САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

на правах рукописи

Мозговой Максим Владимирович

Машинный семантический анализ русского языка и его применения

Специальность 05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель —
доктор физико-математических наук,
профессор Тузов В.А.

Санкт-Петербург – 2006

Оглавление

Оглавление 2
Введение 4
О структуре диссертации 7
Глава 1. Задача формализации естественного языка 10
Формальные грамматики Н. Хомского 11
Модель «смысл ó текст» И. Мельчука 15
Семантический анализатор В. Тузова 18
Глава 2. Семантический анализ в вопросно-ответных системах 27
Принципы организации простой вопросно-ответной системы 30
Классификация вопросительных предложений 32
Примеры вопросов и ответов 43
Глава 3. Семантический анализ в задачах информационного поиска и рубрикации 44
Поиск и рубрикация с помощью словарей классов 47
Усовершенствованный механизм поиска 49
Дополнительные возможности для существующих поисковых машин 52
Глава 4. Спеллчекер и тезаурус 59
Семантический анализатор как модуль проверки правописания 60
Контекстно-ориентированный тезаурус на основе семантического анализатора 62
Глава 5. Поиск частично совпадающих документов и выявление плагиата 65
О задаче выявления плагиата и поиске частичных совпадений 65
Технические особенности систем выявления плагиата 67

Использование семантического анализатора в задаче выявления плагиата 70
Глава 6. Введение в машинный перевод 74
Морфологический и синтактико-семантический уровни анализа текста 76
Семантический уровень анализа текста 82
Адаптация семантического анализатора для различных языков 83
Схема простейшей системы машинного перевода на основе семантического анализатора 87
Практический пример: русско-финский перевод 90
Глава 7. Технические детали 97
Текущая реализация семантического анализатора и её перспективы . 97
Формат выходных данных семантического анализатора 98
Заключение 107
Литература 110

Введение

Обработка текстов на естественных языках (natural language processing, NLP) — тема, не теряющая своей актуальности на протяжении десятилетий. Системы информационного поиска, диалоговые системы, инструменты для машинного перевода и автореферирования, синтезаторы речи (выполняющие хотя бы базовую интонационную разметку), рубрикаторы и модули проверки правописания так или иначе выполняют анализ текстов, написанных на естественных языках. Важность и амбициозность задачи научить вычислительную машину понимать человеческий язык привлекала внимание исследователей уже на заре компьютерной эры. Так, в 50-х годах появляются первые публикации о системах машинного перевода [1]. В наши дни успехи направления можно охарактеризовать как переменные. С одной стороны, почти все коммерческие текстовые редакторы содержат спеллчекер, а переводчики фирмы Promt [2] успешно рекламируются и продаются. С другой стороны, несовершенство существующих систем проверки правописания и низкое качество машинного перевода общеизвестны.

В книге [3] перечисляются типичные этапы, проходимые человеком, самостоятельно изучающим иностранный язык: «На первом этапе он, как правило, бурно и эмоционально переживает свои первые успехи в движении от незнания к знанию: всё кажется легко, просто и быстро. На втором этапе работы у него появляются сомнения в себе, в своей памяти, в силе воли и даже в своих способностях, а заодно и в качестве учебников, с которым он работает… На третьем этапе он приходит к выводу и вполне философскому, а именно: “я знаю, что ничего не знаю!” Именно на этом-то этапе и начинается труд по изучению иностранного языка с увлечением, который, подобно спорту, захватывает человека».

По всей видимости, отношение специалистов к таким сложным и трудно формализуемым областям компьютерной науки, как искусственный интеллект, обработка текстов на естественном языке и распознавание образов[1], эволюционирует сходным образом. Первые успехи на игрушечных примерах заканчиваются экзальтированными заявлениями о скором нахождении полного решения задачи. Затем наступает неизбежное разочарование. Переход к более масштабным проектам наглядно свидетельствует: усложнение системы не есть механическое наращивание функциональности. Уметь распознавать десять слов — не значит написать программу, распознающую речь. Самообучаться игре в крестики-нолики ещё не значит уметь учиться игре в шахматы. Основанную на правилах экспертную систему нельзя просто так расширить в несколько раз, механически добавляя новые и новые правила.

Изучая литературу, можно убедиться, что первые два этапа эволюции отношения к задачам сферы искусственного интеллекта пришлись, соответственно, на 50-60-е и 70-80-е годы прошлого столетия[2]. Вот лишь некоторые цитаты:

- 1956г.: «Задача заключается в том, чтобы работать на основе предположения, что любой аспект обучения или другой функции разума может быть описан так точно, чтобы машина смогла его симулировать. Мы попытаемся определить, как сделать так, чтобы

---

[1] И обработка текстов, и распознавание образов тоже могут быть отнесены к задачам искусственного интеллекта в широком смысле слова.

[2] Разумеется, это не означает, что все фундаментальные работы по обработке естественного языка и искусственному интеллекту были сделаны в 50-60-е, а последующие исследователи лишь критиковали предшественников. Речь здесь идёт лишь об общей тенденции.

машины могли пользоваться языком, формулировать абстракции и концепции, решать задачи, которыми сейчас занимаются только люди, а также заниматься самообучением» [4].

- «В 1960-е гг. <…> сильный ИИ[3] продолжал оставаться главной темой в разработках ИИ» [5].
- «Первая публичная демонстрация переводящего устройства имела колоссальный успех. Это был знаменитый Джорджтаунский эксперимент, проведенный в Нью-Йорке в 1954 году. Тогда все смотрели на возможности компьютерного перевода сквозь розовые очки. Профессиональным переводчикам пророчили в недалеком будущем голодную смерть. Однако вскоре выяснилось, что многие аспекты языка чрезвычайно далеки от формализации, необходимой для успешной работы компьютера с текстом. Многие проблемы казались неразрешимыми, и интерес к машинному переводу сильно упал» [6].
- «1970-е гг. показали резкий спад интереса к ИИ после того, как исследователям не удалось выполнить нереальные обещания его успеха». «1980-е продемонстрировали как рост, так и спад интереса к ИИ. Основной причиной этого были сбои экспертных систем <…> Также были идентифицированы ограничения в работе экспертных систем, поскольку их знания становились всё больше и сложнее» [5].

Начиная с 90-х годов XX века отношение к задачам искусственного интеллекта вообще и к обработке текстов на естественном языке в частности становится всё более прагматичным. Если не удаётся сделать компьютер интеллектуальным, пусть он поступает разумно хотя бы в чём-

---

[3] То есть программное обеспечение, благодаря которому компьютеры смогут думать так же, как люди.

либо. Если не получается создать полноценную систему перевода, пусть автоматический переводчик станет помощником переводчика-человека. Если нельзя добиться большего, пусть программа, анализирующая отсканированный текст, распознаёт хотя бы печатные буквы.

Данная работа посвящена изучению возможных применений *семантического анализатора*, созданного проф. В. Тузовым. Семантический анализатор, с одной стороны, позволяет сравнительно малыми усилиями повысить качественный уровень решений многих задач сферы NLP (что вполне согласуется с современным подходом: если не удаётся достичь революционного улучшения, сделайте хотя бы шаг вперёд). С другой стороны, принципы, заложенные в семантический анализатор, теоретически позволяют добиться весьма значительных результатов, хотя и ценою гораздо больших затрат времени и труда.

О структуре диссертации

Первая глава знакомит читателя с формальными моделями естественного языка. Попытки строго научного описания языков предпринимаются, по крайней мере, с пятидесятых годов XX века (если не считать единичных работ XIX столетия и даже более раннего времени). Лишь немногие из них, однако, оказали существенное влияние на современное состояние NLP. Мы рассмотрим три возможных подхода: грамматики Хомского как наиболее влиятельную модель, оказавшую большое воздействие на теорию компиляции, модель «смысл **ó** текст» И. Мельчука, охватывающую самые разные пласты языкознания, и функциональную теорию языка В. Тузова, на основе которой был разработан семантический анализатор. Теории, посвящённые частным аспектам языка (морфологии, синтаксису) в работе не рассматриваются.

Вторая глава иллюстрирует, как семантический анализатор может быть применён в задаче разработки вопросно-ответных систем, предназначенных для организации полноценного интерфейса на естественном языке между человеком и компьютером. Во второй главе также рассматривается классификация вопросительных предложений, имеющих смысл в контексте диалога с компьютером.

Третья глава посвящена задачам информационного поиска и рубрикации документов. Современные системы поиска и рубрикации обычно основываются на статистическом анализе текстов и анализе различных эвристических показателей (таких как популярность документа и количества ссылок на него, если речь идёт о странице в интернете). Это делает используемые алгоритмы независимыми от языка документов, но не позволяет использовать информацию, напрямую заложенную в слова. Семантический анализатор способен сделать поиск более интеллектуальным, что доказывается на примерах применения *словаря классов* и *деревьев разбора предложений*.

В четвёртой главе описывается механизм использования семантического анализатора в задачах проверки правописания и подбора синонимов слов. Семантический анализатор основан примерно на тех же принципах, что и компилятор языка программирования, поэтому (в частности) проверка правильности структуры входных предложений является его прямой задачей. Кроме того, в состав анализатора входит *семантический словарь*, которым можно воспользоваться как словарём синонимов.

В пятой главе рассматривается задача поиска частично совпадающих документов и выявления плагиата. Алгоритмы, разработанные для её решения, оказываются особенно эффективными при обработке

информации, имеющей некоторую структуру. Неструктурированные данные приходится сравнивать достаточно простыми средствами, в то время как для файлов, поддающихся структурному анализу, можно создать более качественную специализированную процедуру. Семантический анализатор способен структурировать тексты на естественном языке, расширяя возможности для разработки эффективных алгоритмов их сравнения.

В шестой главе изучается возможный подход к решению задачи машинного перевода с помощью семантического анализатора. Машинный перевод изобилует неожиданными трудностями, поэтому говорить о возможности полноценного его осуществления с помощью применения какой-либо технологии не приходится. Однако принципы, на которых основан семантический анализатор, позволяют естественным образом решать задачи, оказывающиеся весьма сложными для других методов построения автоматизированных систем перевода.

Седьмая глава фокусирует внимание на некоторых технических аспектах, связанных с использованием семантического анализатора. Анализатор представляет собой сложную систему, предназначенную для решения нетривиальных задач, и способ его общения с внешним миром сам по себе заслуживает внимания. Также здесь обсуждаются перспективы развития семантического анализатора как программного продукта.

Глава 1. Задача формализации естественного языка

Очевидная сложность разбора текстов, написанных на естественных языках, вызвана их неформальностью. Даже задача анализа формального языка (возникающая, например, при компиляции программ на Паскале) не является тривиальной: она была полностью решена лишь в 60-70-е гг. после появления работ Н. Хомского, Т. Бэкуса, А. Ахо, Дж. Ульмана и других. Неудивительно, что проблема приведения предложений естественного языка к строгому формальному виду изучается со времён возникновения NLP.

Среди наиболее известных и влиятельных работ, посвящённых формальному описанию языков, можно выделить теорию формальных грамматик Н. Хомского [7] и модель «смысл ó текст» И. Мельчука [8]. Идеи Хомского лежат в основе известных алгоритмов анализа текстов компьютерных программ. Многие синтаксические анализаторы англоязычных текстов (см., напр., [10]) так или иначе используют грамматики Хомского. Модель Мельчука изначально предназначалась для изучения проблемы формализации естественных языков, и на теорию компиляции влияния не оказала. Кроме того, как отмечается в работе [11], модель «смысл ó текст» не обладает достаточной языковой независимостью (т.е. ориентирована на русскоязычные тексты), что препятствует её распространению на Западе. Однако в работах российских исследователей по теоретической и компьютерной лингвистике результаты Мельчука широко цитируются.

Как уже было указано выше, данная работа базируется на модели формализации русского языка, предложенной проф. В. Тузовым [12]. В настоящее время подход В. Тузова ещё не получил широкого

распространения, но программные продукты, его использующие, уже существуют (см. [13] и [14]). В следующем разделе мы кратко ознакомимся с идеями Н. Хомского и И. Мельчука и чуть подробнее — с основными положениями теории В. Тузова. Знакомство с базовыми принципами работы семантического анализатора необходимо для понимания дальнейшего материала работы; модели же Хомского и Мельчука описаны с целью доказательства оригинальности теории В. Тузова[4].

В настоящее время существуют также проекты по созданию универсальных языков представления текстов (см., напр., [16]), однако алгоритмы преобразования текстов на естественных языках к подобному универсальному виду и наоборот ещё очень несовершенны и в данной работе не рассматриваются.

Формальные грамматики Н. Хомского

В книге [17] формальные грамматики определяются следующим образом. Если V — некоторый алфавит (то есть множество символов), то:

- V+ — множество всех цепочек (строк) над алфавитом V без пустой цепочки λ;
- V* — множество всех цепочек над алфавитом V, включая λ.

Грамматика состоит из множества терминальных символов VT, множества нетерминальных символов (переменных) VN, множества правил вывода грамматики P и начального символа S ∈ VN. Для грамматик наиболее

---

[4] Некоторые авторы (см., напр., [15]) несправедливо утверждают, что теория В. Тузова основана на упрощённых идеях Мельчука.

общего типа 0 правила вывода имеют вид $\alpha \rightarrow \beta$, где $\alpha \in (VN \cup VT)+$, $\beta \in (VN \cup VT)*$. В наиболее часто применяемых на практике контекстно-свободных (тип 2) грамматиках используются лишь правила вида $A \rightarrow \beta$, где $A \in VN$, $\beta \in V*$.

Любая грамматика описывает некоторый язык (то есть конечное или бесконечное множество строк над алфавитом VT), и может использоваться как для проверки принадлежности данной строки языку, так и для генерирования строк описываемого языка.

Синтаксическая структура, например, английского языка может быть описана с помощью формальных грамматик следующим образом:

- терминальными символами являются отдельные слова языка («the», «cat», «to», «walk»);
- в качестве нетерминальных символов указываются члены предложения («подлежащее», «сказуемое», «дополнение»), грамматические структуры («инфинитивный оборот», «сложное дополнение») и отдельные части речи («глагол», «существительное», «причастие»);
- начальным символом грамматики является символ «предложение».

Правила вывода — наиболее сложная часть грамматики. Некоторые правила лежат на поверхности:

СУЩЕСТВИТЕЛЬНОЕ → **cat | dog | table | ... (**существительные языка**)**[5]
ПРЕДЛОЖЕНИЕ → ПОДЛЕЖАЩЕЕ СКАЗУЕМОЕ ДОПОЛНЕНИЕ
ПРЕДЛОЖЕНИЕ → ПОДЛЕЖАЩЕЕ СКАЗУЕМОЕ ДОПОЛНЕНИЕ ОБСТОЯТЕЛЬСТВО
...

[5] Выражение $A \rightarrow a \mid b$ является сокращённой формой записи $A \rightarrow a$, $A \rightarrow b$.

Однако в целом разработка правил вывода даже для любого языка программирования, не говоря уж о естественных языках, — задача весьма непростая.

Если интересующий нас язык задан контекстно-свободной грамматикой, всегда можно применить какой-либо общий алгоритм разбора (например, Кока-Янгера-Касами [18]) для определения принадлежности изучаемой строки языку. Если строка опознаётся как корректная, алгоритм разбора строит дерево, иллюстрирующее последовательность применения правил вывода к начальному символу, приводящую к получению входной строки (см. рис. 1.1).

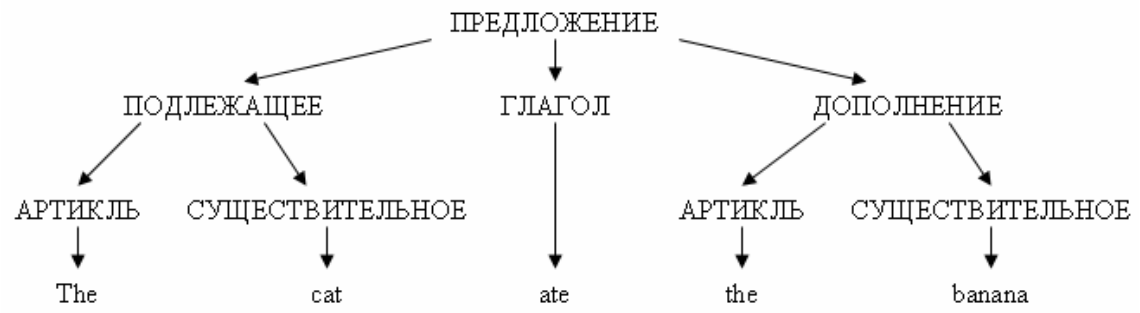

Рис. 1.1. Дерево разбора предложения «The cat ate the banana».

На практике синтаксические анализаторы английского языка нередко до сих пор используют эвристические статистические алгоритмы[6] (см., напр., библиотеку OpenNLP [19]). По всей видимости, задача нахождения полного набора формальных правил вывода для английского

[6] К сожалению, ситуация с русским языком ничуть не лучше.

языка так ещё и не решена[7]. Впрочем, независимо от способа получения дерева, общая идея его организации остаётся неизменной (см. рис. 1.2).

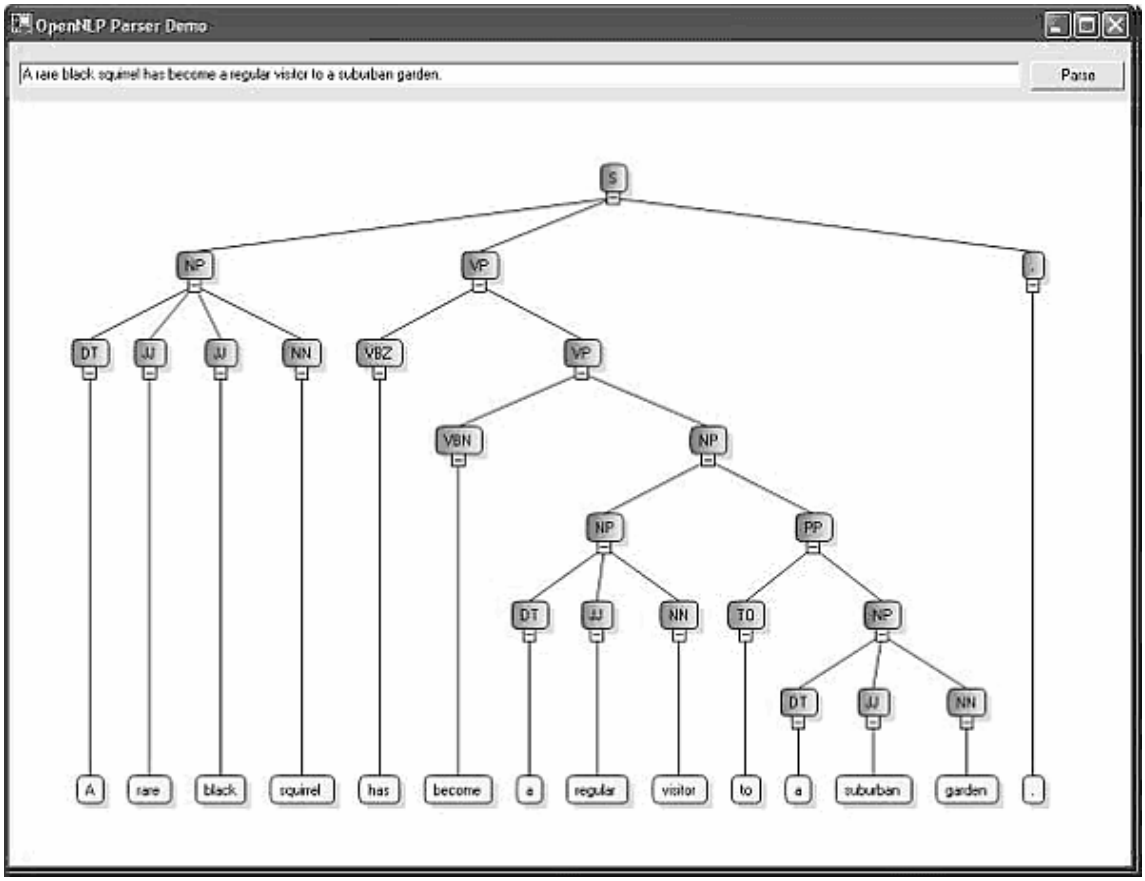

Рис. 1.2. Дерево разбора предложения, полученное при помощи OpenNLP.

Названия нетерминальных символов в современных синтаксических анализаторах обычно совпадают с обозначениями, используемыми в проекте Penn Treebank [20]. В частности:

---

[7] Полагая модель В. Тузова более совершенной, задачу описания естественного языка при помощи грамматик Хомского можно считать в принципе малополезной; однако и у этого подхода есть свои приверженцы.

S – начальный символ **(sentence)**

NP – именная группа **(noun phrase)**

VP – глагольная группа **(verb phrase)**

PP – предложная группа **(prepositional phrase)**

JJ – прилагательное **(adjective)**

NN – noun, singular (существительное**,** единственное число**)**

DT – determiner (определяющее слово**)**

VBZ – verb, 3rd person s.p. (глагол**, 3-**е л.**,** наст. время**,** ед.ч.**)**

TO – particle to (частица **to)**

Заметим, что грамматики Хомского предназначены, прежде всего, для описания структуры предложения. Не менее важный вопрос описания смыслов отдельных слов остаётся за пределами их возможностей.

Модель «смысл ó текст» И. Мельчука

Модель «смысл **ó** текст», описанная впервые в работе [8], оказала большое влияние на взгляды лингвистов на формализацию естественных языков. Согласно [9], центральный постулат теории «смысл **ó** текст» гласит: «Естественный язык есть система, устанавливающая соответствия между любым заданным смыслом и всеми выражающими его текстами; соответственно, лингвистическое описание некоторого языка должно представлять собой множество правил, ставящих в соответствие всякому смыслу все тексты данного языка, несущие этот смысл».

В основе модели лежит, в первую очередь, так называемый «толково-комбинаторный словарь современного русского языка». И. Мельчук утверждает, что толково-комбинаторный словарь «претендует на чисто НАУЧНОЕ описание русской лексики, не делая никаких уступок педагогическим, коммерческим, типографским и тому подобным соображениям» [9]. Действительно, данный словарь является смелой

попыткой применить математические стандарты определений в языкознании. Наиболее важными особенностями словаря Мельчука являются *активность* и *формализованность*.

Под активностью понимается возможность использования словаря для синтеза текста (от смысла к предложению). В противоположность этому, другие толковые словари ориентированы на анализ (выявление смысла того или иного слова в данном предложении). В частности, в словарные статьи включаются сведения о том, как выбрать слово, подходящее к текущему контексту, и о том, как правильно скомбинировать элементы фразы.

Под формализованностью подразумевается способ описания понятий в стиле математики. Сложные понятия определяются через более простые с помощью строгого формального языка, похожего на язык формулировки теорем. На базовом уровне перечисляются «элементарные смыслы» (аксиомы), не подлежащие дальнейшей трактовке. Такое устройство толково-комбинаторного словаря делает его пригодным (по крайней мере, в теории) и для использования компьютером.

Важно отметить, что в каждой словарной статье описывается не отдельное слово, а так называемая *пропозициональная форма*, состоящая из лексемы (некоторого значения слова) и переменных, составляющих её контекст. Например, слову «восхищаться» соответствует, в частности, такая запись: «X *восхищается* Y-ом = X находится в достаточно интенсивном положительном эмоциональном состоянии, каузированном тем, что X считает воспринимаемый им Y очень хорошим <…>».

При составлении словаря нередко используются лексические функции, отделяющие некоторый смысл от его конкретного воплощения в слове. Так, отмечается, что в сочетаниях «тяжёлая болезнь» и «бурные

аплодисменты» слова «тяжёлая» и «бурные» по существу обозначают одно и то же — интенсификацию процесса. Введение лексической функции «интенсификация» (Magn) позволяет добиться большей степени обобщённости при описании слов. На уровне словаря сочетание «тяжёлая болезнь» будет описано как Magn(болезнь), а сочетание «бурные аплодисменты» — как Magn(аплодисменты). Всего в словаре используется несколько десятков лексических функций.

Помимо словаря, в модель «смысл ó текст» входит целая иерархия моделей, описывающих естественный язык на разных уровнях — от фонетики и морфологии до семантики и уровня концепций. Среди этих моделей нельзя обойти вниманием теорию *линеаризации синтаксических деревьев*, ярко иллюстрирующую активность и формализованность модели Мельчука в противовес большинству классических подходов лингвистики. Теория линеаризации предлагает набор правил, с помощью которых можно преобразовать синтаксическое дерево предложения в корректную фразу на русском языке. В отличие от модели Мельчука, в грамматиках Хомского порядок слов указывается непосредственно, поэтому проблема линеаризации вообще не возникает. Однако тем самым резко снижается выразительная мощь модели. Для английского языка с его строгим порядком слов в предложении ограничения метода Хомского не являются критическими, однако при работе с русским языком их уже нельзя игнорировать.

Правила, разработанные Мельчуком, применяются последовательно: на первом этапе отдельные слова объединяются в словосочетания, на втором этапе словосочетания группируются в члены предложения, и, наконец, на третьем этапе члены предложения используются для составления полноценных предложений (возможен также четвёртый этап, на котором простые предложения вводятся в состав сложной фразы).

Семантический анализатор В. Тузова

Подход В. Тузова изначально был связан с попытками практического решения задачи формализации русского языка. Поэтому в книге [21] указывается, что формализация может считаться осуществлённой лишь тогда, когда разработаны следующие компоненты модели:

- формальный («семантический») язык, позволяющий в строгом однозначном виде записать любое предложение естественного языка;
- семантический словарь, содержащий формальные представления слов естественного языка;
- анализатор, преобразующий текст на естественном языке к формальному виду.

В настоящее время все три компонента для русского языка успешно реализованы, однако формализация не является самоцелью: для построения практических систем, использующих модель Тузова, необходимо ещё суметь разумно применить полученные результаты.

Основное отличие описываемой теории от подходов Хомского и Мельчука заключается в изучении текста на естественном языке как *выполняемой* структуры. Проводится чёткая аналогия между обычным текстом и компьютерной программой: разница лишь в том, что семантика в предложениях естественного языка спрятана гораздо глубже. На этом вопросе необходимо остановиться немного подробнее.

Основные этапы взаимодействия компьютера с текстом на языке программирования (например, с программой на Паскале) показаны на рис. 1.3. Аналогичная схема для текста на естественном языке изображена на рис. 1.4.

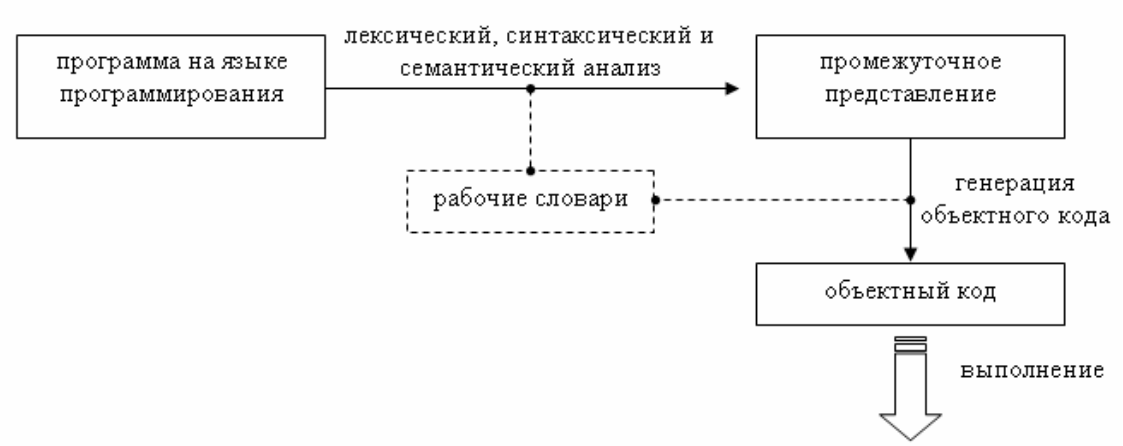

Рис. 1.3. Этапы взаимодействия компьютера с программой (упрощённая схема).

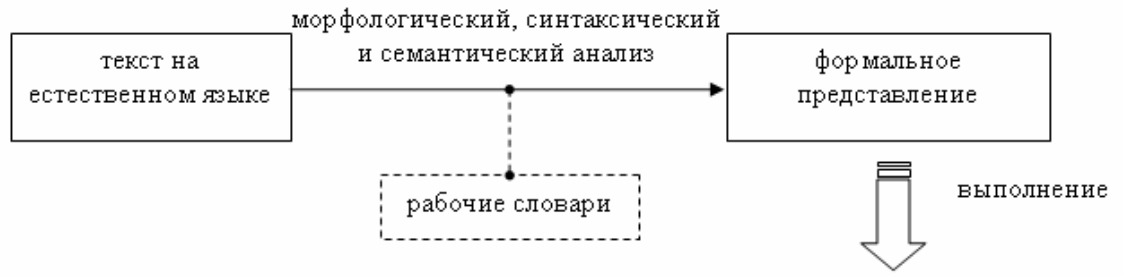

Рис. 1.4. Этапы взаимодействия компьютера с текстом на естественном языке.

Наиболее очевидная сложность в схеме на рис. 1.4 связана с этапом выполнения. Процессор непосредственно умеет выполнять инструкции машинного кода; цель и результат выполнения той или иной команды чётко определены. Для естественного (например, русского) языка смысл «выполнения» не столь очевиден. Ответ на этот вопрос лежит уже в сфере *прагматики*, то есть способа использования естественного языка человеком, а не семантики языка. Цель программы на Паскале (находить корень уравнения, сортировать массив) чётко определена; цель фразы на естественном языке («сарай горит», «Президент поддержал инициативу законодателей») в общем случае неизвестна. Например, может

подразумеваться, что «выполнение» фразы приводит к изображению описываемой ею ситуации на экране компьютера («сарай горит»). С тем же успехом можно полагать, что «выполнение» предложения печатает список найденных в базе данных документов, содержащих сходную информацию. Вероятно, наиболее общим результатом «выполнения» текста можно считать пополнение базы знаний (содержащей легко извлекаемые сведения, почерпнутые из текста).

Поскольку под «выполнением» могут подразумеваться самые различные действия, семантический анализатор не должен переходить в область прагматики, выходя за рамки чистой семантики. Таким образом, суть работы анализатора заключается в получении формального представления текста.

Для того чтобы корректно откомпилировать компьютерную программу, необходимо понимать смысл ключевых слов, таких как *if* или *repeat*. Аналогично, для выполнения текста на естественном языке требуется словарь, содержащий (в частности) описания смысла отдельных слов языка. Заметим, что этот факт с очевидностью следует из аналогии между естественными и компьютерными языками. Никому не приходит в голову создавать компилятор, основанный на статистическом анализе, цепях Маркова или каком-либо виде «поверхностного» семантического разбора. Когда же речь заходит о естественных языках, подобные методы почему-то рассматриваются как вполне подходящие.

Самой объёмной частью анализатора является синтактико-семантический словарь. Его разработка может занять месяцы и годы, так как придётся формально описать все распознаваемые системой слова языка (по определению: программа не может вычислить смысл текста, не зная смысла каждого отдельного слова).

Слова русского языка делятся на две группы: базовые (около 20000 слов в текущей реализации) и производные (около 90000). Базовые слова, такие как *Иванов*, *радость* или *абразив* не выражаются через другие. Для базового слова можно только указать его *класс* (см. табл. 1.1) и, в ряде случаев, краткое формальное описание. Например, слово *абразив* принадлежит классу «поделочные материалы» и описано как «вещество, используемое для обработки металлов»:

```
$12127(Usor_01(ВЕЩЕСТВО$1211,Caus(#,Lab(МЕТАЛЛ$12122,ОБРАБОТКА$15131)))
```

Таблица 1.1. Примеры базовых слов

Слово	Класс
Иванов	Существительное / Физический объект / Живой / Человек / Индивид / ФИО / Фамилия
радость	Существительное / Психика / Душа / Чувства / Довольство / Радость
абразив	Существительное / Физический Объект / Неодушевлённый / Материалы / Поделочные

Классификация объектов — задача сложная и неоднозначная. Можно разработать альтернативную классификацию (вплоть до того, чтобы разбить все объекты на три класса — «маленькие», «большие» и «неопределённого размера») и аргументировать её превосходство. На практике всё определяется изучаемой предметной областью: классы лишь сообщают некоторые сведения о словах, доступные для дальнейшего использования. Если предметная область требует разделения всех объектов на «большие» и «маленькие», систему классов можно адаптировать, не внося при этом никаких изменений в программу семантического анализа. Используемая в настоящее время система, состоящая более чем из 1600

классов (см. рис. 1.5), претендует если не на универсальность, то, по крайней мере, на применимость в типичных случаях.

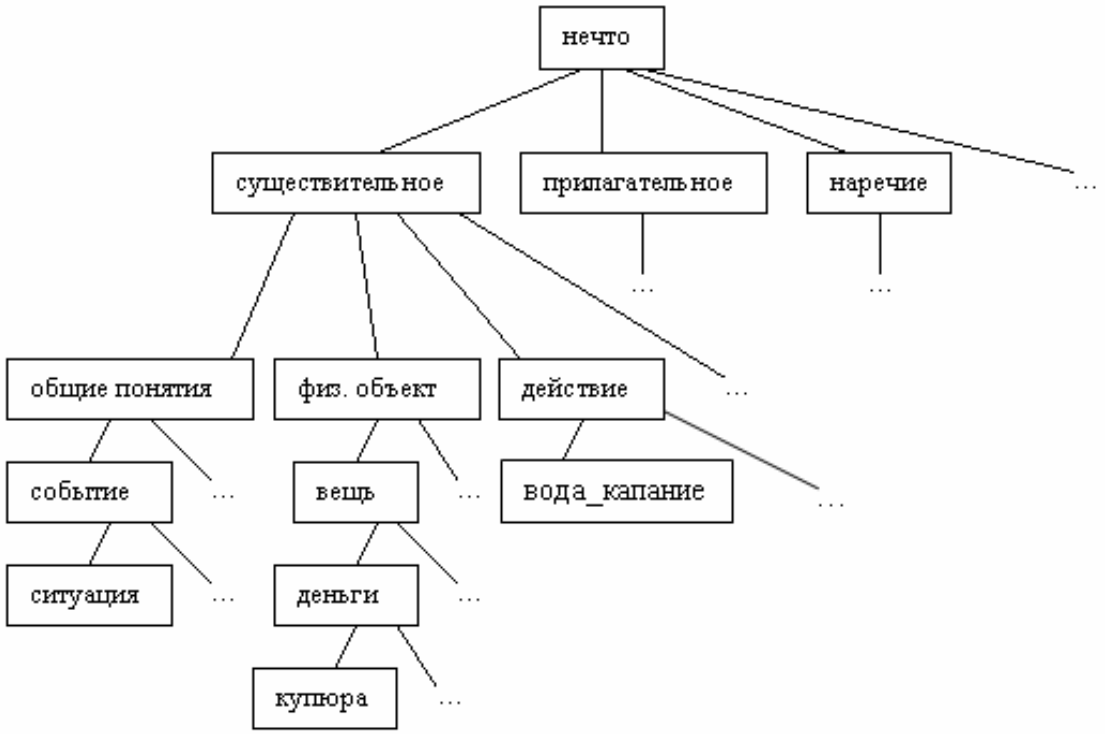

Рис. 1.5. Фрагмент иерархии классов базовых понятий.

Производные слова определяются через базовые путём применения к ним базисных функций[8] (см. табл. 1.2, табл. 1.3).

[8] Это не отменяет необходимости приписывать производным словам соответствующие им классы.

Таблица 1.2. Некоторые базисные функции[9]

Название	Описание
Caus(*x*, *y*)	*x* делает так, чтобы *y* (*x* каузирует *y*)
Copul(*x*, *y*)	*x* есть *y*
Lab(*x*, *y*)	*x* подвергается действию *y*
Ne(*x*)	отрицание *x*

Таблица 1.3. Примеры производных слов

Название	Описание
жечь	Caus(*z1*, Lab(*z2*, ОГОНЬ))
аморфный	NeHab(*z1*, ФОРМА)
мулатка	Copul(МУЛАТ, ЖЕНЩИНА)

Таким образом, формулу слова «жечь» можно интерпретировать следующим образом: *z1* делает так, что *z2* подвергается действию огня. Аналогично, «аморфный» — это «*z1*, не имеющий формы», а «мулатка» — это «мулат, который является женщиной». Словам, имеющим несколько значений, сопоставляется несколько описаний в семантическом словаре. В процессе анализа будет выбрана альтернатива, согласующаяся с реально встретившимся контекстом. Контекст определяется при помощи синтактико-семантической формулы слова, в которой всегда указываются

[9] Названия базисных функций в словаре Тузова позаимствованы у Мельчука, однако сами функции здесь интерпретируются не как лексические, а как смысловые операции над понятиями, предназначенные для получения новых понятий.

классы и падежи сочетающихся с ним слов. Например, описание глагола *разбить* включает в себя, в частности, следующие строки:

```
РАЗБИТЬ      PerfCaus(Z1,IncepPrepar(Z2,ИСПОЛЬЗОВАНИЕ$15301(Z3)))
             ПРИГОТОВЛЕНИЕ$152310(Z1: !Им,Z2: !Вин,Z3: !Для)
РАЗБИТЬ      PerfCaus(Z1,FinFunc(Lab(Z2,УДАР$1520310(Z3))))
             РАЗБИВАНИЕ$11031(Z1: !Им,
                  Z2: ВЕЩЬ$1213\ОКНО$123372~!Вин,Z3: !Тв\!обВин)
```

Описание каждой трактовки слова состоит из двух формул — смысловой и синтактико-семантической. Смысловая формула объясняет значение слова. В процессе анализа предложений она не используется, однако описание смысла может пригодиться при дальнейшей работе с разобранным текстом, например, при пополнении базы знаний. Синтактико-семантическая формула непосредственно нужна анализатору для построения правильного дерева разбора.

Смысловая формула из первого описания определяет слово *разбить* как «приготовить для дальнейшего использования» («разбить палатку»). Синтактико-семантическая формула указывает, что слово *разбить* принадлежит классу ПРИГОТОВЛЕНИЕ$152310 и сочетается с именительным (кто разбивает) и винительным (что разбивает) падежами, а также с предлогом *для* (для чего разбивает). Аналогично, смысловая формула из второго описания определяет слово *разбить* как «прекратить существование объекта вследствие удара» («разбить чашку»). Соответствующая синтактико-семантическая формула сообщает, что слово *разбить* принадлежит классу РАЗБИВАНИЕ$11031 и сочетается с именительным (кто разбивает), винительным (что разбивает) и творительным (чем разбивает) падежами. Вместо творительного может также встретиться винительный падеж с предлогом *об*/*обо* (обо что разбивает). В формуле также указано, что слово, соответствующее винительному падежу, должно принадлежать классу ВЕЩЬ$1213, либо

классу ОКНО$123372. Более подробно устройство словаря обсуждается в шестой главе.

Итоговая распечатка, предоставляемая анализатором, состоит из двух частей. Первая часть описывает структуру каждого предложения в виде суперпозиции слов-функций. Вторая часть содержит информацию об отдельных словах предложений, взятую из синтактико-семантического словаря.

Идея представления предложения в виде суперпозиции слов продолжает аналогию между естественными языками и языками программирования. Выражаясь точнее, в соответствии с моделью В. Тузова предложения естественного языка имеют *функциональную* природу, то есть адекватно описываются с помощью суперпозиции функций (аналогичным образом устроены функциональные языки программирования, такие как Haskell и Standard ML). Так, предложение *Иван купил у Петра книгу* преобразуется семантическим анализатором к следующему виду:

```
купил(Иван, у(Петра), книгу)
```

Полученную информацию можно использовать различными способами, но, повторимся, первоначальная идея заключается в *выполнении* (или *вычислении*) предложений текста, представленных в виде семантических формул.

Во введении уже упоминалось, что семантический анализатор может применяться и как дополнительный механизм, улучшающий качество работы какой-либо существующей системы, и как мощный фундамент для нового проекта, способный в конечном итоге радикально улучшить достижения в некоторых областях NLP. «Косметические» улучшения достигаются сравнительно малыми усилиями, что делает применение

анализатора ещё более привлекательным. Данная работа посвящена в основном изучению способов внедрения семантического анализатора в существующие проекты. Пересмотр всего подхода к решению задачи с использованием теории В. Тузова в качестве фундамента требует солидных трудозатрат, но может окупиться впечатляющими достижениями. Далее мы рассмотрим как теоретические возможности для новых проектов в сфере NLP, так и некоторые практически полученные результаты. Однако говорить о каком-либо по-настоящему серьёзном законченном проекте, полностью основанном на функциональной модели В. Тузова, ещё рано.

Глава 2. Семантический анализ в вопросно-ответных системах

Вопросно-ответная система — это программа, которая умеет обрабатывать введённый пользователем вопрос на естественном языке и печатать осмысленный ответ. Таким образом, вопросно-ответные системы образуют подмножество хорошо известных диалоговых приложений, к которым относятся такие разные примеры, как командная строка DOS, администратор СУБД на основе языка SQL или экспертная система. Некоторые диалоговые системы (восходящие к знаменитой программе ELIZA [22]) пытаются создать атмосферу полноценного «общения» между человеком и компьютером на естественном языке. На практике «интеллектуальность» большинства подобных программ ещё весьма неудовлетворительна; с другой стороны, не каждый текстовый интерфейс нуждается в мощной поддержке технологиями ИИ.

Поскольку здесь обсуждаются вопросы использования семантического анализатора В. Тузова, мы остановимся лишь на системах, которые занимаются обработкой запросов, сформулированных на естественном языке. Как правило, под «обработкой» подразумевается поиск релевантной запросу информации в некотором банке данных. Эта задача родственна, но не идентична *информационному поиску* (information retrieval). В задаче поиска результатом обработки запроса является список релевантных документов, то есть считается, что пользователь ищет некоторый *документ*. В вопросно-ответных системах в качестве результата выводится только конкретный ответ на конкретно поставленный вопрос («Когда начнётся матч?» — «в 18:00 по московскому времени»).

В работе [23] вопросно-ответные системы характеризуются «уровнями понимания». На первом уровне ответы формируются только на основе прямого содержания текста: система сумеет извлечь релевантные данные, только если в вопросе присутствуют те же синтаксические конструкции, которые уже присутствуют во входном тексте. На втором уровне к механизму системы добавляются средства логического вывода, позволяющие делать заключения о месте и времени происходящих событий, казуальной зависимости и т.п. Третий уровень вносит стереотипные знания о ситуациях реального мира, позволяющие «достраивать» базу фактов на основании «здравого смысла», то есть знаний более общего рода.

В настоящее время вопросно-ответные системы, как правило, остаются на уровне экспериментальных разработок. Описанные выше «уровни понимания» к существующим проектам могут быть применены лишь условно: одни системы умеют делать умозаключения, другие систематизируют знания об окружающем мире (вспомним, например, классическую программу SHRDLU [24]). Однако о полноценных вопросно-ответных системах, реализующих полностью первый и хотя бы в некоторой степени второй уровни понимания, говорить пока не приходится.

Лучшие из существующих проектов, ориентированные на английский язык (см., напр., [25] и [26]), используют связку из синтаксического анализатора, словаря и статистического модуля. Для английского языка уже существуют свободно распространяемые синтаксические анализаторы [19], [27] и электронные словари (самым известным из которых является WordNet [28]). Синтаксические анализаторы, как правило, используют уже упомянутую систему разметки Penn Treebank [20]. Словарь WordNet по сути является толковым словарём,

содержащим, помимо объяснений смысла слов (в виде обычного английского текста), некоторую дополнительную информацию, такую как тип слова (часть речи), список синонимов, список других форм того же слова (актуально для английских глаголов) и т.п. Хотя разметка слов выдаётся в хорошо структурированном виде, удобном для анализа компьютером, всё-таки WordNet ориентирован в первую очередь на использование человеком. При анализе слов не учитывается контекст, и распечатка данных, скажем, по слову *table* включает в себя шесть значений, пять из которых относятся к существительным и одно — к глаголам. Список синонимов наряду с пользой добавляет сложностей. Например, согласно WordNet синонимами слова *table* являются такие разные понятия, как *mesa* (столовая гора) и *tabular array* (таблица).

Здесь уделено так много внимания словарю WordNet лишь с одной целью: показать типичные проблемы, связанные с автоматическим использованием стандартных словарей. Обычно их пытаются (с той или иной степенью успеха) решать с помощью статистических методов (см., напр., сборник статей по задаче определения корректного смысла слов [29]). Синтактико-семантический словарь В. Тузова лишён подобных недостатков. В выражениях «девушка с русой косой» и «девушка со стальной косой» анализатор с помощью словаря выберет правильное значение слова *коса*. Само собой, слово *косой* в качестве прилагательного заведомо не будет фигурировать в распечатке разбора как не сочетающееся с контекстом.

Основной работой, подводящей теоретическую базу под использование семантического анализа в вопросно-ответных системах, является уже цитированное исследование [23]. В статье [30] описывается практический подход к реализации вопросно-ответной системы первого уровня понимания на основе модели В. Тузова.

Принципы организации простой вопросно-ответной системы

Перед тем, как изложить основные тезисы [30], отметим, что исследования систем различных уровней понимания могут вестись независимо друг от друга. С одной стороны, первый уровень понимания совершенно необходим для построения вопросно-ответных систем второго и третьего уровней; с другой стороны, механизмы извлечения информации из базы знаний (то есть реализующие высшие уровни понимания) могут разрабатываться независимо, поскольку анализ предложений текста на естественном языке — не единственный возможный способ её пополнения.

Поскольку наши исследования связаны в первую очередь с семантическим анализом, а не с организацией баз знаний, мы в основном ограничивались первым уровнем понимания. Так, система [30] обрабатывает лишь так называемые специальные вопросы (то есть вопросы, задаваемые к тому или иному члену предложения) с вопросительными словами *кто*, *кого*, *кому*, *кем*, *о ком*, *что*, *чего*, *чему*, *чем*, *о чём*, а также *где*, *куда* и *откуда*. В качестве базы знаний выступает обычный текстовый документ, рассматриваемый как последовательность предложений. Чтобы получить корректный ответ, в большинстве случаев достаточно выполнить синтаксический анализ вопросительного предложения, а также предложений входного текста. Часть утвердительного предложения, содержащего ответ, можно найти по падежу. Например, вопросу *кто* соответствует именительный падеж. Однако три вопросительных слова (из рассматриваемых) требуют уже базового семантического разбора. Речь идёт о словах *где*, *куда* и *откуда*. В принципе, можно сказать, что вопросительному слову *где* соответствует предложный падеж, вопросительному слову *куда* — винительный, а

вопросительному слову *откуда* — родительный. Однако на вопросы *где прошла жизнь?*, *куда пошёл Сергей?* и *откуда пришёл Иван?* нельзя ответить сочетаниями *в трудах*, *на риск* и *из вежливости* соответственно, хотя с точки зрения падежей ответы являются корректными. Чтобы выявить, к примеру, принципиальное отличие в значениях сочетаний *из вежливости* и *из ресторана*, необходим семантический анализ. Другим, менее ярким примером использования семантической информации в вопросно-ответной системе, является проверка (с помощью семантического словаря) на одушевлённость. В частности, слово *собака* не может быть ответом на вопрос *что лежит на диване?*

Функционирование системы основывается на некоторых предположениях[10]:

1. Центральное слово (то есть корень дерева) вопросительного предложения является глаголом (*куда собираются экспортировать нефть?* ⇒ собираются(куда, экспортировать(нефть)); *на кого окажет давление Путин?* ⇒ окажет_давление(на(кого), Путин)).
2. Вопросительное слово является первым аргументом центрального слова вопросительного предложения. Специальной обработки требует только случай предложного падежа. Скобочная форма типичного вопроса с конструкцией о ком / о чём выглядит следующим образом: центр_глагол(о(ком), $арг_2$, …, $арг_n$).
3. Центральное слово предложения, содержащего ответ, совпадает с центральным словом вопроса. Один (реже несколько) аргумент центрального слова этого предложения является собственно ответом на

[10] Вытекающих из более общих выводов о структуре вопросительных предложений, см. раздел «Классификация вопросительных предложений».

вопрос. В случае «падежных» вопросов определить требуемый аргумент можно по падежу: падеж корневого элемента аргумента-ответа совпадает с падежом вопросительного слова. Если же вопрос требует семантического анализа, потребуется произвести поиск элемента, к которому этот вопрос может быть применён (семантический анализатор предоставляет такую информацию).

Если в каком-либо предложении входного текста вопросно-ответной системе удалось найти семантическую форму, соответствующую форме запрашиваемой сущности, производится проверка соответствия дерева утвердительной части вопроса (QT) дереву потенциального ответа (AT) по схеме $QT \subseteq AT$ (алгоритм описан в [30]). В случае успеха в качестве ответа печатается подходящий по типу аргумент центрального слова найденного релевантного предложения.

Классификация вопросительных предложений

При разработке экспериментальной вопросно-ответной системы [30] нами была изучена задача классификации вопросительных предложений. Из доступных лингвистических изданий лишь [31] предлагает системный подход к описанию разновидностей вопросов в русском языке. Однако и он представляется недостаточно формальным для непосредственного использования в компьютерных программах. На основе примеров из [31] мы разработали собственную классификацию вопросительных предложений, уделяя особое внимание конструкциям, имеющим практический смысл в контексте вопросно-ответных систем. Как правило, для вопросительных предложений семантический анализатор генерирует простые, удобные при дальнейшей обработке структуры.

Общие вопросы (требующие ответа в форме «да» / «нет»)

Общие вопросы, как правило, образуются при помощи интонации (см. табл. 2.1). Несколько реже используются частицы — либо для акцентирования отдельных слов, либо для придания экспрессивности.

Таблица 2.1. Образование при помощи интонации

Утвердительное предложение	Вопросительное предложение
Он встретил Ваню.	Он встретил Ваню?
Сергей вернулся домой из гостей.	Сергей вернулся домой из гостей?

Для получения утвердительной части образованного при помощи интонации вопросительного предложения достаточно лишь заменить в нём вопросительный знак точкой.

Таблица 2.2. Образование при помощи частиц *ли*, *разве*, *неужели*, *ужели*, *что*, *что ли*

Утвердительное предложение	Вопросительное предложение
Ученик пишет.	Пишет ли ученик? Ученик ли пишет?
Учиться необходимо.	Разве учиться необходимо? Неужели учиться необходимо?
Поезд опять опаздывает.	Что, поезд опять опаздывает (что ли)?
Она красивее тебя.	Что она, красивее тебя (что ли)?
Почтальон тебе письмо принёс.	Что это почтальон — тебе письмо принёс?

Очевидно, что виды вопросов, перечисленные в табл. 2.2, используются в основном в устной речи, и при построении вопросно-ответных систем могут быть вообще проигнорированы без большого ущерба. С другой стороны, вопросительные частицы могут быть легко удалены, и предложение преобразуется к уже рассмотренному виду вопросов, образованных при помощи интонации. Удаление частиц не представляет сложности, поскольку их расположение в дереве разбора известно заранее: частицы *ужели*, *неужели* и *разве* занимают место корневого узла; частицы же *ли*, *что* и *что ли* являются непосредственными потомками корня (см. рис. 2.1).

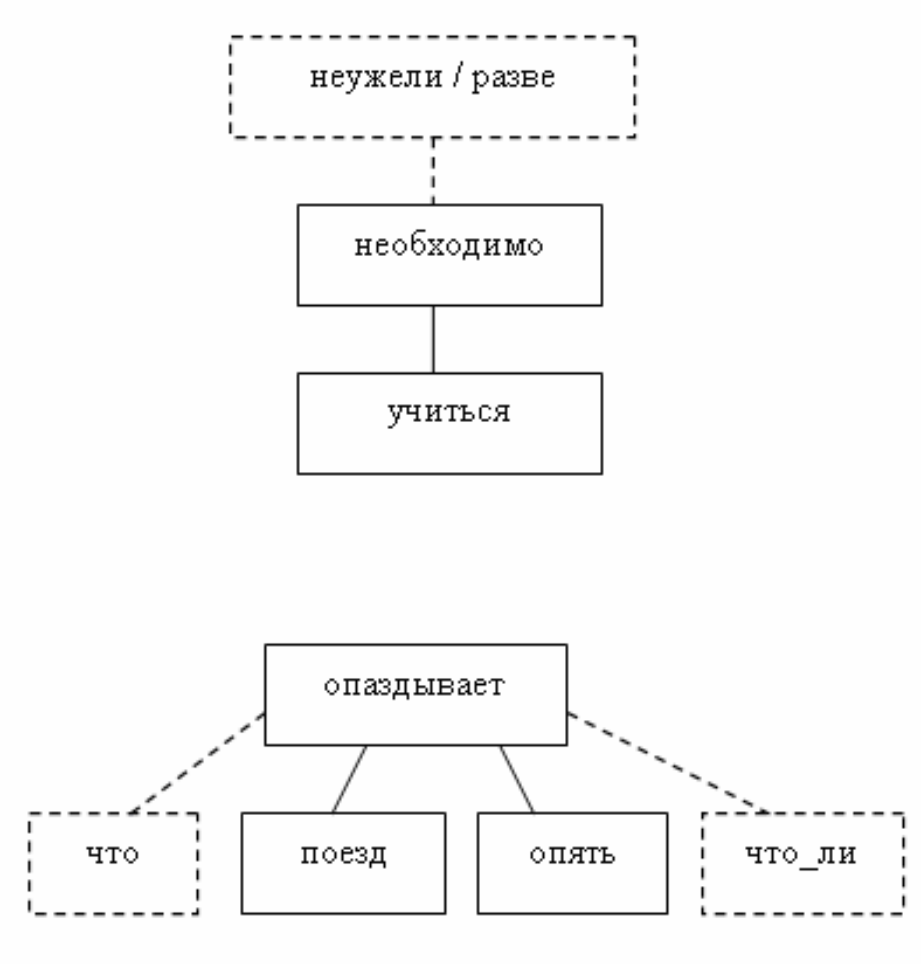

Рис. 2.1. Расположение вопросительных частиц в деревьях разбора.

К общим вопросам относятся ещё два небольших класса: вопросы с *не* и вопросы схемы *не* <предложение> *ли* (см. табл. 2.3, табл. 2.4).

Таблица 2.3. Использование частицы *не* для акцентирования слов

Вопрос без *не*	**Вопрос с *не***
Отец приехал?	Отец не приехал? (*по смыслу вопросы эквивалентны*)

Таблица 2.4. Использование схемы *не* <предложение> *ли*

Образованный интонацией вопрос	**Вопрос по схеме *не* <предложение> *ли***
Вы влюблены?	Не влюблены ли вы?
Он за меня заступится?	Не заступится ли он за меня?
Там дым за рекой?	Не дым ли там за рекой?

Здесь необходимо отметить, что для ответа на вопрос с частицей *не* уже недостаточно ограничиться простой формой «да» / «нет». Ответ должен сопровождаться пояснением:

Ты не встретил Ваню**?** ⇒ Да**,** встретил **(**Нет**,** встретил**) /**
Да**,** не встретил **(**Нет**,** не встретил**).**
Не встретил ли ты Ваню**?** ⇒ Да**,** встретил **/** Нет**,** не встретил**.**

При поиске ответов на специальный вопрос вопросно-ответная система должна извлечь факт, соответствующий запрашиваемой сущности. Отвечая же на общий вопрос, система либо подтверждает, либо опровергает некоторое утверждение. Для первого уровня понимания может быть предложен следующий простой алгоритм. Если найдено

предложение, соответствующее вопросу, результатом будет ответ «да». Если найдено предложение, в котором центральный глагол отрицается, результатом будет ответ «нет». В любом ином случае в качестве результата возвращается значение «неизвестно» (см. табл. 2.5).

Таблица 2.5. Примеры анализа общих вопросов

Вопрос	Предложение из базы знаний	Ответ
Иван встретил отца?	Иван встретил отца.	да
Сергей купил автомобиль?	Сергей не купил автомобиль.	нет
Пётр починил крышу?	Пётр починил дверь. Иван починил крышу.	неизвестно

Специальные вопросы (требующие развёрнутого ответа)

По смыслу специальные вопросы могут быть разделены на несколько различных типов, но с точки зрения семантического анализа эти типы мало различаются. Практически все специальные вопросы образуются с помощью вопросительных слов. К вопросительным словам относятся:

1. местоимения-существительные *кто*, *что*;

2. местоимённые прилагательные *какой*, *который*, *чей*, *каков*;

3. наречия *где*, *куда*, *откуда*, *когда*, *зачем*, *отчего*, *почему*, *как* и *сколько*.

В контексте диалога специальные вопросы могут состоять даже из одного слова: «Придётся ждать. — *Сколько?*» При использовании вопросно-ответных систем первого уровня понимания подобные вопросы не имеют смысла.

Наиболее многочисленная (и наиболее важная в контексте вопросно-ответных систем) группа вопросов образуется при помощи схемы <вопросительное слово> <предложение> (см. табл. 2.6).

Таблица 2.6. Использование схемы <вопросительное слово> <предложение>

Вопросительное предложение
Кто (это) пришёл? Что (это) случилось? Сколько народу собралось? От кого письмо? Какую книгу вы читаете? Что (это) он на меня так смотрит? Почему шум? Откуда сквозит?

Учитывая важность данного типа предложений, имеет смысл рассмотреть их особенности в зависимости от вида вопросительного слова.

1. Деревья вопроса со словом *кто* и предложения, являющегося потенциальным ответом, изображены на рис. 2.2.

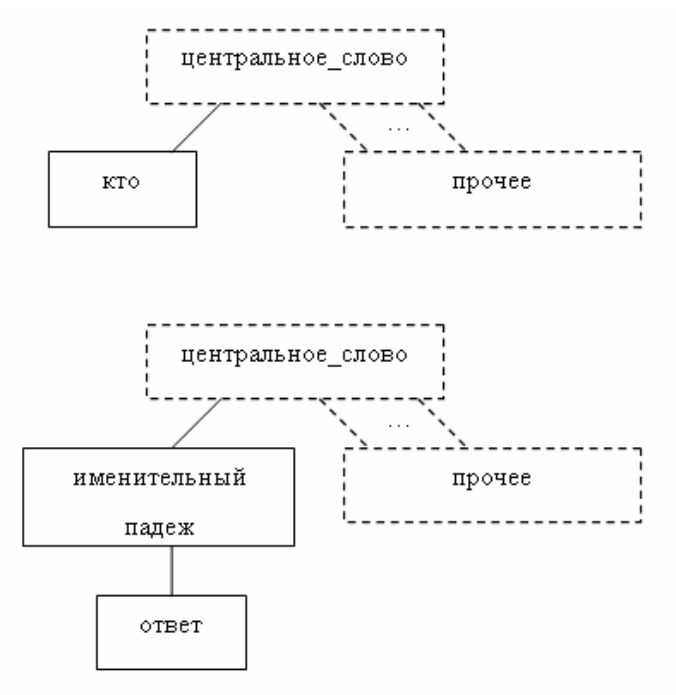

Рис. 2.2. Деревья вопроса со словом *кто* и потенциального ответа на него.

Вопрос со словом *кто* всегда задаётся к именительному падежу. Если в базе знаний найдено предложение, имеющее требуемую структуру, производится дополнительная проверка на соответствие правой части дерева ответа AT правой части дерева вопроса QT (на рисунке они выделены пунктиром) по схеме $QT \subseteq AT$. В случае успеха печатается ответ.

2. Вопрос со словом *что* обрабатывается аналогично, только кроме именительного падежа в дереве потенциального ответа допустимым считается также винительный (*что увидел Саша?*).

3. Вопрос со словом *какой* задаётся к определению. Определение — это свойство (или свойства) объекта, на уровне семантического анализа выделяемое в аргументы слова-функции, соответствующего этому объекту (см. рис. 2.3).

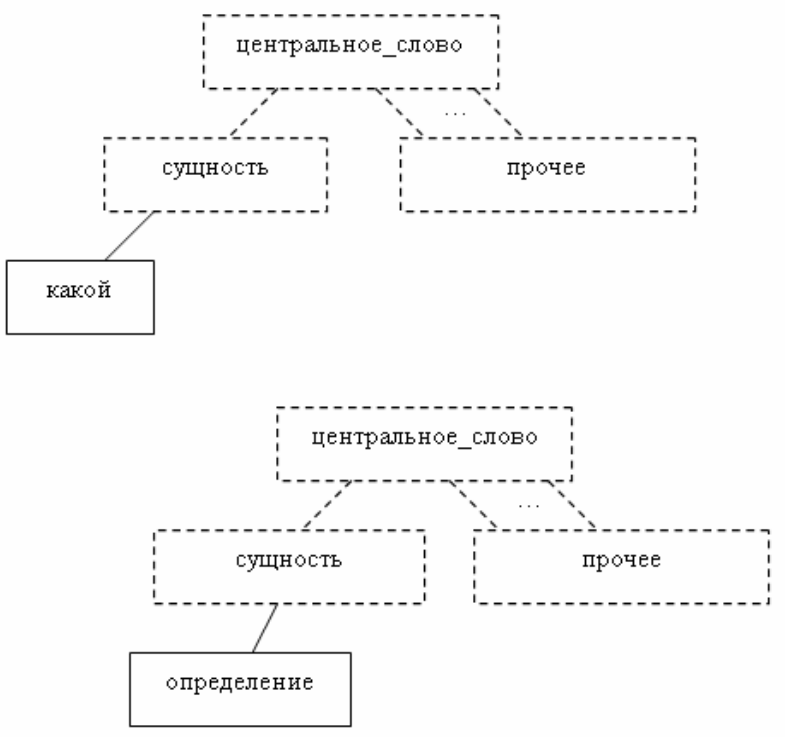

Рис. 2.3. Деревья вопроса со словом *какой* и потенциального ответа на него.

Если дерево ответа соответствует дереву вопроса (QT ⊆ AT), в качестве результата печатаются все найденные определения (В доме стоял итальянский дубовый шкаф с резными ножками. *Какой шкаф стоял в доме?* ⇒ итальянский, дубовый, с резными ножками).

4. Вопросительное предложение со словом *чей* обрабатывается аналогично случаю со словом *какой*. Добавляется единственное новое ограничение: слово, соответствующее запрашиваемой сущности, должно быть одушевлённым.

5. Вопрос со словом *каков* семантически эквивалентен вопросу со словом *какой*. Разница заключается лишь в синтаксисе: *Какой шкаф стоял в доме?* ⇒ Каков был шкаф, стоявший в доме?

6. Если не брать в расчёт устойчивое сочетание *который час* (малоосмысленное в контексте вопросно-ответной системы), слово *который* также используется в качестве замены слову *какой* в некоторых контекстах: *Который из двух братьев — старший?*

7. При поиске ответа на вопрос *где* применяется схема, аналогичная той, которая используется с вопросом *кто*. Разница в том, что корневой элемент поддерева, являющегося ответом, должен иметь вид указания на какое-либо место. В настоящий момент семантический анализатор выявляет такие случаи, помечая корень атрибутом @Где.

8. Вопрос со словом *куда* разбирается аналогично вопросу со словом *где*. Семантический анализатор в этом случае также распознаёт указание на место (атрибут @Куда).

9. Наречие *откуда* в вопросах имеет два смысла. Либо вопрос как-то связан с указанием месторасположения (слово, обратное *куда*), либо слово *откуда* используется как эквивалент конструкций *по какой причине*, *каким образом*. В первом случае (*откуда приехал Иван? —*

Иван приехал из деревни) семантический анализатор вырабатывает атрибут @Откуда. Извлечение ответа при этом аналогично ситуации с атрибутом *куда*. Второй случай менее интересен для вопросно-ответных систем. Вопросы со словом *откуда*, если речь не идёт о месторасположении кого или чего-либо, носят либо ярко выраженный риторический характер, либо не могут быть легко сопоставлены тому или иному утвердительному предложению (необходим контекст диалога): Сколько ему лет? — *Откуда я знаю?*; *Откуда он взял деньги на машину?* — Да вот же.

10. По своей структуре вопрос со словом *когда* аналогичен предыдущим рассмотренным вопросам (*откуда*, *куда*, *где*…) В данном случае запрашиваемая сущность связана со временем, и семантический анализатор вырабатывает для неё атрибут @Когда.

11. Вопросительные слова *зачем*, *отчего* и *почему* встречаются в одних и тех же контекстах и обозначают примерно одни и те же вопросы. Существуют оттенки значений: так, слово *почему* может быть направлено на причину (*почему ты это сделал?*), а слово *зачем* — на цель действия (*зачем ты это сделал?*). Однако в контексте вопросно-ответной системы подобная разница не является слишком существенной. По синтаксической форме вопросы со словами *зачем*, *отчего* и *почему* совершенно аналогичны предыдущим типам вопросов. Поддерево ответа можно определить по словам *чтобы*, *дабы*, *ибо*, *поскольку*, *так как*, *потому что*.

12. Вопрос со словом *как* — это вопрос к наречию или к конструкции, центральным словом которой является наречие. Общая форма вопроса сохраняется неизменной, не меняется и способ поиска ответа. В некоторых случаях слово *как* служит синонимом сочетания *каким*

образом: *Как Сергей вернулся домой?* — Сергей вернулся домой на автобусе. В подобных ситуациях для выявления ответа потребуется найти конструкцию с предложным падежом в корне дерева.

13. Вопрос со словом *сколько* обращён к числительному. В простых случаях общая форма совпадает с ранее рассмотренными: *Сколько лет Ивану?* — Ивану 30 лет. Однако очень часто слова из вопроса не фигурируют в ответе, что затрудняет нахождение соответствия: *Сколько у Петра с собой денег?* — У Петра с собой пятьсот рублей; *Сколько прошло времени?* — Прошло два часа.

Следующая группа вопросительных предложений образуется при помощи инфинитивных конструкций (см. табл. 2.7).

Таблица 2.7. Использование схемы <вопросительное слово> <инфинитивная конструкция>

Вопросительное предложение
Кому сюда прийти? Как нам поступить? Сколько ещё ждать? Куда поставить чемодан? Что же делать? Какой толк спорить?

Для корректной обработки вопросительных предложений этого вида первого уровня понимания уже недостаточно. Кроме того, приведённые вопросы, скорее всего, малополезны для использования в вопросно-ответных системах. Аналогичным образом дело обстоит с вопросами, образуемыми с помощью схем, приведённых на табл. 2.8 – табл. 2.11. Единственное исключение связано с формой <что такое / что значит> <имя / инфинитив>, которая может быть использована для извлечения определений понятий (*что такое «брандмауэр»?*; *что есть «криогенная установка»?*).

Таблица 2.8. Использование схемы <как/что> <имя> (именительный или косвенный падеж, возможно, с предлогом) или <наречие>

Вопросительное предложение
Как здоровье? Как семья? Что на работе? Как с уроками?

Таблица 2.9. Фразеологизированные схемы

Схема	Примеры
<что такое / что значит> <имя / инфинитив>	Что такое справедливость? Что значит верить?
что за <имя в именительном падеже>	Что за шум? Что за разговоры?
<что если / а что если / а если / а вдруг> <предложение>	Что если враг рядом? А что если он не придёт?
что <родительный падеж>	Что новенького? Что пользы плакать?
<что / какое дело / какая забота> <местоимение> *до* <род. падеж>	Что мне до них? Какое дело ему до тебя?

Таблица 2.10. Использование инфинитивного предложения (возможно, с частицей) в качестве вопроса

Вопросительное предложение
Начинать? Спросить? Послать его к вам? Разве уже обедать? Не начинать ли? Неужели начинать?

Таблица 2.11. Использование схемы *a* <имя> / *a* <инфинитивное предложение>

Вопросительное предложение
А бедный брат? А друзья? А к нам? А читать? А петь?

Примеры вопросов и ответов

В завершение раздела уместно привести примеры ответов системы (см. табл. 2.12) на задаваемые пользователем вопросы. Входным текстом будет небольшой фрагмент рассказа Чехова «Поцелуй» (пример взят из [30]):

```
Загремел рояль; грустный вальс из залы полетел в настежь открытые окна, и
все почему-то вспомнили, что за окнами теперь весна, майский вечер.
Все почувствовали, что в воздухе пахнет молодой листвой тополя,
розами и сиренью.
```

Таблица 2.12. Вопросы и ответы

Вопрос	Ответ
Что загремело?	Рояль
Что полетело в окна?	Грустный вальс
Куда полетел вальс?	Из залы
Чем пахнет в воздухе?	Молодой листвой тополя, розами и сиренью

Глава 3. Семантический анализ в задачах информационного поиска и рубрикации

Задача информационного поиска (information retrieval), связанная с алгоритмами быстрого извлечения требуемых документов из некоторого банка данных, считается одной из самых актуальных и широко исследуемых тематик компьютерной науки последнего десятилетия. В первую очередь это связано, разумеется, с бурным ростом интернета — по своей форме гигантской неструктурированной коллекции самых разнообразных данных, поиск в которой представляет серьёзную научную и техническую проблему. В настоящее время ведутся исследования по самым разным направлениям, включающим, например, поиск мультимедийных данных [32], улучшение методик вычисления релевантности [33] и изучение структуры Web [34]. Теоретически при разработке методик информационного поиска приоритет должен, вероятно, отдаваться обеспечению качества извлекаемых документов, то есть разработке адекватных критериев релевантности. На практике для интернетовских поисковых машин зачастую важнее полнота (то есть охват как можно большего количество документов Web), скорость поиска и предоставление дополнительных служб, таких как поиск по электронным библиотекам, магазинам или нахождение информации об объектах на географической карте. Поэтому при разработке алгоритмов извлечения релевантных данных более надёжными можно считать эксперименты, проводимые с так называемыми настольными поисковыми системами, предназначенными для нахождения информации на локальных дисках пользователя. Подобные системы существуют и развиваются уже не первый год (см., напр., проекты The Sleuthhound! [35] и Архивариус 3000 [36]). Изучение настольных поисковых систем позволяет

также сконцентрироваться на «чистых» алгоритмах нахождения релевантности, определяющих ранг документа прямо на основании его соответствия запросу, а не на характерных для интернета косвенных признаках, таких как популярность страницы или количестве ссылок на неё.

Существующие алгоритмы поиска (даже учитывающие косвенные признаки) так или иначе сводятся к извлечению документов, содержащих введённые пользователем слова. Эта модель связана, по крайней мере, с двумя проблемами.

Во-первых, она не оставляет возможности объяснить системе, чтó именно желает найти пользователь. Так, система Яндекс в ответ на запрос *Лев Толстой. Война и мир* выдаёт большое количество ссылок на рефераты. Текст романа (наиболее релевантный документ) в первой тридцатке результатов занимает лишь 11-е и 29-е места (состояние на март 2006г.) Сложно найти и документ, содержащий неизвестное слово (например, узнать название бывшей улицы Пушкинской в Москве). В любом случае, само существование конкурсов по поиску в интернете (Кубка Яндекса, к примеру) наглядно свидетельствует о том, что поиск требуемой информации с помощью существующих систем — задача нетривиальная.

Вторая проблема связана со слабой поддержкой особенностей естественных языков. Существующие системы зачастую игнорируют изменения падежных окончаний слов, а также возможную словесную омонимичность и многозначность. Поддержка словоформ иногда приводит к забавным казусам. Например, вся первая десятка ссылок, возвращаемая поисковиком Яндекс в ответ на запрос *мыть автомобиль*, найдена по сочетанию *мой автомобиль*, хотя слово *мой* в каждом из выбранных

документов является местоимением, а не повелительной формой глагола *мыть*.

Задача рубрикации документов тесно связана с задачей информационного поиска. Под рубрикацией здесь понимается каталогизация документов по темам, предоставляемая любой современной поисковой системой. Чтобы провести параллель между этими двумя задачами, обратимся к популярной схеме поиска и рубрикации, восходящей к классической модели векторного пространства [37].

Документы в этой схеме рассматриваются как элементы некоторого метрического пространства. Полагая пользовательский запрос документом, можно (с помощью метрики) найти список документов, наиболее близких к нему. Кроме того, для любого метрического пространства работают общие алгоритмы классификации и кластеризации (такие как KNN и C-Means [38]), с помощью которых можно решить задачу рубрикации. Сначала вручную создаётся множество кластеров, и в каждый из них помещается некоторое количество релевантных документов. Затем запускается алгоритм классификации, определяющий корректный кластер для каждого добавляемого в коллекцию файла. С помощью метода C-Means можно сформировать систему кластеров и автоматически, но её осмысленность не гарантируется.

Основная сложность заключается в подборе адекватной функции расстояния (метрики). В модели векторного пространства используется довольно простая схема. Документы рассматриваются как векторы размерности T, равной размеру словаря языка документа. Таким образом, любому слову языка сопоставлен некоторый компонент вектора каждого документа. Полагая, что векторы состоят из численных весовых коэффициентов, соответствующих словам языка, можно использовать

обычное скалярное произведение векторов в качестве функции расстояния. В простейшем случае весовые коэффициенты могут представлять собой нули и единицы (слово не присутствует в документе / слово присутствует в документе). Более сложные алгоритмы вычисления весовых коэффициентов учитывают частотное распределение слов как в отдельно взятых документах, так и в целом по коллекции.

Семантический анализ, вероятно, может помочь при разработке метаязыка поисковой машины (решение проблемы объяснения системе цели поиска), но эта задача пока что всерьёз не изучается. Поддержка же особенностей естественных языков — прямая цель семантического анализа. Возможности анализатора по извлечению смысла слов могут быть сравнительно легко встроены в любую поисковую машину (с особенной уверенностью это можно утверждать для локальных поисковых систем). Далее мы подробнее рассмотрим способы использования семантического анализа в задачах информационного поиска и рубрикации.

Поиск и рубрикация с помощью словарей классов

Простейшими сведениями о словах, возвращаемыми семантическим анализатором, являются их классы и начальные формы. Так, анализатор сообщит, что слово *банкноту* имеет начальную форму *банкнота* и принадлежит классу *физический_объект / неодушевлённый / деньги / купюра*. Для слова, не являющегося базовым, анализатор выведет семантическое описание, включающее в себя, в частности, классы понятий, через которые данное слово определяется. Например, *сыщик* — это профессия (класс *физический_объект / живой / человек / индивидуум / профессия*), имеющая отношение к сыскному учреждению (класс *физический_объект / учреждения / государственные*).

Предоставляемая анализатором информация может быть использована двумя очевидными способами: каждое слово в документах коллекции (и в пользовательском запросе) можно заменить либо его начальной формой, либо идентификатором класса (для производных слов — списком идентификаторов классов слов, входящих в семантическое описание).

В первом случае получится поисковая система, способная находить в документах заданные слова независимо от их формы, что очень важно для любого языка, в котором есть падежи (в частности, для русского). Нельзя не вернуться к тому факту, что семантический анализатор рассматривает слова в контексте предложения, поэтому приведённый выше пример с сочетаниями «мыть автомобиль» — «мой автомобиль» уже не сработает.

Во втором случае будут найдены не только явным образом перечисленные в запросе слова, но и слова, им синонимичные. Например, в ответ на запрос *банкнота* система напечатает также документы, содержащие слово *купюра*. Иногда классы оказываются слишком широкими для точного поиска. К примеру, слово президент является представителем класса *физический_объект / живой / человек / индивидуум / профессия / гражданская / глава*. В тот же самый класс попадают также такие разные слова как *вождь*, *атаман* и *бундесканцлер*. Здесь эта проблема обсуждаться не будет, поскольку нами был построен более совершенный механизм информационного поиска (рассматриваемый в следующем разделе), для которого проблема излишней широты классов почти не актуальна.

Для задачи рубрикации широта классов уже не столь пагубна, поскольку сами по себе рубрики в большинстве существующих каталогов сами по себе весьма обширны.

Усовершенствованный механизм поиска

Некоторые возможности по улучшению качества поиска с помощью семантического анализа можно рассмотреть на примере нашей экспериментальной системы [39]. Для удобства оценки качества поиска мы несколько отошли от классической модели извлечения информации на уровне целых документов: наша система определяет рейтинг каждого предложения документа по отдельности. Правила вычисления рейтинга достаточно просты:

1. Если в предложении встретилось искомое слово, и оно является именем собственным (это можно узнать по классу слова: например, к именам собственным относятся слова из классов *физический_объект / поселения / страны / название* и *физический_объект / живой / человек / индивидуум / ФИО*), рейтинг предложения повышается на 8 баллов.
2. Если в предложении встретилось искомое слово, причем его значение в контексте документа совпадает со значением в контексте запроса, рейтинг предложения повышается на 5 баллов. Так, при запросе *русая коса* фраза *девушка со стальной косой* не получит ни одного балла, поскольку значения слова *коса* в запросе и в документе различаются.
3. Если в предложении встретился класс, к которому принадлежит какое-либо слово запроса, рейтинг предложения повышается на 3 балла. Такая ситуация возникает, например, во фразе *председатель совета подтвердил участие в переговорах* при определении соответствия запросу *руководитель*, поскольку слова *председатель* и *руководитель* принадлежат одному и тому же классу.
4. Если в предложении есть слово, в семантическом описании которого встречается та же операция, что и в описании некоторого слова запроса, рейтинг предложения повышается на 1 балл. К примеру, в формальном

описании слова *удалось* встречается запись Hab\$12411/033, что означает «иметь успех» (т.к. строка $12411/033 является идентификатором класса *физический_объект / живой / человек / заслуга / успех*). Та же самая запись может быть найдена в описании слова *получаться*, поэтому рейтинг предложения, содержащего слово *получаться*, будет повышен на единицу, если в запросе встретилось слово *удалось*.

Таким образом, простая поисковая система, основанная на классах, использует только правило №3. В программе же [39] используется более гибкая градация, включающая в себя как более важные (нахождение имени собственного или точного слова), так и менее важные (совпадение операций) случаи.

В табл. 3.1 и табл. 3.2 приведены примеры ответов системы на запросы пользователя[11].

Таблица 3.1. Результат запроса *глава учреждения сообщил*

(рейтинг: 16) Такое мнение высказал в воскресенье РИА «Новости» глава комитета Госдумы по международным делам Константин Косачёв. *Соответствия:* глава — ГЛАВА учреждения — РИА_НОВОСТИ, ГОСДУМА, КОМИТЕТ сообщил — ВЫСКАЗАТЬ
(рейтинг: 11) «Положение в стране крайне нестабильное» — отметил в интервью РИА «Новости» вице-спикер Госдумы Вячеслав Володин.

[11] В качестве тестовой коллекции была использована подборка документов с новостного сервера NEWSru.com

Соответствия: глава — ВИЦЕ-СПИКЕР учреждения — РИА_НОВОСТИ, ГОСДУМА сообщил — ОТМЕТИТЬ, ИНТЕРВЬЮ
(рейтинг: 10) Такой неутешительный прогноз дал в интервью ИТАР-ТАСС заместитель директора Института океанологии РАН Леопольд Лобковский. *Соответствия:* глава — ЗАМЕСТИТЕЛЬ, ДИРЕКТОР учреждения — ИТАР-ТАСС, ИНСТИТУТ сообщил — ИНТЕРВЬЮ
(рейтинг: 3) Кроме того, в регионе ЮВА работают два российских госпиталя. *Соответствия:* учреждения — ГОСПИТАЛЬ

Таблица 3.2. Результат запроса *сложные погодные условия*

(рейтинг: 20) По словам собеседника агентства ситуацию осложняют погодные условия — сильный боковой ветер, минусовая температура и непрекращающийся снегопад. *Соответствия:* сложные — ОСЛОЖНЯТЬ погодные — ПОГОДНЫЙ, ВЕТЕР, СНЕГОПАД условия — УСЛОВИЯ
(рейтинг: 3) Аэропорт Ростова-на-Дону закрыт из-за сильного

обледенения. *Соответствия:* погодные — ОБЛЕДЕНЕНИЕ
(рейтинг: 3) В выходные буран в Москве стихнет. *Соответствия:* погодные — БУРАН
(рейтинг: 3) В дневные часы термометр покажет минус 8-10 градусов в Москве и 7-12 градусов мороза в её окрестностях. *Соответствия:* погодные — МИНУС, МОРОЗ

Как видно из примеров, большинство выданных системой предложений не содержат в точности указанных в запросе слов, тем не менее, все они справедливо сочтены релевантными.

Дополнительные возможности для существующих поисковых машин

Популярные поисковые машины практически никогда не останавливаются на простом извлечении текстовых документов, релевантных документу-запросу. Почти всегда авторы таких систем предоставляют какие-либо дополнительные инструменты, позволяющие добиться более качественных результатов при поиске. Среди наиболее типичных дополнительных средств поиска можно отметить, в частности, следующие:

- логические операции при поиске (и/или/не);

- поиск точной фразы (цитаты);
- возможность указать слова, которые обязательно должны содержаться в документе;
- поиск изображений, mp3-файлов;
- поиск файлов с известным именем;
- поиск в документах со сложной структурой (DOC, PDF);
- применение расстояния между словами.

Используя предоставляемую семантическим анализатором информацию об устройстве предложений естественного языка, можно добавить к этому списку ещё один инструмент: *поиск связанных слов*, идеологически близко связанный с применением расстояния между словами.

Когда поисковая система индексирует некоторый документ, каждому его слову присваивается порядковый номер. Расстояние между двумя словами — это просто разница их порядковых номеров. Если в документе, допустим, встречается сочетание *красная шапочка*, расстояние между словами *красная* и *шапочка* равно единице, а расстояние между словами *шапочка* и *красная* — минус единице. Требуемое расстояние можно ввести прямо в строке поиска. К примеру, если в Яндексе ввести запрос *поставщики /2 кофе*, то найдутся документы, в которых содержится и слово *поставщики*, и слово *кофе*, а расстояние между этими словами будет не больше 2. Таким образом, релевантными окажутся документы, содержащие фразы вида *поставщики колумбийского кофе*, *поставщики кофе из Колумбии* и т.п.

Основное назначение этого инструмента — поиск слов, связанных по смыслу. Если нас интересует чёрный кофе, простой поиск по этим двум

словам выдаст множество документов, содержащих оба слова, хотя и никак не связанных между собой. С другой стороны, поиск по точной фразе *чёрный кофе* окажется слишком ограниченным: сочетания вроде *чёрный молотый кофе* или *чёрный бразильский кофе* не будут найдены. Поиск с расстоянием — разумная альтернатива. Например, в ответ на запрос *чёрный /2 кофе* можно получить и просто *чёрный кофе*, и *чёрный молотый кофе*, и *чёрный бразильский кофе*. Однако документ, содержащий сочетание *чёрный молотый бразильский кофе* уже не будет считаться релевантным. Можно ещё увеличить расстояние (*чёрный /3 кофе*), но при этом увеличится доля нерелевантных документов, содержащих фразы вроде *чёрный пакет с кофе*.

Проблема заключается в том, что расстояние между словами — плохой критерий их взаимной зависимости. Два слова могут стоять рядом в предложении и при этом не относиться друг к другу напрямую; с другой стороны, порою находящиеся далеко друг от друга слова оказываются близко связанными по смыслу. Семантический анализатор, строящий дерево разбора предложения, позволяет воспользоваться более надёжными и обоснованными критериями зависимости.

Изучение дерева разбора даёт возможность выделить, по крайней мере, два типа связей между словами: *связь-свойство* и *связь-отношение*.

Связи первого типа возникают, когда одно слово является свойством (или *как бы* является свойством) второго. Например, именно так связаны слова *чёрный* и *кофе* во фразе *я пью чёрный кофе*» или слова *Большой* и *театр* в предложении *Я иду в Большой театр*.

На уровне дерева предложения это означает выполнения простого требования: одно из слов пары является аргументом другого.

Рассмотрим, к примеру, дерево фразы *я купил чёрный бразильский кофе* (см. рис. 3.1).

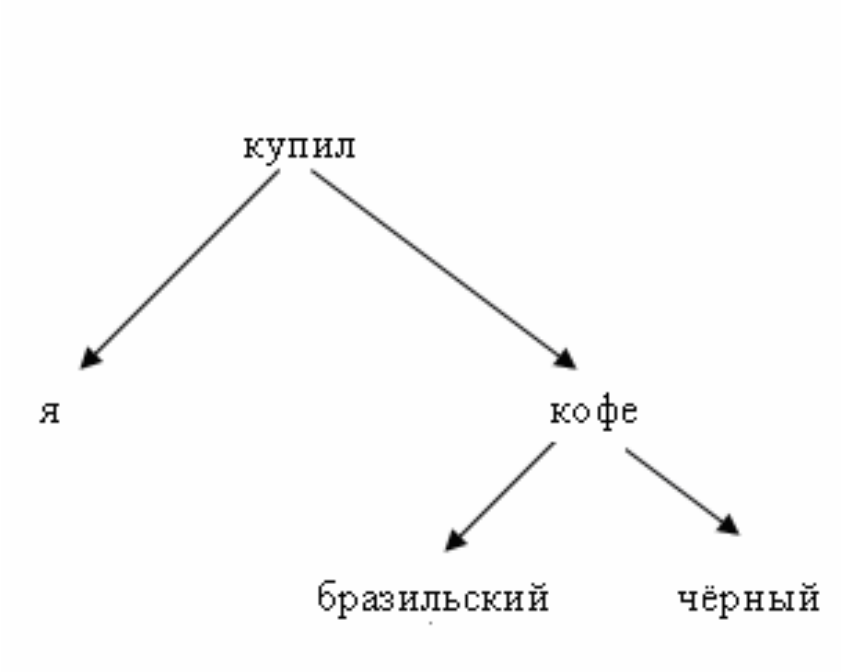

Рис. 3.1. Дерево фразы *я купил чёрный бразильский кофе*.

Слова *чёрный* и *кофе*, *чёрный* и *бразильский* связаны по смыслу, причём связью первого типа.

1. Связь-отношение возникает между двумя словами-потомками некоторого третьего слова-предка А.

В приведённом примере связь такого типа существует между словами *я* и *кофе*, *я* и *бразильский*, *я* и *чёрный*. Можно сказать, что слова *я* и *кофе* находятся в отношении *купил*.

Рассмотрим другие примеры возникновения связей в предложениях:

1. *Маша любит Сашу* (см. рис. 3.2).

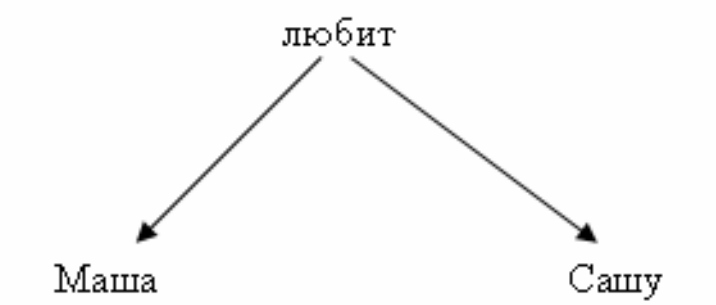

Рис. 3.2. Дерево фразы *Маша любит Сашу*.

Эта фраза — простейший пример возникновения связи-отношения (в данном случае между словами Маша и Сашу). Скобочная форма записи предложения хорошо отражает этот факт: *любит(Маша, Сашу)*. Других связей в предложении нет.

2. *Большой театр находится в Москве* (см. рис. 3.3).

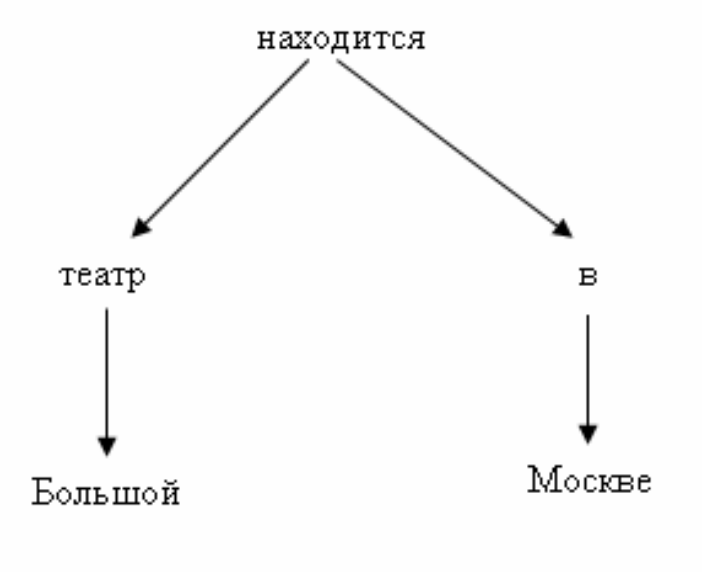

Рис. 3.3. Дерево фразы *Большой театр находится в Москве*.

Слова *Большой* и *театр*, а также *в* и *Москве* связаны зависимостью первого типа. Каждое из слов *Большой* и *театр* также связано с каждым из слов *в* и *Москве* зависимостью второго типа.

3. *Я купил кофе в чёрном пакете* (см. рис. 3.4).

Это предложение — хороший пример того, как семантический анализатор корректно распределяет не связанные между собой (хотя и стоящие почти рядом) первым типом зависимости слова *кофе* и *чёрном* по разным ветвям дерева. Здесь слово *чёрном* связано первым типом зависимости лишь со словами *пакете* и *в*. Со словом *кофе* оно тоже связано, но тип зависимости здесь — второй.

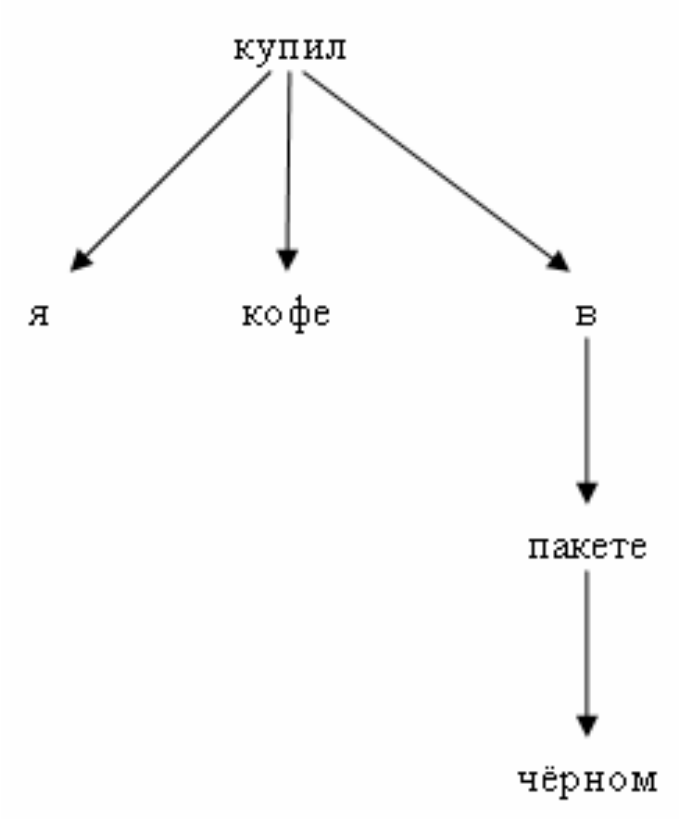

Рис. 3.4. Дерево фразы *я купил кофе в чёрном пакете*.

Мы не будем обсуждать аспекты реализации интернетовской поисковой машины, использующей описанные типы зависимостей,

поскольку, как уже было сказано выше, поиск в интернете неразрывно связан с индексированием больших объёмов данных, распараллеливанием алгоритмов и тому подобными вещами, непосредственно не относящимися к информационному поиску. Наша экспериментальная система, предназначенная для поиска релевантных документов на локальных дисках пользователя, работает следующим образом. Пользователь задаёт запрос в виде набора слов. Некоторые пары набора могут быть обрамлены круглыми (связь-свойство), либо квадратными (связь-отношение) скобками (напр.: *я купил (чёрный кофе)*). Далее специальный модуль с помощью семантического анализатора отбирает из коллекции лишь те документы, в которых заключённые в скобки слова состоят в требуемом типе связи. Затем для отобранных документов вызывается алгоритм ранжирования на основе классической модели векторного пространства, и полученный список возвращается пользователю. Качество результатов (вернее, правильность определения зависимостей между словами) полностью зависит от качества семантического анализатора и достигает 90-95%.

Глава 4. Спеллчекер и тезаурус

Практически любой текстовый процессор, претендующий на звание современного, включает в себя два полезных инструмента — спеллчекер и тезаурус. Спеллчекер (более грамотно называемый *модулем проверки правописания* за неимением более короткого русскоязычного термина) в своей простейшей разновидности занимается поиском слов документа, не внесённых в словарь используемого языка. Например, при анализе фразы *чашка стояла на стле* любой спеллчекер должен указать, что строка *стле* не является корректным русским словом. Более совершенный продукт выдаст список возможных альтернатив (например, *столе* и *стуле* в данном случае). Ещё более развитый спеллчекер может попытаться проанализировать синтаксическую и семантическую сочетаемость слов в предложении, а также правильность расстановки знаков препинания. В этом случае фразы вроде *чашка стояла на столом* и *чашка стояла, на столе* также будет признаны некорректными. Тезаурус представляет собой простой электронный словарь синонимов, помогающий сделать язык документа более выразительным и избавиться от постоянных повторений одних и тех же слов.

Семантический анализатор как модуль проверки правописания

Мощные спеллчекеры на практике встречаются редко. Как правило, реализуется простой просмотр словаря языка[12] с дальнейшим выбором похожих по написанию слов с помощью расстояния Левенштейна [40]. Проверка сочетаемости слов требует уже гораздо большего объёма работы, и лишь немногие программы её осуществляют с той или иной степенью успеха (к таковым относится, например, спеллчекер из поставки Microsoft Word). На практике же подобная функциональность весьма полезна, по крайней мере, по двум причинам. Во-первых, при перефразировке предложений русского языка очень легко забыть изменить падеж одного-двух слов. Во-вторых, при распознавании сканированного текста отдельные буквосочетания иногда определяются неверно вследствие их графического сходства с другими буквосочетаниями. Например, сочетание Г! похоже на букву П. При этом полученное слово может по-прежнему иметь смысл, и несовершенный спеллчекер его пропустит.

Семантический анализатор сам по себе является достаточно мощным средством проверки правописания. Аналогично компилятору любого языка программирования, выполняющего проверку синтаксиса входной программы, семантический анализатор производит контроль корректности входных предложений. Следует отметить, что текущая версия анализатора не разрабатывалась с целью использования в качестве спеллчекера. Анализатор полагает, что входные предложения являются корректными, и

---

[12] Для некоторых языков, однако, даже эта задача может быть весьма нетривиальной. Например, в немецком отдельные слова часто «склеиваются» в одно длинное слово, и поиск в словаре оказывается связанным с поиском тех или иных его частей.

будет пытаться собирать отдельные слова в допустимые конструкции, даже если их смысл окажется несколько экстравагантным. Более того, анализатор попытается извлечь хоть какую-то структуру из предложения, содержащего неизвестные слова или неправильно расставленные запятые. Это объясняется тем, что семантический анализатор был создан для обработки реальных текстов, возможно, содержащих ошибки (впрочем, программирование иных приоритетов — задача вполне посильная).

Рассмотрим несколько примеров, иллюстрирующих работу семантического анализатора в качестве модуля проверки правописания:

1. Предложение *у меня дома на полке книги* разбирается без ошибок.
2. Предложение, в котором по ошибке удалено слово *книги* (получается *у меня дома на полке*) также разбирается без каких-либо замечаний. Смысл его подозрителен, однако допустим (на полке могут стоять дома́).
3. Предложение *чашка стояла на стле* разбирается, однако слово *стле* помечается как неизвестное.
4. Предложение *чашка стояла на столом* разбирается только до предлога *на*, для которого системе не удаётся найти соответствующее существительное.
5. Предложение *чашка стояла на, столе* разбирается полностью, поскольку предлог всегда соединяется с существительным, и запятая в данном случае игнорируется (то есть при прочих равных условиях анализатор попытается успешно разобрать предложение, а не вывести сообщение об ошибке).

Возможность использования семантического анализатора в качестве спеллчекера ещё слабо изучена (в основном потому, что в наших исследованиях есть гораздо более приоритетные задачи). Однако

приведённые примеры доказывают потенциальную перспективность этого направления.

Контекстно-ориентированный тезаурус на основе семантического анализатора

С технической точки зрения электронный словарь синонимов разработать гораздо проще, чем модуль проверки правописания. Тезаурус — это просто программа, выполняющая поиск слова в базе данных и печатающая список сопоставленных ему слов-синонимов. Однако и здесь семантический анализатор находит своё применение. С его помощью можно сделать электронный тезаурус более интеллектуальным.

Типичный сценарий взаимодействия с тезаурусом выглядит следующим образом. В процессе чтения текста пользователь решает заменить какое-либо слово более подходящим синонимом. Пользователь выделяет слово курсором и с помощью соответствующего пункта меню вызывает тезаурус. Тезаурус выводит список синонимов выделенного слов, и пользователь щелчком по элементу списка производит замену.

Недостаток существующих тезаурусов заключается в отсутствии анализа контекста заменяемого слова. Поэтому выводимый словарём список нередко оказывается избыточным. Например, слово *разбить* имеет четыре различных смысла:

1. сломать, испортить (разбить чашку);

2. приготовить для использования (разбить палатку);

3. стать причиной душевных переживаний (разбить сердце);

4. разделить целое на части (разбить задачу на подзадачи).

Программа, способная определить смысл слова в данном контексте, могла бы выводить список только подходящих синонимов. Семантический анализатор вполне пригоден для такой работы.

Чтобы создать качественный контекстно-ориентированный тезаурус, следует применить семантический анализатор в связке с хорошим словарём синонимов (например, с [41]). Можно воспользоваться и встроенным словарём семантического анализатора. Каждое слово семантического словаря описано некоторой смысловой формулой. Близким по значению словам (то есть синонимам) соответствуют близкие (по некоторому критерию) смысловые формулы.

Изучению методов автоматического извлечения синонимов из семантического словаря посвящены работы [42] и [43]. В статье [42] приводятся примеры SQL-запросов, извлекающих синонимы для некоторых классов слов. Также описывается методика построения таких запросов.

В работе [43] рассматривается общий критерий семантической близости, определяемый как расстояние Левенштейна над множеством упрощённых семантических формул, включающих только номера классов и названия базисных функций из исходного описания. Эксперименты показывают, что расстояние, равное нулю (точное совпадение упрощенных формул) почти всегда сигнализирует о найденных синонимах. Расстояние, равное единице, в ряде случаев также обеспечивает хорошее смысловое приближение. Расстояние, превосходящее или равное двум, обычно уже свидетельствует об отсутствии синонимичности (см. табл. 4.1).

Таблица 4.1. Примеры найденных синонимов

Слово	Найденные синонимы
коса (песчаная)	0: — 1: крутобережье крутояр рабатка
косой (косоглазый)	0: косоглазый 1: кривоглазый
девушка	0: дивчина 1: кралечка русалочка снегурочка (и др.)
ключ (источник)	0: родничок фонтанчик 1: прудик ручеёк речка (и др.)

Интересно отметить, что «близость» в терминах семантического словаря далеко не всегда совпадает со смысловой близостью, отраженной в обычных словарях синонимов. Например, слову *косой* (в смысле *косоглазый*) достаточно близки слова *криворогий* и *кареглазый*. Семантическая формула слова *косой* может быть расшифрована как «некто, имеющий косые глаза». Эта формула достаточно близка к схожим формулам «некто, имеющий карие глаза» и «некто, имеющий кривые рога».

Таким образом, с одной стороны, можно сделать вывод о пригодности встроенного словаря семантического анализатора в качестве словаря синонимов. С другой стороны, необходимо отметить, что семантические описания из словаря анализатора не были специально разработаны для поиска синонимов. Поэтому для практического использования лучше рекомендовать адаптацию и подключение к семантическому анализатору какого-либо готового качественного словаря синонимов.

Глава 5. Поиск частично совпадающих документов и выявление плагиата

В главе, посвящённой информационному поиску, мы уже обсуждали задачу разработки адекватной функции расстояния (метрики) для определения степени подобия документов произвольной коллекции. В задачах поиска частично совпадающих документов и выявления плагиата также требуется выявлять наборы «близких» (в некотором смысле) текстов, однако специфика проблемы требует использования иных критериев близости. Поскольку данная тематика занимает значительное место в научной работе автора, мы рассмотрим её подробнее.

О задаче выявления плагиата и поиске частичных совпадений

Проблема плагиата в учебных заведениях по всему миру стоит достаточно остро. Так, по результатам исследования, проведённого в университете Дьюка в 2001 году, 40% студентов практиковали копирование предложений без указания источника, 11% копировали без изменений обширные фрагменты текста, 9% выдавали компьютерные программы, написанные другими, за свои [44]. Плагиат нередко встречается и в научных работах, публикуемых в журналах и в сборниках трудов конференций [45].

К сожалению, в то время как на Западе этой проблеме уделяется пристальное внимание, в России её практически игнорируют. На Западе плагиат считается очень серьёзным нарушением устава университета (или особого «кодекса чести», см., напр., [46], [47]) и может повлечь за собой строгие меры вплоть до исключения студента из учебного заведения.

Учёному плагиат также может стоить карьеры. В нашей стране проблема плагиата накладывается на всеобщее снисходительное отношение к этому явлению. Как отмечают авторы статьи [48], «Проблема [плагиата] в России осознаётся многими преподавателями, но изменить к лучшему ситуацию непросто. [Российская] система образования страдает от недостатка финансирования. Поэтому многие преподаватели бедны и не имеют мотивации следить за тем, что происходит у них за спинами и перед глазами. Взяточничество — обычный способ заработать больше. Российское общество основано на званиях. Звания открывают двери. Не имеет значения, что вы изучали, если вы окончили известный университет. Высокие ожидания провоцируют некоторых избрать лёгкий путь».

Внимание к плагиату (по крайней мере, на Западе) выражается, в частности, в большом количестве статей, посвящённых изучению способов его предотвращения, обучению студентов корректному цитированию источников, разработке «кодексов чести» и методам выявления плагиата в уже сданных студенческих работах. С технической точки зрения последняя задача наиболее хорошо поддаётся автоматизации.

Системы выявления плагиата прежде всего можно разделить на две категории — «онлайновые» и «оффлайновые» (или «герметичные») [49]. «Онлайновые» системы пытаются найти оригинальный источник исследуемого текста в интернете, либо во встроенной обширной базе статей. Подобные службы выявления плагиата (к ним относится, например, Turnitin [50]) почти всегда являются платными и рассчитаны на широкое использование на уровне целых университетов. «Герметичные» системы производят поиск заимствований только в пределах локальной коллекции пользователя. Как правило, «онлайновые» системы предназначены для обработки текстов на естественном языке. Напротив,

«герметичные» детекторы обычно специальным образом разрабатываются для анализа компьютерных программ.

Этому факту можно дать простое объяснение. Как правило, задания для курсовых работ или сочинений в пределах группы достаточно разнообразны, и воспользоваться чужой работой не так просто. Кроме того, достаточно высока вероятность разоблачения. Поэтому сочинение (курсовая) обычно компилируется из интернетовских источников. Задания по программированию редко бывают столь же индивидуальны. Обычно готовые программы сокурсников служат хорошей помощью при решении собственной задачи. Плагиат при этом можно попытаться скрыть с помощью переименования идентификаторов и замены управляющих структур их эквивалентами. Найти же в интернете готовую программу, решающую поставленную задачу, весьма непросто.

Строгая структурированность компьютерных программ позволяет системам, их обрабатывающим, применять ряд специальных методик, делающих поиск более интеллектуальным. Например, система может бороться с переименованием идентификаторов.

Автоматические детекторы плагиата в компьютерных программах находят применение и в задаче поиска частично совпадающих документов. В частности, компьютер может помочь найти дублирующиеся фрагменты кода в большом программном проекте. Таким образом, инструменты поиска совпадений оказываются весьма полезными при рефакторинге.

Технические особенности систем выявления плагиата

На первый взгляд система выявления плагиата работает примерно так же, как и программа поиска или рубрикации — собирает отдельные файлы в группы «подобных» на основании функции расстояния. В

действительности же метрику в явном виде используют сравнительно редко. Дело в том, что при поиске плагиата «степень подобия» документов не так важна. Важно наличие общих подстрок, доказывающее факты «заимствования». Таким образом, типичный детектор плагиата представляет собой программу, сравнивающую содержимое файлов коллекции. Системы, основанные на иных принципах (подсчёт атрибутов, латентный семантический анализ), менее надёжны и широкого распространения не получили.

Как уже отмечалось, автоматические детекторы плагиата в компьютерных программах нередко используют специальные интеллектуальные методики, позволяющие повысить качество поиска. Среди них можно упомянуть *токенизацию*, *параметризованный поиск* и *сравнение деревьев разбора*.

С помощью токенизации [51] можно бороться со всеми видами переименования идентификаторов и управляющих структур. Токенизирующий модуль заменяет различные элементы компьютерных программ более общими «токенами». Например, вместо каждого идентификатора может быть подставлен токен <IDT>, а вместо каждого значения — токен <VALUE>. Таким образом, строка

```
a = b + 45;
```

будет заменена строкой

```
<IDT>=<IDT>+<VALUE>;
```

Теперь переименование переменных ничего не даст, поскольку любая последовательность вида «идентификатор = идентификатор + значение;» преобразуется в одну и ту же токенизированную последовательность. Если заменить начало и конец оператора цикла токенами <BEGIN_LOOP> и

<END_LOOP>, плагиатору не поможет и столь обычная методика сокрытия заимствований как замена типа цикла (например, for на while).

Токенизация применяется очень широко, однако её использование связано с двумя проблемами. Во-первых, для каждого языка программирования требуется свой токенизатор. Во-вторых, система начинает находить ложные совпадения, не соответствующие действительным заимствованиям.

Параметризованный поиск [52] призван решить проблему с переименованием идентификаторов с помощью более тонкой методики. Две строки будут признаны этим алгоритмом идентичными, если в них наблюдаются *регулярные* изменения в именах идентификаторов. Иными словами, алгоритм отслеживает употребление имён, и сочтёт строки идентичными, только если переменные в них используются одинаковым образом. Например, все три строки

```
a = a + 1
x = x + 1
a = b + 1
```

будут признаны любым простым алгоритмом токенизации совпадающими. Алгоритм же параметризованного поиска укажет только на совпадение первых двух строк. Параметризованный поиск может быть реализован достаточно эффективно [53].

Сравнение деревьев разбора [54] — наиболее развитая методика для оценки схожести компьютерных программ, существующая на сегодняшний день. Её суть заключается в поуровневом рекурсивном сравнении деревьев разбора, генерируемых синтаксическим анализатором. Эта процедура достаточно сложна, и в настоящее время используется редко. Кроме того, нет прямых доказательств того, что сравнение деревьев

по качеству результатов заметно превосходит более простые методики сравнения подстрок.

Использование семантического анализатора в задаче выявления плагиата

Системы, предназначенные для выявления плагиата в текстах на естественном языке, обычно устроены проще систем, сравнивающих листинги программ. Как правило, входные тексты не подвергаются никакой обработке. Заметим, что даже простое перефразирование русского предложения приводит к изменениям в падежных окончаниях. Это явление может серьёзно ухудшить способность системы к поиску заимствований, однако все существующие детекторы ориентированы на английский язык, для которого проблема падежных окончаний неактуальна.

Используя семантический анализатор В. Тузова, можно адаптировать методики токенизации и сравнения деревьев разбора для систем выявления плагиата в текстах на естественном языке. Результаты этого опыта описаны в статье [55].

В качестве исходной системы выявления плагиата была использована программа [56], созданная при непосредственном участии автора данной работы. Программа [56] представляет собой «герметичный» детектор плагиата со встроенным токенизатором для текстов на языке Java. Особенностью системы является пониженная алгоритмическая сложность по сравнению со стандартной «наивной» реализацией попарного сравнения файлов коллекции. Система хранит файлы коллекции в особой структуре на основе массива суффиксов [57]. Далее для каждого файла коллекции применяется специальный механизм эвристического поиска,

позволяющий быстро найти совпадения со всеми прочими файлами изучаемого набора.

Наш эксперимент заключался в замене токенизатора Java-файлов модулем на основе семантического анализатора, реализующим аналогичную функциональность для русского языка. Токенизацию можно рассматривать как процесс замены элемента текста его *классом*. Так, элементы x, y, step программы на языке Java могут быть отнесены к классу «ИДЕНТИФИКАТОР», а элементы 10, 15.3, -2 — к классу «ЗНАЧЕНИЕ». Как уже отмечалось, различным словам русского языка семантический анализатор также сопоставляет соответствующий класс. Более того, классы слов объединены в иерархию, поэтому токенизацию можно производить на различных уровнях абстракции.

Токенизацией нулевого уровня можно назвать замену слова его начальной формой. Эта процедура уже приводит к улучшению качества системы, поскольку перестают учитываться падежные окончания слов. Как уже указывалось, семантический анализатор учитывает информацию о контексте, поэтому, например, слово *мой* во фразе *я и мой автомобиль* останется без изменений (как стоящее в начальной форме). Возможность его трактовки как формы глагола *мыть* будет отвергнута.

На первом уровне слово заменяется его непосредственным классом. Например, слово *лиса* будет заменено классом «ЖИВОТНЫЕ-ДИКИЕ», а слово *собака* — классом «ЖИВОТНЫЕ-ДОМАШНИЕ». На втором уровне классы становятся более общими. Теперь и *лиса*, и *собака* попадают в один и тот же класс «ЖИВОТНЫЕ». Токенизация более высоких уровней уже лишена практического смысла, поскольку в результате избыточной широты классов семантически абсолютно разные слова начинают трактоваться как эквивалентные.

Для оценки качества системы была использована коллекция из 350 документов, взятых с сервера новостного агентства NEWSru.com. Каждый документ имеет осмысленное название и включён в одну из следующих категорий: *В России*, *В мире*, *Экономика*, *Религия*, *Криминал*, *Спорт*, *Культура*. Размер статей варьируется от 450 байт до 19 килобайт (медианный размер составляет 2 КБ). Типичная статья состоит из 8-12 абзацев, содержащих достаточно строгие повествовательные предложения и цитаты. Мы не ставили цели уличить авторов статей в плагиате, однако ожидали извлечь пары документов, либо посвящённых одному и тому же событию, либо прямо цитирующих одни и те же источники.

Выходные данные исходной системы [56] сравнивались с распечатками других аналогичных программ. Совпадения же в документах, обнаруженные новой системой, оценивать можно лишь вручную, поскольку других токенизирующих детекторов для русского языка пока не существует.

Без применения токенизации программа обнаружила 20 пар частично совпадающих документов (перекрывающихся, по крайней мере, на 4% от общего размера). Ниже перечислены типичные примеры найденных пар (цит. по [55]):

- Пара документов, посвящённых солнечному затмению 20 марта 2006г. В первом документе перечислены страны, в которых наблюдалось затмение; второй документ полностью посвящён наблюдению затмения на территории России.
- Пара документов о погодных условиях в Европе. В первом говорится о наводнениях в странах Евросоюза; второй представляет собой прогноз погоды в Европе (предсказывающий, в том числе, последующие наводнения).

- Пара документов, посвящённых слухам о помощи, оказанной российской разведкой бывшему иракскому руководству в марте 2003г. Первый указывает позицию Москвы, второй цитирует реакцию Вашингтона.

Токенизация первого уровня привела к повышению степени соответствия для ранее найденных пар. В большинстве случаев степень соответствия возросла в полтора раза. Были также отмечены четыре новых пары, оказавшиеся неверными. Однако степень соответствия для них оказалась низкой (4-5%). Токенизация оказала влияние на оценку степени подобия во многих ситуациях, относящихся к одному из двух случаев:

- Система перестала различать падежные формы одних и тех же слов.
- Система указала один и тот же класс для различных (но сходных) слов. Например, в выражениях *резиденция в Греции* и *резиденция в Афинах* слова *Греции* и *Афинах* были преобразованы в класс «ПОСЕЛЕНИЕ».

Возникновение второго случая иногда приводит и к неправильным результатам. Например, фразы *Владимир Путин заявил* и *Джордж Буш заявил* распознаются как одна и та же токенизированная последовательность <ИМЯ><ФАМИЛИЯ><ЗАЯВЛЕНИЕ>. Необходимо отметить, однако, что классы, используемые семантическим анализатором, не были специально разработаны для задачи выявления плагиата или поиска частичных совпадений. Можно ожидать, что более специализированная иерархия классов приведёт к лучшим результатам.

Помимо извлечения номеров классов отдельных слов семантический анализатор строит дерево разбора каждого предложения текста. Теоретически можно воспользоваться методикой [54] для сравнения деревьев, и тем самым ещё повысить качество детектора, но о практических результатах говорить ещё рано.

Глава 6. Введение в машинный перевод

Проблема машинного перевода очень сложна для обсуждения. На протяжении десятилетий создавались самые разные автоматические переводчики, включая коммерческие. Однако даже сейчас качество их работы в целом весьма неудовлетворительно, и единственная причина, по которой подобные программы выживают, заключается в полном отсутствии более качественных альтернатив.

С одной стороны, говорить о возможности достижения значительных результатов в решении задачи машинного перевода благодаря семантическому анализатору было бы в высшей степени самонадеянно. С другой стороны, высокое качество автоматического анализа текстов системой В. Тузова позволяет выразить надежду на существенное улучшение наиболее слабых мест всех программ перевода — модулей выбора корректной альтернативы слова и подходящей падежной формы.

Типичные проблемы существующих систем (на примере англо-русского перевода) иллюстрирует табл. 6.1. Перевод сделан с помощью пакета Promt [2], тестовые предложения взяты с вебсайта корпорации Microsoft.

Таблица 6.1. Типичные ошибки машинного перевода

Исходный текст	Microsoft is the worldwide leader in software, services and solutions that help people and businesses realize their full potential. We are committed to helping countries improve their global competitiveness, promote local economic growth and development, and drive innovation.

	We believe that every successful corporation has a responsibility to use its resources and influence to make a positive impact on the world and its people.
Перевод	Microsoft — международный лидер в программном обеспечении, услугах и решениях, которые *помогают людям, и фирмы понимают их полный потенциал**. Мы посвящаем себя помощи странам улучшить их глобальную конкурентоспособность, продвигать местный экономический рост и развитие, *и новшество двигателя***. Мы полагаем, что каждая успешная корпорация имеет ответственность использовать ее ресурсы и влияние, чтобы оказать положительное влияние *на мире и ее людях****.
Правильный перевод	* помогают людям и бизнесу полностью реализовать свой потенциал ** и стимулировать инновации *** на мир и его людей

Обе проблемы (выбор значения слова и определение падежной формы) связаны с неполноценным алгоритмом построения дерева предложения. Семантический анализатор специально предназначен для конструирования деревьев разбора, и его принципы с большой вероятностью могут послужить удачной базой для системы автоматического перевода.

На данный момент почти все разработки по семантическому анализу ориентированы на русский язык. Однако уже появляются публикации, в которых описывается обработка англоязычных текстов [58], [59].

В настоящее время задаче применения семантического анализа в области машинного перевода посвящены сразу несколько проектов. Основные сложности связаны с необходимостью привлечения носителей целевых языков (даже очень хорошее владение иностранным языком не поможет разобраться со всеми лингвистическими тонкостями) и с гигантским объёмом работы, требующимся для того, чтобы вывести разработки из области игрушечных экспериментов в сферу реальной жизни.

Хотя говорить о больших достижениях семантического анализа в области автоматического перевода ещё рано, основные принципы, на которых базируются наши исследования, заслуживают некоторого внимания. Прежде всего, однако, придётся несколько глубже ознакомиться с некоторыми аспектами функционирования самого семантического анализатора (подробно механизм его работы описан в книге [21]).

Морфологический и синтактико-семантический уровни анализа текста

В предыдущих главах мы всецело полагались на выходные распечатки, предоставляемые семантическим анализатором. Задача машинного перевода требует более тонкого взаимодействия с анализатором на различных этапах обработки текста.

Когда на вход семантическому анализатору поступает очередное предложение, первым делом в работу включается модуль *морфологического анализа* (см. рис. 6.1). Его задача — извлечь

морфологическую информацию об отдельных словах предложения. Например, встретив слово *стола*, морфологический анализатор сообщит, что оно является существительным мужского рода с начальной формой *стол*, стоящим в родительном падеже, единственном числе. Разумеется, здесь возможны неоднозначности. Слово *берегу* может быть как формой дательного или предложного падежа существительного *берег*, так и формой первого лица (настоящего времени, единственного числа) глагола *беречь*. В таких случаях морфологический анализатор должен вернуть полный список возможных альтернатив. Технически морфологический анализатор представляет собой достаточно простую программу, основанную на электронном варианте словаря А. Зализняка [60].

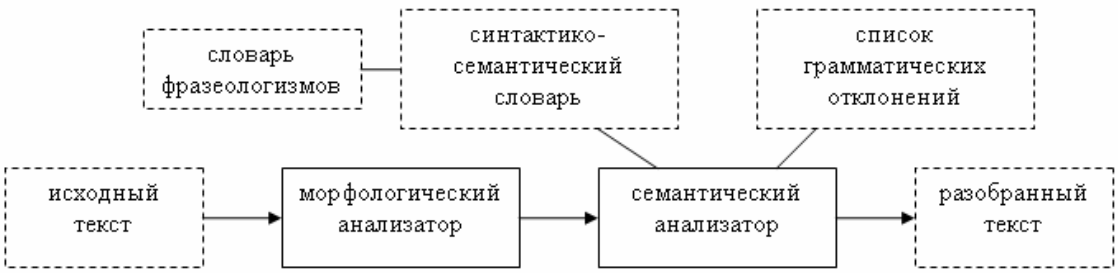

Рис. 6.1. Этапы обработки текстовой информации.

Дальнейшая сборка деревьев предложений производится при помощи *синтактико-семантического словаря*. Рассмотрим фразу

```
В четверг во Владимире состоится церемония вручения золотых медалей
выпускникам-отличникам Владимирской области.
```

Соответствующее ей дерево, построенное анализатором, изображено на рис. 6.2.

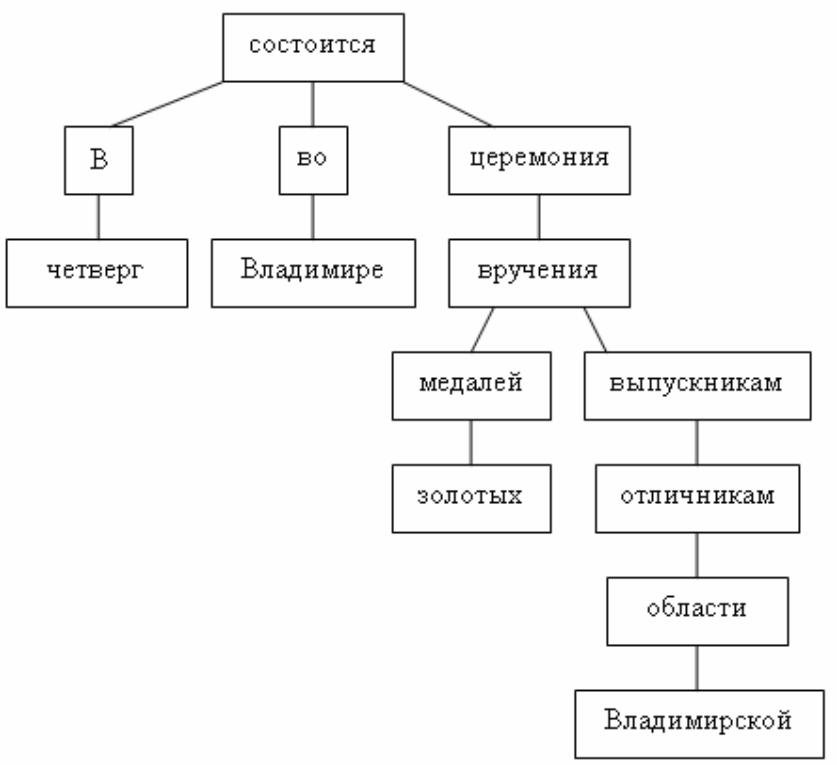

Рис. 6.2. Построенное анализатором дерево разбора предложения.

Система нашла три аргумента для глагола *состоится* (*в четверг*, *во Владимире*, *церемония*). Это удалось сделать потому, что семантический анализатор изначально пытался отыскать именно *три* аргумента для слова *состоится*; данный факт отражён в его синтактико-семантическом описании:

```
СОСТОЯТЬСЯ {Глагол.} N%~СОБЫТИЕ$11(PerfFut Z1: ! !ОНЪ$17\!ОНА$17\!ОНО$17,
                                   Z2: НЕЧТО$1~!Когда, Z3: НЕЧТО$1~!Где)
```

Состояться может *что-то* (аргумент Z1), *когда-то* (аргумент Z2) и *где-то* (аргумент Z3). Информация о сочетаемости слов берётся, в первую очередь, из морфологии и предлогов. Например, вопрос *где* соответствует, в частности, предложному падежу с предлогами *в*/*во* и *на*. Некоторые аргументы могут так и остаться ненайденными (например, так было бы во

фразе *Во Владимире состоится церемония вручения медалей*, в которой опущен блок *когда*).

Количество и типы аргументов варьируются от слова к слову. Например, слово *четверг* присоединяет один аргумент, стоящий в родительном падеже, а слово *Владимир* вообще не присоединяет аргументов[13]:

```
ЧЕТВЕРГ {Сущв._Муж_Неодуш $1602~@ОНЪ$17@Вин} $1602(Z1: ВРЕМЯ$16~!Род)
ВЛАДИМИР {Сущв._Муж_Неодуш $12314000~@ОНЪ$17@Пред}
                                    $12314000(Z0:s> ГОРОД$12314)
```

В анализируемой фразе аргумент для слова *четверг* найден не был, однако присоединение родительного падежа возникает, например, в контексте *первый четверг месяца* (в качестве аргумента выступает слово *месяца*)[14].

Здесь необходимо сделать два важных замечания. Во-первых, одно и то же слово может быть описано двумя или более разными синтактико-семантическими формулами, соответствующими различным смыслам слова. Во-вторых, чистой морфологической и синтаксической информации уже оказывается недостаточно для полноценного анализа предложения.

Рассмотрим, к примеру, фразу *Иван пришёл из магазина*. Пользуясь результатами морфологического анализа, можно установить, что родительный падеж слова *магазин* (с предлогом *из*) соответствует смысловому типу «откуда». Поскольку этот тип присутствует в

---

[13] Символ Z0 не является обозначением аргумента.

[14] Семантический анализатор автоматически присоединяет к существительным согласующиеся с ним прилагательные и числительные, поэтому слово *первый* будет присоединено к слову *четверг*, даже если это не указано явно в синтактико-семантической формуле.

синтактико-семантической формуле слова *прийти* (в качестве аргумента Z2), анализатор может сделать вывод о наличии связи.

В большинстве случаев такой подход срабатывает, поскольку синтаксическая структура предложения, как правило, хорошо отражает структуру семантическую. Однако иногда возникают трудности, неразрешимые средствами синтаксиса. В частности, фраза *Иван пришёл из вежливости* с точки зрения синтаксиса идентична фразе *Иван пришёл из магазина*. Чтобы выбрать правильный тип сочетания *из вежливости* («почему», а не «откуда»), приходится помимо предполагаемого типа аргумента указывать и класс понятия, к которому он принадлежит.

Вернёмся к приведённому выше описанию слова *четверг*. В качестве его аргумента выступает не просто слово, стоящее в родительном падеже, но обязательно слово из класса ВРЕМЯ$16. Такие сочетания, как *первый четверг месяца*, *второй четверг января* будут корректно разобраны анализатором, поскольку слова *месяц* и *январь* входят в класс ВРЕМЯ$16. Если же попытаться обработать малоосмысленное выражение *вручение состоялось в первый четверг двери*, анализатор присоединит слово *двери* к слову *вручение* (по смыслу получится «вручение двери состоялось в первый четверг»).

Таким образом, синтаксические формулы должны содержать и базовую семантическую информацию (номер класса описываемого слова и номера классов слов-аргументов). Этих знаний оказывается достаточно для

разбора подавляющего большинства предложений, встречающихся в реальных текстах[15].

Несколько особняком стоят процедуры, обрабатывающие устойчивые сочетания (фразеологизмы) и стандартные отклонения от нормы в грамматических конструкциях. Фразеологизм рассматривается анализатором как единая конструкция, которой сопоставляется некоторая синтактико-семантическая формула. Таким образом, чтобы добавить в систему информацию о каком-либо устойчивом сочетании, требуется лишь внести изменения в синтактико-семантический словарь. Следует отметить, что под фразеологизмом в семантическом анализаторе можно понимать не только фигуральные выражения вроде *сесть в лужу* или *работать спустя рукава*, но и любые сочетания из двух или более слов, приобретающие собственную семантическую целостность: *автомобильный завод*, *учебное заведение*, *за рубежом*. Под стандартными отклонениями здесь подразумеваются, прежде всего, устойчивые конструкции неполной формы, в некотором количестве присутствующие в языке. Например, во фразах *Москва — столица* и *он студент* опущен глагол *есть*. Аналогично, в английском языке опускается местоимение *myself* в конструкции *I feel fine*. Подобные неполные формы должны быть указаны анализатору в явном виде (в текущей реализации это сделано на уровне кода программы).

---

[15] Качество анализатора очень сильно зависит от качества используемого синтактико-семантического словаря. Хотя в целом создание словаря можно считать законченным, работа над его совершенствованием постоянно ведётся.

Семантический уровень анализа текста

Хотя в настоящее время анализ текста ограничивается синтактико-семантическим уровнем, его возможности с трудом позволяют разбирать некоторые изредка встречающиеся конструкции, основанные на семантической сочетаемости слов. Проблема заключается в том, что для семантического анализатора ни один из корректных способов разбора фразы не является предпочтительным, в то время как человек подсознательно отбрасывает неправдоподобные варианты, даже не задумываясь над ними.

Рассмотрим, к примеру, фразу *он увидел её перед своими глазами*. Для человека очевиден следующий вариант разбора: *он увидел* (кого?) *её* (где?) *перед своими глазами*. Компьютер же находит и другой вариант прочтения: *он увидел* (что?) *перед* (чей?) *её* (как?) *своими глазами*. Определение слова *перед* как существительного в данном случае неоправданно, но с формальной точки зрения разбор был произведён корректно. Аналогично, фраза *норка вылезла из норки* может трактоваться анализатором не только как *норка* (зверёк) *вылезла* (вышла наружу) *из норки* (норы), но и весьма своеобразным способом: *норка* (норковый мех) *вылезла* (облезла) *из норки* (зверька), то есть на норке-зверьке облез норковый мех.

В большинстве случаев подобные трудности можно разрешить с помощью системы приоритетов (именно так сейчас и происходит). Анализатор перебирает альтернативы в порядке, указанном в синтактико-семантическом словаре. Работа прекращается после нахождения первого корректного варианта разбора. Таким образом, перемещая маргинальные трактовки слов в конец списка альтернатив, мы можем гарантировать, что более разумные варианты будут изучены раньше. Некоторые трактовки слов на практике оказываются столь редкими, что их вообще можно

исключить из рассмотрения. Например, к такому случаю относится определение слова *ушла* во фразе *она ушла от него* как краткой формы прилагательного *ушлая*. Сократить количество неправильных случаев разбора можно и при помощи уточнения словаря классов. Чем строже их иерархия, тем более качественный синтактико-семантический словарь можно создать.

Некоторые неоднозначности вообще неразрешимы на уровне анализа отдельных предложений. *Джон передал ему карту*. О какой карте идёт речь — об игральной или о географической? *The cat jumped out*. Как перевести на русский язык слово *cat* — *кошка* или *кот*?

Чтобы ответить на этот вопрос, придётся проанализировать более широкий контекст, выходящий за рамки предложения. Однако подобные задачи не входят в компетенцию семантического анализатора. Высокоуровневым анализом объектов текста должен заниматься отдельный модуль[16].

Адаптация семантического анализатора для различных языков

Адаптация технологии семантического анализа для обработки текстов на иностранных языках может оказаться трудоёмкой, но в целом не очень сложной задачей. По крайней мере, об этом свидетельствуют работы [58] и [59].

В предыдущих разделах было указано, что семантический анализатор опирается на модель текста, включающую морфологический и синтактико-

---

[16] Работа анализатора объектов вообще находится за рамками обработки текстовой информации, поскольку такой анализатор будет иметь дело со структурированным описанием входного текста, построенным семантическим анализатором.

семантический уровни, а также процедуры обработки фразеологизмов и грамматических отклонений. Рассмотрим теперь элементы анализатора с точки зрения адаптации для различных естественных языков. Забегая вперёд, можно отметить, что наиболее трудоёмкой и сложной частью работы является составление синтактико-семантического словаря, но именно словарь содержит универсальные знания, мало зависящие от используемого языка.

Морфологический анализатор неразрывно связан со структурой языка, и адаптация этого модуля невозможна. Для поддержки любого нового языка его придётся переписать. С другой стороны, общая идеология морфологического анализатора обычно проста (морфологический словарь плюс программная оболочка), и его разработку вряд ли можно считать сложной задачей.

Нерегулярности грамматических конструкций придётся внести в анализатор в явном виде. Их количество обычно невелико (меньше 50 случаев для русского). Фразеологизмы также перечисляются в списке.

Синтактико-семантический словарь претендует на роль универсальной коллекции смысловых формул, в минимальной степени зависящих от языка. Рассмотрим ещё раз описание глагола *состояться* в контексте *состоится церемония вручения*:

```
СОСТОЯТЬСЯ {Глагол.} N%~СОБЫТИЕ$11(PerfFut Z1: ! !ОНЪ$17\!ОНА$17\!ОНО$17,
                                   Z2: НЕЧТО$1~!Когда, Z3: НЕЧТО$1~!Где)
```

Здесь указано, что слово *состояться* относится к классу СОБЫТИЕ$11 и что состояться может что-то, когда-то и где-то. Аналогично будет выглядеть и описание для английского глагола *to_take_place(something, when, where)*. В этом и заключается идея семантических описаний: мы говорим об отношениях между понятиями (не зависящих от особенностей естественных языков), а не между словами. В подавляющем большинстве

случаев универсальные смысловые описания хорошо отражаются словами самых разных языков. Все люди на Земле относятся к одному биологическому виду, сталкиваются с одинаковыми ситуациями и испытывают схожие чувства. Трудно вообразить себе язык, в котором не было бы слов для обозначения солнца, работы или любви.

Затрагивая тему семантического сходства различных языков, нельзя проигнорировать существование гипотезы Сепира-Уорфа [61], утверждающей наличие сильной взаимосвязи между языком и мышлением. В частности, Э. Сепир пишет: «“Реальный мир” в значительной степени неосознанно строится на основе языковых привычек той или иной социальной группы. Два разных языка никогда не бывают столь схожими, чтобы их можно было считать средством выражения одной и той же социальной действительности. Миры, в которых живут различные общества, — это разные миры, а вовсе не один и тот же мир с различными навешанными на него ярлыками... Мы видим, слышим и вообще воспринимаем окружающий мир именно так, а не иначе главным образом благодаря тому, что наш выбор при его интерпретации предопределяется языковыми привычками нашего общества» [62].

Иногда на основе этой гипотезы делаются достаточно радикальные заявления о существовании принципиально непереводимых языковых конструкций, столь сильно связанных с особенностями окружающей среды говорящего, что человек из другого общества будет неспособен их понять. Независимо от степени истинности гипотезы Сепира-Уорфа она в равной степени относится как к ручному, так и к машинному переводу. Поскольку в реальности людьми переводы осуществляются (то есть устанавливается близость между семантическими описаниями словесных конструкций языков), нет причин считать, что эта задача непосильна для компьютера.

На практике возникают ситуации, когда некоторая осмысленная семантическая формула не может быть сопоставлена какому-либо слову целевого языка. Обычно это означает одно из двух: либо некоторое понятие исходного языка переводится словесной конструкцией, либо оно специфично для данной языковой среды и должно быть раскрыто по смыслу или даже оставлено без перевода.

Например, английское слово *advocacy* может быть переведено на русский как *адвокатская практика*. При обратном переводе сочетание *адвокатская практика* должно быть распознано как единая конструкция, отображаемое в слово *advocacy*. Таким образом, понятие *адвокатская практика* является неделимым и должно быть описано одной семантической формулой. В английском языке этой формуле будет сопоставлено одно слово, а в русском — фразеологизм.

Если понятие в принципе отсутствует в словаре целевого языка, его можно оставить в исходном виде, снабдив кратким описанием. Так, невозможно адекватно перевести на английский слово *самовар*.

Существуют и более тонкие сложности, связанные с использованием неочевидных фразеологизмов. Рассмотрим, к примеру, фразу *я не знаю, что за этим стоит*. Мы употребляем подобные конструкции не задумываясь, хотя по логике слово *стоит* в данном контексте не уместнее слов *сидит* или *лежит*. Действительно, аналогичное предложение в английском будет иметь вид *I don’t know what lies behind it* — «я не знаю, что за этим лежит». Буквальный перевод здесь оказывается неправильным, поэтому конструкцию *за этим стоит* следует рассматривать как фразеологизм, переводимый схожей фразеологизированной конструкцией английского языка.

Таким образом, концентрируясь на смыслах, а не на словах, можно создать словарь понятий, пригодный не только для русского, но и для других языков. Однако кроме описания аргументов, в синтактико-семантическое описание понятия входит и морфологическая информация, на которую опирается анализатор в процессе разбора предложения. К сожалению, эта часть описания специфична для используемого языка и должна быть изменена. Для русского языка указываются сочетающиеся с описываемым словом предлоги и падежные формы. В описаниях слов английского — языка с очень простой морфологией — почти везде будут фигурировать лишь предлоги и сочетания вида прилагательное + существительное. В словосочетаниях финского, напротив, участвуют в основном падежные формы.

Имея качественный синтактико-семантический словарь и морфологический анализатор, ценой сравнительно небольших усилий можно адаптировать семантический анализатор для обработки текстов на иностранных языках.

Схема простейшей системы машинного перевода на основе семантического анализатора

Ограничившись непосредственным выводом семантического анализатора и не проводя дополнительной работы по анализу объектов текста в рамках глобального контекста, можно построить схему простейшей системы машинного перевода (см. рис. 6.3). Важно отметить, что перевод будет осуществляться только в одну сторону (с языка L1 на язык L2). Для получения обратного перевода в общем случае понадобится построить новый синтактико-семантический словарь и реализовать сразу

несколько нетривиальных функций, не требующихся для прямого перевода.

Схема системы содержит единственный блок, работу которого мы ещё не рассматривали — блок преобразования семантического дерева обратно в текст.

Грамотно составленным пословным переводом текста работа системы не заканчивается. Полученные слова, которые хранятся в дереве разбора, необходимо поставить в правильную падежную форму, а затем сформировать предложение в соответствии с нормами целевого языка.

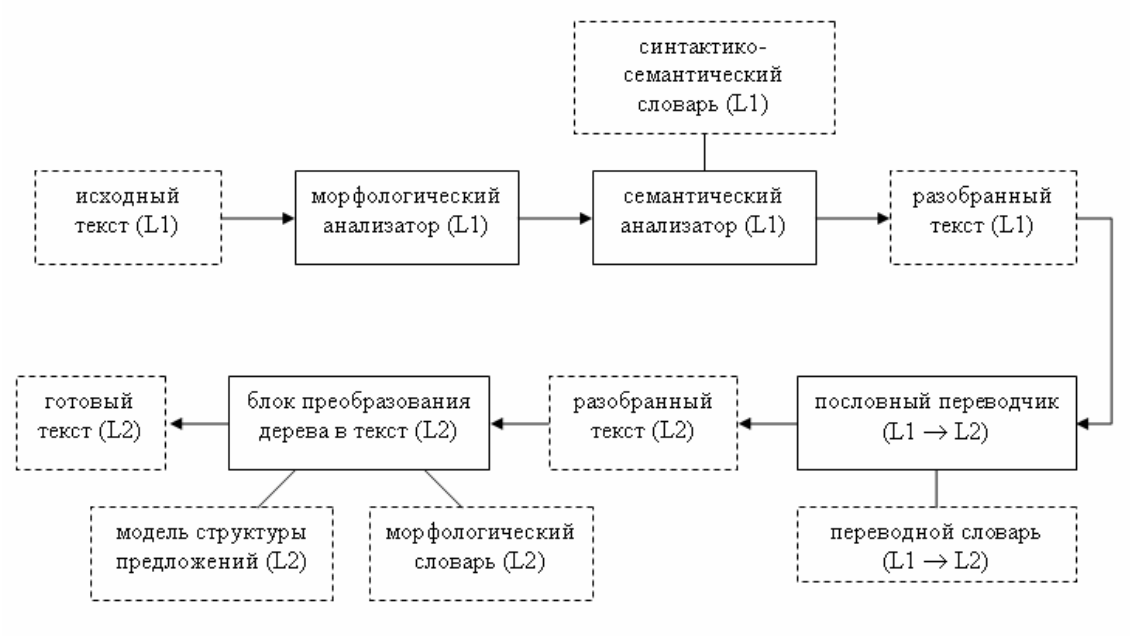

Рис. 6.3. Схема простейшей системы машинного перевода[17].

Первая задача — изменение формы слова — решается средствами морфологического анализатора языка L2. В процессе семантического

[17] Для экономии места схема анализа была несколько упрощена по сравнению с изображённой на рис. 6.1.

анализа морфологический анализатор используется для нахождения начальной формы слова и морфологического описания по известному слову текста. Теперь перед нами стоит обратная задача: по описанию и начальной форме слова сгенерировать словоформу требуемого падежа, времени, рода и числа. Имея готовый морфологический анализатор (по сути — простую «обёртку» вокруг морфологического словаря), несложно адаптировать его для работы в обратном направлении.

Сложность решения задачи преобразования дерева в текст сильно зависит от особенностей языка L2. В настоящее время наше внимание сосредоточено в основном на переводных словарях, и всерьёз обсуждать алгоритмы линеаризации текста было бы преждевременно. В простейшем случае можно воспользоваться готовой структурой исходного предложения на языке L1. По крайней мере, при переводе с английского на русский этот подход даёт неплохие результаты, поскольку порядок слов в английском строг и не противоречит модели предложения русского языка. В общем случае алгоритмы линеаризации представляют собой отдельную научную проблему. Так, в книге [9] И. Мельчук указывает, что линеаризация почти не изучается лингвистами, и предлагает собственный алгоритм преобразования дерева разбора в текст для некоторых классов предложений русского языка. Решение задачи линеаризации не ограничивается одним подбором правильного порядка слов. Иногда логика смысловой конструкции неочевидным образом преобразуется в грамматические конструкции языка. Например, во фразе *мы отправимся на прогулку, если погода будет хорошей* дважды используются глаголы будущего времени. В аналогичной фразе английского ***we will go for a walk if the weather is fine*** будущее время используется только в первом случае («отправимся на прогулку»). Во второй части фразы глагол употребляется в настоящем времени (буквально: «если погода хороша»).

Практический пример: русско-финский перевод

В настоящее время автор совместно с небольшой группой единомышленников из университета г. Йоэнсуу (Финляндия) изучает проблему создания системы русско-финского (а в перспективе и финско-русского) перевода. Объём диссертации не позволяет подробно описать все тонкости, связанные с необходимостью как-то отображать понятия русского языка на соответствующие явления финского, однако дать представление о том, как функциональная теория языка позволяет справиться с возникающими трудностями, можно.

Наиболее яркой особенностью финского языка является практически полное отсутствие предлогов. Их функции берут на себя падежные формы. Например, вопросам *где?* и *куда?* в русском языке соответствуют сочетания предлога *в* с предложным и винительным падежами. В финском же в аналогичных ситуациях используются так называемые *местные падежи*[18]. Некоторые примеры этого явления показаны на табл. 6.2.

Если речь не идёт о физических предметах, имеющих ярко выраженные «внутренние» и «внешние» стороны, между финским и русским языком возможны расхождения. Так, мы говорим «на концерт», в то время как финн скажет «в концерт». Выражение «в сентябре» будет передано инессивом, то есть наиболее ожидаемым для русскоговорящего способом, а сочетание «в понедельник» — *эссивом*, служащим, вообще говоря, для указания на состояние объекта («он пришёл с работы *уставшим*»).

---

[18] Разумеется, это весьма упрощённый взгляд на вещи. На практике установление соответствия изобилует всевозможными тонкостями и исключениями.

Таблица 6.2. Местные падежи финского языка

Падеж	Пример	Аналог в русском языке
иллатив	saun*a**an***	(куда?) в сауну
инессив	saun*a**ssa***	(где?) в сауне
элатив	saun*a**sta***	(откуда?) из сауны
аллатив	tuol*i**lle***	(куда?) на стул
адессив	tuol*i**lla***	(где?) на стуле
аблатив	tuol*i**lta***	(откуда?) со стула

Как и в русском, в финском по падежам изменяются не только объекты, но и слова, их определяющие. Ср.: iso sauna (большая сауна) — iso*ssa* saunas*sa* (в большой сауне).

Перед тем, как перейти к примерам перевода, необходимо отметить ещё одну интересную особенность финского языка: как и в русском языке, финские глаголы изменяются по лицам и числам (см. табл. 6.3).

Таблица 6.3. Изменение глаголов по лицам и числам

Форма	Русский глагол	Финский глагол
инфинитив	говорить	puhu***a***
1-е лицо, ед. число	(я) говорю	puhu***n***
1-е лицо, мн. число	(мы) говорим	puhu***mme***
2-е лицо, ед. число	(ты) говоришь	puhu***t***
2-е лицо, мн. число	(вы) говорите	puhu***tte***
3-е лицо, ед. число	(он/она/оно) говорит	puhuu
3-е лицо, мн. число	(они) говорят	puhu***vat***

Таким образом, даже для перевода простейших фраз вроде *я живу в большой квартире* требуются:

1. умение правильно выбрать глагольную форму в зависимости от вида субъекта;
2. способность определить по сочетанию предлог + падеж русского языка соответствующий падеж финского;
3. знание семантических отличий фраз вида *в сентябре* и *в понедельник*.

Приведённые задачи могут быть успешно решены с помощью семантического анализатора. Субъект можно определить по типу связи с центральным глаголом (именительный падеж). Информация о лице и числе входит в синтактико-семантическое описание слова-субъекта. Используемый финский падеж может быть связан с синтактико-семантической формулой предлога *в*, поскольку различным его смыслам сопоставлены различные описания.

Рассмотрим, к примеру, фрагмент словаря, при помощи которого можно перевести фразы

```
Я еду в Москву.
Они едут в Москву.
Я еду в Москву в сентябре.
Я еду в Москву в понедельник.
Я живу в Москве.
Я уезжаю из Москвы.
Иван живёт в большой красивой квартире.
Я живу в большой красивой квартире.
```

Словарь представляет собой таблицу, каждая строка которой описывает перевод какой-либо пары (слово, синтактико-семантическая формула) на финский язык (см. табл. 6.4).

Таблица 6.4. Фрагмент русско-финского переводного словаря

Слово и формула	Перевод
БОЛЬШОЙ $12/11401/05(Z0:a> НЕЧТО$1, Z1: НЕЧТО$1~!поДат)	iso
В (Z0:y> @Где,Z1: ПОСЕЛЕНИЕ$123~!Пред)	*INESS
В (Z0:y> @Когда,Z1: ВОСКРЕСЕНЬЕ$1602\ОТСУТСТВИЕ$1101/0/15~!Вин)	*ESS
В (Z0:y> @Когда,Z1: ВРЕМЯ$16\ПРОЦЕСС$16132\ВОЗРАСТ$124/2~!Пред)	*INESS
В (Z0:y> @Куда,Z1: ПОСЕЛЕНИЕ$123~!Вин)	*ILLAT
ЕХАТЬ N%~ПОЕЗДКА$15402(Z1: !ОНИ$17,Z2: ПРИЧИНА$1/37/05\ПРИКАЗ$1526031~!Почему,Z3: НЕЧТО$1~!поДат,Z4: НЕЧТО$1~!Откуда,Z5: НЕЧТО$1~!Куда,Z6: !Тв\!наПред)	mennä
ЕХАТЬ N%~ПОЕЗДКА$15402(Z1: !Я$17,Z2: ПРИЧИНА$1/37/05\ПРИКАЗ$1526031~!Почему,Z3: НЕЧТО$1~!поДат,Z4: НЕЧТО$1~!Откуда,Z5: НЕЧТО$1~!Куда,Z6: !Тв\!наПред)	mennä
ЖИТЬ N%~ЖИЗНЬ$10(Z1: НЕЧТО$1~!ОНЪ$17\!ОНА$17\!ОНО$17,Z2: НЕЧТО$1~!сТв,Z3: ТЕЛО$12\РАССТОЯНИЕ$12/32~!Где,Z4:	asua

ДЕЙСТВИЕ$15\ИДЕЯНИЕ$14~!Тв,Z5: !наВин,Z6: ЧЕЛОВЕК$1241~!подТв)	
ЖИТЬ N%~ЖИЗНЬ$10(Z1: НЕЧТО$1~!Я$17,Z2: НЕЧТО$1~!сТв,Z3: ТЕЛО$12\РАССТОЯНИЕ$12/32~!Где,Z4: ДЕЙСТВИЕ$15\ИДЕЯНИЕ$14~!Тв,Z5: !наВин,Z6: ЧЕЛОВЕК$1241~!подТв)	asua
ИВАН $12413/11000(Z0:s> ЧЕЛОВЕК$1241)	Ivan
ИЗ (Z0:y> @Откуда,Z1: ЗЕМЛЯ$121252\ПОСЕЛЕНИЕ$123\ОБЛАСТЬ$12311\УКРЫТИЕ$123361\КОМПАНИЯ$123411\МАССЫ$12411/0/11~!Род)	*ELAT
КВАРТИРА $123314(Z1: ЧЕЛОВЕК$1241\ПОСЕЛЕНИЕ$123~!Род)	asunto
КРАСИВЫЙ $12/120310/05(Z0:a> НЕЧТО$1,Z1: !Для)	kaunis
МОСКВА $12314000(Z0:s> ГОРОД$12314)	Moskova
ОНИ $1241()	he
ПОНЕДЕЛЬНИК $1602(Z1: ВРЕМЯ$16~!Род)	maanantai
СЕНТЯБРЬ $1603(Z1: ГОД$1605~!Род)	syyskuu
УЕЗЖАТЬ N%~ПОЕЗДКА$15402(Z1: !Я$17,Z2: НЕЧТО$1~!Откуда,Z3: НЕЧТО$1~!Куда\!Ото)	lähteä
Я $1241(Z1: ЖИВОЙ$124~!сТв)	minä

Для предлогов в столбце *Перевод* указан падеж, передаваемый данным предлогом. Так, предлогу *в* с формулой

(Z0:y> @Где**,Z1:** ПОСЕЛЕНИЕ**$123~!**Пред**)**

(вопросительное слово *где*, связь с объектом из класса ПОСЕЛЕНИЕ$123 с помощью предложного падежа) соответствует инессив. Следующая строка сопоставляет предлогу *в* в смысле *в <день недели>* падеж эссив.

Программа-переводчик работает следующим образом. Сначала строится дерево исходного предложения. Затем вызывается рекурсивная процедура обхода дерева в глубину. Одним из аргументов процедуры является *текущий падеж*; при первом вызове в качестве текущего падежа передаётся именительный (номинатив). На очередном шаге алгоритм переводит на финский язык слово из просматриваемого узла с помощью словаря. Если слово является глаголом, требуемая форма лица и числа устанавливается с помощью просмотра описания потомка узла, выполняющего функцию субъекта. Если слову соответствует падеж (содержимое графы *Перевод* отмечено звёздочкой), оно никак не переводится, но при вызове процедуры обхода поддерева текущего узла в качестве аргумента *текущий падеж* будет передано новое значение. К переведённому слову присоединяется соответствующее падежное окончание[19] (в случае глаголов присоединяется личное окончание), и слово помещается в результирующий список. По окончании обхода дерева слова в списке упорядочиваются в соответствии с их исходным месторасположением (таким образом, порядок слов в текущей версии переводчика сохраняется) и предложение выдаётся пользователю.

Полученные с помощью системы переводы показаны на табл. 6.5.

---

[19] В настоящее время мы просто присоединяем к слову наиболее типичное окончание со знаком дефиса. В общем случае для конструирования корректных словоформ требуется качественный морфологический анализатор. Для финского такие программы существуют, но практически все они коммерческие.

Таблица 6.5. Примеры переведённых предложений

Исходное предложение	Перевод
Я еду в Москву.	minä mennä-n Moskova-_n
Они едут в Москву.	he mennä-vat Moskova-_n
Я еду в Москву в сентябре.	minä mennä-n Moskova-_n syyskuu-ssa
Я еду в Москву в понедельник.	minä mennä-n Moskova-_n maanantai-na
Я живу в Москве.	minä asua-n Moskova-ssa
Я уезжаю из Москвы.	minä lähteä-n Moskova-sta
Иван живёт в большой красивой квартире.	Ivan asua__ iso-ssa kaunis-ssa asunto-ssa
Я живу в большой красивой квартире.	minä asua-n iso-ssa kaunis-ssa asunto-ssa

Глава 7. Технические детали

Семантический анализатор В. Тузова является самостоятельной программой, генерирующей деревья разбора предложений и синтактико-семантические описания слов для любого входного текстового документа. Выходные данные, таким образом, сами по себе имеют достаточно сложный вид, и их использование вряд ли можно назвать тривиальной задачей. В предыдущих главах мы несколько дистанцировались от вопроса организации интерфейса между семантическим анализатором и другими программами, его использующими. Поскольку данная работа посвящена практическим приложениям семантического анализатора, игнорировать такую важную (хотя и чисто техническую) тему было бы неправильно.

Текущая реализация семантического анализатора и её перспективы

Семантический анализатор запрограммирован на языке FORTH с использованием компилятора Win32Forth [63]. Исторически система представляет собой автономное приложение, работающее по следующей схеме. В конфигурационном файле info.txt пользователь задаёт имена входного и выходного файлов. Затем анализатор запускается с помощью команды semLPB.exe. После окончания работы семантического анализатора выходной файл будет содержать все данные о разборе предложений в текстовом виде.

Подходящий для изучения работы самого анализатора механизм оказывается достаточно громоздким при автоматизированном вызове процедуры семантического анализа из других приложений. К счастью, последние версии компилятора Win32Forth позволяют создавать не только

исполняемые файлы, но и DLL-модули. Используя эту возможность, мы планируем сделать взаимодействие анализатора с внешним миром более простым и удобным[20].

Формат выходных данных семантического анализатора

Изучить выходной файл, генерируемый семантическим анализатором, проще всего на примере. Допустим, на вход поступил текстовый документ, содержащий два предложения:

```
В четверг во Владимире состоится церемония вручения золотых медалей
выпускникам-отличникам Владимирской области. В этом году медалистов
больше, чем в прошлом.
```

В процессе работы семантический анализатор создаёт приведённую ниже распечатку:

```
c:\analiz\infile.txt

==================================================
 В четверг во Владимире состоится церемония вручения золотых медалей
выпускникам -
отличникам Владимирской области.
..................................................
состоится<X005.001>
  (@Когда В<X001.130>
    (@Вин четверг<X002.001>),
   @Где во<X003.046>\<X003.085>
    (@Пред Владимире<X004.001>\<X004.002>\<X004.003>),
   @Им церемония<X006.001>
    (@Род вручения<X007.001>
```

---

[20] Перейти от запуска EXE-файла к вызову функции DLL-модуля нетрудно. Основная задача заключается в том, чтобы грамотно организовать передачу во внешний мир структур, в которых записаны деревья разбора предложений и сведения из синтактико-семантического словаря. Если же оставить без изменения нынешний формат передачи данных в текстовом виде, большого преимущества DLL-модуль не даст.

```
      (@Род медалей<X009.001>\<X009.002>
        (@Род золотых<X008.001>\<X008.005>),
       @Дат выпускникам<X010.001>
        (@Дат отличникам<X012.001>
          (@Род области<X014.001>\<X014.003>
            (@Род Владимирской<X013.001>)
           )
         )
       )
     )
   )
.
==================================================
В
  **
  <X001.130> В {Предлог. $1602~@Когда} (Z0:y> @Когда,Z1:
ВОСКРЕСЕНЬЕ$1602\ОТСУТСТВИЕ$1101/0/15~!Вин)
  s: <X001.130> В Y1>Temp(Y1:,РОД:Z1) \\ <129>
  v: Z1: $1602~@Вин => <X002.001>
     Z0:y> $16~@Когда <= <X005.001>
четверг
  **
  <X002.001> ЧЕТВЕРГ {Сущв._Муж_Неодуш $1602~@ОНЪ$17@Вин} $1602(Z1:
ВРЕМЯ$16~!Род)
  s: <X002.001> ЧЕТВЕРГ (РОД:Z1) \\ <1>
  v: Z1: $1602~@Вин <= <X001.130>
во
  **
  <X003.046> ВО {Предлог. $12413/11~@Где} (Z0:y> @Где,Z1:
ЖИВОЙ$124\МЕСТО$12/3\ГЛОТКА$104/022\ЦЕНТР$12/15/05\МРАК$12/1121/17~!Пред)
  s: <X003.046> ВО Y1>Loc(Y1:,ВНУТРИ$12/313/05(ПРЕД:Z1)) \\ <46>
  v: Z1: $124~@Пред => <X004.002>
     Z1: $124~@Пред => <X004.003>
     Z3: $1~@Где <= <X005.001>
  **
  <X003.085> ВО {Предлог. $12314~@Где} (Z0:y> @Где,Z1:
ПОСЕЛЕНИЕ$123~!Пред)
  s: <X003.085> ВО Y1>Loc(Y1:,ВНУТРИ$12/313/05(ПРЕД:Z1)) \\ <85>
  v: Z1: $123~@Пред => <X004.001>
     Z3: $1~@Где <= <X005.001>
Владимире
  **
  <X004.001> ВЛАДИМИР {Сущв._Муж_Неодуш $12314000~@ОНЪ$17@Пред}
$12314000(Z0:s> ГОРОД$12314)
  s: <X004.001> ВЛАДИМИР S1>Hab(S1:ГОРОД$12314,S0:НАЗВАНИЕ$12/1172) \\
```

<1>
  v: Z1: $123~@Пред **<= <X003.085>**
  ******
  <X004.002> ВЛАДИМИР **{**Сущв**.**_Муж_Одуш **$12413/11000~@**ОНЪ**$17@**Пред**}**
$12413/11000(Z0:s> ЧЕЛОВЕК**$1241)**
  s: <X004.002> ВЛАДИМИР **S1>Hab(S1:**ЧЕЛОВЕК**$1241,S0:**ИМЯ**$1241/11) \\ <2>**
  v: Z1: $124~@Пред **<= <X003.046>**
  ******
  <X004.003> ВЛАДИМИРА **{**Сущв**.**_Жен_Одуш **$12413/11000~@**ОНА**$17@**Пред**}**
$12413/11000(Z0:s> ЧЕЛОВЕК**$1241)**
  s: <X004.003> ВЛАДИМИРА **S1>Hab(S1:**ЧЕЛОВЕК**$1241,S0:**ИМЯ**$1241/11) \\ <1>**
  v: Z1: $124~@Пред **<= <X003.046>**
состоится
  ******
  <X005.001> СОСТОЯТЬСЯ **{**Глагол**.} N%~**СОБЫТИЕ**$11(PerfFut Z1:**
!ОНЪ**$17\!**ОНА**$17\!**ОНО**$17,Z2:** НЕЧТО**$1~!**Когда**,Z3:** НЕЧТО**$1~!**Где**)**
  s: <X005.001> СОСТОЯТЬСЯ **PerfFunc(Loc(Temp(**ИМ**:Z1,**КОГДА**:Z2),**ГДЕ**:Z3)) **
<1>
  v: Z1: @ОНА**$17 => <X006.001>**
     Z3: $1~@Где **=> <X003.046>**
     Z3: $1~@Где **=> <X003.085>**
     $16~@Когда **=> <X001.130>**
церемония
  ******
  <X006.001> ЦЕРЕМОНИЯ **{**Сущв**.**_Жен_Неодуш **$1241/400131/05~@**ОНА**$17@**Им**}**
N%~ЦЕРЕМОНИАЛ**$1241/400131/05(Z1: !**Род**,Z2:** НЕЧТО**$1~!**сТв**)**
  s: <X006.001> ЦЕРЕМОНИЯ
S0>Oper00(РОД**:Z1,**ЦЕРЕМОНИАЛ**$1241/400131/05(**СТВ**:Z2)) \\ <1>**
  v: Z1: @Род **=> <X007.001>**
     Z1: @ОНА**$17 <= <X005.001>**
вручения
  ******
  <X007.001> ВРУЧЕНИЕ **{**Сущв**.**_Сред_Неодуш **$15310/0/04~@**ОНО**$17@**Род**}**
$15310/0/04(Z1: !Тв**,Z2: !**Род**,Z3: !**Дат**)**
  s: <X007.001> ВРУЧЕНИЕ
S0>Caus(ТВ**:Z1,IncepLoc(Z2,**ВНУТРИ**$12/313/05(**РОД**:**РУКА**$104/03(**ДАТ**:Z3)))) **
<1>
  v: Z2: @Род **=> <X009.001>**
     Z2: @Род **=> <X009.002>**
     Z3: @Дат **=> <X010.001>**
     Z1: @Род **<= <X006.001>**
золотых
  ******
  <X008.001> ЗОЛОТОЙ **{**Прил**. $1~@**ОНИ**$17@**Род**} $12/1121/03(Z0:a>** НЕЧТО**$1)**
  s: <X008.001> ЗОЛОТОЙ

A1>Hab(A1:НЕЧТО**$1,**ЦВЕТ**$12/112/05(**ПРИСУЩИЙ**:**ЗОЛОТО**$121450)) \\ <1>**
v: Z0:a> *@КАКОЙ**:**ПрилСущ**7 <= <X009.001>**
Z0:a> *@КАКОЙ**:**ПрилСущ**7 <= <X009.002>**
******
<X008.005> ЗОЛОТОЙ **{**Прил**. $1~@**ОНИ**$17@**Род**}** **N%~**ЗОЛОТО**$121450(Z0:a>** НЕЧТО**$1)**
s: <X008.005> ЗОЛОТОЙ **A1>Rel(A1:**НЕЧТО**$1,**ЗОЛОТО**$121450) \\ <6>**
v: Z0:a> *@КАКОЙ**:**ПрилСущ**7 <= <X009.001>**
Z0:a> *@КАКОЙ**:**ПрилСущ**7 <= <X009.002>**
медалей
******
<X009.001> МЕДАЛЬ **{**Сущв**.**_Жен_Неодуш **$121368~@**ОНИ**$17@**Род**}** **$121368(Z1: !**Род**)**
s: <X009.001> МЕДАЛЬ **(**РОД**:Z1) \\ <1>**
v: *@КАКОЙ**:**ПрилСущ**7 => <X008.001>**
***@**КАКОЙ**:**ПрилСущ**7 => <X008.005>**
Z2: @Род **<= <X007.001>**
******
<X009.002> МЕДАЛЬ **{**Сущв**.**_Жен_Неодуш **$14133~@**ОНИ**$17@**Род**}** **$14133(Z0:s>** ЗНАК**$14133,Z1: !**заВин**)**
s: <X009.002> МЕДАЛЬ **S1>Usor(S1:**ЗНАК**$14133,**НАГРАДА**$1241/131/03(**ЗАВИН**:Z1)) \\ <2>**
v: *@КАКОЙ**:**ПрилСущ**7 => <X008.001>**
***@**КАКОЙ**:**ПрилСущ**7 => <X008.005>**
Z2: @Род **<= <X007.001>**
выпускникам
******
<X010.001> ВЫПУСКНИК **{**Сущв**.**_Муж_Одуш **$12413213~@**ОНИ**$17@**Дат**}** **$12413213(Z1:** ШКОЛА**$123401~!**Род**,Z0:s>** ЧЕЛОВЕК**$1241)**
s: <X010.001> ВЫПУСКНИК **S1>Oper01(**РОД**:Z1,**ВЫПУСК**$152314(S1:**ЧЕЛОВЕК**$1241)) \\ <1>**
v: @Дат **=> <X012.001>**
Z3: @Дат **<= <X007.001>**
отличникам
******
<X012.001> ОТЛИЧНИК **{**Сущв**.**_Муж_Одуш **$1241~@**ОНИ**$17@**Дат**}** **$1241(Z0:s>** ЧЕЛОВЕК**$1241,Z1: !**Род**)**
s: <X012.001> ОТЛИЧНИК **S1>Hab(S1:**ЧЕЛОВЕК**$1241(**РОД**:Z1),**ЛУЧШИЙ**$1/311/06(Mult(**ОЦЕНКА**$1/31))) \\ <1>**
v: Z1: @Род **=> <X014.001>**
Z1: @Род **=> <X014.003>**
@Дат **<= <X010.001>**
Владимирской
******
<X013.001> ВЛАДИМИРСКИЙ **{**Прил**.**_Жен **$1~@**ОНА**$17@**Род**}**

```
N%~ВЛАДИМИР$12314000(Z0:a> НЕЧТО$1)
  s: <X013.001> ВЛАДИМИРСКИЙ A1>Rel(A1:НЕЧТО$1,ВЛАДИМИР$12314000) \\ <1>
  v: Z0:a> *@КАКОЙ:ПрилСущ7 <= <X014.001>
     Z0:a> *@КАКОЙ:ПрилСущ7 <= <X014.003>
области
  **
  <X014.001> ОБЛАСТЬ {Сущв._Жен_Неодуш $100/5~@ОНА$17@Род} $100/5(Z1:
НЕЧТО$1~!Род)
  s: <X014.001> ОБЛАСТЬ (РОД:Z1) \\ <1>
  v: *@КАКОЙ:ПрилСущ7 => <X013.001>
     Z1: @Род <= <X012.001>
  **
  <X014.003> ОБЛАСТЬ {Сущв._Жен_Неодуш $12311~@ОНА$17@Род} $12311(Z1:
ЗЕМЛЯ$12272\ПОСЕЛЕНИЕ$123~!Род)
  s: <X014.003> ОБЛАСТЬ (РОД:Z1) \\ <3>
  v: *@КАКОЙ:ПрилСущ7 => <X013.001>
     Z1: @Род <= <X012.001>
=====================================================
 В этом году медалистов больше, чем в прошлом.
.....................................................
@Сравн больше<X005.004>
  (@Род медалистов<X004.001>
    (@Когда В<X001.127>
      (@Пред году<X003.003>
        (@Пред этом<X002.002>)
       )
     ),
   @Сравн чем<X007.008>
    (@Когда в<X008.127>
      (@Пред прошлом<X009.001>)
     )
   )
.
=====================================================
В
  **
  <X001.127> В {Предлог. $1605~@Когда} (Z0:y> @Когда,Z1:
ВРЕМЯ$16\ПРОЦЕСС$16132\ВОЗРАСТ$124/2~!Пред)
  s: <X001.127> В Y1>Temp(Y1:,ПРЕД:Z1) \\ <127>
  v: Z1: $16~@Пред => <X003.003>
     Z0:y> @Когда <= <X004.001>
этом
  **
  <X002.002> ЭТОТ {ПрилМ_МжСр $1~@ОНЪ$17@Пред} $1/45/05(Z0:a> НЕЧТО$1)
  s: <X002.002> ЭТОТ A1>Copul(A1:НЕЧТО$1,Ne(ДРУГОЙ$1/45/15)) \\ <1>
```

v: Z0:a> *@ПрилМСущ**2 <= <X003.003>**
году
**
<X003.003> ГОД **{**Сущв._Муж_Неодуш **$1605~@**ОНЪ**$17@**Пред**} $1605(Z1:**
БОРЬБА**$151**ВЛАСТИ**$12413205**СУЩЕСТВОВАНИЕ**$1101~!**Род**)**
s: <X003.003> ГОД **(**РОД**:Z1) \\ <3>**
v: *@ПрилМСущ**2 => <X002.002>**
Z1: $16~@Пред **<= <X001.127>**
медалистов
**
<X004.001> МЕДАЛИСТ **{**Сущв._Муж_Одуш **$1241~@**ОНИ**$17@**Род**} $1241(Z0:s>**
ЧЕЛОВЕК**$1241)**
s: <X004.001> МЕДАЛИСТ **S1>Hab(S1:**ЧЕЛОВЕК**$1241,**МЕДАЛЬ**$121368) \\ <1>**
v: @Когда **=> <X001.127>**
@Род **<= <X005.004>**
больше
**
<X005.004> БОЛЬШОЙ **{**Прил**. $1/~@**Сравн**} $1/293(Z0:a>** АСВОЙСТВО**$1/)**
s: <X005.004> БОЛЬШОЙ **A1>Hab(A1:**АСВОЙСТВО**$1/,Magn(**СТЕПЕНЬ**$1/293)) **
<1>
v: *@СРАВНЕНИЕ**:3 => <X007.008>**
@Род **=> <X004.001>**
чем
**
<X007.008> ЧЕМ **{**Прил**. $1~@**Сравн**} N%~**СРАВНЕНИЕ**$142/31@**Сравнение**(Z0:a>**
НЕЧТО**$1)**
s: <X007.008> ЧЕМ **A1>Oper01(A1:**НЕЧТО**$1,**СРАВНЕНИЕ**$142/31) \\ <1>**
v: *@СРАВНЕНИЕ**:**Чем**1 => <X008.127>**
Z0:a> *@СРАВНЕНИЕ**:3 <= <X005.004>**
в
**
<X008.127> В **{**Предлог**. $16/10/15~@**Когда**} (Z0:y> @**Когда**,Z1:**
ВРЕМЯ**$16**ПРОЦЕСС**$16132**ВОЗРАСТ**$124/2~!**Пред**)**
s: <X008.127> В **Y1>Temp(Y1:,**ПРЕД**:Z1) \\ <127>**
v: Z1: $16~@Пред **=> <X009.001>**
Z0:y> *@СРАВНЕНИЕ**:**Чем**1 <= <X007.008>**
прошлом
**
<X009.001> ПРОШЛОЕ **{**Сущв._Сред_Неодуш **$16/10/15~@**ОНО**$17@**Пред**}**
$16/10/15(Z0:s> ВРЕМЯ**$16,Z1: !**Род**)**
s: <X009.001> ПРОШЛОЕ **S1>AntauT(S1:**ВРЕМЯ**$16(**РОД**:Z1),**НАСТОЯЩЕЕ**$16/10/00)**
\\ <1>
v: Z1: $16~@Пред **<= <X008.127>**

!!.....это конец файла**.....!!**

Структура результирующего файла анализатора (на самом высоком уровне абстракции) выглядит следующим образом:

```
имя входного файла

==================================================
 исходное предложение 1
..................................................
дерево разбора предложения 1
==================================================
описание первого слова предложения 1
описание второго слова предложения 1
...
описание последнего слова предложения 1
==================================================
...
...
...
==================================================
 исходное предложение N
..................................................
дерево разбора предложения N
==================================================
описание первого слова предложения N
описание второго слова предложения N
...
описание последнего слова предложения N

!!.................это конец файла.................!!
```

Дерево разбора каждого предложения записывается в виде скобочной формы (ср. скобочную запись первого предложения приведённой выше распечатки с рис. 6.2). Перед каждым словом, кроме корневого, записывается тип грамматической связи с его словом-предком. Обычно типом является название падежа (@Род, @Дат, @Им), либо вид вопроса (@Где, @Когда, @Как). После слова в угловых скобках записывается его порядковый номер в исходном предложении и номер выбранной альтернативы в синтактико-семантическом словаре. Так, запись *состоится<X005.001>* означает, что в исходном предложении слово

состоится — пятое по счёту, и что анализатор выбрал его первое (и единственное) описание из синтактико-семантического словаря в качестве корректного. Если в данном контексте подходят сразу несколько описаний, анализатор выводит их все: *области*<*X014.001*>\<*X014.003*>.

Сразу за скобочной структурой дерева следует секция описаний отдельных слов предложения. Для каждого слова анализатор указывает сведения из синтактико-семантического и смыслового словарей, а также выводит список связей, в которых данное слово участвует. Рассмотрим, к примеру, описание слова *вручения*:

```
вручения
  **
  <X007.001> ВРУЧЕНИЕ {Сущв._Сред_Неодуш $15310/0/04~@ОНО$17@Род}
                                $15310/0/04(Z1: !Тв, Z2: !Род,Z3: !Дат)
  s: <X007.001> ВРУЧЕНИЕ S0>Caus(ТВ:Z1,IncepLoc(Z2,
                    ВНУТРИ$12/313/05(РОД:РУКА$104/03(ДАТ:Z3)))) \\<1>
  v: Z2: @Род => <X009.001>
     Z2: @Род => <X009.002>
     Z3: @Дат => <X010.001>
     Z1: @Род <= <X006.001>
```

Строка, начинающаяся с <*X007.001*>, содержит краткую характеристику слова (часть речи, род, класс) и сигнатуру из синтактико-семантического словаря. В следующей строке (начинающейся с *s:*) находится описание слова из смыслового словаря. В списке, помеченном знаком *v:*, перечисляются связи, относящиеся к слову *вручения* (см. табл. 7.1).

Таблица 7.1. Список связей слова *вручения*

Связь	Описание
Z2: @Род => <X009.001>	присоединяет (=>) в качестве аргумента Z2 слово *медалей* (009.001)
Z2: @Род => <X009.002>	присоединяет (=>) в качестве аргумента Z2 слово *медалей*[21] (009.002)
Z3: @Дат => <X010.001>	присоединяет (=>) в качестве аргумента Z3 слово *выпускникам* (010.001)
Z1: @Род <= <X006.001>	само присоединяется (<=) к слову *церемония* (006.001) как аргумент Z1

Изучая записи выходного файла семантического анализатора, любое приложение может восстановить исходную структуру дерева разбора любого предложения, а также возможные трактовки отдельных слов. Разные задачи требуют разного уровня детальности изучения распечатки анализатора. Так, при разработке контекстно-ориентированного тезауруса знать структуру предложений необязательно — достаточно лишь найти строку, содержащую описание слова, для которого подбираются синонимы. Не требуется анализ структуры предложений и в задаче токенизации (см. главу 5). Напротив, при разработке вопросно-ответных систем именно деревья разбора ставятся во главу угла.

[21] То есть присоединяется другая трактовка слова *медаль* (украшение, внешне напоминающее медаль).

Заключение

Задача обработки текстов на естественном языке является одной из наиболее актуальных проблем компьютерной науки последних десятилетий. В простых задачах (распознавание языка документа, составление частотного словаря) от компьютера не требуется понимания содержания текстов. Если же речь заходит о системах машинного перевода или диалоговых программах, без определения смысла фраз уже не обойтись. Любой алгоритм, выполняющий анализ структуры текста, опирается на какую-либо модель языка. Даже простейшие утверждения наподобие «наиболее часто встречающиеся слова документа определяют его тематику» по сути являются простыми моделями сложных языковых явлений.

Семантический анализатор В. Тузова представляет собой полноценную систему анализа текста, опирающуюся на оригинальную функциональную теорию языка. Нельзя не отметить, что при ближайшем рассмотрении функциональная теория оказывается на редкость простой и изящной, что является хорошим признаком её адекватности (хотя и ничего не доказывающим с формальной точки зрения).

В то время как в мире существует довольно много разработок, основанных на классических теориях (в первую очередь, на грамматиках Хомского), проекты, использующие теорию В. Тузова, пока ещё практически не выходят за рамки лабораторных экспериментов. В этом нет ничего удивительного, учитывая относительную молодость функциональной теории, недостаток литературы и ориентацию на русский язык текущей версии семантического анализатора. Кроме того, нельзя игнорировать тот простой факт, что любая сколько-нибудь серьёзная разработка, основанная на семантическом анализаторе, требует солидных

трудозатрат и, следовательно, капиталовложений. Малочисленная группа энтузиастов продукт промышленного уровня не осилит.

Целью данной работы была попытка показать, что семантический анализатор может быть применён при решении самых различных задач, где требуются технологии NLP. На нынешний момент нам представляется, что именно широта охвата предметной области могла бы привлечь внимание к алгоритмам семантического анализа и помочь понять, где анализатор может быть особенно эффективен.

В рамках исследований изучались такие направления, как создание вопросно-ответных систем, информационный поиск и рубрикация, инструменты проверки правописания и подбора синонимов, поиск частичных совпадений и выявление плагиата, а также машинный перевод. Были разработаны:

- экспериментальная вопросно-ответная система первого уровня понимания;
- классификация вопросительных предложений, пригодная для последующего использования в диалоговых приложениях;
- система информационного поиска, опирающаяся на семантические формулы слов документов коллекции;
- модуль поиска связанных слов;
- контекстно-ориентированный электронный тезаурус;
- система поиска плагиата в текстах на русском языке, использующая систему классов как основу модуля токенизации;
- рабочая модель системы машинного перевода.

В настоящее время наиболее приоритетным направлением исследований автора данной работы является машинный перевод. Мы пытаемся привлечь внимание зарубежных специалистов к нашим идеям. Сотрудничая с носителями иностранных языков, мы надеемся достичь более глубокого понимания проблем машинного перевода и добиться качественных результатов.

Литература

[1] MacDonald N. Language Translation by Machine — a Report of the First Successful Trial // Computers and Automation. — 1954. — Vol. 3(2). — P. 6-10.

[2] Вебсайт компании: http://www.promt.ru

[3] Разинов П.А., Афанасьева В.Н. Финский язык для начинающих. — СПб: М. Г. В., 2001. — 270 с.

[4] McCarthy J., Minsky M.L., Rochester N., Shannon C.E. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. — Dartmouth, 1955.

[5] Джонс М.Т. Программирование искусственного интеллекта в приложениях. — М.: ДМК Пресс, 2004. — 312 с.

[6] Афонин В.Л., Макушкин В.А. Интеллектуальные робототехнические системы. — М.: ИНТУИТ, 2005. — 208 с.

[7] Хомский Н. Аспекты теории синтаксиса. — М.: Изд-во БГК им. И.А. Бодуэна Де Куртенэ, 1999. — 235 с.

[8] Мельчук И.А. Опыт теории лингвистических моделей «смысл ó текст»: семантика, синтаксис. — М.: Наука, 1974. — 314 с.

[9] Мельчук И.А. Русский язык в модели «смысл ó текст». — М.: Языки русской культуры, 1995. — 682 с.

[10] Charniak E. Statistical Parsing with a Context-free Grammar and Word Statistics // In Proc. of the 14th National Conference on Artificial Intelligence, CA, USA. — 1997. — P. 598-603.

[11] Соловьёв В.Д. Возможный подход к универсализации модели «смысл ó текст» // Труды международной конференции «Диалог». — 2003.

[12] Тузов В.А. Математическая модель языка. — Л.: Изд-во Ленингр. ун-та, 1984. — 176 с.

[13] Дерновой Г. О пользе случайностей // Компьютерра. — 2002. — N 25.

[14] Вебсайт проекта «SemLP-технология»: http://www.semlp.com

[15] Чеповский А. Неразрешимая проблема компьютерной лингвистики // Компьютерра. — 2002. — N 30.

[16] Uchida H., Zhu M., Della Senta T. The UNL, a Gift for a Millennium. — Tokyo: UNU Press, 1999.

[17] Молчанов А. Системное программное обеспечение: учебник для вузов. — СПб.: Питер, 2003. — 396 с.

[18] Мозговой М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. — СПб.: Наука и Техника, 2006. — 320 с.

[19] Вебсайт проекта OpenNLP: http://opennlp.sourceforge.net

[20] Marcus M.P., Santorini B., Marcinkiewicz M.A. Bulding a Large Annotated Corpus of English: the Penn Treebank // Computational Linguistics. — 1993. — Vol. 19. — P. 313-330.

[21] Тузов В.А. Компьютерная семантика русского языка. — СПб.: Изд-во СПбГУ, 2004. — 400 с.

[22] Weizenbaum J. ELIZA — a Computer Program for the Study of Natural Language Communication between Man and Machine // Communications of the ACM. — 1966. — Vol. 9(1). — P. 35-36.

[23] Корхов А.В. Метод построения вопросно-ответной системы с использованием математической формализации русского языка // Труды XXXII научной конференции факультета ПМ-ПУ СПбГУ. — 2001.

[24] Winograd T. Five Lectures on Artificial Intelligence / In Zampolli A. (ed.). Linguistic Structures Processing. — Amsterdam: North-Holland, 1977. — P. 399-520.

[25] Scott S., Gaizauskas R. QA-LaSIE: a Natural Language Question Answering System // In Proc. of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence. — 2001. — P. 172-182.

[26] Moldovan D., Harabagiu S., Pasca M., et al. Lasso: a Tool for Surfing the Answer Net // TREC-8. — 1999. — P. 175-183.

[27] Grinberg D., Lafferty J., Sleator D. A Robust Parsing Algorithm for Link Grammars // In Proc. of the 4th International Workshop on Parsing Technologies, Prague, Czech Republic. — 1995. — P. 111-125.

[28] Fellbaum C.D. (ed). WordNet: an Electronic Lexical Database. — Cambridge: The MIT Press, 1998. — 423 p.

[29] Edmonds Ph., Kilgarriff A. (eds). Journal of Natural Language Engineering (Special Issue Based on Senseval-2). — 2003. — Vol. 9(1).

[30] Мозговой М.В. Простая вопросно-ответная система на основе семантического анализатора русского языка // Вестник СПб университета. — 2006. — сер. 10. — вып. 1. — С. 116-122.

[31] Грамматика современного русского литературного языка / Под ред. Шведовой Н.Ю. — М.: Наука, 1970. — 768 с.

[32] Nürnberger A., Detyniecki M. (eds). Adaptive Multimedia Retrieval. — Hamburg: Springer, 2004. — 227 p.

[33] Page L., Brin S., Motwani R., and Winograd T. The PageRank Citation Ranking: Bringing Order to the Web / Technical Report 1999-66, Stanford Digital Library Technologies Project. — 1999.

[34] Broder A. et al. Graph Structure in the Web // Computer Networks. — 2000. — Vol. 33. — P. 309-320.

[35] Вебсайт проекта: http://www.isleuthhound.com

[36] Вебсайт проекта: http://www.wizetech.com/ru/document-search

[37] Salton G., Wong A., Yang C.S. A Vector Space Model for Information Retrieval // Journal of the American Society for Information Science. — 1975. — Vol. 18(11). — P. 613-620.

[38] Witten I.H., Frank E. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Ed. — San Francisco: Morgan Kaufmann, 2005. — 525 p.

[39] Мозговой М.В. Семантический анализатор и задача информационного поиска // Вестник СПб университета. — 2005. — сер. 10. — вып. 3. — С. 54-59.

[40] Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклад АН СССР. — 1965. — Т. 163. — вып. 4. — С. 845-848.

[41] Новый объяснительный словарь синонимов русского языка / Под ред. Апресяна Ю.Д. — М.: Языки славянской культуры, 2003. — 624 с.

[42] Коробейникова О.В., Порошин В.А. Использование компьютерных словарей русского языка — поиск синонимов посредством SQL-запросов // Процессы управления и устойчивость: Труды XXXIV научной конференции аспирантов и студентов / Под ред. Н.В. Смирнова, В.Н. Старкова. — СПб.: Изд-во СПбГУ. — 2003. — С. 379-384.

[43] Мозговой М.В. Контекстно-ориентированный тезаурус русского языка // Процессы управления и устойчивость: Труды 37-й международной научной конференции аспирантов и студентов / Под ред. А. В. Платонова, Н. В. Смирнова — СПб.: Изд-во СПбГУ. — 2006. — С. 379-383.

[44] Bliwise R. A Matter of Honor // Duke Magazine. — 2001. — May-June Issue. — P. 2-7.

[45] Brumfiel G. Physicist Found Guilty of Misconduct // Nature. — 2002. — September Issue. — P. 419–421.

[46] Armstrong Atlantic State University Honor Code: http://www.sa.armstrong.edu/Activities/hccoc.htm

[47] Gettysburg College Honor Code: http://www.gettysburg.edu/academics/acad/honor_code/constitution.html

[48] Alaoutinen S., Kontro-Vesivalo N., Medvedev D., Voracek J., and Uteshev A. Academic Honesty in Cross-Border Education — Opinions of Involved Students // In Proc. of the 34th Frontiers in Education Conference, Savannah, Georgia, USA. — 2004. — P. 20-25.

[49] Mozgovoy M. Desktop Tools for Offline Plagiarism Detection in Computer Programs // Informatics in Education. — 2006. — Vol. 5(1). — P. 97-112.

[50] Вебсайт службы: http://www.turnitin.com

[51] Joy M., Luck M. Plagiarism in Programming Assignments // IEEE Transactions on Education. — 1999. — Vol. 42(2). — P. 129-133.

[52] Baker B.S. On Finding Duplication and Near-Duplication in Large Software Systems // In Proc. of 2nd IEEE Working Conference on Reverse Engineering. — 1995. — P. 86-95.

[53] Fredriksson K., Mozgovoy M. Sublinear Parameterized Single and Multiple String Matching. Technical Report A-2006-2, Department of Computer Science, University of Joensuu, March, 2006.

[54] Belkhouche B., Nix A., Hassell J. Plagiarism Detection in Software Designs // In Proc. of the 42nd annual Southeast Regional Conference. — 2004. — P. 207-211.

[55] Mozgovoy M., Tusov V., Klyuev V. Fast Semantics-Powered Plagiarism Detection System // Submitted for 2006 IEEE International Conference on Computer and Information Technology, Seoul, Korea, 2006.

[56] Mozgovoy M., Fredriksson K., White D., Joy M., and Sutinen E. Fast Plagiarism Detection System // Lecture Notes in Computer Science. — 2005. — Vol. 3772. — P. 267-270.

[57] Manber U., Myers G. Suffix Arrays: a New Method for On-line String Searches // In Proc. of SODA'90. — 1990. — P. 319-327.

[58] Богдановский А.Е. Семантический анализ текстов на английском языке // Процессы управления и устойчивость: Труды 37-й международной научной конференции аспирантов и студентов / Под ред. А.В. Платонова, Н.В. Смирнова — СПб.: Изд-во СПбГУ. — 2006. — С. 286-293.

[59] Кутарба А.Ю. Обработка англоязычных текстов на основе семантического словаря // Вестник СПб университета. — 2005. — сер. 10. — вып. 3. — С. 46-53.

[60] Зализняк А.А. Грамматический словарь русского языка. — М.: Русские словари, 2003. — 800 с.

[61] Whorf B. Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf. — Cambridge: The MIT Press, 1964. — 290 p.

[62] Сепир Э. Статус лингвистики как науки / Сепир Э. Избранные труды по языкознанию и культурологии. — М.: Прогресс, 1993. — С. 259-265.

[63] Вебсайт проекта: http://www.win32forth.org