Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

ТЕХНОЛОГИИ ОБРАБОТКИ ТЕКСТА И ЗВУЧАЩЕЙ РЕЧИ

сборник учебно-методических материалов
для направления подготовки 45.03.03 – Фундаментальная
и прикладная лингвистика

Благовещенск, 2017

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Составители: Андросова С. В., Андросов Е. Ю.

Технологии обработки текста и звучащей речи: сборник учебно-методических материалов для направления подготовки 45.03.03. – Благовещенск: Амурский гос. ун-т, 2017.

ВВЕДЕНИЕ

В начале было слово...

Когда-то язык был слабым отражением вещей в сознании человека. Прозрачной зыбкой гранью, отделяющей человека от мира неживой материи. Сейчас язык развился настолько, что сам проявляет тенденцию к независимому движению и управляет развитием разума. Язык — самое древнее и самое загадочное приобретение человека, а лингвистика — наука о языке — древнейшая из наук. Развиваясь, язык раскрывает методы собственного познания.

Язык и мышление человека неразрывно связаны, поэтому разгадывая алгоритмы языка, человек тем самым пытается расшифровать алгоритмы разума. (по А. В. Анисимову). Эти многочисленные и настойчивые попытки привели в оформлению нового направления в лингвистики — прикладной лингвистики, неотъемлемой частью которой являются информационные технологии.

В наши дни владение информационными технологиями становится в один ряд с такими качествами, как умение читать и писать. Сегодня лингвист должен ориентироваться в мировом информационном пространстве и обладать необходимыми знаниями, умениями и навыками поиска, обработки и хранения информации с использованием информационных технологий, компьютерным систем и сетей.

В задачи настоящего курса входит углубленное изучение проблем алгоритмизации, моделирования лингвистических задач, современных языков программирования. Я попытаюсь дать вам представление о том, как ставится и решается лингвистическая задача с помощью компьютера: от ее словесной формулировке к алгоритму и компьютерной программе.

Для того, чтобы решать эти задачи знания высшей математики и построения сложных математических моделей не нужны, потому что компьютер — это языковая машина и основа его могущества заключается в способности манипулировать языковыми знаками — символами, которым приписывается некоторый смысл. Фактически, естественный язык в информатике занимает центральное место.

ПРИКЛАДНАЯ ЛИНГВИСТИКА

Любая достаточно развитая область знаний имеет три основных аспекта:
-теорию;
-эксперимент;
-практику.

Исходя из этого можно выделить три направления в лингвистике как науке о языке: теоретическую, экспериментальную и прикладную лингвистику.

Прикладная лингвистика — это понятие, до сих пор не имеющее четкого определения и конкретного перечня решаемых ею задач.

Термин прикладная лингвистика появился в конце 20 гг. 20 в., когда была осознана необходимость строгого научного решения прикладных задач с использованием методов формального лингвистического анализа письменных и акустико-лингвистического анализа устных сообщений. Одно из первых определений принадлежит В.В. Звегинцеву.
Определение В.В. Звегинцева: ПЛ — это новая область в лингвистике, которая осуществляет реализацию лингвистических знаний с целью решения всякого рода практических задач.

Определение из Энциклопедического лингвистического словаря: ПЛ — это направление в языкознании, занимающееся разработкой методов решения практических задач, связанных с использованием языка.
Определение А. Е. Кибрика: ПЛ — это раздел языкознания, в котором разрабатываются методы решения практических задач, связанных с оптимизацией использования языка как важнейшего средства человеческой коммуникации.
Итоговое определение: ПЛ — это комплексная научная дисциплина, изучающая язык в различных ситуациях его применения и разрабатывающая методы совершенствования языковых систем и языковых процессов, учение о методах решения разнообразных практических задач с использованием знаний о языке, учение о совершенствовании языковой способности человека и общества в целом.

За рубежом под ПЛ часто понимают совершенствование методов преподавания языка (дидактическая лингвистика). В нашей стране ПЛ понимают как компьютерную лингвистику, которая становится сейчас все более широкой дисциплиной почти синонимом ПЛ.

Синонимы ПЛ:
компьютерная Л, структурная Л, машинная Л, статистическая Л, математическая Л, искусственный интеллект (ИИ) и т.д.

ПЛ требует строгого структурного подхода к языку и отводит важную роль математике. ПЛ рассматривает *язык как сложную алгоритмическую систему* со своими алгоритмами функционирования и законы развития.

Информационные технологии (ИТ)

Определение

ИТ — это совокупность законов, методов и средств получения, хранения, передачи, распространения, преобразования информации с помощью компьютеров.

Предпосылки появления и развития ИТ (Зубов А.В., Зубова И.И., с. 8)

1. Создание ПК с большой памятью и большой скоростью выполнения операций.
2. Разработка звуковых плат, дающих возможность воспроизводить и записывать речь, звуки и музыку в большом диапазоне частот.
3. Изобретение видеоплат, позволяющих выводить на экран компьютеров изображение с телеэкранов и видеомагнитофонов.
4. Разработка мультимедийных компьютеров, позволяющих воспроизводить на экране дисплеев цвет, звук, музыку, движение.
5. Создание процессоров и устройств, способных передавать информацию в сети от одного компьютера к другому.
6. Разработка модемов, позволяющих передавать информацию на далекие расстояния.
7. Создание оргтехники, позволяющей осуществлять высокоскоростную печать документов и их копирование.

ИТ в лингвистике — это совокупность законов, методов и средств получения, хранения, передачи, распространения, преобразования информации о языке и законах его функционирования с помощью компьютеров.

ИТ в основном соотносятся с задачами ПЛ.

Существуют разные перечни задач. Приведем два.

Перечень 1

1. Создание автоматических систем анализа и синтеза речи
2. Автоматические методы переработки текстовой информации
3. Создание автоматизированных систем информационного поиска
4. Составление автоматических словарей и систем машинного перевода
5. Разработка методов автоматического аннотирования, реферирования и перевода
6. Разработка экспертных систем
7. Лингвистическое обеспечение АСУ(автоматизированные системы управления)
8. Стандартизация научно-технической терминологии

Перечень 2 (Зубов, Зубова, 2004: 8-9)

Комплексные задачи в соотнесении с задачами ПЛ

Создание систем

1) искусственного интеллекта;
2) автоматического перевода;
3)автоматического аннотирования и реферирования;
4) порождения текстов;
5) обучения языку;
6) понимания устной речи;
7) генерации устной речи;
8) автоматизированных информационно-поисковых систем;
9) атрибуции и дешифровки анонимных и псевдоанонимных текстов;

Разработка

10) баз данных (электронные словари, каталоги, реестры);
11) программных оболочек электронных словарей;
12) систем передачи информации в сети Интернет.

Частные задачи в соотнесении с задачами ПЛ

Автоматизация процессов

1) построения словарей текстов;
2) морфологического анализа слова;
3) определения значения многозначного слова;
4) синтаксического анализа предложения;
5) поиска слова в словаре;
6) порождения предложения и т.д.

Структура ПЛ

1. Структурная лингвистика – это совокупность взглядов на язык и методов его исследования, в основе которых лежит понимание языка как знаковой системы с четко выделенными структурными элементами (единицами языка, их классами и пр.) и стремление к строгому (как в точных науках) формальному описанию языку. Свое название СЛ получила благодаря особому вниманию к структуре языка, которая представляет собой сеть отношений

(противопоставлений) между элементами языковой системы, упорядоченных и находящихся в иерархической зависимости в пределах определенных уровней.
Структурное описание языка предполагает такой анализ реального текста, который позволяет выделить обобщенные инвариантные единицы (схемы предложений, морфемы, фонемы) и соотнести их с конкретными речевыми сегментами на основе строгих правил реализации Эти правила определяют границы допустимого варьирования языковых единиц в речи. В зависимости от уровня анализа правила реализации формулируются как правила позиционного распределения конкретных, например, принцип дополнительной дистрибуции в фонологии и морфологии (дистрибутивный анализ), или как трансформационные правила в синтаксисе (при трансформационном анализе) регулирующие переход от инвариантной глубинной структуры предложения к множеству ее реализации.
На базе СЛ развилась порождающая грамматика (генеративная лингвистика); идеи структурного анализа во многом определили постановку и решение задач, связанных с машинным переводом; СЛ открыла дорогу для широкого проникновения в лингвистику математических методов (математическая лингвистика).
На СЛ оказали влияние: Сепир, Блумфилд. Ф.де Соссюр, Н.С. Трубецкой, Р. Якобсон, Реформатский.
Успешно применяли методы СЛ: Апресян, Арутюнова, Гак, Зализняк, Звегинцев, Мельчук, Успенский и др.
2. Контрастивная лингвистика (сопоставительная лингвистика) – сопоставительное изучение двух, реже нескольких языков для выявления их сходств и различий на всех уровнях языковой структуры с целью типологической классификации языков. Как правило, контрастивная лингвистика оперирует материалами на синхронном срезе языка.
КЛ появилась и интенсивно развивалась в 50 гг. 20 в., однако ее появление подготовили работы Е.Д. Поливанова, Бодуена де Куртенэ, Л.В. Щербы с изложением теоретических основ сравнения родного и ин. Языков.
В 70 гг. контрастивные исследования в отдельных странах (гл. образом в США) использовали порождающую модель Хомского, с возведением явлений двух сопоставляемых языков к общей глубинной структуре; в наст. время наблюдается отход от этой методики в пользу сруктурно-функционального подхода.
3. Математическая лингвистика (МЛ) — это математическая дисциплина, предметом которой является разработка формального аппарата для описания строения естственных и некоторых искусственных языков.
Возникла в 50 годы 20 в. Одним из главных стимулов появления математической лингвистики послужила назревшая потребность в уточнения основных лингвистических понятий.
Методы МЛ имеют много общего с с методами мат. логики — мат. дисциплины, занимающейся изучением строения мат. рассуждений, - и в особенности таких ее разделов, как теория алгоритмов и теория автоматов.
4. Компьютерная лингвистика (КЛ) задает общую ориентацию на использование компьютеров для решения разнообразных научных и практических задач, никак не ограничивая способы решения этих задач.
5. Вычислительная лингвистика (ВЛ) может пониматься более узко, чем КЛ, так как даже при широкой трактовке понятия «вычисление» за его пределами остаются такие стороны решения лингвистических задач, как, например, представление знаний, организация банков языковых данных, психолингвистические аспекты взаимодействия человека и компьютера и другие.

Модели

Метод моделирования - центральный исследовательский метод в науке. Моделирование в науке - это выяснение свойств какого-либо предмета при помощи построения его модели.

Моделью можно назвать образ какого-либо объекта, используемый в определенных условиях в качестве его заместителя (фотография в паспорте - модель человека).

Свойства моделей:

· условность

· образ может быть не только материальным, но и мысленным и передаваться посредством знаковой системы

· моделью может быть не только образ, но и прообраз оригинала

·модель чаще всего является гомоморфной оригиналу (то есть многим элементам оригинала соответствует меньшее количество элементов модели в отличие от изоморфизма).

Типы моделей: модели объяснительного типа или объясняющие модели и модели воспроизводящего типа. От первых не ожидается порождения языкового продукта; эта модель должна непротиворечиво объяснять действие напр, языка в целом, если он моделируется, или каких-либо его частей моделируемого явления, если моделируются эти части (Марчук, 200...: 21). Вторые обретают ценность, когда воспроизводимый ими результат подобен тому, который был бы получен в результате деятельности человека (там же).

Понятие лингвистической модели возникло в структурной лингвистике, но вошло в научный обиход в 60-70 гг. 20 в. с возникновением мат. лингвистики и проникновением в лингвистику мат. Методов. Содержание термина "модель" в современной лингвистике в значительной степени охватывалось ранее термином "теория" (особенно Ельмслевым). Считается, что наименования модель заслуживает лишь такая теория. которая достаточно эксплицитно изложена и в достаточной степени формализована (в идеале каждая модель должна допускать реализацию на ЭВМ).

Главная цель моделирования в лингвистике - это моделирование целостной языковой способности человека.

Устройство и функционирование языка недоступны непосредственному наблюдению, поэтому изучение языка и продуктов его деятельности — текстов разного рода — осуществляется главным образом с помощью моделирования (Марчук, 200...: 20-21. Марчук Ю.Н. Компьютерная лингвистика.).

Собственно лингвистические модели:

- модели речевой деятельности, процессуальный модели (самые сложные)
- модели языковой системы, языковой структуры (тоже очень сложные)
- модель памяти и др.

Лингвистическое моделирование необходимо предполагает использование абстракции и идеализации. Отображая релевантные существенные (с точки зрения исследования) свойства оригинала и отвлекаясь от несущественных, модель выступает как некоторый абстрактный идеализированный объект.

Всякая модель строится на основе гипотезы о возможном устройстве оригинала и представляет собой функциональный аналог оригинала. что позволяет переносить знания с модели на оригинал.

Критерием адекватности модели является эксперимент.

В идеале модель должна быть формальной (т.е. в ней должны быть в явном виде и однозначно заданы исходные объекты, связывающие их отношения и правила обращения с ними) и обладать объяснительной силой (т.е. не только объяснять факты или данные

экспериментов, необъяснимые с точки зрения уже существующей теории, но и предсказывать неизвестное раньше, хотя и принципиально возможное поведение оригинала, которое позднее должно подтверждаться данными наблюдения или экспериментов).

Конструирование модели - не только одно из средств отображения языковых явлений, но и объективный практический критерий проверки истинности знаний о языке. В единстве с другими методами изучения языка моделирование выступает как средство углубления познания скрытых механизмов речевой деятельности, его движения от относительно примитивных к более содержательным моделям, полнее раскрывающим сущность языка.

Внутри языка как системы существует принцип моделирования: одни его подсистемы моделируют другие, например, система письменной речи является моделью устной речи; внутри письменной речи мы имеем дело с несколькими моделями (печатной, рукописной); план выражения является моделью плана содержания.

Метод моделирования обычно опирается на знаковые систем, но язык - сам знаковая система, т.е. слова мы моделируем при помощи слов.

ТЕМА 1. ОСНОВНЫЕ СОСТАВЛЯЮЩИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

1. Структура информационных технологий
2. Теоретические основы информационных технологий.
3. Будущее информационных технологий.
4. Алгоритмы и их свойства.
5. Элементы языка программирования Turbo Pascal.

Структура информационных технологий

В составе современных информационных технологий можно выделить следующие составляющие:

1) теоретические основы информационных технологий;
2)методы решения задач информационными технологиями;
3)средства решения задач, используемые в информационных технологиях:
   а) аппаратные средства;
   б) программные средства.

Рассмотрим подробнее эти составляющие.

2. Теоретические основы информационных технологий.

Теоретическую основу информационных технологий составляют важнейшие понятия и законы информатики. В свою очередь понятие «информатика» тесно связано с понятием «информация».

Слово информация (от лат. informatio — разъяснение, изложение) в обычном житейском понимании обозначает некоторые сведения о внешнем и внутреннем мире, которые мы используем Для регулирования своего поведения. Более строго это понятие раскрывается разными способами. Выберем один из них, который более всего подходит к рассматриваемым в пособии лингвистическим задачам, и определим информацию как определенным образом связанные сведения, данные, понятия, отраженные в нашем сознании и изменяющие наши представления о реальном мире.

Информация обладает разными свойствами. Наиболее важными из них являются: ценность, достоверность, полнота, актуальность, логичность, компактность.

Ценность информации определяется тем, насколько она важна для позволяющих получателю информации достичь своей цели.

Полнота — насколько много имеется сведений, позволяющих получателю достичь своей цели.

Актуальность информации определяется необходимостью ее немедленного использования для достижения какой-либо цели.

Компактность информации — способность представить ее в наиболее сжатом виде. Достоверность и логичность (не требуют особых пояснений).

Выделяют различные виды информации. При этом для ее классификации по видам разработано много подходов, использующих разнообразные признаки и особенности информации. Так, в зависимости от того, какими органами чувств воспринимается информация, ее делят на визуальную, аудиальную (звуковую, фонетическую), аудиовизуальную, тактильную. По направленности информации всем членам общества или каким-то его группам различают информацию массовую, предназначенную для всех членов общества, и специальную — для специалистов в различных областях науки, техники,

культуры, производства. Специальную информацию подразделяют на научную, техническую, производственную, эстетическую и т.п.
В каждом виде специальной информации выделяют подвиды. Например, в зависимости от области науки и научной информации выделяют информацию физическую, математическую, биологическую, лингвистическую и т.д. Так, лингвистической информацией называют множество определенным образом связанных сведений, данных, понятий о языке и правилах его функционирования, отраженных в нашем сознании и влияющих на наше речевое поведение.

Слово «информатика» также не имеет единого определения. С современной точки зрения информатика — это наука о законах и методах получения, хранения, передачи, распространения, преобразования и использования информации в естественных и искусственных системах с применением компьютера.
В зависимости от вида информации выделяют различные типы информатики. Так, различают информатику социальную, экономическую, научную научно-техническую, статистическую, биологическую, медицинскую и т. п.
Наука, изучающая законы и методы организации и переработки с помощью компьютера лингвистической информации, называется лингвистической информатикой. Вспомнив, что понимается под лингвистической информацией, можно сказать, что объектом исследования лингвистической информатики будет структура слов, словосочетаний, предложений, текстов. Ее интересуют правила, объединяющие нижестоящие языковые единицы в вышестоящие, правила перевода предложений и текстов, способы построения рефератов и аннотаций, пути обучения языкам и целый ряд других вопросов, связанных с языком и речью.

Аппаратное и программное обеспечение ит

- Средства аппаратного обеспечения: компьютер и периферийные устройства
- Программное обеспечение (ПО):

1. Системное ПО (операционные системы — ОС, дополнительные системные программы — утилиты и драйверы);
2. Прикладное ПО;
3. Прикладные инструментальные средства.

ОС — главная программа, загружаемая в оперативную память компьютера после его включения.

Функции ОС

1. Управление работой ПК (внутренние функции ПК, контроль за выполнением операций, распределение памяти и т.д.)
2. Запуск и выполнение прикладных программ.
3. Обеспечение пользователю удобного способа общения с ПК.

Утилита — программа, расширяющая возможности ОС.

Виды утилитов

1. Программы-архиваторы;
2. Программы для создания резервных копий;
3. Антивирусные программы;
4. Программы для диагностики копьютера.

Драйверы — программы для управления устройствами компьютера (чаще ввода-вывода): драйверы клавиатуры, мыши, принтера, сканера и т.д.

Прикладные программы

1. Текстовые процессоры (MS Words, Open Office writer, Notepad etc)
2. Программы автоматического преобразования графической информации в текстовую (скан в текстовый файл) (FineReader etc);
3. Системы машинного перевода;
4. Системы автоматического аннотирования и реферирования;
5. Настольно-издательские системы (PageMaker, Scribus);
6. Обучающие программы;
7. Экспертные системы, напр. Для диагностики неисправности приборов, при определении болезни и метода ее лечения и т.п.;
8. Табличные процессоры (MS Excel, OpenOffice Calc) для обработки экономических и статистических данных;
9. СУБД (Access для Windows, MySQL для Linux) позволяют осуществлять создание и модификацию больших совокупностей структурированных определенным образом данных (баз данных), а так же поиск в них информации.

Прикладные инструментальные средства. Языки программирования

Изначально никаких языков программирования не существовало. Программирование осуществлялось в машинных кодах процессора (любой набор цифр), под который писалась программа. Это было неудобно, поскольку слабо читаемо. И писать такие программы, и находить в них ошибки тоже было сложно. Для упрощения этого процесса был придуман язык низкого уровня Ассемблер. В нем машинные команды записывались командами естественного языка. Напр., команда сложения обозначалась тремя буквами «ADD», команда вычитания «SUB» (subtract), команда сравнения «CMP» (compare), команда перехода к другому адресу «JMP» (jump) и т.п.

Проблема состояла в том, что каждый процессор имел свой собственный набор машинных команд и большинство семейств процессоров имели сильно отличающийся друг от друга язык Ассемблера. Поэтому стали разрабатывать так называемые языки программирования высокого уровня, которые в больше походили на естественный язык общения, программы на этом языке могли писаться быстрее, ошибки на этих языках было легче обнаруживать и исправлять. Но перед тем как запускать такие программы, они должны были быть переведены в язык, понятный для компьютера, т.е. язык двоичных машинных кодов. Процесс такого перевода называется трансляцией. Трансляторы бывают двух типов:

1) программы-интерпретаторы;
2) программы-компиляторы.

Интерпретаторы берут программу на языке высокого уровня и исполняют ее строчка за строчкой, останавливаясь в случае ошибки.

Достоинства и недостатки: интерпретаторы позволяют быстро разрабатывать небольшие программы, но вместе с тем требуют больших накладных расходов, поскольку помимо собственно программы пользователя в памяти компьютера должна находиться еще и программа-интерпретатор. Написание сложных программ поэтому затруднено.

Компиляторы сначала переводят всю программу в машинные коды, которые исполняются уже без участия компилятора. Перед исполнением программы необходимо пройти стадию компиляции, т.е. перевода программы на языке высокого уровня в машинные коды.

Достоинства и недостатки: исполняется быстро, поскольку не требует дополнительных накладных расходов, но исправление ошибок в этих программах требует

исправления первоначальной версии высокого уровня, а затем проведения повторной компиляции.

Одним из самых распространенных *интерпретаторов* до недавнего времени являлся язык BASIC. В настоящее время в связи с распространением интернета широкое распространение получили языки JAVA, PERL (Practical Extraction and Report Language — практический язык для извлечения данных и составления отчётов), PYTHON, PHP (Hypertext Preprocessor) — для оформления и динамического создания страниц HTML (hyper text markup language).

В 60-70 гг. самыми распространенными *компиляторами* были COBOL (Common Business Oriented Language), FORTRAN (Formula Translator), PL/1 (Program Language 1). В настоящее время для программирования в основном используется универсальный язык C/C++ (C с добавлением объектно-ориентированного программирования), а для обучения программированию — язык PASCAL.

Компилятор обрабатывает программу, написанную на языке Free Pascal в два этапа.

1. Компилятор анализирует, какие внешние библиотеки3 нужно подключить, разбирает текст программы на составляющие элементы, проверяет синтаксические ошибки и в случае их отсутствия формирует объектный код (в Windows — файл с расширением .obj, в Linux — файл с расширением .o). Получаемый на этом этапе двоичный файл (объектный код) не включает в себя объектные коды подключаемых библиотек.

2. Компоновщик подключает к объектному коду программы объектные коды библиотек и генерирует исполняемый код программы. Этот этап называется компоновкой или сборкой программы. Полученный на данном этапе исполняемый код программы можно запускать на выполнение.

Состав алгоритмического языка

В письменном тексте на естественном языке выделяют четыре основных элемента: символы, слова, словосочетания и предложения. Аналогично и в алгоритмическом языке с тем отличием, что слова там принято называть лексемами, словосочетания — выражениями, а предложения — операторами.

1. Символы языка = алфавит — основные неделимые знаки (прописные и строчные латинские буквы, знак подчеркивания; знаки препинания; различные виды скобок; слэши, проценты, больше, меньше, плюс, тильда и т.п.).

2. Лексемы. Состоят из символов алфавита. К лексемам относят идентификаторы, ключевые / зарезервированные слова, знаки операций, константы, разделители: скобки, точка, запятая, пробельные символы.

2.1. Идентификаторы.

Идентификатор — это имя программного объекта. Общий для разных языков момент — в имени могут быть использованы только буквы, цифры и знак нижнего подчеркивания. Детали могут различаться. Например, на языке C / C++ чередование или выбор строчных и прописных букв образует разные имена, а на языке Free Pascal — нет.

2.2. Зарезервированные / ключевые слова

Это идентификаторы, которые имеют специальное значение для компилятора. Это значение строго определено, использование в другом значении невозможно. Ключевые слова варьируют от языка к языку, хотя ряд слов совпадают. Например, do, while, const, for, if и ряд других совпадают у C++ и Free Pascal.

ТЕМА 2. ПРИНЦИПЫ РЕШЕНИЯ ЛИНГВИСТИЧЕСКИХ ЗАДАЧ МЕТОДОМ МОДЕЛИРОВАНИЯ

Экспериментальные методы и модели

При использовании информационных технологий в лингвистике на первый план выходит метод компьютерного моделирования.

Выделяют следующие типы моделей.

1. Структурные модели, служащие для изучения и описания внутреннего строения некоторого объекта. Например, такая модель строится, если необходимо изучить систему согласных какого-либо языка или устройство речевого аппарата человека.
2. Функциональные модели, позволяющие изучать поведение некоторого объекта, течение некоторого процесса или же этапы реализации некоторого явления. Например, функциональная модель строится, если необходимо смоделировать процесс создания некоторого текста человеком. Такая же модель создается для объяснения процесса перевода текста с одного языка на другой.
3. Динамические модели, которые создаются при необходимости найти объяснение некоторых процессов или явлений в их временном развитии. Так, если требуется узнать, как со временем менялось произношение некоторого слова, строят динамическую модель такого процесса.

Модели различаются по направленности на определенный объект и по используемым средствам моделирования — алгоритму или исчислению.

Алгоритм - строгая последовательность предписывающих правил .

Исчисление - множество разрешающих правил (порядок выполнения не важен) .

Понятие лингвистической модели возникло в структурной лингвистике, но вошло в научный обиход в 60-70 гг. 20 в. с возникновением математической лингвистики и проникновением в лингвистику математических методов.

Особая роль в лингвистике отводится функциональным моделям, позволяющим раскрыть суть функционирования языка, механизма производства и восприятия речи и текста. Нельзя заглянуть в мозг человека и посмотреть, как в нем осуществляются операции с буквами, звуками, словами, предложениями при всевозможном использовании языка. Поэтому для решения таких задач в рамках функциональных моделей выделяют воспроизводящие инженерно-лингвистические модели (ВИЛМ). Они представляют собой компьютерные системы, поведение которых, с одной стороны, имитирует поведение реальных лингвистических объектов, а с другой — позволяет хотя бы частично воспроизвести эти реальные объекты.

Типы лингвистических моделей:

1. По охвату структуры языка:

а) общие (глобальные) стремятся охватить весь язык: <VG> (vocabulary, grammar) ;

б) частные: фонетическая модель русского языка, модель системы гласных .

2. По типологическому статусу:

а) универсальные стремятся охватить все языки мира: <VG> ;

б) специфические характерны для определенного языка или группы языков: мягкость - твердость согласных рус. языка (не действует в англ., нем., франц.) .

3. По гносеологическому статусу:

а) модели языка ;

б) модели лингвистических знаний различные фонетические школы ;

в) модели деятельности лингвиста .
4. По отраженному аспекту языка и речевой деятельности:
а) анализирующие модели моделируют процесс понимания, используют логическое средство — алгоритм ;
б) синтезирующие модели моделируют процесс вербализации, смысла речевого отрезка
в) порождающие модели: порождающая грамматика Н. Хомского объект моделирования — множество правильных речевых отрезков составляются правила различения приемлемого и неприемлемого; логический средство - исчисление; не служат выражением смысла; на выходе - цепочки элементов (грамматически правильных предложений);
г) собственно структурные модели — основа всех остальных; объект моделирования - структура языка как таковая; логический аппарат — логика отношений и классов. Пример: грамматический словарь Зализняка.
5. По конечной цели исследования:
а) теоретические;
б) описательные;
в) прикладные.
6. По используемым методам
а) математические модели;
б) психологические модели;
в) социологические модели.
7. По функциональному статусу:
а) абстрактно обобщающие модели;
б) действующие.
8. По используемым материальным средствам:
а) графические;
б) символьные;
в) компьютерные.

Частная модель обычно входит в набор частных моделей, описывающий определенный уровень языка: фонологический, морфологический, синтаксический, лексико-семантический.

Лингвистическое моделирование необходимо предполагает использование абстракции и идеализации: отображает релевантные свойства оригинала и отвлекается от несущественных свойств. Всякая модель строится на основе гипотезы о возможном устройстве оригинала и представляет собой функциональный аналог оригинала. что позволяет переносить знания с модели на оригинал.
Критерием адекватности модели является эксперимент.
В идеале модель должна быть формальной (т.е. в ней должны быть в явном виде и однозначно заданы исходные объекты, связывающие их отношения и правила обращения с ними) и обладать объяснительной силой (т.е. не только объяснять факты или данные экспериментов, необъяснимые с точки зрения уже существующей теории, но и предсказывать неизвестное раньше, хотя и принципиально возможное поведение оригинала, которое позднее должно подтверждаться данными наблюдения или экспериментов).

Основные теоретические требования к модели:
1. Полнота модели - способность отражать все факты, на которые она рассчитана, на охват которых она претендует.
2. Простота - удобство, использования как можно меньшего числа средств (символов, правил) для достижения поставленной научной цели.

3. Объяснительная сила - способность модели вскрывать причины наблюдаемых фактов и предсказывать новые факты (например. модели исторического изменения слова; !! системы машинного перевода в очень малой степени объяснительные).
4. Адекватность - свойство максимальной похожести на моделируемый объект, на оригинал, можно свести к объяснительной силе или теоретико- множественному соответствию.
5. Экономность - экономичное использование энергетических и временных ресурсов при применении модели.
6. Точность - возможность выполнения операций представляемым моделью формальным аппаратом.
7. Эстетические свойства - красота модели.

Главный прикладной критерий - удобство модели.

Для моделирования языка очень важны логические средства реализации модели (компьютерное воплощение модели).

Конструирование модели - не только одно из средств отображения языковых явлений, но и объективный практический критерий проверки истинности знаний о языке. В единстве с другими методами изучения языка моделирование выступает как средство углубления познания скрытых механизмов речевой деятельности, его движения от относительно примитивных к более содержательным моделям, полнее раскрывающим сущность языка. Внутри языка как системы существует принцип моделирования: одни его подсистемы моделируют другие, например, система письменной речи является моделью устной речи; внутри письменной речи мы имеем дело с несколькими моделями (печатной, рукописной); план выражения является моделью плана содержания.

Метод моделирования обычно опирается на знаковые систем, но язык - сам знаковая система, т.е. слова мы моделируем при помощи слов.

Компьютерное моделирование на Free Pascal

Задача 1

На языке Паскаль представьте алгоритм выбора варианта прочтения диграфа «ea».

Условия задачи

Вариант 1: [ei]: только для great, break, steak

Вариант 2: [e]: heavy, heaven, breast, meadow, weapon, wealthy, ready, read (Past Simple), dead, death, breath, measure, pleasure, health, healthy.

Вариант 3: [iei]: только create

Вариант 4: [i]: все остальные.

Простейший вариант обработки строк на языке Pascal представлен десятью стороками (стр.)

```
1. program ReadDigraf;
2. var
  3. sInp : String;
4. begin
 5. ReadLn(sInp);
  6. If sInp = ´read´
    7. then begin
      8. WriteLn ('r -[i]- d, Infinitive and Present Simple, if other  - [e]') ;
    9. end;
10. end.
```

1стр. – ключевое слово **program**, за которым через пустое место (whitespace – пробел или энтер или таб) следует название (имя) программы.

!! Имя должно начинаться с буквы, состоять только из букв, цифр и знака нижнего подчеркивания; минус, плюс, точка и т. д. нельзя.

!! В алгоритмическом языке различия между большими и маленькими буквами в зарезервированных словах и индентификаторах отсутствует), после названия программы точка с запятой, после которой можно поставить или enter, или пробел, или вообще ничего (точка с запятой сама по себе является разделителем и не является ни к ключевым словом, ни идентификатором). Но для эстетических задач лучше поставить enter, поскольку эстетичность – одно из требований к разрабатываемым моделям, нарушение которого, однако, не мешает корректной сборке программы и правильному ее выполнению.

!! К случаям обязательного использования пустого места относятся:

- использование после ключевых (зарезервированных слов);
- использование после идентификаторов.

2 стр. – секция определений, в данном случае – блок определения переменных: ключевое слово **var**, за которым через пустое место (это ключевое слово) следует описание переменных (**3 стр.**): имя, двоеточие, тип, далее точка с запятой. Если вводится несколько переменных одного типа, то можно записать их в одной строке, разделив запятыми, затем двоеточие, тип и точка с запятой.

!! Имеются разные области видимости переменных: глобальные (видимые для всей программы) и локальные (видимые в пределах конкретного *программного блока – в Pascal программные блоки – это процедуры и функции*). Если **var** стоит после слова **program**, то это глобальная область, а если после слова **function** или **procedure**, то это локальная область.

Типы переменных

Основные: integer (целые числа), real (числа с плавающей запятой — вещественные, напр. $5.08\text{-}1.3\cdot10^{7}$), char (символ, напр. 'b' 'ї'), string (строка)

Более сложные: array (массив одномерный, двухмерный), record (запись: если в массиве можно задавать элементы только одного типа, то в записи – разных), object (используется в объектно-ориентированном программировании (ОП) наряду с данными разных типов содержит процедуры и функции), file (внешние файл на диске).

4 стр. – **begin** и пустое место – начало программы.

5 стр. – **Readln** – стандартная процедура чтения с устройства ввода (напр. с клавиатуры), а в скобках указываются параметры, куда заносится результат чтения (в данном случае строковая переменная **sInp**).

6 стр. – условный оператор. Начинается с зарезервированного слова **if**, за которым следует условие (выражение на языке Pascal, вычисляемое по правилам Булевой алгебры – то есть с результатом «Истина или ложь»). За условием следует (**7 стр.**) зарезервированное слово **then**, за которым идет оператор Pascal, за которым может следовать (а может и отсутствовать вторая часть – **else**). Если необходимо ввести более одного оператора, используется так называемые операторные скобки – пара **begin end**, между членами которой можно использовать любое количество операторов (несколько командных строк).

8 стр. – **Writeln** – стандартная процедура вывода (в файл или на экран). В скобках дается, что именно выводить.

9 стр. – конец блока (того, что начинался с **then**), точка с запятой.

10 стр. – конец программы: **end.** (с точкой без пробелов)

!! Зарезервированные слова – слова, которые используются в строго определенных случаях и не могут быть использованы в качестве имен переменных, функций, процедур.

Список зарезервированных слов:

absolute, and, array, asm, begin, case, const, constructor, destructor, div, do, downto, else, end, file, for, function, goto, if, implementation, in, inherited, inline, interface, label, mod, nil, not, object, of, on, operator, or, packed, procedure, program, record, reintroduce, repeat, self, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor

Простейший вариант обработки предложен только для одного слова read с двумя способами прочтения – через [i] и через [e]. Это лишь малая часть, предполагаемая условиями задачи. Чтобы выполнить все автоматическое транскрибирование необходимы дальнейшие усложнения.

Вариант обработки строк на языке Pascal: усложнение 1

```
program prog1;
var
  sInp : String;
begin
 ReadLn(sInp);
  If sInp = 'read'
    then begin
      WriteLn ('r -[i]- d, Infinitive and Present Simple, if other  - [e]') ;
    end;
  if sInp = 'create'
  then begin
  WriteLn ('cr -[iei] - te') ;
  end;
end.
```

По сравнению с предыдущим простейшим вариантом добавлено еще одно ветвление для слова *create* (см. условия задачи и выделенный голубым цветом фрагмент данной программы).

Вариант обработки строк на языке Pascal: усложнение 2

```
program prog2;
const
countEI = 3;
TarrayEI : array [1..countEI] of string = ('great', 'break', 'steak');
var
  sInp : String;
begin
 ReadLn(sInp);
  If sInp = 'read'
    then begin
      WriteLn ('r -[i]- d, Infinitive and Present Simple, if other  - [e]') ;
    end;
  if sInp = 'create'
  then begin
  WriteLn ('cr -[iei] - te') ;
  end;
  If sInp = TArrayEI[1]
```

```
  then begin
    WriteLn ('gr -[ei]- t');
  end;
  If sInp = TArrayEI[2]
  then begin
    WriteLn ('br -[ei]- k');
  end;
  If sInp = TArrayEI[3]
  then begin
    WriteLn ('st -[ei]- k');
  end;
end.
```

По сравнению с предыдущим вариантом добавлена константа из трех составляющих (константы удобно применять для исключений из правил чтения, поскольку их количество всегда ограничено), на каждую из которых добавлено ветвление (см. голубую заливку).

Такая запись неудобна по причине избыточности кода (слишком много почти одинаковых строк).

Чтобы сделать запись более экономной, напишем функцию по замене одних символов в строке на другие символы.

```
1. function Replace(sInput,sSearch,sReplace:String):String;
2. var i1:Integer;
   3. sTmp:String;
4. begin
  5. i1:=Pos(sSearch,sInput);
  6. sTmp:=sInput;
  7. Delete(sTmp,i1,Length(sSearch));
  8. Insert(sReplace,sTmp,i1);
   9. Replace:=sTmp; // функции Replace присвоить значение временной переменной sTmp
10. end; // end Replace function
```

1 стр.: У данной функции три переменных/аргумента на входе (указаны в круглых скобках) – исходная строка (заданное слово), строка для поиска (дираф в заданном слове) и строка замены (на что заменить найденный дираф – один из транскрипционных знаков [i], [e], [ei]). Аргументов может и не быть, если это функция без аргументов. В скобках через двоеточие указывается тип переменной/аргумента, в данном случае – строковая переменная – **string**. После скобки через двоеточие указан тип возвращаемого функцией результата, в данном случае это тоже строка. Поэтому у нас дважды употребляется слово **string**: для типа переменной и для типа результата, возвращаемого функцией.

2 стр. – **var i1:Integer** – это описание переменных, используемых внутри данной функции. Ключевое слово **var** показывает, что наступает начало блока, описывающего переменные. **i1** – произвольно выбранное имя, которое используется для названия переменной. В нашем случае это индекс начала диграфа. Через двоеточие указан тип переменной – **integer** – целое число, потому что номером символа в строке может быть только целое число. Другие примеры: z8 – номер ряда в кинотеатре, тип – integer – целое число; m22 – цена продукта, тип – real – вещественная переменная с плавающей запятой и т. д.

3 стр. – **sTmp:String** – временная переменная для хранения промежуточного значения строки. Имя произвольно. В нашем случае это то слово, в котором мы хотим прочесть дираф -ea-.

4 стр. и **10 стр.** – операторные скобки.

В программе используются четыре библиотечные функции: *Pos, Delete, Insert, Length* (**5–9 стр.**).

Функция Pos возвращает индекс подстроки строке, напр. Для строки 'read' и подстроки 'ea' функция возвращает индекс 2 – дираф начинается со второго символа в строке (в слове).

Функция Delete удаляет из строки определенное количество символов. Строка определяется первым аргументов, с какого символа начинать удалять — вторым аргументов, сколько символов удалять – третьим аргументом (аргументы в примере даны в скобках). Например, для случая Delete(s, 3, 2), где s = create, строке s присваивается значение *crte* (начиная с третьего символа два удаляются). Обратите внимание, что в процедуру *Delete* можно передавать только строковые переменные. Если пытаться ввести напрямую Delete('create', 3, 2), то компилятор выдаст сообщение об ошибке.

Функция Length возвращает значение длины строки (количество символов в строке). Напр. Length ('break')=5. В нашем случае – длина диграфа = 2 (два символа).

Функция Insert вставляет первый аргумент во второй аргумент, начиная с символа, который определяется третьим аргументом. Напр. s1=rd, s2=ea, тогда после выполнения процедуры Insert (s2, s1, 2), s1 = read, а s2 не меняется.

!! В итоговом варианте программы обратить внимание на отсутствие знака «;» после **end** в строке 51 презентации. Причина – препозиция к ключевому слову **else**.

Итоговый вариант программы

```
1. program ReadDigraf;
2. const
  3. CountEI = 3;
  4. TArrayEI : array [1..CountEI] of string = ('great','break','steak');
  5. CountE = 19;
  6. TArrayE : array [1..CountE] of string =('heavy', 'heaven', 'breast', 'meadow', 'weapon', 'wealth',
'wealthy', 'ready',  'dead', 'bread', 'death', 'breath', 'measure', 'pleasure', 'treasure', 'head', 'thread',
'threat','meant');
  7. TStringI = 'read';
  8. TStringIEI = 'create';
9. var
  10. sInp, sRpl, sTail : String;
  11. i : Integer;
12. function Replace(sInput,sSearch,sReplace:String):String;
13. var i1:Integer;
    14. sTmp:String;
15. begin
  16. i1:=Pos(sSearch,sInput);
  17. sTmp:=sInput;
  18. Delete(sTmp,i1,Length(sSearch));
  19. Insert(sReplace,sTmp,i1);
  20. Replace:=sTmp;
21. end;
22. begin
  23. sRpl := '';
  24. sTail := '';
```

```
  25. ReadLn(sInp);
  26. If sInp = TStringI
    27. then begin
      28. sRpl := ' -[i]- ';
      29. sTail := ', Infinitive and Present Simple, if other  - [e]';
    30. end;
31. If sInp = TStringIEI
    32. then begin
      33. sRpl := ' -[iei]- ';
    34. end;
  35. For i := 1 to CountEI do begin
    36. If sInp = TArrayEI[i] then begin
      37. sRpl := ' -[ei]- ';
    38. end;
  39. end;
  40. For i := 1 to CountE do begin
    41. If sInp = TArrayE[i] then begin
      42. sRpl := ' -[e]- ';
    43. end;
  44. end;
  45. If sRpl =" then sRpl := ' -[i]- ';
  46. i := Pos ('ea',sInp);
  47. If i>0 then begin
    48. Delete (sInp,i,2);
    49. Insert (sRpl, sInp, i);
    50. If Length(sTail)>0 then sInp := sInp + sTail;
  51. end
  52. else begin
    53. WriteLn ('WARNING!!! "ea" NOT FOUND!!!');
  54. end;
  55. WriteLn(sInp);
56. end.
```

Примечания к программе

Добавлено: Tail — для комментария про Infinitive and Present Simple глагола *read*.

Строки с 47 по 53: знак больше нуля вводится для того, чтобы отсечь случаи, где искомого диграфа нет.

Программирование алгоритма решения задачи 1 – довольно сложный случай. Самый простой случай – когда транскрипционные знаки совпадают со знаками клавиатуры и когда имеется только один вариант прочтения той или иной буквы. Программирование таких случаев на примере английской буквы «m» представлено ниже.

Автоматическое транскрибирование буквы «m»

```
1.program Read_m;
2.var
3.sInp, SRpl: string;
4.i : Integer;
5. begin
```

```
6. sRpl := '';
7. ReadLn(sInp);
8. for i:=1 to Length(sInp) do begin
9.  If (sInp[i] ='m')
10.    then begin
11.      sRpl := sRpl+' -[m]- ';
12.  End else begin
13.    sRpl := sRpl+sInp[i];
14.  end;
15.  end;
16.   WriteLn(sRpl);
17.end.
```

В данной программе не представлены константы, только область переменных, поскольку имеется только одно правило прочтения и исключения отсутствуют. Соответственно, программа маленькая – состоит только из 17 строк. Схожей простотой программирования характеризуются очень немного букв для английского языка (вспомним известную поговорку: «Пишем – Манчестер, читаем – Ливерпуль»), пожалуй, только «l», «v», «r» (последний – для американского и канадского вариантов). С остальными возникают более или менее серьезные сложности. Для каких-то букв их меньше, а для каких-то больше. Например, для «b», «n», в дополнение к приведенному примеру,

ЗАДАЧА 2

Составить алгоритм чтения синонимичных диграфов -ou-, ow.

Условия задачи

Основной тип [aʊ];

Исключение 1: [əʊ]: soul, poultry, shoulder, low, bow (лук), bowl, owe, own.

Исключение 2: [ʌ]: country, cousin, young, touch.

Исключение 3: [ʊ]: should, would, could, wounded;

Исключение 4: [u]: route, rouge.

В отличие от предыдущего алгоритма (см. задачу 1), в данном, согласно условиям задачи, предполагается использование знаков, не имеющихся на клавиатуре – транскрипционных знаков (кроме исключения 4). Это требует введения строк с поддержкой юникода. Для этого вместо типа данных string везде в программе необходимо использовать ansistring. Во 2 стр. через две косы черты (так можно обозначить любое примечание) дан соответствующий комментарий:

// uses AnsiString;

В области констант описаны четыре массива для работы с исключениями из правила. Перед описанием массива указывается количество элементов в нем (4, 6, 8 и 10 стр.). Затем следует описание массива: его имя (идентификатор), указание на тип данных (массив) и счетчик элементов [от первого элемента и до конца массива], тип каждого элемента (строка со знаками Юникода) и после знака равенства в круглых скобках и одинарных кавычках через запятую все эти элементы перечисляются (см. 5, 7, 9, 11 стр.). Такое описание легко можно модифицировать – убрать или добавить элементы, – не забыв при этом поменять указание на количество элементов.

Помимо четырех констант типа массив, имеются две строковые константы для задания одиночных уникальных исключений (12–13 стр.).

Программа

```
1. program ReadDigraphsouow;
2. // uses AnsiString;
3. const
4. Countou = 7;
5. TArrayou : array [1..Countou] of AnsiString = ('soul', 'poultry', 'shoulder', 'bowl', 'low',
   'own', 'owe');
6. Counta = 4;
7. TArraya : array [1..Counta] of AnsiString = ('country', 'cousin', 'young', 'touch');
8. Countu1 = 4 ;
9. TArrayu1 : array [1..Countu1] of AnsiString = ('should', 'would', 'could', 'wounded');
10. Countu = 2;
11. TArrayu : array [1..Countu] of AnsiString = ('route', 'rouge');
12. TAnsiStringou = 'bow';
13. TAnsiStringu1 = 'wound';
14. var
15. sInp, SRpl, sTail : AnsiString;
16. i : Integer;
17. function Replace (sInput, sSearch, sReplace : AnsiString) : AnsiString ;
18. var i1 : Integer;
19. sTmp : AnsiString;
20. begin
21.  i1:= Pos(sSearch, sInput);
22.  sTmp:= sInput;
23.  delete(sTmp, i1, Length(sSearch));
24.  Insert(sReplace, sTmp, i1);
25.  Replace:= sTmp;
26.  end;
27.  begin
28.  sRpl := '';
29.  sTail := '';
30.  ReadLn(sInp);
31.  if sInp = TAnsiStringou
32.  then begin
33.  sRpl := ' -[əʊ]- ';
34.  sTail := ', в значении "Лук", if other - [aʊ]';
35.  end;
36. if sInp = TAnsiStringu1
37.  then begin
38.  sRpl := ' -[ʊ]- ';
39.  end;
40.  For i := 1 to Countou do begin
41.  if sInp = TArrayou[i] then begin
42.  sRpl := ' -[əʊ]- ';
43.  end;
44.  end;
45.  For i:= 1 to Counta do begin
46.  if sInp = TArraya[i] then begin
```

```
47.  sRpl := ' -[ɵ]- ';
48.  end;
49.  end;
50.  For i := 1 to Countu1 do begin
51.  if sInp = TArrayu1[i] then begin
52.  sRpl := ' -[ʊ]- ';
53.  end;
54.  end;
55. For i := 1 to Countu do begin
56.  if sInp = TArrayu[i] then begin
57.  sRpl := ' -[u]- ';
58.  end;
59.  end;
60.  If (sRpl ='') And (Pos('ou',sInp)>0)
61.     then begin
62.       sRpl := ' -[au]- ';
63.   End;
64.   If (sRpl ='') And (Pos('ow',sInp)>0)
65.     then begin
66.       sRpl := ' -[au]- ';
67.   End;
68.   If Pos('ou',sInp)>0
69.     then begin
70.       sInp:=Replace(sInp,'ou',sRpl)+sTail;
71.     end;
72.     If Pos('ow',sInp)>0
73.     then begin
74.       sInp:=Replace(sInp,'ow',sRpl)+sTail;
75.     end;
76.     WriteLn(sInp);
77. end.
```

ЗАДАЧА 3

Составить алгоритм прочтения диграфа -th-

Условия задачи

1. **/ɵ/:** thin, thaw, teeth (at the beginning and at the end of the words)

!/ɵ/: the, this, these, that, those, they, them, their, there, then, than, ***with***

!/t/: Themes, Thomas

2. **/ð/:** breathe, wither, teethe (between vowel graphemes)

Дополнительная сложность – необходимость учитывать положение в слове (начало, конец, середина) и положение между гласными (задать, какие графемы являются гласными).

Задаем гласные через константу, тип которой — набор символов: **set of char ['a',]** и т.д. -- в квадратных скобках, в одинарных кавычках.

Далее будем проверять оператором **In** нахождение символа до и после диграфа в этом наборе гласных.

Проверим, чтобы перед и после -th- была хотя бы одна буква (дираф не в начале и не в конце строки).

Для этого позиция диграфа должна быть больше единицы (не в начале слова) и одновременно с этим (оператор булевой алгебры **and**) позиция должна быть как минимум на два меньше, чем длина слова (диграф занимает две позиции, потому что он диграф — состоит из двух символов). Еще одно **and** для того, чтобы проверить, что **sRpl** по-прежнему пустая строка (sRpl = ' ').

Строки 35-38: в переменной **i** находится позиция -th-, которая засылается туда функцией **Pos.**

i-1 — это позиция перед диграфом

i+2 — это позиция после диграфа. Предположим, что позиция **i=3**. Это означает, что буква -t- третья, а буква -h- четвертая, позиция за диграфом, таким образом, будет пятой.

Выполнение через **консоль** с вызовом **mc** (midnight commander) — аналог Norton Commander в Dos. Быстрый вызов списка папок через **CTRL верхняя косая.** Просмотр результатов через **CTRL O**.

Программа

```
1.  program ReadDigraphTH;
2.  const
3.  CountTH1 =  11;
4.  TArrayTH1 : array [1..CountTH1] of AnsiString = ('the', 'this', 'these', 'that', 'those', 'they',
    'them', 'their', 'there', 'then', 'than');
5.  CountTH2 = 2;
6.  TArrayTH2 : array [1..CountTH2] of AnsiString = ('Themes', 'Thomas');
7.  vowel : set of char = ['a', 'e', 'i', 'o', 'u', 'y'];
8.  var
9.  sInp, SRpl : AnsiString;
10. i : Integer;
11. cBefore, cAfter: char;
12. function Replace (sInput, sSearch, sReplace : AnsiString) : AnsiString ;
13. var i1 : Integer;
14. sTmp : AnsiString;
15. begin
16.  i1:= Pos(sSearch, sInput);
17.  sTmp:= sInput;
18.  delete(sTmp, i1, Length(sSearch));
19.  Insert(sReplace, sTmp, i1);
20.  Replace:= sTmp;
21. end;
22. begin
23.  sRpl := '';
24.  ReadLn(sInp);
25.  For i := 1 to CountTH1 do begin
26.    if sInp = TArrayTH1[i] then begin
27.      sRpl := ' -[ð]- ';
28.    end;
29.  end;
30.  For i:= 1 to CountTH2 do begin
31.    if sInp = TArrayTH2[i] then begin
32.      sRpl := ' -[t]- ';
33.    end;
34.  end;
```

```
35.  i := Pos('th', sInp);
36.  if (i>1) and (i<length(sInp)-1) and (sRpl = '') then begin
37.    cBefore := Char(sInp[i-1]);
38.    cAfter := Char (sInp[i+2]);
39.    if (cBefore In vowel) and (cAfter In vowel) then begin
40.      sRpl := ' -[ð]- ';
41.    end;
42.  end;
43.  if sRpl = '' then sRpl := ' -[θ]- ';
44.  If Pos('th',sInp)>0
45.   then begin
46.     sInp:=Replace(sInp,'th',sRpl);
47.  end;
48.  WriteLn(sInp);
49. end.
```

ТЕМА 3. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБРАБОТКЕ ТЕКСТОВ

План

1. Основные задачи обработки текстов.
2. Анализ отдельных слов.
3. Анализ отдельных предложений.
4. Семантический анализ.

1. Основные задачи обработки текстов

Задачи обработки текстов возникли практически сразу вслед за появлением вычислительной техники. Но, несмотря на полувековую историю исследований в области искусственного интеллекта, огромный скачок в развитии ИТ и смежных дисциплин, удовлетворительного решения большинства практических задач обработки текста пока нет. Решением задач обработки текстов с применением ИТ занимается компьютерная лингвистика.

Компьютерная лингвистика – это раздел науки, изучающий применение математических моделей для описания лингвистических закономерностей. Эту область можно разделить на две большие части. Первая изучает способы применения вычислительной техники в лингвистических исследованиях – применение известных математических подходов (например, статистической обработки) для выявления закономерностей. Данные закономерности используются второй частью компьютерной лингвистики, изучающей вопросы осмысления текстов, написанных на естественном языке – создание математических моделей для решения лингвистических задач и разработка программного обеспечения, функционирующего на основе этих моделей.

Вторая часть компьютерной лингвистики тесно соприкасается с разделом искусственного интеллекта, занимающимся разработкой систем обработки текстов, написанных на естественном языке (NLP, Natural Language Processing).

Существует наиболее общая схема обработки текстов (см. рис. 1), инвариантная по отношению к выбору естественного языка (т.е. независимо от того, на каком языке написан исходный текст, его анализ будет проходить все указанные в схеме стадии): 1. Разбиение текста на отдельные предложения; 2. Разбиение текста на отдельные слова; 3. Определение характеристик отдельных слов; 4. Синтаксический анализ

Первые две стадии практически одинаковы для большинства естественных языков. Единственное, где могут проявляться черты, специфичные для выбранного языка - это обработка сокращений слов и обработка знаков препинания (точнее, определение того, какие из знаков препинания являются концом предложения, а какие нет).

Последующие две стадии (определение характеристик отдельных слов и синтаксический анализ), наоборот, очень сильно зависят от выбранного естественного языка. Последняя стадия (семантический анализ), как и первые стадии, мало зависит от выбранного языка, но, это проявляется только в общих подходах к проведению анализа.

Семантический анализ полностью основывается на результатах работы предыдущих фаз обработки текста, которые всегда специфичны для конкретного языка. Следовательно, способы представления их результатов тоже могут сильно варьироваться, оказывая большое влияние не реализацию методов семантического анализа. Результаты анализа, произведенного на ранних стадиях, могут быть многозначны - для выходных параметров указывается не одно, а сразу несколько возможных значений (например, может существовать несколько способов трактовки одного и того же слова). В таком случае последующие стадии должны определять наиболее вероятные значения результатов ранних стадий анализа и уже на их основе проводить дальнейший анализ текста.

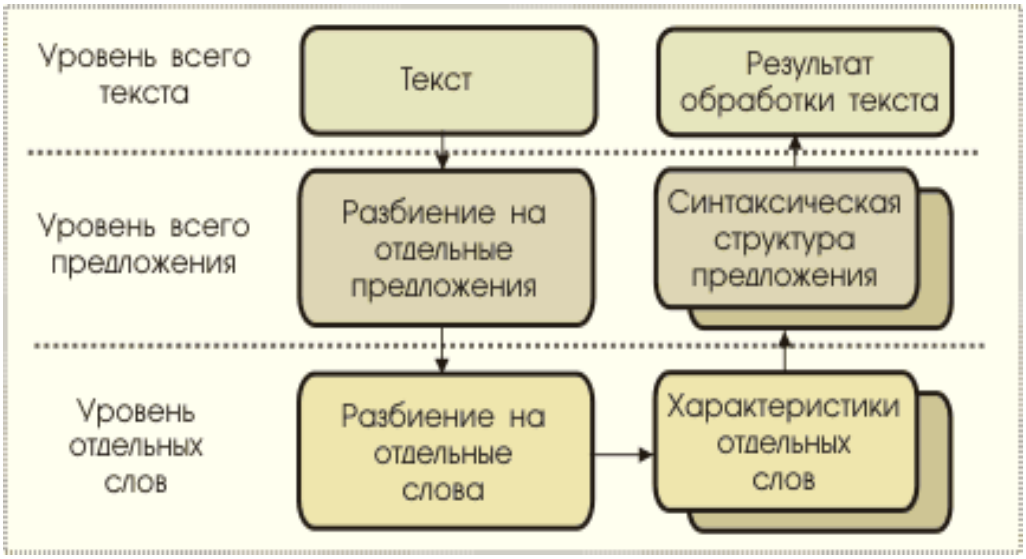

Рисунок 1. Общая схема обработки текстов

Специфические моменты обработки текстов на естественном языке

- обработка сокращений слов;
- обработка знаков препинания: определение того, какие из знаков препинания являются
- концом предложения, а какие нет;
- определение характеристик отдельных слов;
- синтаксический анализ;
- семантический анализ.

Рассмотрим более подробно каждую из стадий анализа текста после разделения текста на отдельные слова и предложения. К первой стадии (анализ отдельных слов) относится морфологический анализ (определение морфологических характеристик каждого слова: часть речи, падеж, склонение, спряжение и т.д.) и морфемный анализ: приставка, корень, суффикс и окончание. Ко второй стадии относится синтаксический анализ текста, к третьей - различные задачи семантического анализа: поиск фрагментов текста, формализация, реферирование и т.д.

Анализ отдельных слов

В эту стадию обработки текста входит морфологический и морфемный анализ слов. Входным параметром этих методов является текстовое представление исходного слова. Целью и результатом морфологического анализа является определение морфологических характеристик слова и его основной словоформы. Перечень всех морфологических характеристик слов и допустимых значений каждой из них зависят от естественного языка. Тем не менее, ряд характеристик (например, название части речи) присутствуют во многих языках. Результаты морфологического анализа слова неоднозначны, что можно проследить на множестве примеров.

Существует три основных подхода к проведению морфологического анализа. Первый подход часто называется ***"четкой" морфологией***. Для русского языка он основан на словаре А.А. Зализняка. Второй подход основывается на некоторой системе правил, которые по заданному слову определяют его морфологические характеристики. В противоположность первому подходу, его называют ***"нечеткой" морфологией***. Третий, ***вероятностный***

подход, основан на сочетаемости слов с конкретными морфологическими характеристиками. Он широко применяется при обработке аналитичных языков со строго фиксированным порядком слов в предложении и практически неприменим при обработке текстов на русском языке. Рассмотрим каждый из указанных способов морфологического анализа более подробно.

Словарь Зализняка (см. рис. 2) содержит основные словоформы слов русского языка, для каждой из которых указан некоторый код. Известна система правил, с помощью которой можно построить все формы данного слова, отталкиваясь от начальной словоформы и соответствующего ей кода. Помимо построения каждой словоформы, система правил автоматически ставит ей в соответствие морфологические характеристики. При проведении четкого морфологического анализа необходимо иметь словарь всех слов и всех словоформ языка. Этот словарь на входе принимает форму слова, а на выходе выдает его морфологические характеристики.

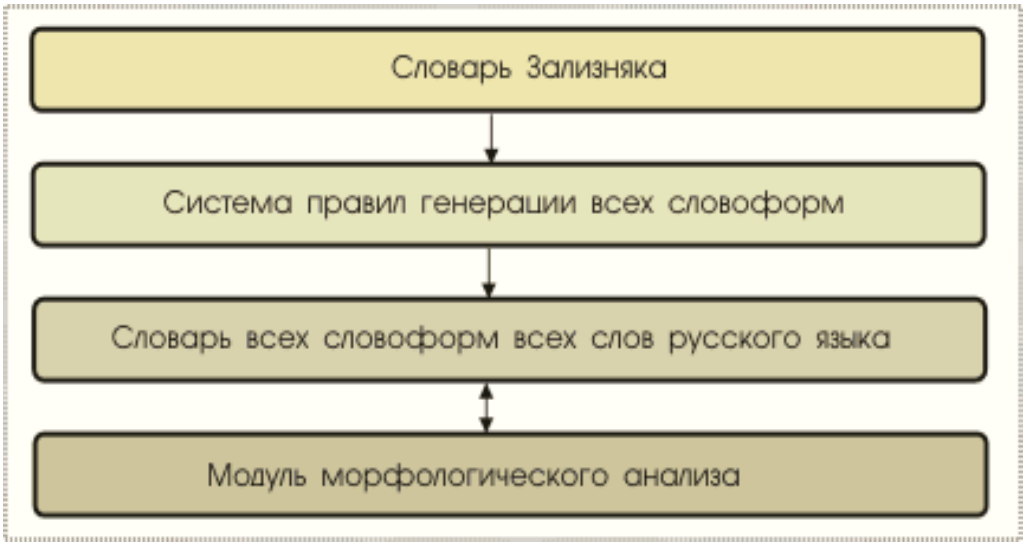

Рисунок 2. Точный морфологический анализ

Данный словарь можно построить на основе словаря Зализняка по очевидному алгоритму: необходимо перебрать все слова из словаря, для каждого из них определить все возможные их словоформы и занести их в формирующийся словарь. Эта задача решена.

При таком подходе получается, что для проведения морфологического анализа заданного слова необходимо просто найти его в словаре, где уже хранятся точные, "окончательно известные" значения всех морфологических характеристик заданного слова. Возможно, что для одного и того же входного слова будут храниться сразу несколько вариантов значений его морфологических характеристик.

К сожалению, этот способ применим не всегда, поскольку слова, поступающие на вход, могут не входить в словарь всех словоформ. Такая ситуация может возникнуть из-за ошибок ввода исходного текста, наличия в тексте имен собственных и так далее. В случае, когда метод точной морфологии не дает нужного результата, применяется неточная морфология.

Целью **морфемного анализа** слова является разделение слова на отдельные лексемы: приставки, корни, суффиксы и окончания. В словаре морфем русского языка указанно разделение каждого слова на отдельные части, но не указаны типы каждой из получившихся частей (какая из них является приставкой, какая корнем и т.д.). Известно, что множество всех корней слов русского языка открыто, но множество всех возможных приставок, суффиксов и окончаний – ограничено. Кроме того, известно, что в любом слове сначала идут приставки, затем корни, далее суффиксы и окончание. Поэтому на основе словаря морфем

русского языка можно построить другой словарь, который будет содержать не только разбиение каждого слова на части, но и тип каждой из них. В таком случае, для проведения морфемного анализа слова необходимо обратится к этому словарю. Подобная задача также решена (см. рис. 3).

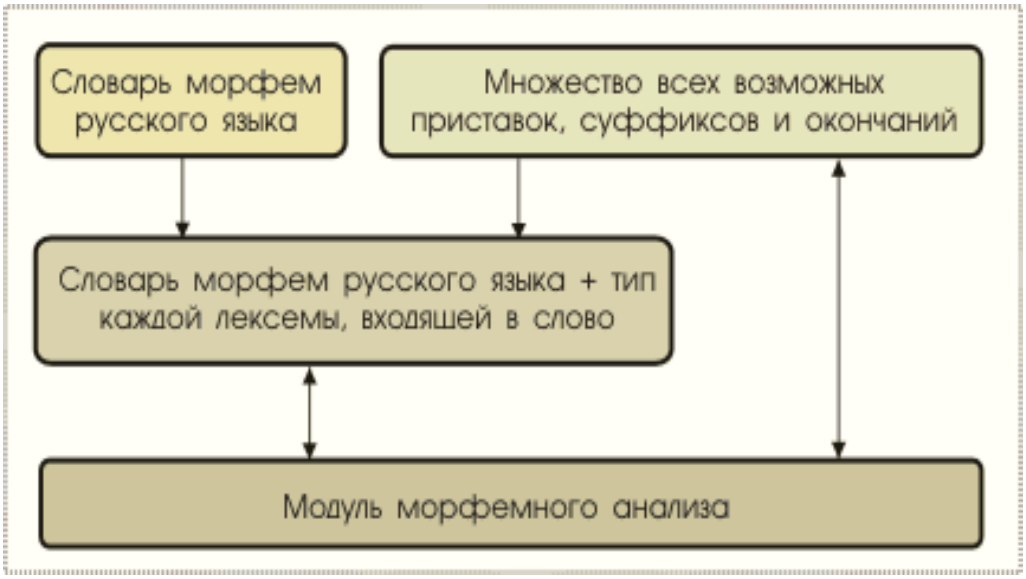

Рисунок 3. Морфемный анализ

Морфемный анализ не ограничивается обращениями к словарю. В ситуации, когда слово отсутствует в словаре, возможно непосредственное проведение анализа, на основе стандартного строения слов русского языка (приставка + корень + суффикс + окончание) и множества всех приставок, суффиксов и окончаний.

Вернемся к морфологическому анализу слова в той ситуации, когда не удалось определить характеристики слова с помощью методов точной морфологии, но удалось расчленить слово на отдельные части. Наличие тех или иных лексем может определять морфологические характеристики этого слова можно построить систему правил, которая будет опираться на наличие или отсутствие каких-либо частей и выдавать одно или несколько предположений о морфологических параметрах. Данный набор правил можно построить двумя способами. Первый основан на морфемном анализе слов, содержащихся в словаре всех словоформ, и их морфологических характеристик. Рассмотрим эту задачу более формально: известны пары значений, состоящие из морфемного строения слова и его морфологических характеристик. Это есть не что иное, как "вход" и "выход" системы правил, которая по морфемному строению слова будет определять его морфологические характеристики. Задача построения такой системы правил может быть решена с помощью самообучающейся системы некоторого типа. В данном случае могут быть использованы деревья решений, ILP (Inductive Logic Programming) и прочие алгоритмы.

Второй подход – **неточный** морфологический анализ – заключается в формировании набора правил вручную. По большому счету, реализация такого подхода ни что иное, как написание экспертной системы диагностирующего типа.

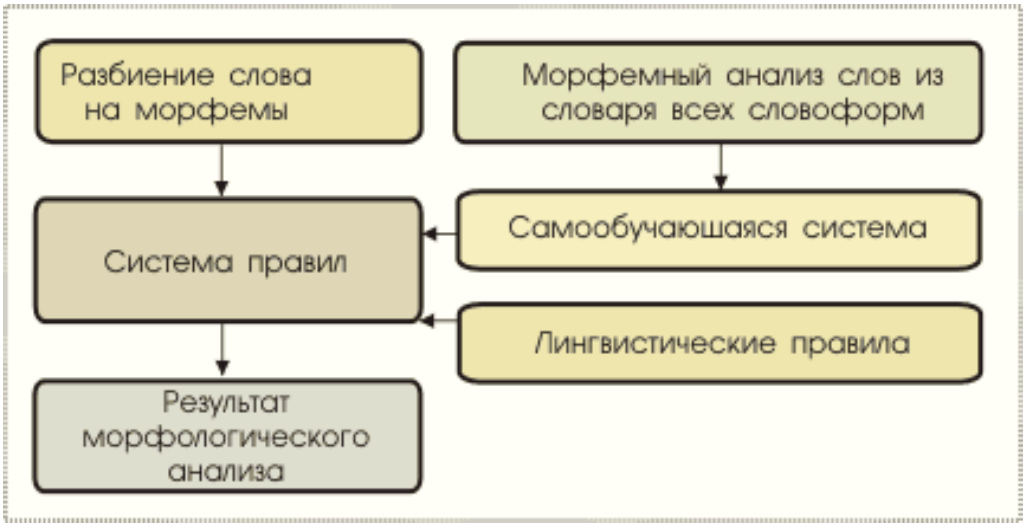

Рисунок 4. **Неточный морфологический анализ**

Третий подход – **вероятностный** способ проведения морфологического анализа слов – заключается в следующем. Одна и та же словоформа может принадлежать сразу к нескольким грамматическим классам. Для каждой словоформы определяются все ее грамматические классы, а также вероятность ее отношения к каждому из этих классов. Это происходит на основе некоторого набора документов, где каждому слову предварительно поставлен в соответствие грамматический класс. После этого вычисляются вероятности сочетаний определенных грамматических классов для слов, стоящих рядом (для двоек, троек, четверок и так далее). На основе этих чисел может проводиться анализ слов, но для него необходимо уже не только само слово, но и стоящие рядом с ним слова.

К методу вероятностного анализа необходимо сделать два важных замечания. Во-первых, он применим только для аналитичных языков, у которых четко фиксирован порядок слов в предложении. Если же порядок слов можно изменять, то данный метод неприменим, поскольку все возможные сочетания грамматических классов будут практически равновероятны. Во- вторых, если первые два способа анализа (точная и неточная морфология) на входе принимают отдельные слова, то вероятностный способ, наоборот, на входе принимает или все предложение или, по крайней мере, несколько слов, стоящих рядом.

Синтаксический анализ

После того, как произведен анализ каждого слова, начинается анализ отдельных предложений (синтаксический анализ) для определения взаимосвязей между отдельными словами и частями предложения (слайд). Результатом такого анализа является граф, узлами которого выступают слова предложения, при этом, если два слова связаны каким-либо образом, то соответствующие им вершины графа связаны дугой с определенной окраской. Возможные окраски дуг зависят, во-первых, от языка, на котором написано предложение, во-вторых, от выбранного способа представления синтаксической структуры предложения.

При синтаксическом анализе предложений русского языка (слайд) в качестве возможных окрасок дуг можно использовать вопросы, задаваемые от одного слова к другому. В вершинах графа пишутся слова не в том виде, в котором они встречаются в предложении, а в их основной словоформе. Некоторым словам (например, предлогам) вообще не соответствует ни одна из вершин графа, но эти слова влияют на вопросы, задаваемые от одного слова к другому.

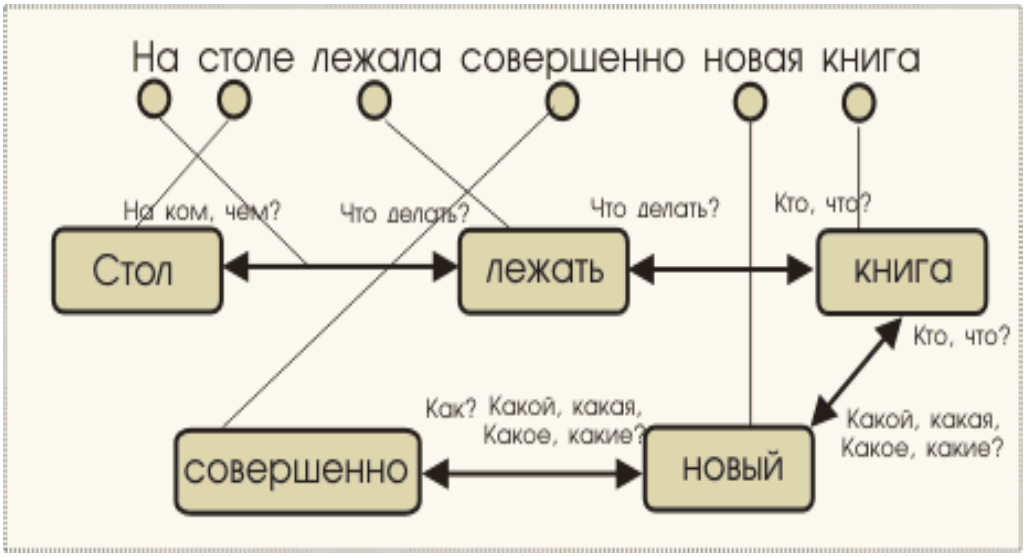

Рисунок 5. **Пример синтаксического анализа предложения**

Дуги графа обладают следующими свойствами. Во-первых, они являются двунаправленными, при этом, вопросы, написанные в начале и в конце дуги, различаются между собой. Во-вторых, вопросы соответствуют морфологическим характеристикам основной словоформы слова и не соответствуют той форме слова, которая употреблена в предложении. В-третьих, они не отражают смысловой нагрузки слов, и это свойство проявляется, прежде всего, в том, что не делается различия между одушевленными и неодушевленными предметами. В-четвертых, образующийся граф является деревом. В-пятых, если вершины графа расположить слева направо в порядке, соответствующем словам в исходном предложении, то дуги графа не будут пересекаться.

Возможны и другие способы представления зависимостей между словами в одном предложении (например, разбор предложения по частям и выделение подлежащего, сказуемого и так далее). На основе системы этих зависимостей могут быть разработаны несколько иные способы представления синтаксической структуры предложения.

Перейдем теперь к **методам** синтаксического анализа предложений. Их можно разделить на две группы: методы с фиксированным, заранее заданным набором правил и самообучающиеся методы. Заметим, что правила представляются не в виде классических продукций ("если ..., то ..."), а в более удобном виде – в виде грамматик, задающих синтаксис языка. Исторически, первым способом описания синтаксиса языка были формальные грамматики.

Формальные грамматики хорошо изучены и широко применяются при описании формальных языков (например, языков программирования), но они непригодны для описания синтаксиса естественных языков.

Трансформационные грамматики разрабатывались уже специально для задания синтаксических правил построения предложений, написанных на естественном языке. Эти грамматики задаются в виде ориентированного графа состояний, всем дугам которого поставлены в соответствие определенные части речи. В начале работы алгоритм синтаксического анализа находится в некотором начальном состоянии, которому соответствует некоторая вершина графа. Алгоритм просматривает предложение слева направо, анализирует встречающиеся слова и делает переходы из одного состояния в другое в соответствии с выходящими из текущей вершины дугами и очередным словом предложения. Работа заканчивается либо когда предложение просмотрено полностью, либо

когда невозможно сделать переход из текущего состояния (нет выходящей дуги с необходимой пометкой).

Основными недостатками трансформационных грамматик являются 1) неспособность задавать рекурсивные синтаксические правила; 2) построение трансформационной грамматики даже для небольшого подмножества языка требует больших усилий.

Два описанных способа заключают в себе четко заданную систему правил, согласно которым производится синтаксический анализ предложения. Помимо уже указанных недостатков, они имели еще один большой минус, который заключается в неспособности этих методов анализировать неправильно построенные предложения. Это привело к созданию новых методов синтаксического анализа, основанных на вероятностном подходе, к которым нужно отнести вероятностные грамматики и вероятностный разбор.

Вероятностные грамматики являются расширением формальных и отличаются от них в следующем: каждому правилу построения указана некоторая вероятность применения этого правила при построении предложения. После того, как произведен синтаксический анализ предложения известно, на основе каких правил было построено это предложение и на основе сопоставленных с ними вероятностей может быть посчитана "суммарная" вероятность.

Очевидно, что одно и то же предложение может быть разобрано несколькими способами. Для каждого из них считается его "суммарная" вероятность и выбирается наиболее вероятный способ построения предложения. Этот метод способен анализировать неправильно построенные предложения, но он, как и два предыдущих способа анализа, включает в себя систему заранее задаваемых правил.

Синтаксический анализ на основе обучающихся систем – это пока еще малоизученный подход. Он заключается в следующем: разрабатывается некоторое множество примеров, каждый из которых содержит пару: исходное предложение и результат его синтаксического анализа. Этот результат вводится человеком, занимающимся обучением системы, в ответ на каждое подаваемое на вход предложение. Затем, при подаче на вход предложения, не входящего в список примеров, система сама генерирует результат. В качестве подхода к реализации такой обучающейся системы используются ряд способов: нейронные сети, деревья вывода, ILP и методы поиска ближайшего соседа.

Это далеко не весь спектр методов синтаксического анализа. Удовлетворительных решений данного вопроса пока еще не найдено, хотя есть методы, дающие неплохие результаты, но работающие только на подмножестве языка. Решение задачи синтаксического анализа текста должно послужить основой, во-первых, для построения более совершенных синтаксических корректировщиков (программных средств, проверяющих правильность построения предложения) и, во-вторых, для построения алгоритмов более качественного семантического анализа текстов.

Семантический анализ

Семантический анализ текста базируется на результатах работы синтаксического анализа, получая на входе уже не набор слов, разбитых на предложения, а набор деревьев, отражающих синтаксическую структуру каждого предложения. В силу того, что методы синтаксического анализа пока мало изучены, решения целого ряда задач семантической обработки текста базируются на результатах анализа отдельных слов, и вместо синтаксической структуры предложения, анализируются наборы стоящих рядом слов.

Большинство методов семантического анализа должны, так или иначе, работать со смыслом слов, а, следовательно, должна быть какая-то общая для всех методов анализа база, позволяющая выявлять семантические отношения между словами. Такой основой является тезаурус языка. На математическом уровне он представляет собой ориентированный граф, узлами которого являются слова (в их основной словоформе). Дуги задают **отношения между словами** и могут иметь следующие окраски.

1. Синонимия. Слова, связанные дугой с такой окраской являются синонимами.

2. Антонимия. Слова, связанные дугой с такой окраской являются антонимами.

3. Гипонимия. Дуги с такой окраской отражают ситуацию, когда одно слово является частным случаем другого (Например, слова "мебель" и "стол"). Дуги направлены от общего слова к более частному.

4. Гиперонимия. Это отношение является обратным к гипонимии.

5. Экванимия. Дугами с такой окраской связаны слова, являющиеся гипонимами одного и того же слова.

6. Омонимия. Слова, связанные таким отношением, имеют одинаковое написание и произношение, но имеют различный смысл.

7. Паронимия. Данный тип дуги связывает слова, которые часто путают.

8. Конверсивы. Слова, связанные такой окраской, имеют "обратный смысл" (Например, "купил" и "продал").

Таким образом, тезаурус задает набор бинарных отношений на множестве слов некоторого естественного языка. В настоящий момент имеется тезаурус английского языка, но для русского языка работа по созданию тезауруса еще не проделана, хотя имеются коммерческие продукты, включающие в себя тезаурус подмножества русского языка, а также отдельные словари синонимов и антонимов для подмножества русского языка. К сожалению, эти словари в электронном виде пока не доступны.

Семантический анализ текста включает в себя ряд практически важных задач и, поэтому, будем рассматривать не методы анализа, а актуальные задачи и уже существующие их решения. Одной из наиболее изученных задач является контекстно-свободный поиск текстовой информации. Ее смысл заключается в следующем: имеется большой набор файлов, содержащих тексты на некотором естественном языке, и задана некоторая строка. Необходимо найти все файлы, в которых она или похожая текстовая информация встречается. В подавляющем большинстве случаев необходим именно "неточный" поиск, то есть по смыслу слова с учетом специфики естественного языка. Большинство существующих систем основываются исключительно на морфологическом анализе слов и не задействуют более сложных схем анализа.

Наиболее важной с практической точки зрения задачей является извлечение информации из текстовых данных и представление ее в виде формальной системы знаний, в частности, в виде семантической сети. На данный момент имеются экспериментальные разработки в данном направлении, ориентированные на конкретную предметную область знаний, однако готовых коммерческих программных продуктов пока нет.

Рассмотрим эту задачу более подробно. Имеется ***семантическая сеть***, состоящая из узлов и связей.

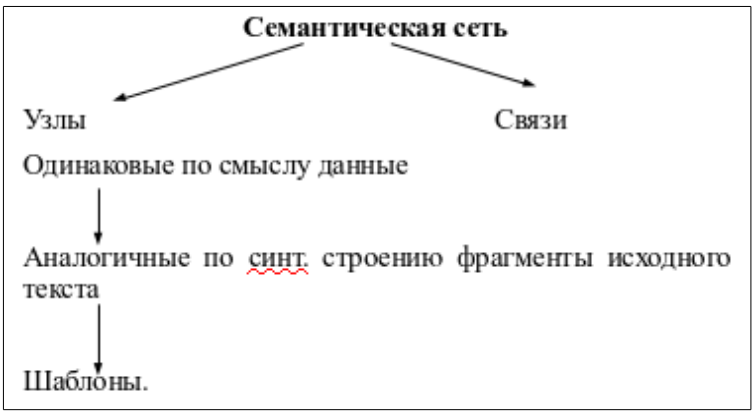

Рисунок 6. **Структура семантической сети**

Каждому из узлов соответствуют некоторые атомарные данные, а каждой дуге — некоторая окраска. Если семантическая сеть построена на основе анализа некоторого текста и является формализованным представлением содержащихся в нем знаний, то каждому ее элементу соответствует некоторый фрагмент исходного текста. Узлам, отражающим одинаковые по смыслу данные (например, вес, возраст, дата рождения), соответствуют аналогичные по синтаксическому строению фрагменты исходного текста. Значит, можно ввести некоторые шаблоны. Каждый из них описывает синтаксическую структуру части исходного текста и создаваемые элементы семантической сети. При описании синтаксической структуры указываются не только связи слов в предложении, но и условия, накладываемые на каждое из слов. Эти условия могут проверять как морфологические или семантические (на основе тезауруса) характеристики слова, так и смысловые пометки этого слова, поставленные при поиске других шаблонов. Если какая-то часть текста удовлетворяет всем условиям, указанным в шаблоне, то происходит ее формализация.

Извлечение информации из текстовых данных является основой для решения двух важных задач: во-первых, "раскопки" текста , во-вторых, создания систем загрузки текстов в хранилища данных. Такие системы существуют и предназначены для интеграции и очистки данных, помещаемых в хранилища, но они не предоставляют никаких средств ввода данных, содержащихся в текстовом виде.

Наряду с извлечением информации существует и обратная задача генерации правильно построенных текстов. Исходными данными для таких систем является четко формализованные знания. На первый взгляд эта задача может показаться странной, поскольку в большинстве случаев формализованные знания можно представлять в виде бланков, имеющих четкую, заранее определенную систему полей. Но это не всегда так: в случае, когда предметная область имеет сложную и разветвленную структуру, то часто оказывается, что большинство полей бланка оказываются пустыми, что сильно затрудняет восприятие информации, и для конченого пользователя было бы гораздо проще и удобнее иметь дело не с таким бланками, а с неформализованным (но корректно построенным) текстовым описанием тех же самых данных.

С позиции двух указанных задач интересно рассмотреть методы обработки текстовой информации, разработанные Роджером Шенком. Они образуют психолингвистический подход к анализу текстовой информации и основываются на двух идеях. Во-первых, для анализа одного предложения не обязательно рассматривать все его слова- смысл предложения можно определить по "ключевым" словам и наличию связей между ними.

Вторая идея заключается в представлении результатов анализа текста в виде концептуальной сети, способной формально описать смысл, содержащийся в исходном тексте и являющейся семантической сетью с предопределенным набором типов узлов и дуг.

В качестве простого примера автоматической обработки грамматических показателей текста приведём программирование автоматического поиска нужных грамматических явлений.

ЗАДАЧА 4

Найти в тексте все формы глагола *to be* и вывести на экран с указанием количества употреблений в тексте в абсолютных единицах и в процентах по сравнению с общим количеством слов в тексте.

1. Необходимо объяснить компьютеру, что такое слово. **Слово** — это последовательность буквенных символов или знака «дефис», при этом слово начинается и заканчивается только на буквенные символы; перед и после слова идут один или более разделителей, в качестве которых выступают все небуквенные символы, а также начало или конец файла.

2. Поскольку язык Free Pascal изначально не содержит в себе средства работы с текстовыми форматами текстовых процессоров (Writer), для обработки текстов в нем необходимо перевести текст, в котором будет производиться поиск, в файл PLAIN TEXT (плоский = простой текстовый файл), чтобы не мешало форматирование (достигается выбором в меню «сохранить как» для writer — текст .txt, для MS Word — текст MS-DOS).

3. Формы глагола to be: is, am, are, was, were, be, been

В области **констант** через массив задаем формы глагола to be с указанием количества форм. Формы записываем большими буквами, поскольку в тексте они могут употребляться в разных регистрах (как большие, так и маленькие), поэтому мы все буквы приводим к верхнему регистру (строки 4 и 28).

В области **переменных** в глобальной области видимости мы описываем следующие переменные: текстовый файл, в котором будет произведен подсчет (строка 6), массив счетчиков для форм глагола to be (строка 7), символ, который читается из файла (читается по одному символу (строка 8)); переменная для промежуточного хранения слова (строка 9); количество всех слов в тексте (строка 10).

В программе используемые следующие **процедуры**.

1. Процедура init – начальная инициализация переменных, т.е. Обнуление счетчиков (строки 11-18) и открытие файлов (строки 19-21).

2. Процедура пропуска разделителей слов (строки 22-30). Строка 24: присваиваем символу (C) небуквенное значение, чтобы нормально работал цикл с предусловием (while – строка 25), который читает символы пока не прочитает буквенный символ или не наступит конец файла.

3. Процедура чтения слова: читает символы из файла, добавляет их в переменную S пока символы являются буквами (строки 31–41). Если слово заканчивается на дефис, то дефис выбрасываем (строки 42–43). Далее прибавляем единицу к счетчику слов (44–45).

4. Процедура сравнения прочитанного слова с формами искомого глагола (46–54). Если строка соответствует i-той форме глагола to be, то к этому счетчику добавляется единица (51).

5. Финальная процедура: закрытие текстового файла и выдача результатов на экран.

Программа

```
1. Program Count_to_be;
2. const
3. Counttobe = 7;
4. Arraytobe: array [1..Counttobe] of string = ('IS', 'AM', 'ARE', 'WAS', 'WERE', 'BE',
   'BEEN');
5. var
6. F : text;
7. Tcounttobe : array [1..Counttobe] of integer;
8. C : char;
9. S : string;
10. CountWord : integer;
11. procedure init;
12. var i : integer;
13. begin
14.    For i := 1 to Counttobe do begin Tcounttobe [i] := 0;
```

```
15. end;
16.    CountWord := 0;
17.    C := ' ';
18.    S := '';
19.    assign (F, 'script1.txt');
20.    reset (F);
21. end;
22. procedure SkipToWord;
23. begin
24.    C:=' ';
25.     while not eof(F) and not (C in ['A'..'Z']) do
26.     begin
27.       read(F, C);
28.       C := UpCase (C);
29.    end;
30. end;
31. procedure ReadWord;
32. begin
33.   if not eof(F) and (C in ['A'..'Z']) then S := C;
34.   while not eof(F) and (C in ['A'..'Z', '-']) do
35.     begin
36.       read(F, C);
37.       C := UpCase (C);
38.       If (C in ['A'..'Z', '-']) then begin
39.         S := S+C;
40.       end;
41.    end;
42.    if S [length (S)] = '-' then begin delete (S, length(S), 1);
43.    end;
44.    CountWord := CountWord + 1;
45. end;
46. procedure CheckToBe;
47. var i : integer;
48. begin
49.   For i := 1 to Counttobe do begin
50.     if S = Arraytobe [i] then begin
51.       Tcounttobe [i] := Tcounttobe [i]+1;
52.     end;
53.    end;
54. end;
55. procedure final;
56. var i : integer;
57. begin
58.   close(F);
59.   Writeln ('words in text ', CountWord);
60.   For i := 1 to Counttobe do begin
61.         Writeln ('Count of ', Arraytobe [i], '=', Tcounttobe [i],', or ',100.0*Tcounttobe
      [i]/CountWord:7:3,'%');
62.   end;
63. end;
```

```
64. begin
65.   init;
66.   while not eof (F) do begin
67.     SkipToWord;
68.     ReadWord;
69.     CheckToBe;
70.   end;
71.   final;
72. end.
```

Программирование тестов

ЗАДАЧА 5

Запрограммировать многовыборный тест.

Условия задачи

Имеется некоторое количество заданий с четырьмя вариантами ответов на каждое. Только один из этих вариантов – правильный. По окончании выполнения теста необходимо дать результаты автоматического подсчета количества правильных ответов. Задания и варианты представлены в следующем виде.

Choose only one correct variant

1. Her eyes are red. She ... all night.
a) cries; b) is crying; c) cried; **d)** has been crying.
2. I cut myself when I ... .
a) shaved; **b)** was shaving; c) had shaven; d) had been shaving.
3. I look ... hearing from you.
a) after; b) for; c) at; **d)** forward to.
4. I ... to him before you arrived.
a) was talking; b) talked; **c)** had talked; d) had been talking.
5. His father is a ... . He works on a big modern ship.
a) sailor; b) seller; c) doctor; d) cop.

Язык программирования – Free Pascal

Пояснения

Имеются две строки: первая – вопроса, вторая – вариантов ответа. На каждый вопрос требуется ввести один символ – букву правильного ответа.

Задачу решаем через введение ***трех массивов***: первый массив – вопрос, тип – строка; второй массив – варианты ответов, тип – строка; третий массив – правильный ответ, тип – символ.

Программа

```
1. program MultipleChoice;
2. const
3. count = 5;
4. Questions:Array[1..count] of string = (
5. '1. Her eyes are red. She ... all night.',
```

```
6.  '2. I cut myself when I ... .',
7.  '3. I look ... hearing from you.',
8.  '4. I ... to him before you arrived.',
9.  '5. His father is a ... . He works on a big modern ship.');
10. AnswerVariants: Array [1..count] of string = (
11. 'a) cries; b) is crying; c) cried; d) has been crying.',
12. 'a) shaved; b)  was shaving; c) had shaven; d) had been shaving.',
13. 'a) after; b) for; c) at; d) forward to.',
14. 'a) was talking; b) talked; c) had talked; d) had been talking.',
15. 'a) sailor; b) seller; c) doctor; d) cop.');
16. CorrectVariant: Array [1..count] of char = (
17. 'd', 'b', 'd', 'c', 'a');
18. var
19. InpChar : char;
20. CorrectAnswer: integer;
21. Counter : integer;
22. begin
23. CorrectAnswer:= 0;
24. For Counter:= 1 to count do begin
25. Writeln('-------------------------------------------------');
26. Writeln('Question No ',Counter);
27. Writeln (Questions [Counter]);
28. Writeln (AnswerVariants [Counter]);
29. repeat
30. Readln (InpChar);
31. if not (InpChar in ['a'..'d']) then begin
32. Writeln('Input error. Please enter character a, b, c or d')
33. end;
34. until InpChar in ['a'..'d'];
35. if InpChar = CorrectVariant [Counter] then begin
36. CorrectAnswer:= CorrectAnswer + 1;
37. end;
38. end;
39. writeln ('Correct answers ',CorrectAnswer,' or ',(CorrectAnswer/count*100):7:3,'%');
40. end.
```

Автоматический перевод

I.Концептно-ориентированная модель памяти переводов.

1. Перевод текстов. Общие понятия. Восемь типов памяти перевода.
2. Выделение терминов и анализ терминологии.
3. Аспекты использования памяти переводов.
4. Основные принципы работы.
5. Пути расширения возможностей.

1. Перевод текстов. Общие понятия

Существуют различные определения понятия перевод текстов. В качестве рабочего определения примем следующее: «Перевод есть вид человеческой языковой деятельности, в результате которой некоторый текст на одном языке ставится в соответствие тексту на другом языке, при этом обеспечивается их смысловая эквивалентность» . Слово «перевод» понимают

двояко: как сам процесс перехода от текста на одном языке к этому же тексту на другом языке, так и результат этого перехода, т.е. тот текст, который получается в результате перевода.

Переводом текстов человек начал заниматься еще в античном мире — более 20 веков назад. Одним из первых основные принципы перевода сформулировал Марк Тулий Цицерон (106—43 гг. до н.э.), древнеримский политический деятель, оратор и писатель. Он переводил произведения древних греков на латинский язык и считал, что переводить следует не слова,

а мысли, не букву, а смысл, в соответствии с условиями и духом своего языка.

Однако такой правильный взгляд на теорию перевода не являлся общепринятым как в древние, так и в последующие средние века (конец V— середина XVII в.). Вред научной теории перевода принесли переводы древнееврейских религиозных текстов на другие языки. При этом любое отступление от оригинала рассматривалось как ересь. Такой подход привел к

возникновению и развитию пословного перевода, буквализму, искажавшему смысл и стиль текста исходного языка.

В эпоху Возрождения (XIV—XVI вв.) появились шедевры мировой литературы: произведения Ф. Рабле, У. Шекспира, С. Сервантеса, Ф. Петрарки, Дж. Боккаччо и целого ряда других писателей и поэтов, что привело к резкому возрастанию количества переводов. В это время особенно отчетливо стали осознаваться трудности перевода художественных текстов.

Как протест против пословного, буквального перевода начинает возникать вольный перевод, при котором на другом языке передается лишь общая идея текста исходного языка. Такой перевод не учитывает стилистические особенности автора исходного текста, реалии места и времени описываемых в нем событий и многие другие особенности переводимого произведения. В процессе вольного перевода тексты исходного языка порой искажались до

неузнаваемости.

В течение многих веков речь шла лишь о переводе художественных текстов. И многие переводчики, литераторы, критики сомневались, что при этом можно передать художественную ценность оригинала. Известный лингвист Вильгельм фон Гумбольдт (1767–1835) писал об этом так: «В сущности всякий перевод представляется мне попыткой решить неразрешимую задачу, ибо переводчик оказывается между Сциллой и Харибдой: либо он в ущерб традиционным пристрастиям и языку своей нации держится слишком близко к оригиналу, либо – напротив – приносит оригинал в жертву особенностям своего языка и национального вкуса. Благополучно миновать обе эти особенности не только трудно, но и

невозможно». Как видно, В. Фон Гумбольдт стоял на позициях «непереводимости»: возможен или буквальный, или вольный перевод. Третьего не дано.

Идеи перевода в значительной степени интересовали И. В. Гёте, Н. В. Гоголя, А. С. Пушкина, В. Г. Белинского и других писателей, поэтов, критиков, переводчиков. Их труды, а также работы их последователей постепенно помогли подойти к созданию истинно научных теорий перевода, утверждающих возможность хорошего перевода текста с любого языка на другой язык. Такие теории начали создаваться в 50—60-е годы XX века. Не останавливаясь подробно на этом вопросе, отметим их одну общую особенность: данные теории опираются на метод моделирования процесса перевода текстов человеком.

Существование нескольких современных теорий перевода объясняется, во- первых, тем, что объект моделирования — умственные действия человека в процессе перевода — сложен и недоступен прямому наблюдению. Во-вторых, теории перевода решают разные задачи, связанные с языками, психологией человека, культурой и традициями страны исходного языка (ИЯ) и переводного языка (ИЯ). Все эти теории (модели) гипотетичны. Степень их «пригодности» для той или иной пары языков проверяется практикой.

Насчитывается большое количество типов и видов переводов, существуют различные подходы к их классификации . Рассмотрим некоторые из них.

В зависимости от *переводного материала* различают:

1) перевод художественной литературы;

2) научно-технический перевод (перевод научно-технических, военных, юридических текстов и т.д.);

3) общественно-политический перевод (перевод газет, политических журналов и т.д.);

4) бытовой перевод (перевод текстов разговорно-бытового характера).

По *форме презентации текста перевода и текста оригинала* выделяют:

1) письменный перевод;

2) устный перевод.

По *признаку основной прагматической функции перевода* (с какой целью делается перевод) различают:

1) практический перевод (в целях получения новой технической, экономической, политической, эстетической и другой информации);

2) учебный перевод (для обучения основам перевода);

3) экспериментальный перевод (например, для оценки умения переводчика и качества работы);

4) эталонный перевод (как образцовый перевод, с которым сравниваются другие переводы).

По *степени механизации процесса* перевода выделяют:

1) традиционный («ручной») перевод, выполняемый человеком;

2)перевод, выполняемый человеком с помощью компьютера (например, когда в памяти электронного устройства находится двуязычный словарь и компьютер по запросу пользователя ищет переводные эквиваленты иностранных слов);

3) перевод, выполняемый компьютером с помощью человека (например, когда компьютер по специальной программе делает перевод, а за справками — неизвестным переводным эквивалентом, неизвестной синтаксической структурой и т.д. — обращается к человеку-интерредактору);

4) машинный или автоматический перевод (выполняется компьютером без вмешательства человека).

Необходимость создания систем машинного перевода

Как было отмечено выше, машинный перевод осуществляется компьютером самостоятельно, без вмешательства человека. Более строго это понятие может быть определено следующим образом: машинный, или автоматический перевод, (МП или АП)| (англ.: machine translation, automatic translation) — это выполняемое компьютером действие по преобразованию текста на одном естественном языке в текст на другом естественном языке при сохранении эквивалентности содержания, а также результат такого действия. Человек, как правило, в той или иной мере участвует в подготовке машинного перевода или в его «доведении» до удобочитаемого вида. Чаще всего до ввода в компьютер переводимый текст специальным образом готовится человеком-предредактором. Он упрощает структуру предложений, выделяет терминологические обороты, указывает класс слов для омографичных форм и т.д. Выданный компьютером перевод для удобства чтения подвергается стилистической правке человеком-постредактором.

В течение последних десятилетий большое внимание во всех странах уделяется научно-техническому переводу — основному источнику новых научных и технических знаний. Европейское; экономическое сообщество имеет свою службу перевода, включающую около 2 тыс. переводчиков. Они переводят в год примерно 600 000 страниц текстов с пяти языков и также не справляются со все возрастающими потоками заказов на переводы. Это приводит к тому, что до специалистов различных стран зарубежная информация доходит с большим опозданием (порой через 5—10 лет).

Единственным способом увеличения скорости перевода является использование в переводческой деятельности современных компьютеров, которые в миллиарды раз быстрее человека могут выполнять необходимые для перевода логические действия. Человек-переводчик тратит 20 % своего времени на перевод, 40 % — на поиск по словарю незнакомых слов и 40 % — на пере печатку и оформление перевода. Компьютер же в процессе перевода тратит 95 % времени непосредственно на перевод и 5 % на пополнение словаря. Если максимальная производительность труда переводчика составляет 4—5 авторских листов в месяц, то такая, например, система машинного перевода, как SYSTRAN, переводит в час до 1 млн словоупотреблений (около 120 авторских листов). Эти количественные характеристики свидетельствуют о преимуществе компьютерных систем.

Однако качество такого перевода значительно уступает переводу, сделанному человеком. И тем не менее проблемами машинного перевода сейчас активно занимаются во всех развитых странах: США, Франции, Японии, Германии, Китае и т.д. Ежегодно по машинному переводу проводится несколько крупных международных конференций, в разных странах постоянно издаются журналы и книги по этой проблеме. Первый эксперимент по машинному переводу был проведен в США в Джорджтаунском университете 7 января 1954 года. ЭВМ перевела с русского языка на английский несколько достаточно простых предложений по физике. В России первый машинный англо-русский перевод был выполнен в 1955 году.

Основные понятия и проблемы машинного перевода

Чтобы понять, какие проблемы стоят перед специалистами по машинному переводу, рассмотрим, что нужно знать для того, чтобы перевести текст с одного языка на другой. Это прежде всего:

1) язык (лексика, грамматика), с которого осуществляется перевод;

2) язык (лексика, грамматика), на который переводится текст;

3) предметное содержание переводимого текста (реалии места и времени, область специальных знаний, индивидуальные особенности переводимого автора и т.п.).

Выше было отмечено, что все теории перевода текстов построены на основе моделирования деятельности человека-переводчика. Это в полной мере относится и к теориям машинного перевода. Однако мало что известно о детальных процедурах перевода текста человеком. В общем виде процесс перевода текста состоит из следующих трех этапов:

1) постижение текста на ИЯ;

2) интерпретация текста на ИЯ;

3) перевыражение текста ИЯ и создание текста на ПЯ.

Суть постижения исходного текста заключается в понимании того, о чем говорится в исходном тексте. При этом выделяют три формы понимания:

а) филологическое, или дословное, понимание текста. При этом предполагается, что человек, читающий текст, понимает значение всех слов этого текста, смысл всех его предложений и содержание всего текста;

б) стилистическое понимание — это понимание настроения героев, их иронии, трагических и комических эффектов и т.д.;

в) понимание идейного замысла автора. Оно выражается в осознании следующего: для чего автор писал свое произведение, что он хотел сказать читателю, какие цели преследовались автором при создании текста.

Переходя к характеристике интерпретации текста, необходимо отметить, что какие бы пары языков ни использовались в качестве исходного и переводного, в таких языках нет полного семантического тождества:

слову ИЯ, как правило, соответствует несколько слов ПЯ;

один тип предложения ИЯ может быть передан несколькими синтаксическими структурами ПЯ;

любая связная последовательность предложений ИЯ передается не сколькими допустимыми последовательностями ПЯ.

Поэтому лингвистически точный перевод текстов невозможен в принципе. Возможна лишь правильная интерпретация текста на исходном языке средствами переводного языка. При этом переводчик должен уметь передать объективный смысл всего произведения, сводя к минимуму свое субъективное отношение к описываемому в тексте.

Процесс перевыражения текста ИЯ в текст ПЯ носит творческий характер. Переводчику надо не только заменить слова и предложения ИЯ словами и предложениями ПЯ, но и сделать это стилистически верно, художественно. Текст перевода должен сохранить и предметное содержание исходного текста, и его идейный замысел.

Результатом процесса перевыражения должен стать такой текст ПЯ, который считается явлением своей литературы, переводного языка и в то же время сохраняет оттенок чужого.

Перевод текстов — задача, которая до сих пор не имеет однозначного алгоритма решения. Многие действия, которые выполняет человек в процессе перевода текста, неясны, и это создает огромные трудности в ходе построения алгоритмов машинного перевода. Тем не менее большинство специалистов по переводу отмечают, что в процессе перевода текста выполняется следующая последовательность действий:

1) морфологический анализ каждого слова предложения ИЯ;

2) синтаксический анализ каждого предложения текста ИЯ;

3) синтаксический синтез каждого предложения ПЯ;

4) морфологический синтез каждого слова предложения ПЯ.

Человек выполняет эти действия, опираясь на знания языка или словари. Очевидно, компьютер, осуществляющий перевод текстов, должен уметь выполнять те же самые действия.

В процессе морфологического анализа слов предложения ИЯ каждое слово получает наборы лексико-грамматических признаков (часть речи, род, число, падеж, время, лицо, управление и т.д.). Компьютер может сформировать такие наборы либо по формальным признакам слов (суффиксам, окончаниям, приставкам), либо с опорой на специальный автоматический словарь. В нем каждой словоформе уже даны соответствующие лексико-грамматические признаки, и в процессе морфологического анализа слова компьютер берет их из словаря в готовом виде.

Синтаксический анализ предложения ПЯ сводится к поиску основных членов предложения (группы подлежащего, группы сказуемого и т.п.).

Синтаксический синтез предложения ПЯ заключается в создании предложения ПЯ определенной синтаксической структуры, определяемой правилами ПЯ и синтаксической структурой предложения ИЯ. Чтобы компьютер мог выполнить это задание, он должен иметь в памяти сведения о синтаксических структурах ИЯ, ПЯ и их соответствиях друг другу. Еще одна задача этапа синтаксического синтеза связана с заменой слов ИЯ их переводными эквивалентами из словаря ПЯ.

Морфологический синтез каждого слова предложения ПЯ сводится к постановке слов ПЯ в нужном числе, роде, падеже, времени и т.д. Для этого компьютер должен владеть знаниями о лексико-грамматических признаках каждого слова ПЯ, которые берутся из упомянутого выше автоматического словаря.

Таким образом, система машинного перевода текстов, например с английского языка на русский, должна состоять из компонентов, показанных на рисунке 7.

Назначение четырех представленных на этой схеме подсистем ясно из вышеизложенного. Рассмотрим более подробно две другие составляющие — англо-русский автоматический словарь и англо-русские синтаксические соответствия.

Рисунок 7. Компоненты при машинном переводе

Основные определения

Концепт — это не зависящее от конкретного языка понятие, соответствующее реальной или абстрактной сущности, свойству, действию, либо иному элементу, отражающему связь между другими понятиями.

Термин — это слово или словосочетание на заданном языке, обозначающее в этом языке конкретный концепт.

Терминология — это множество обозначающих один и тот же концепт терминов из различных языков.

Сегмент — это непрерывный фрагмент текста, состоящего из терминов одного языка, обозначающих связанную по некоторому критерию группу концептов.

Вариант сегмента — это сегмент, похожий на исходный по некоторому критерию.

Исходный язык — это язык, с которого осуществляется перевод.

Целевой язык — это язык, на который осуществляется перевод.

Языковая пара — это упорядоченная пара сегментов, объявленных переводчиком эквивалентными по смыслу, первый из которых содержит термины на исходном языке, а второй — на целевом.

2. Восемь способов применения компьютера при переводе.

В современных профессиональных средах перевода возможности вычислительной техники используются на различных этапах и уровнях. Всего можно выделить восемь способов применения компьютера при переводе (таблица 1).

Таблица 1. **Способы применения компьютера при переводе**

	Уровень терминов	Уровень сегментов
До первода	Выделение терминов Анализ терминологии	Сегментация текста
Во время перевода	Автоматический поиск терминологии	Поиск языковых пар в памяти переводов Машинный перевод
После перевода	Проверка соответствия терминологии	Проверка целостности сегментов, формата и грамматики

Выделение терминов и анализ терминологии.

На этом этапе производится исследование текста с целью выяснения, какие слова или словосочетания могут быть взяты в качестве терминов. После того, как определен термин на исходном языке, осуществляется анализ терминологии на предмет того, какой термин на целевом языке следует выбрать для обозначения нужного концепта. Например, если в исходном тексте встретилось словосочетание "операционная система" то программа должна

проанализировать его в качестве возможного термина, даже если в системе уже определены термины "операционный" и "система".

Автоматический поиск терминологии

Данный процесс может быть сравнен с машинным переводом на уровне отдельных терминов. Суть его заключается в том, что в процессе работы над текстом переводчик имеет возможность видеть варианты перевода для каждого термина, и быстро вставлять нужный перевод в текст на целевом языке, не рискуя допустить опечатку.

Проверка соответствия терминологии

После того, как перевод выполнен, компьютер осуществляет проверку того, что все вхождения каждого из терминов были переведены одинаково. Например, если термин "операционная система" был заменен при своем первом вхождении на "operating system", а при втором вхождении на "operational system", то должно быть выдано соответствующее предупреждении о нарушении единства терминологии.

Сегментация текста

Разбиение текста на сегменты является важным подготовительным этапом для полной или частичной автоматизации перевода. Сегменты должны по возможности содержать фрагменты текста, грамматически независимые друг от друга. Иными словами, должна быть обеспечена возможность корректного перевода каждого сегмента независимо от других. Обычно разбиение на сегменты выполняется по знакам пунктуации.

Поиск языковых пар в памяти переводов

Автоматическая память переводов, или просто память переводов (TranslationMemory), подразумевает, в первую очередь, просмотр ранее переведенных текстов. Она сравнивает переводимый в текущий момент текст с тем, что хранится в базе, "вспоминает" сегменты, которые изменились незначительно, и предлагает использовать их перевод повторно. Разумеется, критерии сходства сегментов могут быть различны, и они играют очень важную роль в расширении возможностей памяти переводов.

Машинный перевод

Данный способ перевода заключается в алгоритмической обработке исходного текста, в ходе которой происходит разбор сегментов, выделяются отдельные термины и отношения между ними, после чего осуществляется замена всех терминов на соответствующие термины целевого языка в нужной форме и взаиморасположении. Машинный перевод (MachineTranslation) применим только в очень узком контексте и требует значительного постредактирования переведенного текста.

Проверка целостности сегментов, формата и грамматики

Данные действия выполняются по окончании перевода и имеют своей целью проверить, все ли сегменты остались на своих местах, сохранилась ли форматирующая информация, и корректен ли результирующий текст с точки зрения грамматики целевого языка.

Среди перечисленных технологий наибольший интерес представляют терминологические словари и память переводов, поскольку именно от их эффективности зависит скорость и качество перевода. Технология построения терминологических словарей достаточно хорошо проработана и основана на принципах, аналогичных тем, что применяются в обычных двуязычных словарях. Разбиение текста на термины обычно осуществляется по пробелам с дополнительным привлечением некоторого морфологического анализа.

Сложнее обстоит дело с организацией памяти переводов. Наряду с тривиальной задачей поиска языковой пары, включающей сегмент, идентичный заданному, память переводов должна обеспечивать возможность поиска сегментов, похожих на данный по некоторому критерию. Таким образом, центральной проблемой классической памяти переводов является построение анализатора таких "нечетких совпадений" (fuzzy matches), характеристики которого и определяют преимущества и недостатки каждой конкретной системы профессионального перевода.

3. Аспекты использования памяти переводов.

Как следует из вышеизложенного, основой функционирования любой системы памяти переводов являются ранее переведенные тексты. Множество этих текстов постоянно пополняется новыми переводами, вследствие чего, процент автоматически переводимых сегментов, постепенно растет. Это означает, что для наиболее эффективного использования памяти переводов, все тексты должны содержать достаточное количество похожих фраз. Такое положение вещей имеет место в документации на различного рода продукты. Это обусловлено двумя факторами. Во-первых, документацию принято составлять максимально простым языком, лаконично и в строгих терминах. Во- вторых, с появлением новых версий и модификаций поставляемого потребителям продукта содержание документации меняется лишь в незначительной степени. Память переводов, в подобных случаях, избавляет переводчика от необходимости по несколько раз переводить идентичные фрагменты текста, входящие в разные документы.
В то же время, использование памяти переводов требует от переводчика специальной подготовки, а также наличия соответствующего аппаратного и программного обеспечения. Другим негативным фактором является то, что для обеспечения ожидаемого эффекта все переводы должны быть сделаны в одной и той же среде, либо в средах, совместимых по формату представления данных.
Наконец, полезный эффект памяти переводов проявляется с заметной отсрочкой во времени, требуя поначалу дополнительных капиталовложений.

Резюмируя вышесказанное, можно выделить три условия применимости рассматриваемой технологии:
1) большой объем перевода;
2) однотипность переводимых текстов;
3) готовность к отсроченному возврату капиталовложений.

Основные принципы работы памяти переводов

Память переводов представляет собой базу данных, хранящую языковые пары, и определенный механизм поиска. Несмотря на то, что различные профессиональные среды перевода, такие как "Translator'sWorkbench" фирмы Trados, "Transit" фирмы Star, "DejaVu" фирмы Atril, имеют, по-видимому, различную реализацию этого механизма ("по-видимому", поскольку алгоритмы не придаются огласке), общая идея становится ясной после изучения примеров. Поэтому с примеров и начнем.
Пусть в исходном тексте встречаются следующие фразы:

"Температура регулируется поворотом ручки."
"Температура регулируется поворотом ручки по часовой стрелке."
"Напор воды регулируется поворотом ручки по часовой стрелке."

Если сегментация выполняется по предложениям, то каждая из приведенных фраз попадет в отдельный сегмент. Пусть первый сегмент был переведен человеком следующим образом:

"The temperature can be adjusted by turning the knob."

Языковая пара, состоящая из исходного и переведенного сегментов, заносится в память переводов. Когда переводчик доходит до второй фразы примера, система определяет сходство и выводит на экран следующую информацию: таблица 2.

Таблица 2. **Результат определения сходства**

Текущий сегмент	Температура регулируется поворотом ручки *по часовой стрелке*
Найденный сегмент	Температура регулируется поворотом ручки
Перевод	The temperature can be adjusted by turning the knob
Степень сходства	70%

Теперь переводчик имеет возможность частично воспользоваться уже сделанным переводом, учтя различия: "The temperature can be adjusted by turning the knob clockwise."
После того, как сегмент, соответствующий второй фразе примера помечается как переведенный, в памяти переводов появляется еще одна языковая пара. Тем самым, когда дело доходит по третьей фразы, система уже имеет возможность показать переводчику два похожих варианта: таблица 3.

Таблица 3. **Третья фаза переводов**

Текущий сегмент	*Напор воды* регулируется поворотом ручки по часовой стрелке
Найденная языковая пара 1	***Температура*** регулируется поворотом ручки по часовой стрелке
	The temperature can be adjusted by turning the knob clockwise
Степень сходства	~65%
Текущий сегмент	*Напор воды* регулируется поворотом ручки *по часовой стрелке*
Найденная языковая пара 2	***Температура*** регулируется поворотом ручки
	The temperature can be adjusted by turning the knob
Степень сходства	~40%

Воспользовавшись, к примеру, первым из предложенных вариантов, переводчик быстро расправляется с оставшейся частью фразы:
"The water head can be adjusted by turning the knob clockwise."

Эффективность работы памяти переводов во многом определяется тем, насколько удачно решены следующие задачи:
1) сегментация;
2) обработка специальных символов и форматирующей информации.

Очевидно, что с увеличением размера сегментов будет уменьшаться число полных совпадений (и увеличиваться число частичных), что сильно повысит ресурсоемкость процедур поиска и потребует от переводчика значительных усилий в изучение предоставленных ему в качестве вариантов перевода языковых пар. С другой стороны, уменьшение размера сегментов сделает их малопригодными для повторного использования, поскольку сильно возрастет влияние контекста на перевод. Оптимальной единицей сегментации чаще всего оказывается фрагмент предложения, ограниченный знаками препинания. Во избежание ошибочной сегментации по точкам внутри аббревиатур и других подобных случаев используют регулярные выражения и списки исключений.

Вторая проблема обусловлена тем, что в тексте кроме букв зачастую присутствуют иные символы, как то: маркеры внедренных в документ объектов, закладки, перекрестные ссылки, переключатели свойств шрифта. Все эти инородные элементы в ряде случаев могут повлиять на перевод. Например, выделенное курсивом слово может при переводе быть взято в кавычки и попасть в результирующий текст в неизменном виде. Для управления поведением анализатора в таких ситуациях во многих программных продуктах предусмотрены специальные настройки, в том числе, основанные на применении регулярных выражений.

Пути расширения возможностей памяти переводов

Поскольку функцией памяти переводов является поиск в базе данных переведенных фрагментов заданного сегмента, то пределом ее возможностей является, очевидно, выборка, максимально покрывающая исходный сегмент и не содержащая никакой избыточной (лишней) информации.
Попытаемся выделить возможные варианты повышения качества памяти переводов, воспользовавшись приведенным ранее примером. Выберем и рассмотрим две языковые пары: таблица 4.

Таблица 4. **Языковые пары 1–2**

Языковая пара 1	Температура регулируется поворотом ручки по часовой стрелке
	The temperature can be adjusted by turning the knob clockwise
Языковая пара 2	Напор воды регулируется поворотом ручки по часовой стрелке
	The water head can be adjusted by turning the knob clockwise

Сходство сегментов на исходном языке позволяет сделать предположение, что их переводы, то есть сегменты на целевом языке также должны быть похожи. Коль скоро это так, что возникает резонное желание выделить из двух приведенных языковых пар общую часть и представить ее в виде новой языковой пары. Выполнив несложную операцию пересечения строк, получаем следующий результат: таблица 5.

Таблица 5. **Языковая пара 3 – результат пересечения**

Языковая пара 3	регулируется поворотом ручки по часовой стрелке
	can be adjusted by turning the knob clockwise

Теперь для любого сегмента, включающего фрагмент " регулируется поворотом ручки по часовой стрелке", может быть выбрана языковая пара номер 3, содержащая только необходимый перевод для фрагмента.
Однако, не всегда все так хорошо. Чуть более внимательный взгляд на этот пример сразу же заставит нас признать, что создание таких "укороченных" языковых пар эквивалентно уменьшению размера сегмента, а мы помним, чем это грозит. Маленький фрагмент текста, в особенности, если он не ограничен никакими знаками препинания, едва ли может быть правильно переведен без учета контекста. Следовательно, при выделении общих частей в двух используемых уже языковых парах необходимо руководствоваться теми же принципами, что и при начальной сегментации исходного текста.

К тому же, не стоит забывать, что пересечение сегментов на исходном языке не обязательно изоморфно пересечению сегментов на целевом языке. Это связано с различиями правил грамматики в разных языках, порядка слов, соответствия слов понятиям. Поэтому осмысленное значение целевого сегмента языковой пары, образованной пересечением, можно ожидать только при:

1) значительном размере обоих сегментов вновь образованной языковой пары;
2) эвристически определенном изоморфизме пересечения сегментов на исходном и целевом языке (например, если пересечение осуществлено по знакам пунктуации);
3) морфологическом и синтаксическом анализе результата пересечения с привлечением технологии машинного перевода.

Еще одной немаловажной задачей при реализации механизма описанных манипуляций с языковыми парами является создание некоторого формализма, позволяющего однозначно определить, какие именно пары должны подвергаться обработке, как именно должен формироваться результат, как должен осуществляться поиск сегмента и каковы критерии сравнения сегментов при поиске. Этому посвящена следующая тема.

II.Многоуровневая модель памяти переводов

План

1. Представление данных.
2. Поиск и добавление языковых пар.
3. Вычисление пересечения языковых пар.

1. Представление данных

Структура реально используемых ныне реализаций памяти перевода является одноуровневой и подразумевает наличие упорядоченного списка языковых пар. С введением механизма выделения в парах общих частей подобная организация данных окажется неудобной. Действительно, для любых двух пересекающихся языковых пар в базе будет создаваться дополнительный элемент, содержание которого будет полностью дублироваться в обеих языковых парах. От избыточности удастся избавиться (Способ 1), если удалить вынесенную

общую часть из обеих пар, а на ее место поставить ссылку на вновь созданную языковую пару (рис. 8).

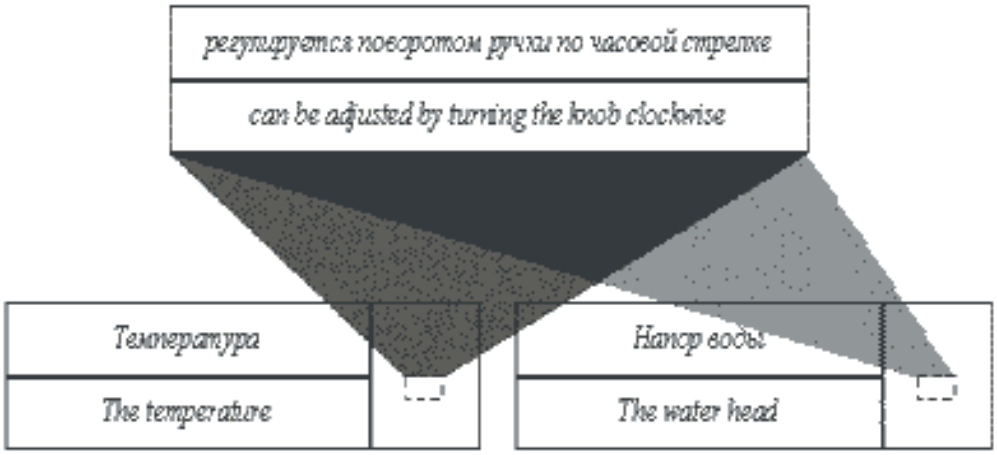

Рисунок 8. **Избавляемся от избыточности – Способ 1**

Повторяя данную процедуру каждый раз, когда обнаруживается очередное пересечение, мы, в конечном счете, получим направленный граф, узлами которого являются языковые пары, а дугами- отношения включения.

Еще одной оптимизацией (Способ 2) является разделение исходного и целевого сегментов каждой языковой пары. Это имеет смысл, поскольку 1) нередки случаи, когда одна и та же исходная фраза переводится на целевой язык по-разному, что порождает две языковые пары с одинаковым первым элементом; 2) пересечения исходных и целевых сегментов не всегда изоморфны, и их результаты не обязательно образуют языковую пару.

Поэтому разделим языковые пары, и получим два ориентированных графа, "синхронизированных" друг относительно друга (рис. 9). При этом отношения, связывающие вершины разных графов, могут быть различной местности, но обязательно обладают свойством симметричности.

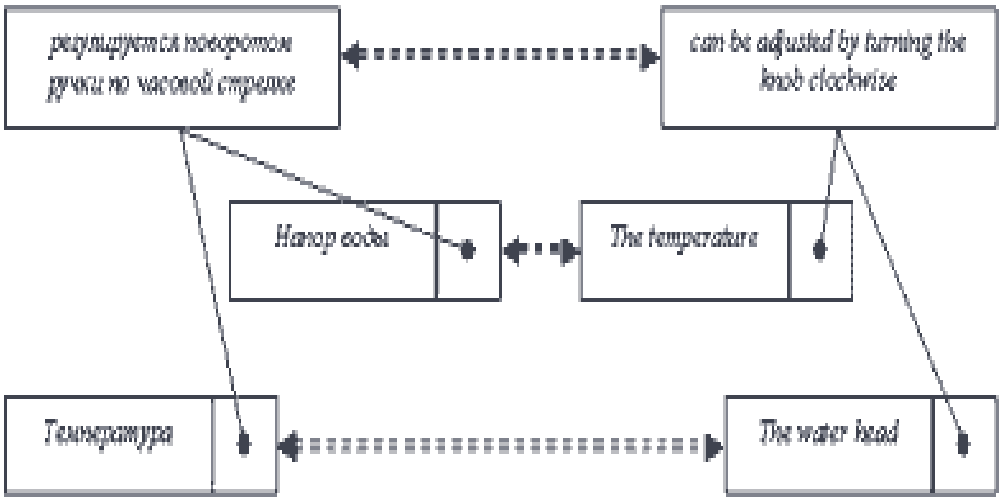

Рисунок 9. **Избавляемся от избыточности – Способ 2**

Развивая идею дальше, вспомним, что в нашем распоряжении уже имеется аппарат, позволяющий формализовать представленную схему. Это — объектно-ориентированный

подход. В самом деле, каждому сегменту в базе может быть сопоставлен отдельный класс, отношение включения по своим свойствам идентично отношению наследования, а горизонтальные связи, формирующие языковые пары могут быть заданы механизмом ассоциирования, либо введением в класс метода Translate ("перевести").

Тем не менее, предложенная модель пока что не описывает внутреннюю структуру сегментов, которую необходимо анализировать при создании новой языковой пары на основе пересечения существующих. Для решения этой задачи разобьем все сегменты на отдельные слова (по пробелам). Теперь каждый сегмент может быть представлен классом, являющимся производным от всех слов, образующих текст сегмента. При этом совершенно необязательно

иметь перевод для каждого слова, поскольку это будет отражено лишь отсутствием соответствующей связи между узлами графов, а метод Translate будет возвращать "нулевое" значение.

Памятуя о том, что в состав среды перевода помимо памяти переводов входит также терминологический словарь, деление сегментов можно осуществлять не по словам, а по терминам. Наличие терминологического словаря дает и еще одну возможность. Для каждого вхождения термина в сегмент можно определить его начальную форму, то есть ту, в которой он входит в базу словаря. Эта начальная форма послужит как бы абстрактным базовым классом для каждого вхождения термина, а конкретное вхождение будет содержать определение значений заданных в базовом классе атрибутов: род, число, падеж и т. п.

Последнее нововведение подталкивает нас к мысли о том, терминологический словарь можно слить воедино с памятью переводов, представив, тем самым, все ресурсы переводчика в виде универсальной модели (рис. 10).

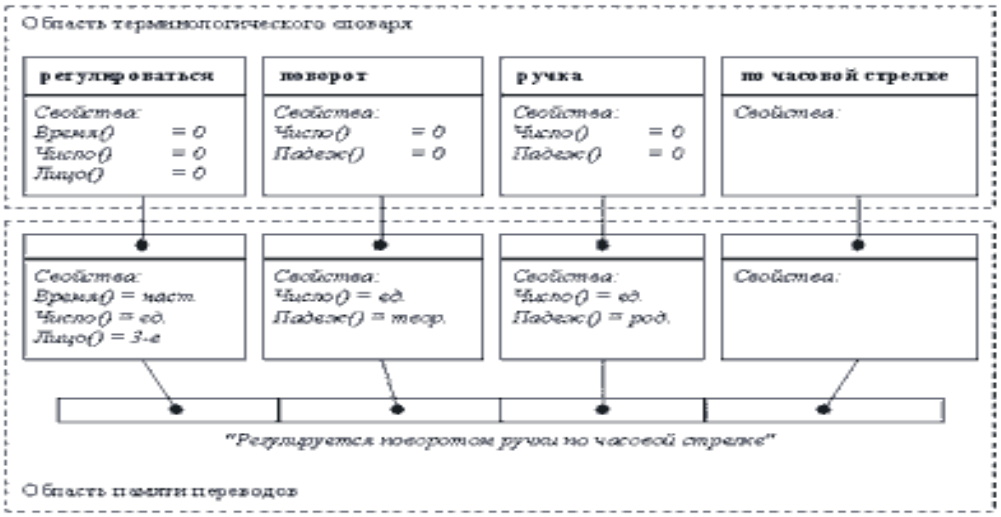

Рисунок 10. **Универсальная модель ресурсов переводчика**

2. Поиск и добавление

Пока память переводов была линейной, сегменты неделимыми, а сравнение строгим, решение задачи поиска сводилось к введению отношения строгого лексикографического порядка над множеством сегментов на исходном языке, т. е. определялся оператор "меньше", на основе которого можно было осуществить обыкновенный двоичный поиск, и проверку на равенство. С введением оператора "нечеткого совпадения", позволяющего оценить степень сходства для любых двух сегментов, решение проблемы поиска резко усложнилось и, без дополнительных ухищрений с различного рода индексацией, стало эквивалентно задаче полного перебора. Предложенная многоуровневая модель памяти переводов предоставляет некоторый механизм неявной индексации: каждое входящее в сегмент слово идентифицирует не-

которое подмножество ориентированного графа памяти переводов, состоящее из узлов, которые можно достичь, начав обход от узла, соответствующего выбранному слову.

Используя особенности выбранной структуры памяти переводов, задачу поиска сегментов, похожих на заданный, можно решить путем выполнения следующих действий (рис. 11):

1) разбить заданный сегмент на слова;

2) найти в памяти переводов все узлы, соответствующие этим словам;

3) спускаясь по графу отношений наследования, помещать в список найденных сегментов все встречаемые узлы.

Резонным представляется вопрос о том, в каком порядке следует предоставлять найденные сегменты переводчику: ведь приведенная процедура поиска выберет из памяти все сегменты, пересекающиеся с заданным по крайней мере по одному слову. Каковы правила фильтрации и сортировки найденных сегментов?

Ответ на этот вопрос лежит за пределами выбранного формализма, однако в этом нет ничего страшного. Дело в том, что результат поиска представляет собой классический вариант одноуровневой памяти переводов, анализ которого может быть произведена методами, формализованными в рамках существующих сред перевода. Для обеспечения эффективности поиска целесообразно осуществлять оценку "пригодности" сегментов по мере их нахождения. Например, если некоторый сегмент полностью совпадает с эталоном, то все его потомки в графе могут быть автоматически исключены из поиска.

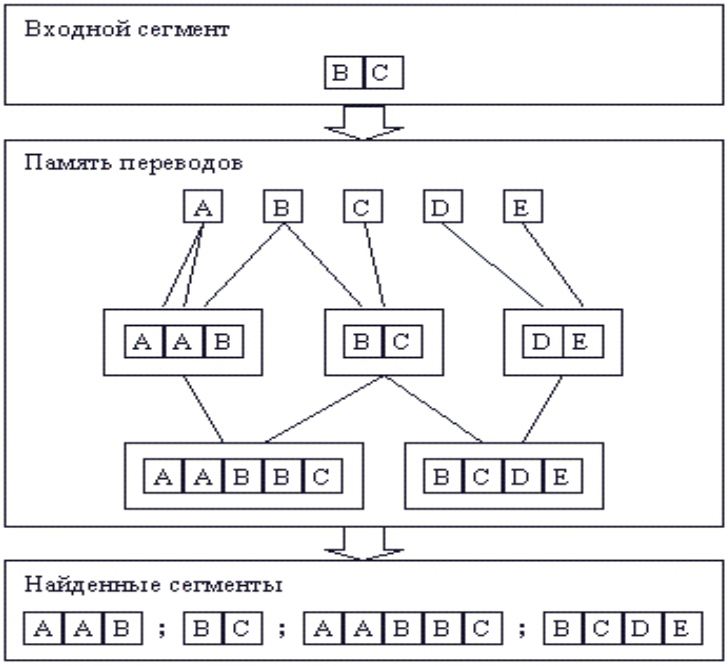

Рисунок 11. Действия при поиске сегментов

Теперь поговорим о задаче добавления нового сегмента в память переводов. Очевидным условием корректности процедуры добавления является обеспечение успешного поиска. Стало быть, добавляемый сегмент должен иметь в числе своих предков (не обязательно прямых) все составляющие его слова. Следуя целям оптимальности, можно заключить, что среди предков должны присутствовать также узлы графа, содержащие фрагменты данного сегмента. Иными словами, если в памяти переводов присутствуют сегменты "AB" и "CD", то сегмент "ABCD" должен стать наследником этих двух сегментов. Аналогично, если в памяти присутствует сегмент "ABCD", то добавляемый сегмент "AB" должен стать его предком. В общем случае при добавлении сегмента в граф памяти

переводов могут существовать альтернативные варианты наследования. В такой ситуации схема добавления заметно усложнится. В любом случае, проблема построения оптимальной иерархии классов решается в рамках объектно-ориентированного подхода, поэтому мы не будем заострять здесь на ней внимание.

3. Вычисление пересечения языковых пар

Поскольку выделение общей части двух сегментов- важный этап предлагаемой технологии перевода, изучим этот вопрос более детально. При этом, помня о том, что сегмент в памяти перевода выступает, чаще всего, не отдельно, а как элемент языковой пары, будем рассматривать именно пересечение пар.

Для начала дадим определения пересечения сегментов. Итак, пересечение сегментов A и B- это множество сегментов Ci, таких что:

1) каждый из Ci содержится и в A, и в B;

2) никакие два Ci не содержат одинаковых фрагментов;

3) не существует такого сегмента D, что и A, и B содержат D, и D содержит один из сегментов Ci.

Приведенное определение не подразумевает выделения из сегментов A и B всех общих фрагментов. Это сделано для того, чтобы можно было использовать алгоритмы различной сложности реализации пересечения.

Теперь перейдем к пересечению языковых пар. Как уже было упомянуто выше, очень важно определить, является ли пересечение изоморфным, иными словами, можно ли считать результаты пересечения исходных и целевых сегментов языковой парой. Два примера иллюстрируют это (рис. 12). В первом случае пару сегментов "достаточно высока" и "is high enough" имеет смысл поместить в память переводов, поскольку она действительно представляет собой вариант перевода, который может быть повторно использован переводчиком. Во втором случае- это совершенно очевидно- сегменту "достаточно" не следует сопоставлять сегмент "is high enough", поскольку данная языковая пара будет некорректной.

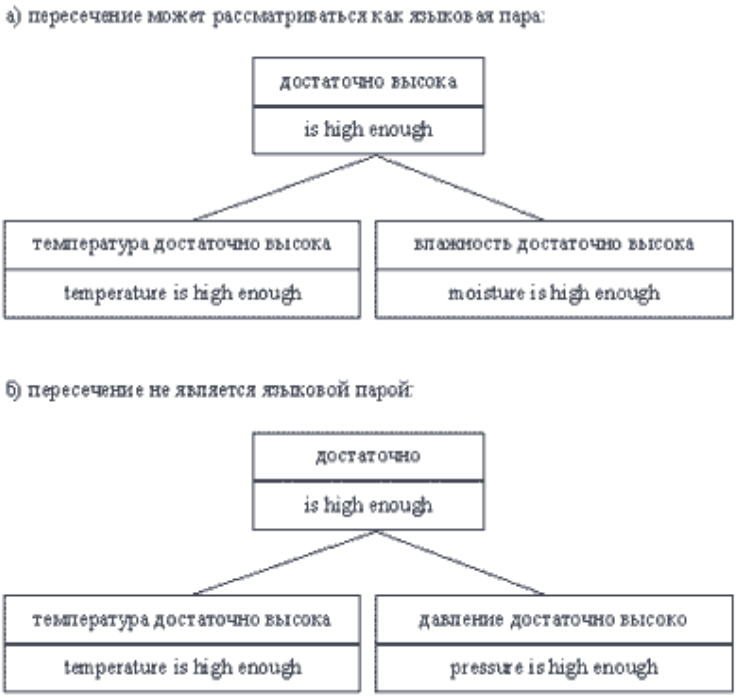

Рисунок 12. **Примеры пересечения языковых пар**

Для проверки изоморфизма пересечений можно использовать подход, основанный на технологии машинного перевода. Его суть в сопоставлении терминов, образующих исходный и целевой сегменты. Для этого необходимо произвести грамматический разбор сегментов с целью выделения терминов и синтаксических связей между ними. После этого можно воспользоваться терминологическим словарем для определения того, какому термину в целевом сегменте соответствует заданный термин в исходном сегменте. Иными словами, изоморфизм можно определить по следующему критерию (рис. 13): пересечение является изоморфным, если всем терминам его исходного сегмента, сопоставлены термины его целевого сегмента, и синтаксические связи между ними идентичны тем, которые присутствуют в сегментах, из которых было получено пересечение.

В общем случае, для оценки изоморфизма можно проверять не только отдельные термины (суть корневые узлы графа памяти переводов), но и родительские сегменты всех уровней. Это повысит надежность оценки, снизив риск неправильного определения синтаксических связей.

Следует обратить внимание на тот факт, что в предлагаемой модели машинный перевод используется только для грамматического анализа текста, образующего сегмент. Слабым местом систем машинного перевода является выбор перевода для терминов сегмента, и именно эта задача решается более надежным способом- с помощью памяти переводов.

Рисунок 13. Критерии определения изоморфизма

От языковых пар к языковым звездам

Нередкой является ситуация, когда перевод приходится осуществлять не только с языка А на язык В, но и, наоборот, с языка В на язык А. Одна и та же память переводов будет одинаково полезна в обоих случаях, поскольку содержит максимально синхронизированные графы сегментов на языке А и на языке В. Однако стоит нам усложнить задачу и

предположить необходимость перевода между несколькими языками, как полезность единой памяти переводов заметно падает. Действительно, если перевод осуществлялся с языка A на языки B и C, то в памяти не будет храниться соответствия между сегментами на языках B и C. Как же обеспечить подобную возможность?

Разумным решением могло бы явиться использование некоторого промежуточного языка X, на который осуществлялся бы перевод, а затем, вторым этапом, выполнялся бы перевод с языка X на целевой язык. В подобном случае все языковые пары в памяти переводов состояли бы из сегмента языка X и сегмента одного из целевых (либо исходных) языков. Тут имеются, однако, подводные камни. Во-первых, как мы уже убедились, пересечение языковых пар не всегда бывает изоморфным, следовательно, не все языковые пары в памяти переводов будут содержать перевод на язык X. Очевидно, такие пары будут бесполезны. Во-вторых, при переводе всегда имеется опасность потери смысла: двойной перевод значительно увеличивает эту опасность.

Каким же должен быть этот гипотетический промежуточный язык X, чтобы им было целесообразно воспользоваться? Его свойства вытекают из двух названных проблем. Во-первых, этот язык должен обеспечивать изоморфное пересечение для любого другого языка. Нарушение изоморфизма (по крайней мере, в родственных языках) обусловлено в значительной степени различием синтаксических правил, приводящим к разному порядку членов предложения, а также к различию форм одного и того же слова. Отсюда следует, что язык X должен быть инвариантен к порядку слов и как-то учитывать их формы в исходном языке. Во-вторых, он должен быть в состоянии передать смысл фразы на любом языке, следовательно- включать в себя специфические понятия всех существующих человеческих языков.

Если такой универсальный язык будет найден, то память переводов можно будет организовать не на основе языковых пар, а на основе языковых звезд, где в центре находится сегмент на языке X, на лучах- варианты переводов его на другие языки. При значительном объеме перевода между большим количеством языков дополнительные затраты на удвоенную работу переводчика с лихвой окупятся гибким механизмом памяти переводов, значительно упрощающим многоязычный перевод.

Осталось только найти язык X. И такой язык существует! Это универсальный сетевой язык UNL (Universal Networking Language), предложенный Институтом Развития Обучения (Institute of Advanced Studies – IAS) при Университете Объединенных Наций (United Nations University – UNU). Им мы и воспользуемся для дальнейшего развития модели памяти переводов.

ТЕМА 4. АВТОМАТИЧЕСКИЙ АНАЛИЗ И СИНТЕЗ РЕЧИ

Синтез звучащей речи представляет собой преобразование печатного текста в цифровой форме в звуковой текст на естественном языке.

Автоматический синтез (генерация) может осуществляться разными способами:

1) параметрический синтез;

2) компилятивный (=конкатенативный) синтез:

- микросегментный (микроволновый);

- аллофонный;
- дифонный;
- полуслоговой;
- слоговой;
- синтез из единиц произвольного размера.

3) полный синтез речи по правилам

- формантный синтез по правилам;

- артикуляционный синтез по правилам.

Параметрический синтез работает только на заранее заданных текстах и применим, когда набор сообщений ограничен. В ходе данного вида синтеза звуковой сигнал представлен определённым числом непрерывно изменяющихся параметров. Для формирования гласных звуков используется генератор тонального сигнала, для согласных – генератор шума.

Компилятивный синтез основывается на составлении текстов из заранее записанного "словаря" элементов. Размер элемента системы может быть разным: аллофон, дифон, слог и более крупные единицы. Если единицей является слово, то обычно запас элементов ограничивается несколькими сотнями слов, а содержание синтезируемых текстов – объёмом словаря.

Полный синтез речи по правилам может воспроизводить речь по заранее неизвестному тексту. Этот метод не использует элементов человеческой речи, а базируется на запрограммированных лингвистических и акустических алгоритмах. ***Формантный*** синтез базируется на формантах – частотных резонансах речевой акустической системы. Алгоритм формантного синтеза моделирует работу речевого тракта человека, работающего как набор резонаторов. Это универсальная и перспективная технология, однако в настоящее время, большинство синтезаторов, в основе работы которых лежит только формантный синтез, понять без подготовки сложно. ***Артикуляторный*** метод пытается доработать недостатки формантного путем добавления в модель фонетических особенностей произнесения отдельных звуков.

Чаще всего используются аллофонный и дифонный методы. Для дифонного синтеза речи базовым элементом являются «половинки» рядом стоящих фонов, а для аллофонного — фоны при учете особенностей левого и правого контекстов. При этом различные типы контекстов объединяются в классы по степени акустической близости. Примерами могут быть 1) придыхательный [k^h] после нейтрального гласного перед ударным открытым гласным заднего или центрального ряда [ɔ], [ɑ] и 2) придыхательный [k^h] после нейтрального гласного перед ударным закрытым или среднезакрытым гласным переднего ряда [i], [ɪ], [e]

Преимущество таких систем состоит в том, что они дают возможность синтезировать текст по не заданному заранее тексту. Их недостаток состоит в том, что качество синтезированной речи зачастую ниже качества естественной речи, поскольку на границах сшивки элементов могут возникать искажения. Проблема решаема при соблюдении правил сегментации – ставить границу между гласным и согласным по нулю перед пиком на осциллограмме и единообразно отделять согласные друг от друга и гласные друг от друга. Полезным оказывается четкое разделение типов контекста согласно особенностям взаимодействия ряда гласного с активным органом согласного и подъема гласного со способом образования согласного (смычка или щель). Также весьма трудно управлять интонационными характеристиками речи, так как характеристики отдельных слов могут изменяться в зависимости от контекста или типа фразы.

На практике, на современном этапе развития, несмотря на активное продвижение в этой области, разработчики технологии синтеза речи всё-таки испытывают некоторые трудности, в основном связанные с искусственностью синтезируемой речи, отсутствием в ней эмоциональной окраски и с низкой помехоустойчивостью.

Дело в том, что любая синтезированная речь, как правило, воспринимается человеком с трудом. Это связано с тем, что пробелы в синтезированном тексте заполняет человеческий мозг, который задействует для этого дополнительные ресурсы, и человек может нормально воспринимать синтезированную речь только около 20 минут.

На восприятие речи также влияет её эмоциональная окраска. В случае с синтезированной речью она отсутствует. Хотя стоит отметить, что некоторые алгоритмы всё же позволяют в некоторой степени имитировать эмоциональную окраску речи путём изменения длительности фонем, пауз и модуляции тембра, но пока их работа далека от идеала.

Что касается третьей названной проблемы - низкой помехоустойчивости, то эксперименты показывают, что восприятию синтезированного текста мешают любые, даже самые небольшие посторонние шумы. Это опять-таки связано с тем, что для обработки синтезированной речи человеческий мозг задействует дополнительные центры, которые не используются при восприятии речи естественной.

Синтезированная речь может применяться в различных областях, среди которых:

1) информационно-справочные системы, которые могут использоваться в том числе для помощи слепым и немым;

2) при объявлениях об отправлении/прибытии поездов/автобусов, в аэропортах, самолетах и т. п.;

3) для выдачи информации о технологических процессах: в военной и авиакосмической технике, в робототехнике, в акустическом диалоге человека с компьютером.