Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Ю. Б. Крапивин

ЕСТЕСТВЕННО-ЯЗЫКОВОЙ ИНТЕРФЕЙС ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-40 03 01 «Искусственный интеллект»

Минск БГУИР 2023

УДК 004.512:004.89(076.5)
ББК 32.813я73
К78

Р е ц е н з е н т ы:

кафедра интеллектуальных информационных технологий учреждения образования «Брестский государственный технический университет» (протокол № 3 от 16.11.2021);

заведующий кафедрой интеллектуальных систем Белорусского государственного университета кандидат физико-математических наук, доцент Е. И. Козлова

Крапивин, Ю. Б.

К78 Естественно-языковой интерфейс интеллектуальных систем. Лабораторный практикум : пособие / Ю. Б. Крапивин. – Минск : БГУИР, 2023. – 64 с. : ил.

ISBN 978-985-543-680-6.

Сформулированы основные положения, касающиеся теории и лабораторного практикума по дисциплине «Естественно-языковой интерфейс интеллектуальных систем», приведен подробный теоретический материал по курсу, даются рекомендации по выполнению лабораторных работ.

УДК 004.512:004.89(076.5)
ББК 32.813я73

ISBN 978-985-543-680-6

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Как показывает современная история развития научного познания мира, наиболее интересные и значимые результаты достигаются в научно-исследовательских предприятиях, объединяющих отдельные дисциплины в рамках некоторой глобальной научной проблемы. Такое объединение имело место, например, в 50-х и 60-х гг. XX в., когда возникло новое научное направление – искусственный интеллект (ИИ), сосредоточившее усилия математиков, психологов, специалистов в области робототехники, электроники с тем, чтобы «научить» электронно-вычислительные машины в определенном смысле думать и вести себя подобно человеку (естественному интеллекту).

Обработка естественного языка – одно из наиболее динамично развивающихся направлений искусственного интеллекта в настоящее время. Его теоретическую основу составляет компьютерная лингвистика (машинная лингвистика, автоматическая обработка текста) – область знаний, связанная с решением задач автоматической обработки информации, представленной на естественном языке (ЕЯ).

Учебная дисциплина «Естественно-языковой интерфейс интеллектуальных систем» предназначена для студентов, обучающихся по специальности «Искусственный интеллект», и имеет выраженный прикладной характер. Изучение дисциплины позволяет сформировать представление об основах компьютерной лингвистики, ее роли в решении основных задач автоматической обработки текста естественного языка, подходах к их решению, а также актуальных способах организации диалога «человек – компьютер» при проектировании и разработке интеллектуальных систем, активно развивающихся и использующихся в настоящее время.

В результате освоения курса «Естественно-языковой интерфейс интеллектуальных систем» студент должен:

знать:

- основные понятия и направления компьютерной лингвистики;
- основные принципы организации диалога «человек – компьютер»;
- лингвистические модели, применяемые для реализации естественно-языкового интерфейса;
- структуру речевого сигнала, методы его анализа и синтеза;
- методы синтеза речи по тексту, общую структуру и принципы работы синтезатора речи;
- основные задачи и методы автоматического распознавания речи, принципы работы современных компьютерных систем распознавания речи;

уметь:

- охарактеризовать цели и задачи разработки естественно-языкового интерфейса в интеллектуальных системах;

- анализировать способы описания морфологических, синтаксических и семантических моделей естественного языка и методики их применения для построения естественно-языкового интерфейса;
- анализировать современный уровень и перспективы развития естественно-языкового интерфейса электронных вычислительных систем;

приобрести навыки:

- морфологического анализа естественного языка;
- синтаксического анализа естественного языка;
- семантического анализа естественного языка;
- построения естественно-языкового интерфейса в интеллектуальных системах;
- использования лингвистических моделей для построения систем естественно-языкового общения.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1. Введение в компьютерную лингвистику

Уровень исследований в области искусственного интеллекта и компьютерной лингвистики, по сути, определяет современный уровень развития такой важнейшей научно-технической отрасли, как информатика. Ключевым объектом исследований указанных научных направлений является естественный язык, рассматриваемый как универсальное средство описания действительности в виде текста и коммуникации с вычислительной системой с помощью ЕЯ-интерфейса. Язык – это знаковая система, предназначенная для порождения, передачи и хранения информации. Информация, передаваемая языковыми средствами, всегда воплощается в некотором тексте, поэтому передача информации есть создание, или порождение, текста, с одной стороны, и восприятие, «прием» текста – с другой [1]. Текст здесь понимается в широком смысле этого слова как множество текстовых документов (книг, статей, патентов, отчетов и т. п.), представленных в электронной форме.

Сам язык выполняет следующие важные функции:

- информационную – используется с целью представления, хранения и передачи информации;
- коммуникативную – связана с потребностью человека в общении, в осуществлении связей (коммуникации) между людьми;
- воспитательную – заключается в том, что человек, имея возможность читать произведения великих мыслителей как в художественной литературе, так и в философской, научной и т. п., подвергается одновременно процессу воспитания;
- развивающую – освоение языка связано с развитием интеллектуальных способностей;
- поэтическую (эстетическую) – позволяет носителю языка реализовать себя в эмоциональной сфере, в сфере чувств и настроений: обеспечивается возможность получать эмоции, которые зафиксированы с помощью языка в поэтических, художественных и других литературных произведениях, и делиться ими [2].

Лингвистика (языкознание) изучает внутреннее строение и функционирование языка, рассматривая их в рамках своих отдельных научных направлений. Так, лексикологию интересуют словарный состав языка и закономерности его развития. Грамматика, разделяясь на морфологию и синтаксис, изучает способы и средства построения и изменения слова и предложения. Семантика – наука о смысле (значениях) слов и предложений, частей речи и членов предложений. Прагматика – учение об условиях и целях коммуникации, влияющих на понимание, так называемое изучение языка в контексте, изучение отношений между средствами языка и теми, кто этими средствами пользуется. Дискурс изучает и анализирует характерные для данного вида текстов (или коммуникантов) особенности лексики, синтаксиса, семантики либо прагматики языко-

вых единиц, проявляющиеся в актуальных коммуникативных актах, речи и письменных текстах (например, спортивный дискурс) [3, 4].

В [5, с. 12–13] отмечается, что одной из наиболее характерных особенностей ЕЯ, в отличие от формальных (искусственных) языков, является неоднородность его правил: с одной стороны, в каждом ЕЯ имеется небольшое число весьма общих правил, охватывающих большое число единиц, которые часто встречаются в текстах, а с другой стороны, в ЕЯ имеется большое число частных правил, каждое из которых охватывает небольшое число единиц, относительно редко встречающихся в текстах [6]. Понятно, что соответствующие модели будут работать тем успешнее, чем более однородны правила, что явно противоречит самой природе языка. Отступление от упорядоченности, регулярности, единообразия в строении и функционировании языковых единиц отражает одну из основных особенностей строения и функционирования естественного языка [7]. Как показывает анализ, разрешение этого противоречия может быть достигнуто путем разбиения класса языковых явлений соответствующего уровня на ряд относительно однородных подклассов, каждый из которых описывается единообразно устроенным множеством правил, т. е. имеет место так называемый принцип эшелонирования правил [8].

Второе важное характеристическое свойство ЕЯ состоит в «недостаточности» его формализации, т. е. в том, что никакое множество правил, относящееся к данному уровню глубины языка, каким бы полным оно ни было, не является достаточным для того, чтобы получить единственную, в общем случае правильную, структуру соответствующего уровня [9]. Необходимый здесь компенсаторный механизм обеспечивается с помощью вероятностных правил, позволяющих выбрать в множестве всех допустимых альтернатив какую-то одну в качестве наиболее подходящей (конечно, с учетом того, что в принципе возможна и менее вероятная альтернатива).

С указанным связано еще одно важное свойство ЕЯ – свойство неабсолютности его правил (прагматических, семантических и даже синтаксических), которые в реальном языке могут нарушаться. Это означает, что ЕЯ не может быть описан целиком непротиворечивым множеством правил, а значит, соответствующие алгоритмы его обработки должны обладать свойством устойчивости, по крайней мере в том смысле, как это определено в [10].

Развитие информационных технологий, и особенно сети Интернет, привело к тому, что в настоящее время поисковые серверы оперируют уже десятками миллиардов документов. Текст стал самым надежным и эффективным источником знаний о предметной области (внешнем мире) [5].

По мнению многих ученых [4], настоящий период развития языкознания характеризуется серьезным влиянием на лингвистическую теорию прикладной лингвистики. По словам известного ученого В. А. Звегинцева, прикладная лингвистика является «полигоном, где проходят испытания как частные гипотезы, так и глобальные теоретические построения» [11]. Кроме того, когда говорят о проверке лингвистических теорий на практике, то имеют в виду не

только, например, создание практических грамматик и словарей, но и их эффективное приложение (использование) в обучении и преподавании.

Так, вследствие внедрения новых информационных технологий во все сферы человеческого общения прикладная лингвистика развивается по направлению автоматизации основных задач, оптимизации коммуникации, т. е. развивается такое направление прикладной лингвистики, которое назвали компьютерной лингвистикой. Свое начало она берет в 50–60-х гг. XX в., а ее появление связано с попыткой решить задачу автоматического перевода текстов с одного языка на другой (Джорджтаунский проект).

В [1, с. 90] отмечается, что истоки компьютерной лингвистики восходят к исследованиям известного американского ученого Н. Хомского в области формализации структуры естественного языка [12], а полученные им результаты на стыке лингвистики и математики заложили основу для теории формальных языков и грамматик (часто называемых генеративными, или порождающими, грамматиками). Эта теория относится ныне к математической лингвистике и применяется для обработки не только ЕЯ, но и искусственных языков, в первую очередь языков программирования. К математической лингвистике относят также и квантитативную лингвистику, изучающую частотные характеристики языка – слов, их комбинаций, синтаксических конструкций и т. п., при этом используются математические методы статистики, так что можно назвать этот раздел науки статистической лингвистикой [13]. Компьютерная лингвистика тесно связана и с такой междисциплинарной научной областью, как искусственный интеллект [14], в рамках которого разрабатываются компьютерные модели отдельных интеллектуальных функций. Существуя уже более полувека и опираясь в своем развитии на результаты в области общей лингвистики (языкознания), компьютерная лингвистика неразрывно связана и с такими научными направлениями, как психология, социология, а это, в свою очередь, еще раз подчеркивает междисциплинарный характер проводимых исследований, связанных с решением следующих основных прикладных задач автоматической обработки текста:

1. Информационный поиск, в том числе автоматическое индексирование документов и запросов пользователя. Основной задачей здесь является поиск необходимой пользователю информации согласно его запросу (релевантной его запросу). С этой точки зрения именно смысловая обработка документов и запросов пользователя становится особенно актуальной.

2. Автоматизация инженерии знаний. Речь идет о том, чтобы на основе автоматического анализа текста выделить из него знания, по крайней мере соответствующие трем основным их типам: объектам (классам объектов), фактам и правилам, отображающим закономерности внешнего мира (предметной области).

3. Автоматическое реферирование текстов. Подразумевает автоматическое выделение на основе анализа текста наиболее важной с определенной точки зрения информации из документа и представление ее пользователю в том или ином виде. Это может быть часть оригинального текста, набор взятых из

него предложений, выделенных по заданному критерию, информация, соответствующая основным типам знаний, – реферат в виде списка объектов (ключевых слов) или фактов, в виде иерархии объектов и т. д.

4. Машинный перевод (МП) текстовых документов с одного ЕЯ на другой (другие). Целью данной задачи является получение адекватного представления входного текста в терминах некоторого другого (других) ЕЯ. Система МП может использоваться как средство собственно перевода документов пользователя, так и как составная часть других информационных систем, например многоязычных систем информационного поиска.

5. Автоматическая классификация текстовых документов. Включает в себя: автоматическую категоризацию, т. е. распределение документов по заранее созданным категориям; автоматическую кластеризацию, т. е. распределение документов по автоматически генерируемым группам или иерархиям групп; генерацию таксономий, т. е. создание иерархий концептов или тематических категорий.

6. Визуализация набора текстовых документов. Целью данной задачи является представление в графической форме, удобной для понимания пользователя, информации о некотором наборе документов и (или) информации, содержащейся в них. Как правило, получаемые представления имеют вид деревьев, графов либо диаграмм. Например, это может быть диаграмма объектов входного набора документов с указанием частоты их встречаемости в документах [15].

2. Естественно-языковой интерфейс

Центральными научными проблемами компьютерной лингвистики являются проблема моделирования процесса понимания смысла текстов (перехода от текста к формализованному представлению его смысла) и проблема синтеза речи (перехода от формализованного представления смысла к текстам на естественном языке).

В то же время сами вычислительные системы, опираясь на базы знаний и функциональные возможности анализа ЕЯ на различных уровнях его глубины, в том числе и на семантическом уровне, способствуют созданию совершенно нового класса информационных систем – интеллектуальных информационных систем, призванных сыграть решающую роль в развитии информационных технологий (речь идет о ЕЯ-интерфейсах пользователя, экспертных системах, системах автоматизации инженерии знаний и решения инновационных задач, семантического поиска информации и т. д.). Под пользовательским интерфейсом понимается система программных решений, реализующих поиск, получение, просмотр и обработку информации из внешнего хранилища, как правило, структурированного источника данных [16]. При этом естественно-языковой интерфейс – это разновидность пользовательского интерфейса, который принимает на вход и обрабатывает запросы на естественном языке, а также может использовать естественный язык для вывода найденной информации пользователю. Разработкой естественно-языкового пользовательского интерфейса ак-

тивно занимаются исследователи по всему миру, предлагая различные подходы, опирающиеся на словари и грамматики, а также пошаговое формирование запроса пользователем [17], построение деревьев зависимостей, использование правил и эвристик при разборе естественно-языкового запроса (проект *NaLIR* [18]), методы машинного обучения для формирования *SQL*-запросов на основе естественно-языкового представления (проект *Sqlizer* [19]) [20]. Таким образом, реализация ЕЯ-интерфейса предполагает удобный для неподготовленного пользователя диалоговый режим коммуникации и должна способствовать достижению целей пользователя, которые определяются его информационными потребностями. Согласно [21] диалог – процесс достижения его участниками определенных согласованных целей путем обмена связанными высказываниями, выраженными в языке, о некотором реальном или гипотетическом мире (проблемной области). Поэтому применительно к диалогу между пользователем и компьютером под общением понимают процесс обмена взаимосвязанными высказываниями, выраженными в языке, направленный на достижение целей пользователя, т. е. на удовлетворение информационных потребностей пользователя (ИПП).

В общем случае процесс общения не может быть сведен к обмену изолированными парами высказываний «вопрос – ответ». Высказывания участников общения образуют связный текст – *дискурс*, имеющий, как правило, достаточно сложную структуру. Связность дискурса обеспечивается как лингвистическими (родовидовыми, анафорическими, модальными, стилистическими согласованиями, согласованиями пресуппозиций и т. п.), так и экстралингвистическими (ситуативными) средствами, т. е. с помощью временных, причинно-следственных и других связей, существующих в проблемной области.

Следует подчеркнуть, что разговорный ЕЯ гораздо более компактен, чем литературный, «письменный» язык, т. к. при общении широко используются разнообразные умолчания (эллипсисы, анафоры, пресуппозиции и др.), восстанавливаемые (раскрываемые) участниками исходя из текущих целей диалога. Если участники достигли цели, поставленной в начале общения, то говорят, что общение завершилось успехом (глобальным успехом), в противном случае – неудачей (глобальной неудачей). В процессе общения могут возникать различные локальные неудачи, вызванные, например, неправильностью (нарушением грамматических норм) высказываний участников, непониманием друг друга из-за различных представлений о теме диалога или о проблемной области на языке общения и т. п. В основном локальные неудачи не приводят к глобальной неудаче, они преодолеваются участниками общения гибкой (в ходе диалога) корректировкой текущих целей. Цели, преследуемые участниками общения, определяют структуру диалога, которая может рассматриваться на трех уровнях: глобальном, тематическом и локальном.

На *глобальном уровне* определяются общие свойства решаемых пользователями задач.

На *тематическом уровне* структура диалога зависит от конкретных особенностей решаемой задачи – от алгоритма ее решения (разбиения задачи на подзадачи) и распределения ролей (активная или пассивная роль) между участниками общения при решении конкретных подзадач.

На *локальном уровне* рассматриваются конкретные шаги диалога, образуемые взаимосвязанными высказываниями его участников (рис. 1).

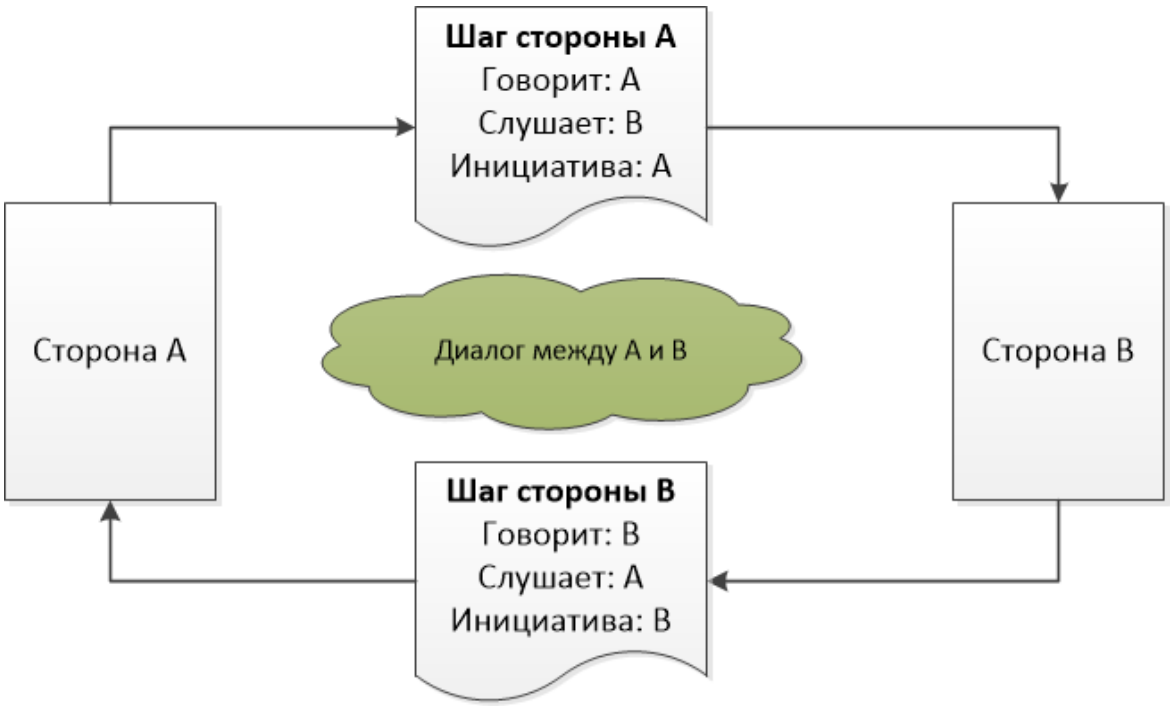

Рис. 1. Структура диалога

Шаг диалога трактуется как пара «действие – реакция», где высказывание активного (т .е. владеющего инициативой) участника соответствует действию, а пассивного – реакции.

Основными параметрами структуры диалога на этом уровне являются:

- инициатор шага и вид инициирования (вид действия);
- способ влияния действия на реакцию;
- способ спецификации задачи (подзадачи), решаемой на данном шаге.

Действие и реакция, образующие шаг диалога, могут в общем случае не соответствовать соседним (во временной последовательности) высказываниям участников. Соответствие нарушается при перехватах инициативы. Перехват инициативы возникает в тех случаях, когда пассивный участник вместо цели (подцели), предложенной активным участником, выбирает иные цели (подцели), в частности подцели, предусматривающие преодоление локальных неудач. Например, вместо ответа на вопрос (что соответствовало бы стандартной реакции) второй участник может задать встречный вопрос (т. е. совершить действие и тем самым взять на себя активную роль) и лишь после получения ответа на него ответить на первоначально заданный вопрос (и тем самым вернуть инициативу). Таким образом, перехват инициативы как бы разрывает первоначально

инициированный шаг диалога и открывает поддиалог – происходит смена цели (темы) диалога, в котором инициативой владеет ранее бывший пассивным участник.

Важными особенностями процесса общения, направленного на удовлетворение информационных потребностей пользователя, которые необходимо учитывать при проектировании и реализации ЕЯ-интерфейсов независимо от специфики решаемых пользователями задач, являются:

1. Изменяемость. Информационная потребность пользователя не может быть заранее четко определена в спецификациях на разработку системы общения. Напротив, она неизбежно изменяется в ходе разработки и эксплуатации системы.

2. Несовпадение взглядов на мир. Представления, имеющиеся у пользователя и системы о языке общения и проблемной области, относительно которой ведется общение, могут не совпадать. Соответственно процесс общения должен предусматривать разъяснение смысла неизвестных терминов, обнаружение и устранение несовпадающих представлений, а также предупреждение ошибочных толкований, т. е. установление общих точек зрения на обсуждаемые в процессе общения сущности.

3. Связность общения. Процесс общения не может быть ограничен обменом изолированными парами «вопрос – ответ», т. к. в большинстве реальных случаев информационная потребность пользователя не может быть выражена в виде одного вопроса (предложения). Часто требуется определить ситуацию, в которой возникла ИПП, т. е. предпослать запросу на решение некоторой задачи контекст, в котором эту задачу необходимо решать. Кроме того, процесс удовлетворения ИПП – решение некоторой задачи в большинстве реальных приложений требует взаимодействия, основанного на смешанной инициативе участников. Поэтому процесс общения должен иметь сложную, разветвленную структуру и состоять из обмена связанными высказываниями.

4. «Неправильность» высказываний пользователя. Для выражения ИПП пользователь может применить как «правильные» предложения, т. е. такие, которые будут однозначно поняты и верно обработаны системой, так и «неправильные». Неправильности возникают, во-первых, из-за того, что пользователь обычно не в состоянии учесть все ограничения системы общения в части ее возможностей и знаний, во-вторых, в связи с использованием умолчаний, характерных для естественного общения и допускающих неоднозначное толкование высказываний, и, в-третьих, из-за отклонения предложений от грамматической нормы [21].

Таким образом, организация ЕЯ-интерфейса между пользователем и компьютером требует реализации в интеллектуальной информационной системе следующих функций:

- ведение диалога – определение его структуры и ранга роли, которую система и пользователь выполняют на текущем шаге диалога;

- понимание – преобразование поступающих от пользователя высказываний на естественном языке в высказывания на языке внутреннего представления;
- обработка высказываний – формирование или определение заданий на решение задач или подзадач на данном шаге диалога;
- генерация – формирование выходных высказываний на ЕЯ.

В соответствии с выделенными функциями обобщенная схема диалоговой подсистемы в системе с ЕЯ-интерфейсом приведена на рис. 2 и может быть представлена в виде четырех основных компонент:

- понимания высказываний;
- обработки высказываний;
- генерации высказываний;
- ведения диалога.

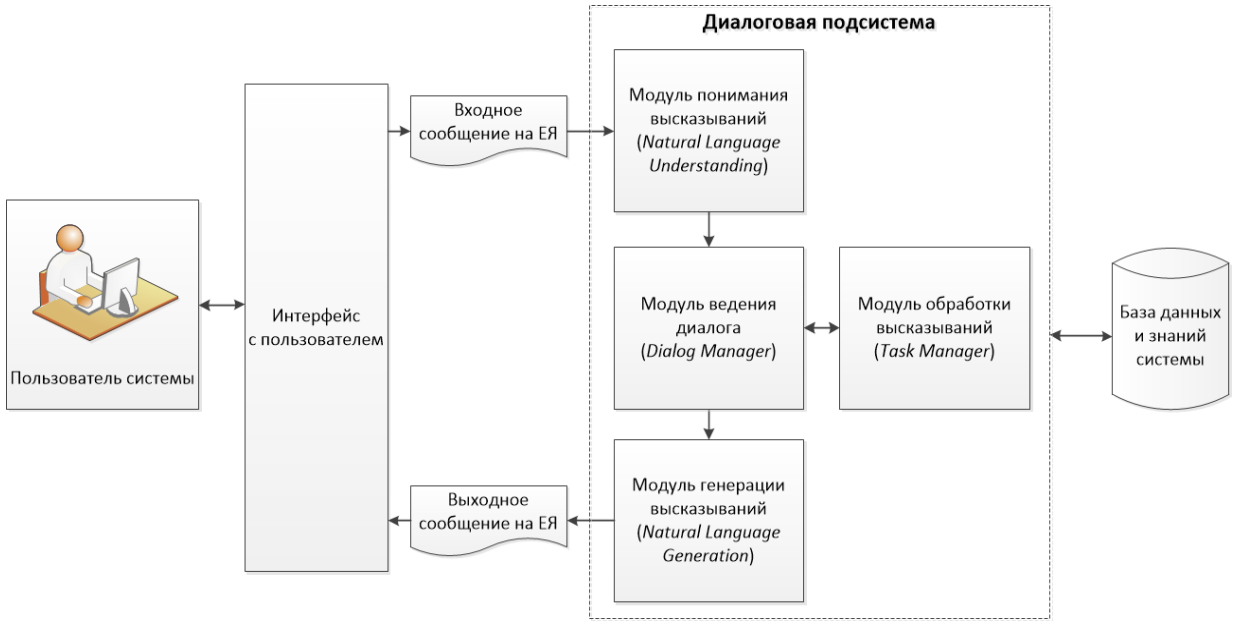

Рис. 2. Обобщенная схема диалоговой подсистемы в системе с ЕЯ-интерфейсом

Основная функциональность диалоговой компоненты заключается в обеспечении решения задачи ведения диалога и формирования или обработки случаев перехватов инициативы. Решение первой задачи состоит в том, чтобы выполнять целесообразные (т. е. способствующие достижению конечных целей пользователя) действия системы на текущем шаге диалога. Это достигается анализом контекстной информации, определяющей общую и тематическую структуру диалога. На основании этих сведений рассматриваемая компонента либо формирует задание (действие), выполняемое системой на текущем шаге (например, формулирование вопроса, выполнение операций, связанных с «пониманием», обработкой и дальнейшим синтезом утверждения), либо определяет адекватное действие-реакцию, в зависимости от того, кому принадлежит инициатива: системе или пользователю. Если инициативу в основном проявляет пользователь, а система только реагирует на его требования, определяя по

виду требования тип задания, то для системы весь диалог сводится к выработке реакции на текущие высказывания пользователя. В противном случае система ведет диалог в соответствии с имеющимися у нее «представлениями» о структуре диалога (т. е. о разбиении задач на подзадачи и о том, кто из участников, когда и какую подзадачу решает) и о способе обмена высказываниями. На рис. 3 приведен фрагмент кода на языке *Python*, иллюстрирующий абстрактный базовый класс *Dialog*, отвечающий за прием сообщений пользователя (метод *listen()*), их анализ (метод *parse()*), интерпретацию результатов (метод *interpret()*), например, с использованием количественных оценок, выражающих корректность интерпретации, и синтез текста ответа (метод *respond()*) [22].

```
import abc

class Dialog (abc.ABC):
    """
    Объект диалога принимает высказывания, выполняет их парсинг
    и интерпретацию, а затем изменяет внутреннее состояние.
    После этого может сформулировать ответ.
    """
    def listen(self, text, response= True, **kwargs):
        """
        Принимает текст высказывания text и выполняет его анализ.
        Передает результаты методу interpret для определения ответа.
        Если ответ необходим, вызывается метод respond, синтезирующий
        текст ответа на основе последнего поступившего текста и текущего
        состояния объекта Dialog.
        """
        # Анализ входного текста
        sents = self.parse(text)

        # Интерпретация
        sents, confidence, kwargs = self.interpret(sents, **kwargs)

        # Формулировка ответа
        if response:
            reply = self.respond(sents, confidence, **kwargs)
        else:
            reply = None

        # Передача инициативы
        return reply, confidence
```

Рис. 3. Абстрактный базовый класс *Dialog*,
формально определяющий ход диалога пользователя с системой

Метод *listen()* получает текст сообщения и логический флаг *response*, указывающий, была ли передана инициатива этому объекту и требуется ли вернутьответ (если передать в аргументе *response* значение *False*, то объект класса *Dialog* лишь воспримет высказывание и обновит свое внутреннее состояние). Метод также принимает произвольное количество именованных аргументов

(*kwargs*), в которых может передаваться дополнительная контекстная информация. Метод *parse()* определяет способ анализа входных данных, используя, например, механизм регулярных выражений или более сложные методы на базе лингвистических правил или машинного обучения. Метод *respond()* на основании входных сообщений, актуального состояния диалога и вспомогательных именованных аргументов, содержащих всю необходимую информацию для формирования ответа (например, результатов работы методов *listen()* и *interpret()*), синтезирует текст ответа пользователю.

Если роли участников неизменны, однозначны и предопределены заранее, то структуру диалога называют *жесткой.* В простейшем случае такая структура диалога сводится к двум взаимосвязанным высказываниям участников (вопрос – ответ) с указанием участника, владеющего инициативой. Примерами реализации подобного взаимодействия являются автоматизированные системы поддержки пользователей с интерактивными голосовыми меню, где существует возможность перевести разговор на человека-оператора для получения более подробной информации (рис. 4).

Рис. 4. Структурно-функциональная схема
автоматизированной системы поддержки пользователей

Как правило, в качестве информационной базы (блок «Ресурсы системы») для формирования реплик диалога используются наборы часто задаваемых вопросов, или *frequently asked questions* (*FAQ*), что обеспечивает достаточно быстрый ответ за счет применения механизмов поиска по шаблону (паттерну) с использованием регулярных выражений и возврата фиксированного (шаблонизированного) ответа в случае обнаружения совпадений [23]. Механизм сопо-

ставления с паттернами встроен в большинство современных языков программирования (например, *C#*, *Python*), а язык сопоставления с паттернами, или язык регулярных выражений, задействуется для написания самих паттернов. На рис. 5 приведен фрагмент кода на языке *Python*, иллюстрирующий процесс задания различных вариантов выражения приветствия (*greeting*), знакомства (*introduction*), прощания (*goodbye*) и переклички (*rollcall*) с помощью регулярных выражений, а также их дальнейшую компиляцию в объекты регулярных выражений с целью эффективного многократного использования: поиска указанных шаблонов во входном тексте (метод *parse()*) [22].

```
class Greeting(Dialog):
    """
    Запоминает участников, вступающих в беседу и покидающих ее, отвечает
    соответствующим приветствием и прощанием. Пример системы,
    основанной на правилах, которая управляет своим внутренним состоянием и
    использует регулярные выражения и логику поддержания диалога
    """
    PATTERNS = {
        'greeting': r'hello|hi|hey|good morning|good evening',
        'introduction': r'my name is ([a-z\-\s]+)',
        'goodbye': r'goodbye|bye|ttyl',
        'rollcall': r'roll call|who\'s here?'
    }

    def __init__(self, participants = None):
        #Participants - это словарь, отображающий пользователя
        #в его настоящее имя
        self.participants = {}

        if participants is not None:
            for participant in participants:
                self.participants[participant] = None

        #Скомпилировать регулярные выражения
        self._patterns = {
            key: re.compile(pattern,re.I)
            for key, pattern in self.PATTERNS.items()
        }

        def parse(self, text):
            """
            Применяет все регулярные выражения к тексту в поисках совпадения.
            """
            matches = {}
            for key, pattern in self._patterns.items():
                match = pattern.match(text)
                if match is not None:
                    matches[key] = match
            return matches
```

Рис. 5. Фрагмент кода реализации класса *Greeting*

Развитием жесткой структуры является альтернативная структура, которая задает множество возможных (но заранее предписанных) направлений течения диалога. Выбор одного из возможных направлений осуществляет пассивный участник. Если роли участников общения распределяются в ходе общения, то структуру диалога называют гибкой. Гибкие структуры подразделяются по степени свободы выбора момента перехвата (предопределенные моменты, произвольные моменты) и по способу перехвата (предопределенный способ перехвата, произвольный способ).

В каждом из рассмотренных выше случаев ЕЯ является объектом моделирования, и учет в этом смысле особенностей ЕЯ существенно влияет на качество работы лингвистических процессоров (ЛП), осуществляющих его автоматическую обработку в целях решения как указанных, так и многих других задач.

3. Лингвистические аспекты в задаче автоматической обработки текста

Как отмечалось выше, язык представляет собой знаковую систему, предназначенную для порождения, передачи и хранения информации. Любой текст T языка L можно рассматривать как конечную цепочку $T = a_1 a_2 ... a_n$, где $a_i \in A,\ i = \overline{1,n}$, A – алфавит языка L, образованную в соответствии с множеством его лексико-грамматических, синтаксических и семантических правил. Иначе говоря, текст на ЕЯ состоит из отдельных единиц (знаков), и возможно несколько способов разбиения (членения) текста на единицы, относящиеся к разным уровням [1].

Общепризнано существование следующих уровней:

- уровень предложений (высказываний) – синтаксический уровень;
- уровень слов (словоформ – слов в определенной грамматической форме, например *ручка*, *дружбой*) – морфологический уровень;
- уровень фонем (отдельных звуков, с помощью которых формируются и различаются слова) – фонологический уровень [24].

Фонологический уровень выделяется для устной речи, а для письменных текстов в языках с алфавитным способом записи (в частности, в европейских языках) он соответствует уровню символов (фонемы приблизительно соответствуют буквам алфавита).

Уровни, по сути, есть подсистемы общей системы ЕЯ (взаимосвязанные, но в достаточной степени автономные), и в них самих могут быть выделены подсистемы. Так, морфологический уровень включает в себя также подуровень морфем. Морфема – это минимальная значимая единица языка (корень, приставка, суффикс, окончание, постфикс).

Вопрос о количестве уровней и их перечне в лингвистике до сих пор остается открытым. Как отдельный может быть выделен лексический уровень – уровень лексем. Лексема – это слово как совокупность всех его конкретных

грамматических форм (например, лексему *лист* образуют формы *лист*, *листа*, *листу*, *листом*). Вернее сказать, лексема – семантический инвариант всех словоформ. В тексте встречаются словоформы (лексемы в определенной форме), а в словаре ЕЯ – лексемы. Точнее, в словаре записывается каноническая словоформа лексемы, называемая также леммой (например, для существительных это форма именительного падежа единственного числа: *лист*).

В рамках синтаксического уровня может быть выделен подуровень словосочетаний – синтаксически связанных групп слов (*видел лес*, *синий шар*).

Выделяют различные типы словосочетаний, которые определяются синтаксическими свойствами отдельных слов и правилами их соединения. В зависимости от степени «слияния» составляющих выделяют:

- свободные словосочетания – в процессе их формирования лексические значения входящих в них слов сохраняются (например, «вычислительная техника»);
- несвободные словосочетания – утрачена или ослаблена лексическая самостоятельность знаменательных слов.

Различают также:

- синтаксически несвободные словосочетания, выполняющие единую синтаксическую функцию, например *процессор повышенной производительности* или *системы искусственного интеллекта* (нельзя сказать *процессор производительности*, поэтому словосочетание *повышенной производительности*, являющееся свободным, неделимо на компоненты в приведенном выше словосочетании);
- фразеологически несвободные словосочетания, которые обладают семантической и грамматической неразложимостью (например, в предложении *Процессор вышел из строя* словосочетание *вышел из строя* семантически неделимо, т. к. означает *испортился*) [25].

При классификации на основе понятия сочетаемости, которая определяет разрешенные связи одного конкретного слова с другим конкретным словом или группой слов, различают сочетаемость:

- морфосинтаксическую;
- лексическую;
- семантическую [26].

Информация о сочетаемости слов является сугубо индивидуальной и обычно фиксируется в виде моделей управления или лексических функций для каждой конкретной единицы в словаре. Конструктивными единицами словосочетания являются слова или словосочетания, имеющие меньшее количество единиц. Пусть слово *A* синтаксически непосредственно или опосредствованно связано со словом (словосочетанием) *B*, тогда:

1. Под морфосинтаксической сочетаемостью слова *A* (или морфосинтаксическими ограничениями на сочетаемость *A*), синтаксически связанного со словом *B*, понимают информацию о части речи или синтаксическом статусе *B* и о грамматической (в частности, предложно-падежной) форме, в которой

должно стоять *B*. Например, разные морфосинтаксические ограничения у неточных синонимов *ошибаться (адресом) – перепутать (адрес)* и т. п.

2. Лексическую сочетаемость *A* (лексические ограничения на сочетаемость *A*) составляет информация о том, каким должно быть слово *B* (или класс слов B_1, ..., B_n), с которым синтаксически связано слово *A*. Например, слово *ошибаться* лексически сочетается только с небольшой конкретной группой существительных, например *адрес*, *дом*, *дверь*, *номер*, *этаж* и т. п. (например, *ошибаться дверью*). Эти существительные можно задать, вероятно, только поименно, а не с помощью какого-либо семантического признака. Так, например, близкий по смыслу глагол *перепутать* сочетается и с другими словами – *книга*, *шляпа*, *дата*, *должность* и т. п.

3. Семантическую сочетаемость *A* (семантические ограничения на сочетаемость *A*) составляет информация о том, какими семантическими признаками должно обладать слово *B*, синтаксически связанное с *A*. О семантических, а не лексических ограничениях на сочетаемость *A* разумно говорить лишь в тех случаях, когда любое слово *B*, имеющее требуемый семантический признак, способно сочетаться с *A*. Например, перевозить можно физические объекты ограниченного размера (станки, детали, лошадей и т. п.) [26].

В зависимости от принадлежности главного слова к той или иной части речи различают лексико-грамматические типы словосочетаний, которые могут быть образованы, например, в соответствии со следующими правилами.

Глагольные словосочетания:

- глагол + существительное или местоимение (с предлогом или без предлога): *купить хлеба*, *обратиться к нему*;
- глагол + инфинитив или деепричастие: *просить приехать*, *сидеть задумавшись*;
- глагол + наречие: *поступать правильно*, *повторять дважды*.

Именные субстантивные словосочетания:

- согласуемое слово + существительное: *ясный день*, *мой мир*;
- существительное + существительное: *город в огнях*, *отрывок из поэмы*;
- существительное + наречие: *шаг вперед*, *лов зимой*;
- существительное + инфинитив: *готовность помочь*, *повод поговорить*.

Именные адъективные словосочетания:

- прилагательное + наречие: *по-праздничному нарядный*, *едва слышный*;
- прилагательное + существительное (местоимение): *широкий в плечах*, *равнодушный ко всему*;
- прилагательное + инфинитив: *способный организовать*, *готовый сопротивляться* [27].

В рамках синтаксического уровня также выделяют надуровень сложного синтаксического целого, которому примерно соответствует абзац текста. Сложное синтаксическое целое, или сверхфразовое единство, – это последовательность предложений (высказываний), объединенных смыслом и лексико-грамматическими средствами [25]. К таким средствам относятся в первую очередь лексические повторы и анафорические ссылки – ссылки на предшествую-

щие слова текста, реализуемые при помощи местоимений и местоименных слов (*они, этот, там же* и т. д.).

Иерархия уровней проявляется в том, что единицы более высокого уровня разложимы на единицы более низкого (например, словоформы – на морфы); более высокий уровень во многом обусловливает организацию нижележащего уровня – так, синтаксическая структура предложения в значительной мере определяет, какие должны быть выбраны словоформы.

Можно также говорить еще об одном уровне – уровне дискурса, под которым понимается связный текст в его коммуникативной направленности [25]. Дискурс – это последовательность взаимосвязанных друг с другом предложений текста, обладающая конкретной смысловой целостностью, за счет чего он выполняет определенную прагматическую задачу. Во многих типах связных текстов проявляется традиционная схематическая (дискурсивная) структура, организующая их общее содержание. Например, определенную структуру имеют описания сложных технических систем, патентные формулы, научные статьи, деловые письма и др.

Особым является вопрос об уровне семантики. В принципе, смысл есть всюду, где имеются знаковые единицы языка (морфемы, слова, предложения). Подтверждением самостоятельности уровня семантики считается то, что человек обычно запоминает смысл высказывания, а не его конкретную языковую форму. Следует отметить, что текст имеет свою микро- и макросемантику, микро- и макроструктуру. Семантика текста обусловлена коммуникативной задачей передачи информации (текст – информационное целое); структура текста определяется особенностями внутренней организации единиц текста и закономерностями взаимосвязи этих единиц в рамках цельного сообщения (текст – структурное целое). Единицами текста на семантико-структурном уровне являются: высказывание (реализованное предложение), межфразовое единство (ряд высказываний, объединенных семантически и синтаксически в единый фрагмент). Межфразовые единства, в свою очередь, объединяются в более крупные фрагменты-блоки, обеспечивающие тексту целостность благодаря реализации дистантных и контактных смысловых и грамматических связей. На уровне композиционном выделяются единицы качественно иного плана – абзацы, параграфы, главы, разделы, подглавки и др. Единицы этих двух уровней взаимосвязаны и взаимообусловлены, в частном случае они могут даже накладываться друг на друга, как, например, межфразовое единство и абзац, хотя при этом они сохраняют свои собственные отличительные признаки. С семантической, грамматической и композиционной структурой текста тесно связаны его стилевые и стилистические характеристики. Каждый текст обнаруживает определенную более или менее выраженную функционально-стилевую ориентацию (научный, художественный и др.) и обладает стилистическими качествами, диктуемыми данной ориентацией и к тому же индивидуальностью автора [28].

Кроме многоуровневости системы ЕЯ сложность его моделирования связана с постоянно происходящими в нем изменениями (что вполне ощутимо по

прошествии одного-двух десятилетий). Изменения касаются не только словарного запаса языка (новые слова и новые смыслы старых), но также синтаксиса, морфологии и фонетики. Как следствие, принципиально невозможно единожды разработать формальную модель конкретного ЕЯ и построить соответствующий лингвистический процессор. Требуются постоянное пополнение знаний о языке на всех его уровнях и коррекция существующих моделей [23].

4. Лингвистический процессор

Согласно [5] лингвистическая обработка текста в составе различных систем его автоматической обработки обычно реализуется с помощью специальных информационно-программных комплексов, называемых лингвистическими процессорами [29]. Независимо от приложения ЛП включает в себя две основные компоненты: лингвистическую базу знаний (ЛБЗ) и информационно обеспечиваемый ею набор программных модулей собственно обработки (целевой обработки) текстовых документов, реализованных в соответствии с определенными, как правило, оптимизированными по времени их выполнения и требуемой памяти ЭВМ, алгоритмами. Первый из указанных модулей ЛП является декларативным, второй – процедурным. Естественно-языковые, а также другие интеллектуальные информационные системы, реализующие решение рассмотренных выше задач автоматической обработки текста, включают в себя в качестве своей обязательной базовой функциональности именно автоматический лингвистический анализ (ЛА) текстовых документов, который в силу этого своего универсального назначения определяет так называемый базовый ЛП ЕЯ-систем. А что касается конкретных приложений, то их реализация в каждом случае требует еще некоторых дополнений как в декларативном, так и в процедурном компонентах лингвистического процессора.

Общепринятой схемой лингвистического анализа текста является его поэтапная обработка: снизу вверх от лексических фактов к семантико-синтаксической цели [30, 31]. На каждом из этапов анализа входное представление текста определенным образом обрабатывается и дополняется лингвистической информацией, т. е. по мере обработки текста выделяются все более и более глубинные его структурные единицы и связи между ними, и, таким образом, входной текст претерпевает последовательные этапы формализации и структуризации [32]. Традиционно выделяют следующие основные этапы (задачи) ЛА текста:

- форматирование;
- лексический анализ;
- лексико-грамматический анализ;
- синтаксический анализ;
- семантический анализ [6].

В соответствии с этой последовательностью входной текст вначале форматируется, затем разбивается на отдельные слова и предложения, далее для слов определяются их лексико-грамматическая и синтаксическая роль и связь с

другими словами из предложения и, наконец, их семантическая роль и смысловая связь с другими предложениями текста в целом, т. е. реализуется вся функциональность так называемого базового ЛП (рис. 6) [33]:

- форматирование текста (на этом этапе проводится преобразование текстовых документов, представленных обычно в различных форматах, в некоторый единый формат, максимально сохраняющий стилистическую и структурную разметку документов; здесь же осуществляется разбиение текста на параграфы, выделение заголовков, подзаголовков и т. п.);
- лексический анализ (предназначен прежде всего для распознавания в обрабатываемом тексте границ слов и предложений);
- лексико-грамматический анализ (ЛГА) (его задачей является определение лексико-грамматического класса каждого слова входного текста с учетом контекста и заранее заданного списка этих классов для ЕЯ; например, для английского языка их количество может достигать около 200 (это, например, *JJ* – прилагательное; *VB* – глагол; *MD* – модальный глагол; *NN* – существительное единственного числа; *RB* – наречие; *ATI* – определенный артикль; *CC* – союз [34]), для белорусского и русского – более 1000);
- синтаксический анализ (на данном этапе традиционно для каждого предложения текста строится синтаксическое дерево, в котором его слова представлены в виде листьев, а другим узлам соответствует, например, простая именная, предложная, глагольная и т. п. группы, в том числе и собственно предложение (корень дерева); при этом устанавливаются синтаксические связи между ними, например фиксируется факт, что простое предложение включает в себя подлежащее, выраженное именной группой, сказуемое, выраженное глаголом в прошедшем времени, и прямое дополнение, также выраженное именной группой);
- семантический или семантико-синтаксический анализ, задачей которого является извлечение из синтаксических деревьев в виде отношений наиболее значимых с точки зрения знаний основных типов, синтаксических структур:

1) *Simple Noun Phrase* (простая именная группа);
2) *Verb Phrase* (глагольная группа);
3) *Noun Phrase Additional* (именная группа, распространенная различными оборотами);
4) *Complex Sentence* (сложноподчиненное предложение) [35].

Что касается ЛБЗ, то основными ее компонентами согласно [5, 33] являются следующие:

- классификаторы лексико-грамматических, синтаксических и семантических свойств ЕЯ (их состав зависит от конкретных свойств ЕЯ и от характера приложения, определяющего степень детализации лингвистического анализа текста);
- базовый (эталонный) словарь (реализуется в виде словаря словоформ ЕЯ и включает в себя максимально возможное их количество; при этом для каждой словоформы указаны все ее возможные вне контекста лексико-грамматические коды (ЛГК); классические базовые словари, например, англий-

ского, французского языков содержат несколько сотен тысяч словоформ, русского и белорусского языков – более одного миллиона словоформ);

- базовый (эталонный) корпус текстов (БКТ) (реализуется в виде определенным образом подобранных текстов, причем как минимум для каждого слова текста указан его единственный с точки зрения контекста ЛГК; минимальный размер БКТ обычно составляет порядка одного миллиона словоупотреблений, однако моделирование ЕЯ на более высоких уровнях его глубины требует разработки БКТ объемом порядка 10^7–10^8 словоупотреблений, причем аннотированных не только ЛГК, но и, возможно, метками синтаксических и семантических отношений; БКТ предназначен прежде всего для получения количественных оценок языка, тестирования лингвистических гипотез и отдельных алгоритмов и систем автоматической обработки текста);

- лингвистические правила анализа текста на различных уровнях глубины ЕЯ (такие правила, получаемые лингвистами-экспертами, являются основой разработки машинных алгоритмов для большинства этапов автоматического лингвистического анализа текста; совокупность этих правил, например, для лексико-грамматического и синтаксического анализа составляет грамматику ЕЯ; их количество в зависимости от глубины лингвистического анализа текста и степени обобщения терминальных символов может колебаться от нескольких десятков до десятков тысяч).

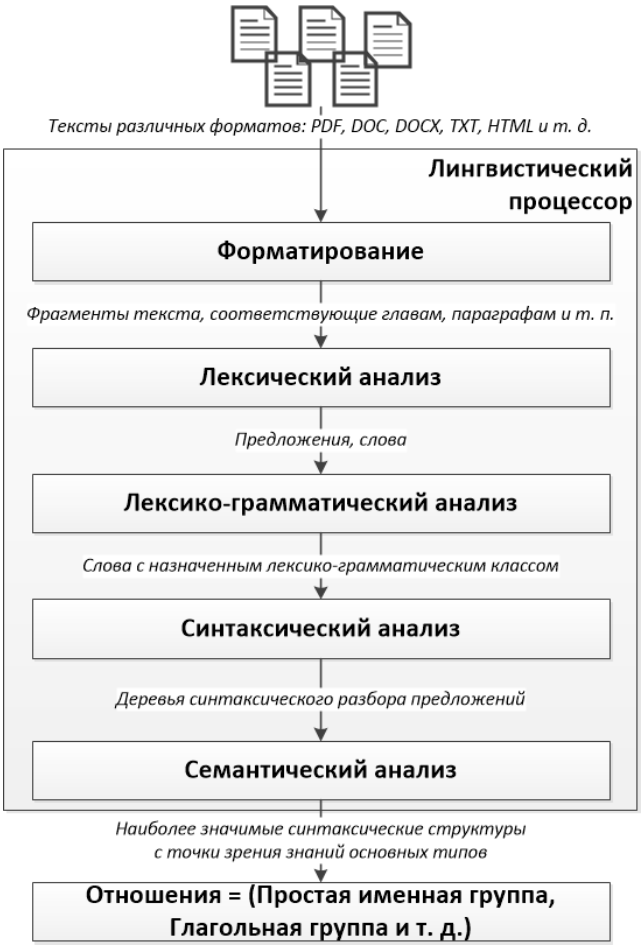

Рис. 6. Задачи ЛА, решаемые с помощью базового ЛП

С целью машинной обработки лингвистических правил должна быть создана некоторая нотация для формального описания этих правил, где они обычно и представляются в ЛБЗ. Причем предлагаемый формализм должен быть максимально соотнесен с требованиями его доступности для использования экспертами, возможностью обобщения разрабатываемых правил и оптимизации скорости их обработки. Последнее особенно важно, учитывая, что даже наиболее известные алгоритмы подобного типа уже не удовлетворяют современным требованиям, предъявляемым к промышленным системам автоматической обработки текста. Это, в частности, относится и к такому известному производителю промышленных лингвистических ресурсов, как *Connexor* [36]. В [5] в качестве такого формализма разработан и успешно внедрен в промышленные приложения так называемый язык расширенных регулярных выражений (*WRE*). В дополнение к основным перечисленным ресурсам в состав ЛБЗ входят различного рода словари (словари идиом, аббревиатур, имен собственных, слов, параметров и т. п.), специальные лексические базы данных, например типа *WordNET* [37], отображающие синонимические, иерархические и ассоциативные отношения в концептах и т. д. Безусловно, ЛБЗ, ориентируясь на обработку текстов на нескольких языках, является многоязычной.

Функциональности представленного выше базового ЛП достаточно как для обеспечения лингвистической составляющей при решении задач компьютерной лингвистики, так и для требуемого для решения всех задач уровня лингво-статистического анализа текста, который традиционно сводится прежде всего к подсчету частот, распознаваемых на приведенных ранее этапах обработки текста лексических единиц и отношений.

В большинстве случаев ЛП применяется пользователями ЭВМ не напрямую, а внутри различных пакетов прикладных программ (систем информационного поиска, проверки орфографии, автоматического реферирования и др.). Причем один и тот же ЛП может использоваться сразу несколькими приложениями. Например, лингвистический процессор *MS Windows*, обеспечивающий проверку орфографии, используется в следующих приложениях: *MS Word, MS Excel, MS PowerPoint* и др. В связи с этим ЛП удобно оформлять в виде независимой внешней библиотеки, доступной различным приложениям. Основным преимуществом этого подхода является независимость реализации ЛП от конкретных приложений, что облегчает процесс его разработки, тестирования и последующей поддержки [5].

5. Корпусы естественно-языковых текстов

Под *лингвистическим*, или *языковым, корпусом текстов* понимается большой, представленный в машиночитаемом виде, унифицированный, структурированный, размеченный, филологически компетентный массив языковых данных, предназначенный для решения конкретных лингвистических задач [38].

Процесс создания корпуса можно представить в виде следующих этапов:

1. Обеспечение поступления текстов в соответствии с перечнем источников.

2. Преобразование в машиночитаемую форму. Тексты в электронном виде для создания корпусов могут быть получены самыми разными способами – ручной ввод, сканирование и др.

3. Анализ и предварительная обработка. На этом этапе все тексты, полученные из разных источников, проходят филологическую выверку и корректировку. Выполняется подготовка «технологического» (библиографического и экстралингвистического) описания текста.

4. Конвертирование и графематический анализ. Некоторые тексты проходят также через один или несколько этапов предварительной машинной обработки, в ходе которых осуществляются перекодировка (если требуется), а также удаление или преобразование нетекстовых элементов (рисунки, таблицы), удаление из текста переносов, «жестких концов строк», обеспечение единообразного написания тире и т. д., выделение и оформление нестандартных (нелексических) элементов, обработка специальных текстовых элементов (имен (имя, отчество), написанных инициалами, иностранных лексем, записанных латиницей, названий рисунков, примечаний, страниц форзаца, зачеркиваний, титульных листов, списков литературы и т. д.). Как правило, эти операции выполняются в автоматическом режиме. Обычно на этом же этапе осуществляется сегментирование текста на его структурные составляющие.

5. Разметка текста. Разметка текста заключается в приписывании текстам и их компонентам дополнительной информации (метаданных): экстралингвистических, относящихся ко всему тексту; данных о структуре текста; лингвистических метаданных, описывающих элементы текста. Метаописание текстов корпуса включает в себя как содержательные элементы данных (библиографические данные, признаки, характеризующие жанровые и стилевые особенности текста, сведения об авторе), так и формальные (имя файла, параметры кодирования, версия языка разметки, исполнители этапов работ). Эти данные обычно вводятся вручную. Структурная разметка документа (выделение абзацев, предложений, слов) и собственно лингвистическая разметка обычно осуществляются автоматически.

6. Корректировка результатов автоматической разметки: исправление ошибок и снятие неоднозначности (вручную или полуавтоматически).

7. Конвертирование размеченных текстов в структуру специализированной лингвистической информационно-поисковой системы (*corpus manager*), обеспечивающей быстрый многоаспектный поиск и статистическую обработку.

8. Обеспечение доступа к корпусу. Корпус может быть доступен в пределах дисплейного класса, может распространяться на электронных носителях или с помощью сети Интернет. Различным категориям пользователей могут предоставляться разные права и разные возможности.

9. Создание документационного обеспечения, в котором описываются различные аспекты создания и использования корпуса, в частности приводятся

сведения о разметке, позволяющие искать по метаданным, язык запросов корпусного менеджера и т. д. [38].

Корпусный менеджер, или корпус-менеджер, – это специализированная поисковая система, включающая в себя программные средства для поиска данных в корпусе, получения статистической информации и предоставления пользователю результатов в удобной форме. Поиск в корпусе текстов (КТ) позволяет по любому слову построить конкорданс – список всех употреблений данного слова в контексте со ссылками на источник. Корпусы могут использоваться для получения разнообразных справок и статистических данных о языковых и речевых единицах. Например, можно получить данные о частоте словоформ, лексем, грамматических категорий, проследить изменение частот и контекстов в различные периоды времени, получить данные о совместной встречаемости лексических единиц и т. д.

Базовые функциональные возможности корпусного менеджера сводятся к обеспечению:

- построения как *KWIC* (*Key Word In Context*), так и полных конкордансных списков;
- поиска отдельных слов и словосочетаний;
- осуществления поиска по шаблонам (реализация сложных запросов);
- сортировки списков по нескольким критериям, выбираемым пользователем;
- отображения найденных словоформ в неограниченном контексте;
- предоставления статистической информации по отдельным элементам корпуса;
- отображения лемм, морфологических характеристик словоформ и метаданных (библиографические, типологические), что зависит от степени покрытия корпуса разметкой;
- сохранения и распечатки результатов;
- работы как с отдельными файлами, так и с корпусами, неограниченными по размеру;
- поддержки различных форматов текстовых данных (*TXT*, *DOC*, *RTF*, *HTML*, *XML* и др.) [38].

Для обработки корпусных данных используются корпусные менеджеры, построенные на основе систем управления базами данных (СУБД) или поисковых систем. Например, корпусный менеджер *Bonito*, разработанный П. Рыхли и группой *NLPlab* (*Natural Language Processing Laboratory*) на факультете информатики Университета им. Масарика (Чехия), реализует клиент-серверную архитектуру и обеспечивает доступ к корпусу английских текстов *SUSANNE* (*Surface and Underlying Structural Analysis of Natural English*) (http://www.grsampson.net/), содержащему более 130 тыс. слов Брауновского корпуса американского английского языка, аннотированного согласно схеме *SUSANNE* (создан в Университете Сассекса, Великобритания). Пример результатов его работы приведен на рис. 7.

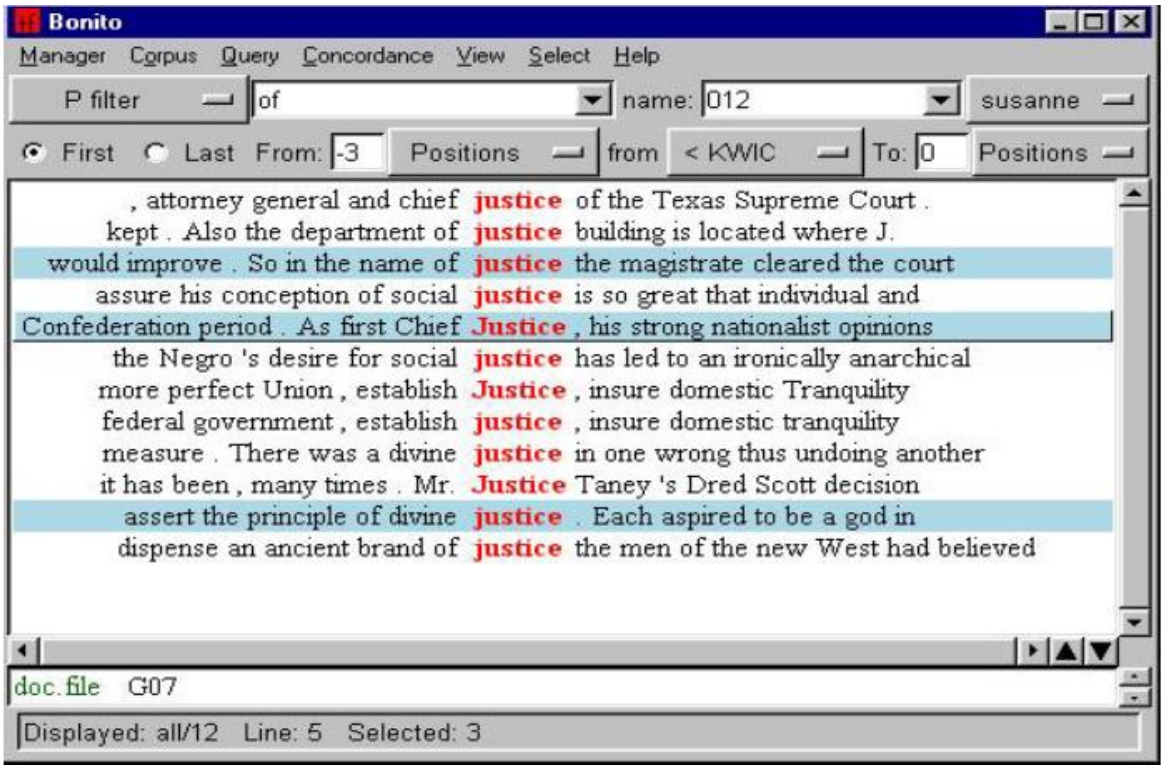

Рис. 7. Конкорданс для словоформы *justice*,
полученный с помощью корпусного менеджера *Bonito*

КТ, содержащий представительный массив языковых данных за определенный период, позволяет изучать динамику процессов изменения лексического состава языка, проводить анализ лексико-грамматических характеристик в разных жанрах и у разных авторов. С течением времени объем и состав корпуса текстов может меняться, однако эти изменения должны либо не изменять его структуру, либо изменять ее обоснованно, обеспечивая сохранение важной характеристики – репрезентативности, или сбалансированности, под которой обычно понимают необходимо-достаточное и пропорциональное представление в корпусе текстов различных периодов, жанров, стилей, авторов и т. п. [38].

Например, Британский национальный корпус (*BNC*) является одним из больших эталонных корпусов, в котором содержится более 100 млн слов разговорного и письменного английского языка. Разработан в Оксфордском университете при участии Ланкастерского университета и Британской библиотеки. Работа над созданием корпуса продолжалась с 1991 по 1994 г. Подкорпус, представляющий письменный английский язык, составляет 90 % всего корпуса и включает в себя тексты из газет, периодических научных изданий и журналов, издаваемых для различных возрастов, произведений популярной научной фантастики, тексты писем, школьных и университетских сочинений и т. д. Тексты отбирались по трем основным критериям: время, область, которую данный текст описывает, и тип издания. По времени все тексты принадлежат примерно одному периоду, начиная с 1975 г., исключения составляют произведения развлекательной литературы. К развлекательной литературе принадлежит 25 % текстов, 75 % письменных текстов были взяты из информативных изданий

(наука, искусство, коммерция и финансы, досуг, социология, мировое обозрение). Учитывались длина (количество слов), тема, обсуждаемая в тексте, имя автора, возраст, пол, место рождения, место жительства, возрастная группа людей, которым предназначен данный текст, а также уровень его сложности. Подкорпус устной речи включает в себя речь добровольцев различных возрастов, социальных классов из разных регионов Великобритании. Разговорная речь присутствует в окружении множества контекстов: от речи формальных деловых или правительственных встреч до радиошоу и телефонных разговоров. Так, весь 10-миллионный подкорпус устной речи разделен на две примерно равные части: *демографическую*, содержащую транскрипции «спонтанных», естественных диалогов, и *контекстно-управляемую часть*, содержащую записи, сделанные на публичных мероприятиях. В работах по сбору материала (записи диалогов) для демографической части участвовали 124 добровольца, действовавших по всей территории Великобритании, фиксируя, при каких условиях, кто являлся собеседником, каковы были взаимоотношения, физическое окружение записываемой речи и т. д. Контекстно-управляемая часть включает в себя следующие категории социального контекста:

- образовательные и информативные собрания (лекции, семинары, программы новостей);
- деловые события (выставки, консультации, интервью, собрания торговых организаций);
- публичные события (проповеди, политические речи, заседания парламента);
- темы, касающиеся досуга (спортивные комментарии, клубные встречи).

Все тексты Британского национального корпуса сегментированы на предложения. Словам внутри предложения присвоены соответствующие коды, обозначающие грамматический класс слова или его часть речи (например, тег *AJ*0 обозначает прилагательное (*good, old, beautiful* и т. д.), а *AJC* – сравнительное прилагательное (*better, older* и т. д.)). Сегментация и присвоение тегов осуществлялись в автоматическом режиме (процент допускаемых ошибок составлял примерно 1,7 %) с помощью утилиты автоматической разметки *CLAWS*, разработанной в университете Ланкастера. Если программа сталкивалась со случаями, когда она не могла однозначно присвоить слову какой-то тег, ему присваивались сразу два тега (например, *VVD-VVN*, из которых первый обозначает глагол прошедшего времени, а второй – причастие прошедшего времени). Работа с корпусом осуществляется с помощью корпусного менеджера *XAIRA* (*XML Aware Indexing and Retrieval Architecture*) [38].

Как отмечалось ранее, основной компонентой ЛБЗ лингвистического процессора является базовый корпус текстов, в котором каждому словоупотреблению указан с учетом контекста его единственный ЛГК. Такой корпус также называют размеченным, индексированным или тегированным. В общем случае в размеченном корпусе словам или предложениям присваиваются метки (теги) в соответствии с характером разметки:

- морфологические;

- синтаксические;
- семантические;
- просодические и др.

Разметка корпусов представляет собой трудоемкую операцию, особенно учитывая размеры современных корпусов. Если для некоторых видов разметки, в частности анафорической, просодической, создание автоматических систем пока представляется довольно сложным и основная часть работы проводится вручную, то для морфологического (лексико-грамматического) и синтаксического анализа существуют соответствующие программные средства:

- теггеры (*taggers*);
- парсеры (*parsers*).

В результате работы программ-теггеров каждой лексической единице приписываются грамматические характеристики, включая часть речи, лемму и набор граммем (например, род, число, падеж, одушевленность/неодушевленность, переходность и т. д.).

В результате работы программ автоматического синтаксического анализа фиксируются синтаксические связи между словами и словосочетаниями, а синтаксическим единицам приписываются соответствующие характеристики (тип предложения, синтаксическая функция словосочетания и т. д.). Однако автоматический анализ естественного языка не всегда безошибочен и, как правило, дает несколько вариантов анализа для одной лексической единицы (слова, словосочетания, предложения). В этом случае говорят о грамматической омонимии. Снятие неоднозначности (морфологической, синтаксической) в целом является одной из важных задач, для решения которой используются как автоматические, так и ручные способы [38].

Один из простых способов автоматизации процесса аннотирования корпуса текстов заключается в том, чтобы словарь, в котором указаны лексические категории для самых распространенных слов или для наибольшего количества слов, совместить с неразмеченным корпусом, а затем каждому слову в неразмеченном корпусе автоматически присвоить тег от соответствующего ему слова в снабженном пометами словаре. Например, если словоформы *information* и *distribution* появились и в корпусе, и в словаре, тег *Noun* (существительное), который сопровождал эти словоформы в словаре, автоматически будет приписан им в корпусе. Подобно этому такие формы, как *lexical* и *frequent*, будут помечены как прилагательные, поскольку они всегда являются членами этой категории, *the* и *a* будут помечены как артикли, *identify* и *see* – как глаголы и т. д. [38, 39]. Однако процесс нахождения соответствующих форм в корпусе и в снабженном пометами словаре не может быть использован для определения категорий всех форм, потому что некоторые формы могут быть членами более чем одной категории. В работе [5] представлена эффективная по трудоемкости и точности технология автоматизации построения базового аннотированного корпуса текстов, основанная на итеративном понижении степени многозначности их лексических единиц.

Предлагаемая принципиальная схема исходит из того, что уже построены лексико-грамматический классификатор и базовый эталонный словарь ЕЯ, образовано множество текстов, которые должны войти в БКТ, а также разработана программная часть многовариантного вероятностного лексико-грамматического анализатора:

- шаг 1: выделить из БКТ «минимально допустимое» по объему подмножество T_1 текстов для аннотирования (таковым является объем порядка 10^6 словоупотреблений);
- шаг 2: из корпуса текстов T_1 выделить его определенную часть T_2 и проаннотировать ее вручную; положить $T_3 = T_1 – T_2$;
- шаг 3: выделить из T_3 его определенную часть T_4 и, используя необходимые статистические данные, полученные из T_2, с помощью модуля многовариантного вероятностного лексико-грамматического анализа получить определенное число вариантов аннотирования для каждого предложения из T_4;
- шаг 4: для каждого слова каждого предложения из T_4 зафиксировать все полученные автоматически попарно различные варианты его аннотирования и на основе экспертного анализа, т. е. вручную, снять лексико-грамматическую многозначность слов, где она имеет место;
- шаг 5: положить $T_3 = T_3 – T_4^*$, где T_4^* – полученное на шаге 4 однозначно проаннотированное множество текстов;
- шаг 6: повторять шаги 3–5 до тех пор, пока получаемое на шаге 5 множество текстов T_3 не окажется пустым.

Было экспериментально установлено, что при объеме T_1 порядка 1,2−1,5 млн словоупотреблений полученные с такого объема статистические данные обеспечивают точность работы алгоритма автоматического лексико-грамматического аннотирования текста порядка 95 % [40], что является высоким показателем и позволяет оставшуюся неаннотированную на шаге 1 часть КТ обработать автоматически.

Отмечается, что объем T_2 на шаге 2 может составлять 100–150 тыс. словоупотреблений, а объем T_4 на шаге 3 фактически соответствует производительности эксперта-лингвиста, например, в один день осуществляющего снятие многозначности слов на шаге 4. Число сохраняемых на шаге 3 вариантов аннотирования зависит от конкретного языка, состава его лексико-грамматического классификатора и соображений компромисса между трудоемкостью (стоимостью) и точностью решения задачи.

Основной эффект при использовании описанного алгоритма достигается за счет того, что получаемое для каждого слова на шаге 4 множество попарно различных вариантов его аннотирования содержит меньше ЛГК, чем то их множество, которое соответствует слову в базовом словаре, причем часто первое из указанных множеств содержит единственный ЛГК.

Предложенный подход позволяет значительно уменьшить количество многозначных слов в представленном эксперту-лингвисту «многовариантном» корпусе, т. е. уменьшить количество ручной работы. По оценкам А. В. Чеусова, использование описанного выше алгоритма позволило уменьшить коли-

чество омонимичных слов русскоязычного корпуса текстов приблизительно в семь раз [5].

Основным назначением КТ является обеспечение тестирования алгоритмов автоматического ЛА текста, а также построения вероятностных алгоритмов такого анализа. В первом случае речь идет о выборе и использовании адекватных метрик при проведении процедуры оценки алгоритмов. Для этого подходят известные в сфере информационного поиска показатели полноты и точности. Например, А. В. Чеусовым в результате выполнения некоторой *i*-функциональности ЛП во входном тексте распознано (выделено) в качестве решения задачи n_i лингвистических объектов/отношений [5]. Из них правильно, как оказалось, распознаны m_i объектов/отношений, а в самом тексте их в действительности насчитывается k_i. Тогда точность P_i решения задачи определяется как

$$P_i = \frac{m_i}{n_i} \cdot 100\ \%,$$

а полнота R_i –

$$R_i = \frac{m_i}{k_i} \cdot 100\ \%.$$

На практике достаточно простые и одновременно обладающие высокой точностью и, как правило, низкой полнотой паттерны для распознавания фактов (семантических отношений: *Subject* – субъект, *Action* – акция, *Object* – объект, *Adjective* – атрибут действия, *Preposition* – обстоятельство действия или объекта, *Indirect Object* – непрямой объект действия, *Adverbial* – атрибут действия с функцией наречия и т. п.) в итоге обеспечивают качественное с точки зрения инженерии знаний решение задачи в очень больших по объему полнотекстовых БД. Это связано с тем, что в таких БД одни и те же факты выражаются многими разными, в том числе и простыми, грамматическими средствами ЕЯ. Во втором случае для построения вероятностных алгоритмов анализа с помощью аннотированного корпуса текстов могут быть получены важные статистические сведения: например, частоты слов, ЛГК, пары (Слово$_i$, ЛГК^j), пары (ЛГК$_i$, ЛГК$_j$) и т. п.

6. Лексический и лексико-грамматический анализ текстов естественного языка

Лексический и лексико-грамматический анализ являются начальными этапами обработки текста ЕЯ лингвистическим процессором. Согласно [5] на этапе лексического анализа текста прежде всего распознаются границы его слов и предложений. Здесь же частично или полностью решаются задачи распознавания имен собственных, аббревиатур, электронных адресов, цифровых и дру-

гих знаковых комплексов. Этот вид ЛА иначе называют сегментацией текста [7]. Существуют два основных подхода к решению этой задачи:

- подход, основанный на экспертных (лингвистических) правилах;
- подход, основанный на самообучающихся алгоритмах.

В первом случае описание конкретных языковых ситуаций, определяющих границы слов и предложений по знакам препинания, некоторому контексту и т. п., дается на основе экспертного анализа в виде так называемых лингвистических правил. В общем случае отдельное такое правило может быть представлено в виде «условие → операция». Здесь условие представляет собой лингвистический паттерн (шаблон), задающий конкретную закономерность. Паттерн – это формальная спецификация свойства набора примеров, определенная в терминах некоторого формального языка, например языка регулярных выражений или формальной грамматики, если число всевозможных исключений из общего правила может оказаться настолько велико, что их нельзя описать путем простого перечисления [41]. Если при анализе текста выполняется определенное условие, то над ним производится соответствующая операция, например фиксируется граница предложения. В машинной реализации решения задачи сегментации текста на основе лингвистических правил обычно используется аппарат теории конечных автоматов и трансдьюсеров [42].

В рамках второго подхода задача решается, например, путем анализа синтаксического контекста потенциальной границы предложения с использованием нейронных сетей [43], использованием модели максимальной энтропии [44], определением границы предложений на этапе лексико-грамматического анализа текста [45].

В любом случае, точность решения задачи лексического анализа должна быть максимально близка к абсолютной, чтобы не допустить ее резкого падения на последующих этапах анализа текста. И с этой точки зрения первый из указанных подходов является более перспективным, поскольку допускает наращивание объема используемых лингвистических правил, а следовательно, дальнейшее повышение точности решения задачи, при условии, что существуют эффективные алгоритмы обработки достаточно большого объема сложных по составу лингвистических правил, разработанных экспертами.

Как отмечалось ранее, задачей лексико-грамматического анализа текста является определение лексико-грамматической категории каждого его слова с учетом контекста. Множество всех лексико-грамматических категорий ЕЯ обычно задается заранее разработанным классификатором его лексико-грамматических свойств. Традиционно лексико-грамматическая классификация ЕЯ основана на разделении слов на части речи [46, 47]. При этом каждой из категорий дается некоторое условное обозначение (код, класс) в соответствии с выбранной системой кодирования. В результате ЛГА входному тексту ставится в соответствие в идеале единственная последовательность лексико-грамматических кодов, по одному для каждого его слова, обычно называемая кодовой цепочкой. Как правило, задача ЛГА решается в два этапа в предположении, что экспертами заранее построен так называемый эталонный словарь

языка, в котором каждому его слову приписаны возможные для него вне контекста ЛГК в соответствии с заданным классификатором лексико-грамматических свойств языка. На первом этапе поиском по эталонному словарю каждому слову входного текста ставятся в соответствие все возможные для него ЛГК, на втором – с учетом контекста снимается омонимия, т. е. для каждого слова определяется его единственно возможный ЛГК. Поскольку требуемый контекст не выходит за рамки одного предложения, то и задача ЛГА текста обычно решается последовательно, начиная с его первого предложения и заканчивая последним.

Рассмотрим в качестве примера следующее предложение на английском языке: *Low-fat vegetarian diet significantly reduces cholesterol level*. Если эталонный словарь построен, например, с использованием известного *LOB*-классификатора лексико-грамматических свойств слов английского языка [34], то словам данного предложения в результате поиска по такому словарю будут поставлены в соответствие следующие ЛГК: *Low-fat* (*JJ*), *vegetarian* (*JJ*, *NN*), *diet* (*NN*, *VB*), *significantly* (*RB*), *reduces* (*VBZ*), *cholesterol* (*NN*), *level* (*NN*, *VB*). А это в результате порождает восемь формально возможных кодовых цепочек:

- *JJ JJ NN RB VBZ NN NN*;
- *JJ NN NN RB VBZ NN NN*;
- *JJ JJ VB RB VBZ NN NN*;
- *JJ NN VB RB VBZ NN NN*;
- *JJ JJ NN RB VBZ NN VB*;
- *JJ NN NN RB VBZ NN VB*;
- *JJ JJ VB RB VBZ NN VB*;
- *JJ NN VB RB VBZ NN VB*.

Здесь *JJ* – имя прилагательное, *NN* – существительное единственного числа, *VB* – инфинитив глагола или личная форма глагола, совпадающая с инфинитивом в написании, *RB* – наречие, *VBZ* – глагол третьего лица единственного числа, «.» – знак препинания «точка». В результате снятия многозначности решением задачи ЛГА данного предложения должна быть названа первая из представленных кодовых цепочек.

Для решения задачи лексико-грамматического анализа текста также существует два основных подхода:

- аналитический;
- лингвистический.

В основу первого, например, может быть положена однородная цепь Маркова первого порядка с конечным числом состояний [48], которая является математической моделью грамматики, порождающей ограниченный естественный язык [45]. Точность ЛГА в этом случае составляет порядка 95 % при размере тренировочного корпуса в 1 млн словоупотреблений [49]. Что касается скорости ЛГА, то задача может быть решена за время, близкое к линейному, что, безусловно, является большим достоинством рассматриваемого подхода, в рамках которого реализованы такие известные системы ЛГА, как *PARTS Tagger*

[50], *Xerox Tagger* [51] и *TnT Tagger* [52]. Однако реализация алгоритма и его настройка требуют достаточно трудоемкой работы с тренировочным корпусом.

Лингвистический подход к ЛГА текста опирается на знания, выявленные экспертами в процессе качественного анализа естественного языка и предельно точно отражающие его истинные механизмы. Языковая компетенция в этом случае формализуется с помощью лингвистических правил, которые эксплицитно описывают конкретные языковые ситуации с указанием того, как необходимо проводить ЛГА текста в данной ситуации.

Каждое такое правило, как и в случае лексического анализа текста, имеет две составные части: условие и операцию. Условие здесь задает конкретную закономерность с использованием отдельных слов, их ЛГК, семантических классов и т. п. Если некоторый фрагмент анализируемого текста удовлетворяет условию, то над ним производится соответствующая операция. Например, правило установки ЛГК может выглядеть следующим образом [33, с. 67]:

$ATI < \_NN > BEZ|BER|BEDZ|BED \rightarrow NN$

Здесь слову, находящемуся между определенным артиклем (*ATI*) и личной формой глагола *be* (*BEx*), ставится в соответствие лексико-грамматический класс *NN* при условии, что он присутствует во множестве ЛГК, назначенных данному слову в эталонном словаре.

Подобным образом задаются правила преобразования слов в пределах парадигмы как с целью получения тех форм, которые отсутствуют в базовом словаре, так и для получения канонической формы слова. Лемматизация текста – это преобразование любой грамматической формы слова (существительного, прилагательного, глагола, и т. д.) в его нормальную (каноническую) форму. Например, для имен существительных в русском языке такой формой будет единственное число, именительный падеж; для прилагательных – единственное число, именительный падеж, мужской род; для глаголов, причастий, деепричастий – неопределенная форма глагола.

Рассмотрим следующее предложение на английском языке:

The companies offered comprehensive economic solutions in all spheres of business.

Результатом его лексико-грамматического анализа будет размеченное с помощью ЛГК предложение, содержащее лексемы в канонической форме:

The (*ATI*) *company* (*NN*) *offer* (*VB*) *comprehensive* (*JJ*) *economic* (*JJ*) *solution* (*NN*) *in* (*IN*) *all* (*ABN*) *sphere* (*NN*) *of* (*IN"*) *business* (*NN*) . (.)

Здесь в скобках также приведены соответствующие ЛГК лексико-грамматического классификатора *LOB Corpus* [34]: *ATI* – определенный артикль, *NN* – существительное единственного числа, *VB* – глагол, *JJ* – прилагательное, *IN* или *IN"* – предлог, *ABN* – определитель, «.» – знак препинания «точка».

Учет синонимических отношений между лексическими единицами становится возможным в случае использования существующих специализированных лингвистических ресурсов, включающих в себя списки синонимов, например, *WordNet* [53], многоязычной лексической БД *MModWN*, аналогичной по своей

структуре WordNet, описывающей концепты внешнего мира в форме пронумерованных понятий (синсетов), выраженных набором синонимичных слов и словосочетаний на всех языках из множества заданных, а также различными семантическими отношениями между концептами («общее – частное», «часть – целое», «группа – элемент» и т. д.), или гибридной базы знаний системы *OSTIS* [54] и модели представлений знаний различных типов в рамках фреймворка данной базы знаний. В этом случае процедура поиска и извлечения синонимов из соответствующего синсета, определяемого заданным словом входного текста, является нетрудоемкой.

В рамках лингвистического подхода, например, реализованы системы ЛГА *Taggit Tagger* [55], *Constraint Based Tagger* [56]. Количество используемых ими лингвистических правил колеблется от нескольких десятков до нескольких тысяч, а точность достигает 96 %, что несколько превосходит показатели статистических систем ЛГА.

7. Синтаксический анализ текстов естественного языка

Задачей синтаксического анализа текста ЕЯ является распознавание в каждом его предложении синтаксических отношений и представление их, как правило, в виде функционального или синтаксического дерева, в котором словам предложения указывается их грамматическая функция и определяется тип синтаксической связи между ними [5, 56]. При этом может иметь место синтаксическая многозначность, т. е. анализируемому предложению может соответствовать несколько вариантов синтаксического дерева. На рис. 8 и 9 представлен именно такой случай, который имеет место для предложения на английском языке *Fruit flies like bananas*.

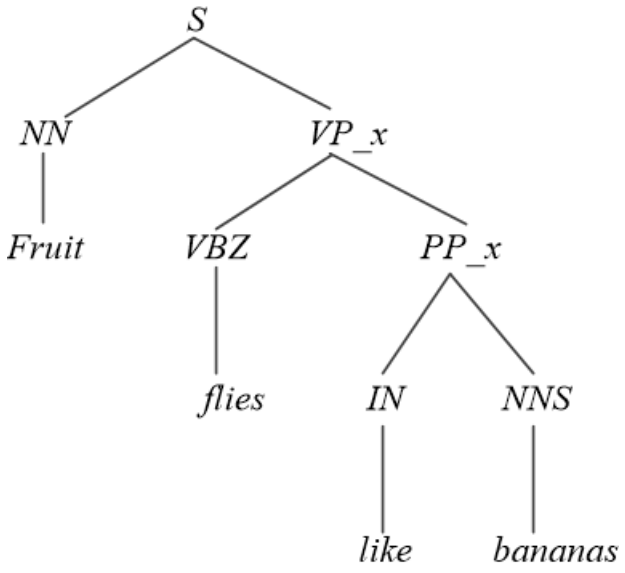

Рис. 8. Пример дерева синтаксического анализа
для предложения ЕЯ *Fruit flies like bananas*

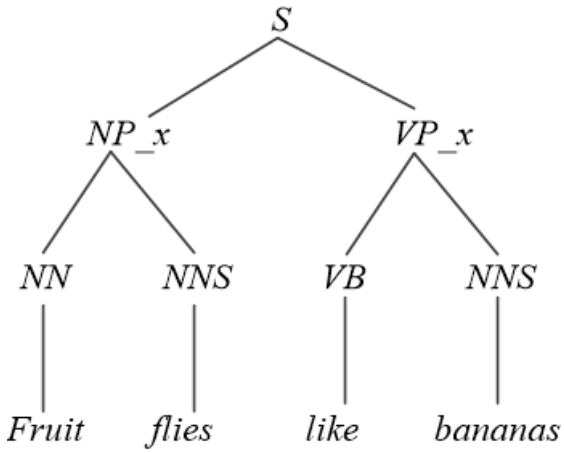

Рис. 9. Пример дерева синтаксического анализа
для предложения ЕЯ *Fruit flies like bananas*

Что касается алгоритмов синтаксического анализа, то большинство из них реализуется в соответствии с некоторой формальной грамматикой [57]. Наиболее исследованными теоретически и получившими широкое применение в практических приложениях являются грамматики Хомского [58, 59], в частности контекстно-свободные (КС) грамматики [7], трансформационные и *GB*-грамматики [60]. Последние описывают два уровня синтаксической структуры (глубинный и поверхностный), связанных посредством трансформаций. Грамматики Хомского по мощности оказались избыточными для описания синтаксиса ЕЯ и в то же время недостаточными для учета семантических аспектов языка.

КС-грамматики представляются в виде продукций (правил), ставящих в соответствие нетерминальным символам в своих левых частях (до знака «=») набор терминальных и нетерминальных символов в правых частях. Пример контекстно-свободных правил для простой грамматики русского языка приведен на рис. 10.

S → NP VP	**ADJECTIVE** → молодой
VP → VERB	**ADJECTIVE** → старого
VP → VERB NP	**ADJECTIVE** → лежащего
NP → NOUN	**NOUN** → лис
NP → ADJECTIVE NP	**NOUN** → волка
PP → PREPOSITION NP	**VERB** → видит
	VERB → лежит

Рис. 10. Пример контекстно-свободных правил

КС-правила в первой колонке описывают структуру нетерминальных символов ((*S* – предложение, *NP* – именная группа, *VP* – глагольная группа,

PP – предложная группа, *ADJECTIVE* – прилагательное, *NOUN* – существительное, *VERB* – глагол, *PREPOSITION* – предлог), во второй – словарь, т. е. соответствие между нетерминальными и терминальными символами. Подобная грамматика описывает, например, такие предложения, как:

- *лис видит волка*;
- *молодой лис видит старого волка*;
- *молодой лис видит старого лежащего волка*;
- *лис лежит*.

Достаточно просто расширить эту грамматику, чтобы представить в словаре лексику языка более полно. В данной грамматике выбор конкретного правила для построения глагольных групп (*VP*-правила) или именных групп (*NP*-правила) задан вариантами, гарантированный выбор между которыми сделать в рамках данного правила невозможно. Подобная грамматика относится к так называемым недетерминированным грамматикам.

Это привело к разработке вероятностных КС-грамматик, в которых каждому правилу ставится в соответствие некоторый параметр, отражающий вероятность того, что нетерминальный символ в левой части правила может быть расширен данным правилом. Такой механизм обеспечивает одно из возможных решений проблемы многозначности синтаксических деревьев. Были разработаны *HPSG* (*Head-Driven Phrase Structure Grammar*) [61], *SFG* (*Systemic Functional Grammar*) [62], *LFG* (*Lexical Functional Grammar*) [63], которые позволяют учитывать семантические признаки и даже всю лингвистически релевантную информацию на этапе синтаксического анализа текста. Использованию КС-грамматик сопутствует одна очень серьезная проблема – многократный анализ одних и тех же поддеревьев. И ее решение лежит в плоскости применения приемов динамического программирования. Одним из известных алгоритмов такого типа является *CYK* (*Cocke Younger Kasami*) [7]. Эффективные алгоритмы синтаксического анализа должны обеспечивать учет многозначности синтаксических деревьев, уметь оперировать элементами и отношениями различных уровней глубины ЕЯ и при этом максимально точно и быстро решать целевую задачу. Среди наиболее известных нынешних систем синтаксического анализа текста можно назвать разработки *Carnegie-Melon University* [64], компании *Connexor* [65], компании *Ling Soft* [66]. Так, на рис. 11 приведен результат выполнения автоматического синтаксического анализа предложения *Lots of people walk well* с помощью *Link Parser* [64]. Тогда как большинство систем синтаксического анализа используют структуры уровня именных и глагольных групп при построении дерева фразы, *Link Grammar*, лежащая в основе *Link Parser*, использует информацию о типах связей, которые каждое слово может иметь со словами, находящимися справа или слева, и несколько общих грамматических правил:

- *Xp* – связь между началом и концом предложения;
- *Sp* – связь между существительным и глаголом;
- *Wd* – связь между началом предложения и предложением;

- *Mp* – связь между именной группой и модифицирующей ее предложно-именной группой;
- *Jp* – связь между предлогом и относящейся к нему именной группой;
- *MVa* – связь между глаголом (прилагательным) и модификатором.

```
++++Time                                          0.00 seconds (6.24 total)
Found 1 linkage (1 with no P.P. violations)
  Unique linkage. cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

    +-----------------Xp-----------------+
    |        +--------Sp--------+        |
    +---Wd---+-Mp-+--Jp-+       +--MVa-+  |
    |        |    |     |       |      |  |
LEFT-WALL lots.n of people.p walk.v well.e .

Constituent tree:

(S (NP (NP Lots)
       (PP of
           (NP people)))
   (VP walk
       (ADVP well))
   .)
```

Рис. 11. Пример результата автоматического синтаксического анализа для предложения ЕЯ *Lots of people walk well* посредством *Link Parser*

8. Семантико-синтаксический анализ текстов естественного языка

Как отмечается в [15], попытка смоделировать понимание человеком текста привела к постановке вопроса о семантических структурах в языке и об их описании в терминах (понятиях) некоторой системы знаний, что в конечном счете и составляет суть семантического анализа текста.

Для представления результатов такого анализа чаще всего используют предикаты первого порядка, семантические сети, фреймы [7] и онтологии [67].

Одним из подходов к семантическому анализу текстов естественного языка (ТЕЯ) является подход, базирующийся на синтаксисе [68]. В данном случае на основе синтаксического дерева предложения и смысла слов строится соответствующая предложению предикатная формула.

Следует отметить, что синтаксические структуры не всегда хорошо подходят для целей семантического анализа. Так, например, синтаксические деревья часто содержат много связей, которые не играют существенной роли для семантического анализа ТЕЯ. Решением подобных проблем является использование семантических грамматик [69]. Правда, такие грамматики ориентированы на семантический анализ определенных предметных областей и правила составляются таким образом, чтобы необходимые семантические единицы опи-

сывались внутри одного правила. Эти правила предназначены сугубо для определенного (необходимого) уровня семантического анализа.

В случае если типы информации заранее фиксированы, семантический анализ текста может производиться согласно относительно простым и фиксированным шаблонам или фреймам, которые заполняются информацией из текста. Информация же, которая не подпадает под шаблоны и фреймы, просто игнорируется [70].

Ключевой проблемой на этапе семантического анализа ТЕЯ является проблема многозначности слов. Одним из классов алгоритмов разрешения данной проблемы являются обучающиеся алгоритмы, основанные, например, на анализе контекста многозначных слов, либо так называемые самонастраивающиеся алгоритмы [71, 72].

В настоящее время в рамках задачи семантического анализа текста речь обычно идет о некоторых частных, но важных ее подзадачах. Так, например, исходя из известных положений в области ИИ, согласно которым основными типами знаний являются объекты (классы объектов), факты и правила (причинно-следственные отношения), отображающие закономерности внешнего мира (предметной области), можно одной из целевых задач семантического анализа текста считать распознавание концептов и семантических отношений между ними типа «субъект – акция – объект» и «причина – следствие» с соответствующими атрибутами и с учетом синонимических и иерархических отношений. Например, предложение на английском языке *Today the user can download 10,000 papers from the Web by typing the word screen* содержит [73]:

1) концепты:
- *user*;
- *10,000 papers*;
- *Web*;
- *word screen*;

2) *CAO*1:
- субъект: *user*;
- акция: *download*;
- объект: *10,000 papers*;
- атрибут-предлог: *from*;
- непрямой объект: *Web*;
- атрибут-прилагательное: –;
- атрибут-наречие: –;

2) *CAO*2:
- субъект: *user*;
- акция: *type*;
- объект: *word ‘screen’*;
- атрибут-предлог: –;
- непрямой объект: –;
- атрибут-прилагательное: –;
- атрибут-наречие: –;

3) причина – следствие:

- *CAO*-причина (*CAO*2): *user – type – word screen*;
- *CAO*-следствие (*CAO*1): *use – download – 10,000 papers – from Web*.

Эти семантические компоненты и отношения являются универсальными, не зависящими от предметной области и конкретного языка, охватывают практически все информативные словоупотребления каждого предложения анализируемого текста, обеспечивают максимально возможное обобщение в виде паттернов лингвистических правил распознавания других требуемых конкретным приложением семантических компонентов и отношений, например таких, как время, параметр, месторасположение, состав, часть/целое и др., и в конечном счете обеспечивают распознавание не только тематического, но и логического содержания текста.

Отметим, что указанные компоненты знаний могут и не фиксироваться в выходной структуре в явном виде. В этом случае, как, например, в [15], фиксируются только отношения *SimpleNounPhrase, VerbPhrase, NounPhrase_additional, ComplexSentence*, предопределяющие знания основных типов. Результаты всех указанных этапов обработки текста образуют его лингвистический индекс (*LI*), который формально может быть представлен в виде

$$LI = \langle W, POS, SYN, REL \rangle,$$

где *W* – множество слов текста; *POS, SYN* и *REL* – отображения слов в множества их соответственно лексико-грамматических и синтаксических классов (меток), а также меток семантико-синтаксических отношений, предопределяющих знания основных типов.

Такая модель обладает одним очень важным свойством – она допускает естественное включение в себя новых компонент в соответствии с разрабатываемым приложением базового ЛП. Так, например, при решении задачи автоматизации инженерии знаний (если текст рассматривать как их основной источник) могут быть добавлены компоненты, соответствующие отображениям слов в множества типов основных и атрибутивных знаний [74]. В этом случае происходит естественный переход от *LI* текста к его семантическому индексу, рассматриваемому в качестве эффективной модели представления автоматически распознаваемых в тексте знаний, которая, во-первых, в отличие от известных моделей, не ориентирована на конкретные механизмы вывода, но может быть при необходимости трансформирована в любую из этих моделей, и, во-вторых, обеспечивает пользователю ЕЯ-доступ к распознаваемым в тексте знаниям [73].

Результаты такого анализа ТЕЯ могут быть представлены с помощью уже упоминавшейся выше и получившей широкое распространение в последнее время модели представления знаний – онтологии. Частным случаем аналогичной базы знаний о внешнем мире является известная лексическая база данных *WordNet* [53], разработанная в 1985 г. Дж. Миллером и его коллегами из Лаборатории когнитологии Принстонского университета (США). Популярность и широкое распространение *WordNet* обусловлены прежде всего ее существен-

ными содержательными и структурными характеристиками. Принстонский *WordNet* и все последующие варианты для других языков направлены на отображение состава и структуры лексической системы языка в целом, а не отдельных тематических областей. Например, актуальная версия *WordNet* охватывает общеупотребительную лексику современного английского языка – более 120 тыс. слов. Для существительных, например, задано 25 семантических групп: вещество, владение, время, действие (деятельность), животные (фауна), знание, количество, лицо, местоположение, намерение, общение, объект, отношение, пища, растение, совокупность, событие, создание человека, состояние, тело, форма, черта (свойство), чувство, явление, а для глаголов выделено 38 основных семантических областей: событие, состояние, действие, путь, образ действия, место, отрицание и т. д.

Основной конструктивной единицей в лексической базе данных является синонимический ряд (синсет) со стандартным определением словарного типа. Предполагается, что синсет в словаре представляет лексикализованное понятие. Если слово появляется более чем в одном синсете, то оно считается многозначным (при этом омонимия рассматривается как разновидность многозначности). Слова и синсеты связаны друг с другом парадигматическими отношениями, например:

- гиперонимии (*breakfast → meal*) (*завтрак → прием пищи*);
- гипонимии (*meal → lunch*) (*прием пищи → обед*);
- *has-member* (*faculty → professor*) (*факультет → профессор*);
- части целого, или *member-of* (*pilot → crew*) (*пилот → экипаж*);
- меронимии: *has-part* (*table → leg*) (*стол → ножка*);
- антонимии (*leader → follower*) (*лидер → последователь*).

Отношение синонимии отражается в объединении слов в класс эквивалентности – синсет, а отношение гипонимии – позволяет организовывать синсеты в виде семантических сетей. Для разных частей речи родовидовые отношения могут иметь дополнительные характеристики и различаться областью охвата.

По аналогии с *WordNet* в рамках проекта *EuroWordNet* было создано несколько подобных систем для ряда европейских языков (например, французского, немецкого, испанского, итальянского и т. д.). Все они были построены по единой модели, хотя могли содержать особые парадигматические отношения. Каждый из национальных тезаурусов *WordNet* отражает все особенности лексической системы соответствующего языка, сходство между языками выражается (неявно) в сходстве структур. Явным же образом национальные тезаурусы связаны при помощи общей понятийной схемы (*Top Ontology*), которая вначале включала в себя 63 общих понятия для существительных и глаголов, а затем была расширена совместными усилиями участников проекта до 1360. Элементы этой общей понятийной схемы связаны между собой при помощи системы межъязыковых индексов (*Inter-Lingual-Index*) в большую многоязычную лексическую базу данных. Посредством этого индекса можно переходить от одной языковой структуры *WordNet* к другой. Лексические противопоставления, не

входящие в обобщенный онтологический базис, сохраняются в конкретных языковых реализациях *WordNet*. Таким образом, рассмотренные выше онтологии, помимо указанной, могут применяться для решения задач автоматической обработки текста, например информационного поиска (*information retrieval*), построения вопросно-ответных систем (*question-answering systems*), автоматического машинного перевода (*machine translation*) и т. п. На выделение же указанных типов знаний, например, ориентирован семантический анализ ТЕЯ, осуществляемый ЛП широко известной современной интеллектуальной информационной системы [75].

9. Инструментальные аспекты в задачах автоматической обработки текста

Для эффективного решения задач автоматической обработки текста естественного языка в зависимости от специфики конкретных задач применяются языки программирования (скриптовые языки, например *JavaScript*, *Python* и т. п., общего назначения высокого уровня, например *Java*, *C#*/*C++* и т. д.), и связанные с ними стеки технологий. Ниже приведены некоторые основные сведения, которые помогут начать решать такие задачи, используя язык программирования *Python* и его расширения в виде специализированных библиотек.

Подходящий интерпретатор можно скачать с веб-сайта *python.org* [76], а с целью упрощения процесса установки и настройки библиотек можно использовать дистрибутивный пакет *Anaconda* [77], который уже включает в себя большинство библиотек, необходимых для работы в области решения задач ИИ и автоматической обработки ТЕЯ.

Для каждого выполняемого проекта может потребоваться определенная комбинация внешних библиотек, а иногда конкретных версий, которые отличаются от версий, используемых для других проектов. Если использовать *Python* для одиночного использования, то эти библиотеки будут конфликтовать, поэтому стандартным решением является применение виртуальных сред, т .е. изолированных сред *Python*, которые поддерживают собственные версии библиотек *Python* (в зависимости от того, как выполняется настройка среды самого *Python*).

Если не использовать *Anaconda*, то можно задействовать встроенный модуль *venv* или установить библиотеку *virtualenv*. Для создания и использования виртуальной среды (*Anaconda*) необходимо выполнить следующие команды, используя оболочку командной строки (*Anaconda Prompt*), например:

- для создания среды *Python 3.6* с именем *dsfs*: *conda create -n dsfs python= 3.6*;
- для получения информации о существующих средах: *conda info –envs*;
- для активации выбранной среды, например *dsfs*: *conda activate dsfs*;
- для деактивации активной среды: *conda deactivate*.

В *Python* используется пробельное форматирование, пример показан на рис. 12. Некоторые библиотеки не загружаются по умолчанию, это касается как

функционала, составляющего часть языка *Python*, так и сторонних компонент. Для того чтобы их можно было использовать, необходимо *импортировать* модули, которые их содержат. Импорт модуля показан на рис. 13, где *re* – это название модуля, содержащего функции и константы для работы с регулярными выражениями.

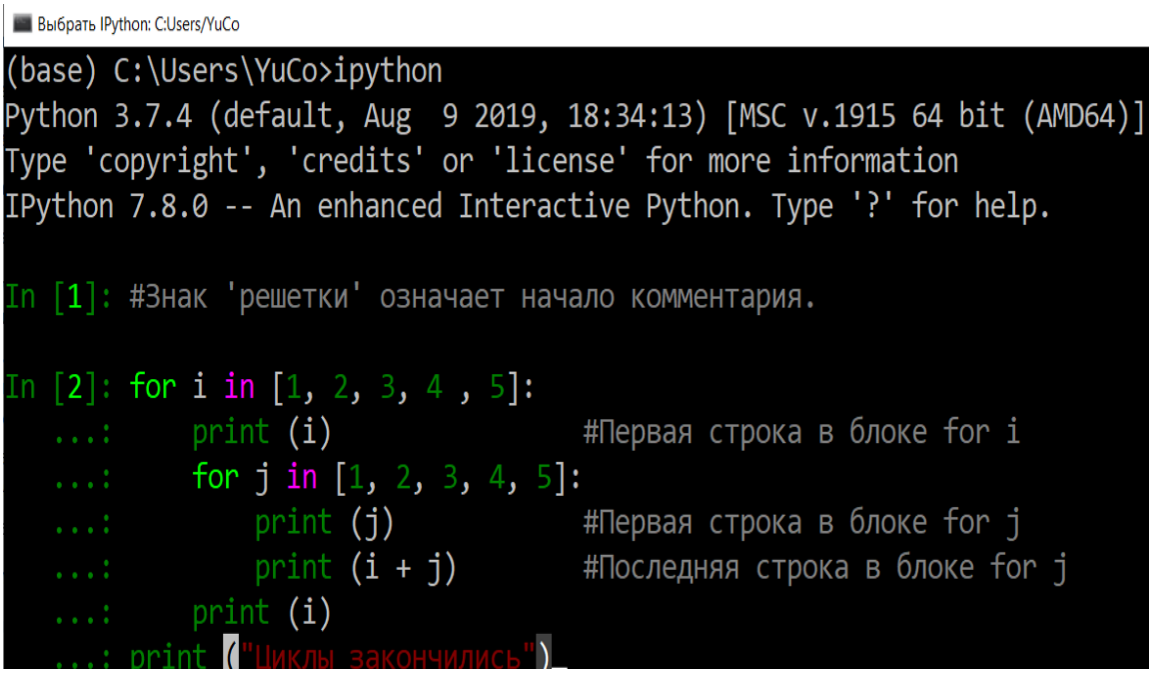

Рис. 12. Пример кода, содержащего вложенные циклы *for*

```
import re
my_regex = re.compile("[0-9]+", re.I)
```
Рис. 13. Пример кода, иллюстрирующий процесс импортирования модуля

Импортировав весь модуль таким способом, можно обращаться к функциям, предваряя их префиксом *re*. Если в коде переменная с именем *re* уже есть, то можно воспользоваться псевдонимом (рис. 14).

```
import re as regex
my_regex = regex.compile("[0-9]+", regex.I)
```
Рис. 14. Пример кода, иллюстрирующий использование псевдонима модуля

В *Python* функции определяются при помощи инструкции *def*, принимают ноль или несколько аргументов на входе и возвращают соответствующий результат на выходе. Их можно назначать переменным и передавать в другие функции так же, как любые другие аргументы. Параметрам функции, помимо этого, можно передавать аргументы по умолчанию, которые следует указывать только тогда, когда ожидается значение, отличающееся от значения по умолчанию (рис. 15).

```
def double(x):
    """
    Когда требуется, здесь размещают
    многострочный документирующий комментарий docstring,
    который поясняет, что именно функция вычисляет.
    Например, указанная функция умножает входящее значение на 2
    """
    return x * 2


def my_print(message = "мое сообщение по умолчанию"):
    print(message)

my_print ("привет") # Напечатает 'привет'
my_print()           # Напечатает 'мое сообщение по умолчанию'
```

Рис. 15. Пример кода, иллюстрирующий использование функций

Важными структурами данных в *Python* являются:

- список;
- кортеж;
- словарь;
- множество.

Кортеж можно рассматривать как упорядоченную совокупность (или коллекцию), похожую на массив в других языках программирования, но с дополнительной функциональностью (рис. 16).

Практически все, что можно делать со списком, не модифицируя его, можно делать и с кортежем. Вместо квадратных скобок кортеж оформляют круглыми скобками или вообще обходятся без них (рис. 17).

В словаре значения ассоциированы с ключами, что позволяет быстро извлекать значение, соответствующее конкретному ключу. Ключи словаря должны быть хешируемыми; в частности, в качестве ключей нельзя использовать списки. Если нужен составной ключ, то лучше воспользоваться кортежем или же найти способ преобразования ключа в строковое значение. Если, например, необходимо подсчитать количество появлений слов в документе и при этом требуется избежать повторяющихся проверок на присутствие ключа в словаре, то возможно использовать словари *defaultdict*, импортируя их из модуля *collections*. Некоторые основные операции работы со словарями показаны на рис. 18.

```
#СПИСКИ
integer_list = [1, 2, 3]                                    # Список целых чисел
heterogeneous_list = ["строка", 0.1, True]                  # Разнородный список
list_of_lists = [integer_list, heterogeneous_list, []]      # Список списков
list_length = len(integer_list)                             # Длина списка равна 3
list_sum =  sum(integer_list)                               # Сумма значений в списке равна 6
x = list (range (10))                                       # Задает список [0, 1, ..., 9]

# Получение доступа к элементу списка выполняется с помощью квадратных скобок
zero = x[0]                                 # равно 0, т. е. индекс 1-го элемента равен 0
one =  x[1]                                 # равно 1
nine = x[-1]                                # равно 9, взять последний элемент
eight = x[-2]                               # равно 8, взять предпоследний элемент
x[0] = -1                                   # Теперь x равен [-1, 1, 2, 3, ..., 9]

# Срез i:j означает все элементы от i (включительно) до j (не включая)
# Если опустить начало среза, то список будет "нарезан" с самого начала,
# а если не указывать конец среза, то список будет "нарезан" до самого конца

first_three = x[ :3]                        # Первые три равны [-1, 1, 2]
three_to_end = x[3:]                        # С третьего до конца равно [3, 4, ..., 9]
one_to_four = x[1:5]                        # С первого по четвертый: равно [1, 2, 3, 4]
last_three = x[-3:]                         # Последние три равны [7, 8, 9]
without_first_and_last = x[1:-1]            # Без первого и последнего равно [1, 2, …, 8]
copy_of_x = x[:]                            # Копия списка x равна [-1, 1, 2, …, 9]

# использование 3-го аргумента, отвечающего за шаг приращения значения
идентификатора элемента
every_third = x[::3]                        # [-1, 3, 6, 9]
five_to_three = x[5:2:-1]                   # [5, 4, 3]

# проверка принадлежности элемента списку
1 in [1, 2, 3]                              # True
0 in [1, 2, 3]                              # False

#Объединение списков
x = [1, 2, 3]
x.extend([4, 5, 6])                         # Теперь x равен [1, 2, 3, 4, 5, 6]
x = [1, 2, 3]
y= x + [4, 5, 6]                    # y равен [1, 2, 3, 4, 5, 6]; x остался без изменений

#Итеративное добавление элементов в список
x = [1, 2, 3]
x.append(0)                                 # Теперь x равен [1, 2, 3, 0]
y= x[-1]                                    # равно 0
z = len(x)                                  # равно 4

#Распаковка списка при известном значении количества его элементов
x, y = [1, 2]                               # Теперь x равен 1, y равен 2
_, y = [1, 2]                               # Теперь y равен 2, а первый элемент не нужен
```

Рис. 16. Пример кода, иллюстрирующий использование списков

```
# Кортежи
my_list = [1, 2]                          # Задать список
my_tuple = (1, 2)                         # Задать кортеж
other_tuple = 3, 4                        # Еще один кортеж
my_list[1] = 3                            # Теперь список my_list равен [1, 3]

# Обработка возникающего исключения (при помощи инструкций try и except)
try:
    my_tuple[1] = 3
except TypeError:
    print("Кортеж нельзя модифицировать")

# Применение кортежей для возврата нескольких значений из функций
# Функция возвращает сумму и произведение двух параметров
def sum_and_product(x, y):
    return (x + y), (x * y)

sp = sum_and_product(2, 3)                # равно (5, 6)
s, p = sum_and_product(5, 10)             # s равно 15, p равно 50

#Множественное присваивание
x, y= 1, 2                                # Теперь x равен 1, y равен 2
x, y= y, x                                # Обмен значениями переменных:
                                          # теперь x равен 2, y равен 1
```

Рис. 17. Пример кода, иллюстрирующий использование кортежей

```
#Словари
empty_dict = {}                                 # Задать словарь
empty_dict2 = dict()                            # Задать словарь
grades = {"Joel" : 80, "Tim" : 95}              # Литерал словаря (баллы за экзамены)

#получить доступ к элементу словаря по ключу
joels_grade = grades["Joel"]                    # равно 80

#если значение отсутствует, то возникнет исключение
try:
    kates_grade = grades["Kate"]
except KeyError:
    print ("Оценки для Кэйт отсутствуют!")

#проверка наличия ключа
kate_has_grade = "Kate" in grades               # False
joels_grade = grades.get("Joel", 0)             # равно 80
kates_grade = grades.get("Kate", 0)             # равно 0
no_ones_grade = grades.get("Никто")             # значение по умолчанию равно None

#присваивание значения по ключу
grades ["Tim"] = 99                             # Заменяет старое значение
grades["Kate"] = 100                            # Добавляет третью запись
num_students = len(grades)                      # равно 3

#возможное представление структурированных данных с помощью словаря
tweet = {
    "user" : "YuCo",
    "text" : "Решение задач AOT",
    "retweet_count" : 100,
    "hashtags" : ["#NLP", "#Python", "#ArtificialIntelligence"]
}

#обработка элементов структурированных данных
tweet_keys = tweet.keys()                       # Итерируемый объект для ключей
tweet_values = tweet.values()                   # Итерируемый объект для значений
tweet_items = tweet.items()                     # Итерируемый объект для кортежей
                                                # (ключ, значение)
"user" in tweet_keys                            # Возвращает True
"user" in tweet                                 # Проверка ключа, с использованием
                                                # быстрого in
"YuCo" in tweet_values      # True (медленный, но единственный способ проверки)

#defaultdict
from collections import defaultdict
word_counts = defaultdict(int)                  # int() возвращает 0
for word in document:
    word_counts[word] += 1
```

Рис. 18. Пример кода, иллюстрирующий использование словарей

Множество представляет собой неупорядоченную коллекцию уникальных значений. Они могут содержать только неизменяемые объекты: строки (*str*), целые числа (*int*), числа с плавающей точкой (*float*), кортежи, содержащие исключительно неизменяемые элементы. Несмотря на то что множества являются итерируемыми объектами, они не относятся к последовательностям и не поддерживают индексирование и сегментацию квадратными скобками []. Выполнение операции *in* происходит очень быстро, поэтому для обработки большой совокупности элементов с целью установления принадлежности некоторой последовательности множество подходит лучше, чем список. Некоторые основные операции работы со множествами показаны на рис. 19.

```
#Множества
s = set ()                          # Задать пустое множество
s.add (1)                           # Теперь s равно { 1 }
s.add(2)                            # Теперь s равно { 1, 2 }
s.add(2)                            # s как и прежде равно { 1, 2 }
x = len(s)                          # равно 2
y = 2 in s                          # равно True
z = 3 in s                          # равно False

# Список стоп-слов
stopwords_list = ["a", "an", "at"] + hundreds_of_other_words + [ "yet", "you"]
"zip" in stopwords_list             # False, но проверяется каждый элемент
# Множество стоп-слов
stopwords_set = set(stopwords_list)
"zip" in stopwords_set              # Очень быстрая проверка

item_list = [ 1, 2, 3, 1, 2, 3]     # Список
num_items = len(item_list)          # равно 6
item_set = set(item_list)           # Множество { 1, 2, 3 }
num_distinct_items = len(item_set)  # равно 3
distinct_item_list = list(item_set) # Список [ 1, 2, 3 ]
```

Рис. 19. Пример кода, иллюстрирующий использование множеств

Стандартная библиотека *Python* предоставляет богатую функциональность обработки текстовых и двоичных данных, проведения математических вычислений, программирования в функциональном стиле, работы с файлами и каталогами, хранения, сжатия, архивирования данных, поддержки сервисных функций операционной системы, функций работы с данными в сети Интернет (в том числе представленных в форматах *JSON* и *XML*), построения графического интерфейса с пользователем. На рис. 20 приведены некоторые модули стандартной библиотеки.

```
import collections  # Дополнительные структуры данных
                    # помимо списков, кортежей, словарей, множеств

import csv          # Обработка файлов с данными,
                    # разделенными запятыми

import datetime     # Операции с датой
import time         # и временем

import decimal      # Вычисления с фиксированной и плавающей точкой,
                    # включая финансовые вычисления

import doctest      # Простое модульное тестирование
                    # с использованием проверочных тестов
                    # и ожидаемых результатов,
                    # закодированных в doc-строках

import json         # Обработка данных в формате JSON
                    # для использования с веб-сервисами и базами данных

import math         # Распространенные математические константы и операции
import os           # Взаимодействие с операционной системой

import queue        # Структура данных, работающая по принципу
                    # "первым вошел, первым вышел" (FIFO)

import random       # Псевдослучайные числа
import re           # Регулярные выражения
import sqlite3      # Работа с реляционной базой данных SQLite
import statistics   # Функции математической статистики
import string       # Обработка строк

import sys  # Обработка аргументов командной строки;
            # потоки стандартного ввода-вывода, стандартный поток ошибок

import timeit       # Анализ быстродействия
```

Рис. 20. Некоторые модули стандартной библиотеки *Python*

На рис. 21 представлены популярные библиотеки *Python*, также используемые для решения задач автоматической обработки текста на различных их этапах.

```
import nltk  # Natural Language Toolkit - непосредственно для
             # решения задач обработки текстов естественного языка

import numpy       # Numerical Python предоставляет высокопроизводительную
                   # структуру данных ndarray для представления
                   # списков и матриц, а также функции обработки таких
                   # структур данных

import scipy       # Scientific Python – библиотека, встроенная в numpy.
                   # Добавляет функции для научных вычислений:
                   # интегралы, дифференциальные уравнения,
                   # расширенную обработку матриц и т.п.

import pandas      # Управление данными с использованием структуры ndarray
                   # и предоставлением одномерной и двумерной структуры
                   # данных Series и DataFrames соответственно

import matplotlib  # Визуализация и построение диаграмм
                   # (графики, точечные диаграммы, гистограммы, текст и т.п.)
```

Рис. 21. Некоторые популярные библиотеки *Python*

ПРАКТИЧЕСКАЯ ЧАСТЬ

Лабораторная работа № 1. Разработка автоматизированной системы формирования словаря естественного языка

Цель работы: освоить принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка.

Задачи лабораторной работы:

- ознакомиться с назначением, структурой и функциональностью, предоставляемой базовым ЛП для решения задачи автоматического лексического и лексико-грамматического анализа ТЕЯ;
- закрепить навыки программирования при решении задач автоматической обработки ТЕЯ.

Методические указания. Требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки, необходимые в рамках следующих базовых требований к разрабатываемому приложению:

- входные данные – текст заданного естественного языка;
- выходные данные – перечень лексем с дополнительной информацией согласно заданию;
- взаимодействие с пользователем посредством графического интерфейса (интуитивно понятного и дружественного пользователю);
- наличие системы средств помощи пользователю;
- обеспечение возможности построения, сохранения, просмотра, редактирования, пополнения, фильтрации и поиска по заданному условию, документирования автоматически получаемого словаря либо заданной его части;
- поддержка различных форматов представления входных данных (*TXT*, *RTF*, *PDF*, *DOC*, *DOCX*).

Рекомендуется использовать функциональность стандартной, а также специализированных библиотек языка программирования *Python* для обработки естественного языка, например *nltk*.

Вариант задания выбирается студентом самостоятельно (табл. 1) и согласовывается с преподавателем. Средства разработки выбираются студентом самостоятельно. Защита лабораторной работы предполагает демонстрацию работоспособности всех реализованных функций в соответствии с требованиями.

Требования к отчету. В отчете представить, в том числе графически, используя такие программные средства, как *Microsoft Visio* или *Draw.io*:

- структурно-функциональную схему разработанного приложения;
- описание структур хранения данных, алгоритмов их обработки, необходимых для реализации базовых требований к разработанной программе;
- оценку быстродействия приложения;
- выводы по работе и по перспективам использования приложения.

Отчет представить для проверки в электронном виде.

Задание 1. Список слов, упорядоченный по алфавиту и включающий в себя как лексемы, так и словоформы, с указанием частоты встречаемости каждой из форм. Для словоформ пользователю должна быть предоставлена возможность вводить дополнительную морфологическую информацию, а именно отнесение слова к соответствующей части речи, указание рода, числа, падежа и т. п. При этом морфологическую информацию допустимо оформить как отдельную неформатированную запись, т. е. просто текст, который пользователь может оформлять произвольным образом.

Задание 2. Список слов, упорядоченный по алфавиту и включающий в себя только лексемы с дополнительно оформленными записями для образования словоформ. В этих записях должна храниться следующая информация: основа слова; часть речи; окончания слова, соотнесенные с соответствующей морфологической информацией (род, падеж, число и т. п.). При работе с таким словарем должны быть обеспечены средства генерации той или иной словоформы в соответствии с введенными «правилами».

Задание 3. Список слов, упорядоченный по алфавиту и включающий в себя только лексемы с дополнительно оформленными записями для образования словосочетаний. В этих записях должны храниться слова (точнее, ссылки на эти слова), с которыми данное слово может сочетаться. При этом возможно обеспечение автоматизированного извлечения из исходных текстов типовых словосочетаний с их последующей обработкой. Дополнительно можно добавить средства синтеза словосочетаний, но при этом в словарь необходимо будет включить правила формирования словоформ (см. задание 2).

Задание 4. Список слов, упорядоченный по алфавиту и включающий в себя только лексемы с дополнительно оформленными записями о месте и роли данного слова в составе предложения. К такой информации относится описание того, каким членом предложения может быть данное слово и в какой форме (падеж, число, время и т. п.). Например, если это существительное в именительном падеже, то оно может выступать в роли подлежащего; если это существительное в родительном падеже, то оно может быть дополнением; если это прилагательное, то оно может быть определением и т. п.

Варианты заданий для выполнения приведены в табл. 1.

Таблица 1

Варианты индивидуальных заданий

Номер индивидуального задания	**Язык текста**	**Формат входного документа**	**Вариант задания**
1	**2**	**3**	**4**
1	Русский	*TXT, RTF*	Задание 1
2	Русский	*PDF*	Задание 1
3	Русский	*DOC, DOCX*	Задание 1
4	Русский	*TXT, RTF*	Задание 2
5	Русский	*PDF*	Задание 2

Окончание табл. 1

1	2	3	4
6	Русский	*DOC, DOCX*	Задание 2
7	Русский	*TXT, RTF*	Задание 3
8	Русский	*PDF*	Задание 3
9	Русский	*DOC, DOCX*	Задание 3
10	Русский	*TXT, RTF*	Задание 4
11	Русский	*PDF*	Задание 4
12	Русский	*DOC, DOCX*	Задание 4
13	Английский	*TXT, RTF*	Задание 1
14	Английский	*PDF*	Задание 1
15	Английский	*DOC, DOCX*	Задание 1
16	Английский	*TXT, RTF*	Задание 2
17	Английский	*PDF*	Задание 2
18	Английский	*DOC, DOCX*	Задание 2
19	Английский	*TXT, RTF*	Задание 3
20	Английский	*PDF*	Задание 3
21	Английский	*DOC, DOCX*	Задание 3
22	Английский	*TXT, RTF*	Задание 4
23	Английский	*PDF*	Задание 4
24	Английский	*DOC, DOCX*	Задание 4

Лабораторная работа № 2. Синтаксический анализ текстов естественного языка

Цель работы: освоить принципы разработки прикладных сервисных программ для решения задачи автоматического синтаксического анализа текста естественного языка.

Задачи лабораторной работы:

- ознакомиться с назначением, структурой и функциональностью, предоставляемой базовым ЛП для решения задачи автоматического синтаксического анализа ТЕЯ;
- закрепить навыки программирования при решении задач автоматической обработки ТЕЯ.

Методические указания. Требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки, необходимые в рамках следующих базовых требований к разрабатываемому приложению:

- входные данные – текст заданного естественного языка;
- выходные данные – структуры, полученные при проведении автоматического синтаксического анализа предложений входного текста согласно варианту задания;
- взаимодействие с пользователем посредством графического интерфейса (интерфейс должен быть интуитивно понятным и дружественным пользователю);
- наличие системы средств помощи пользователю;

- обеспечение возможности построения, сохранения, просмотра, редактирования, документирования автоматически получаемого результата либо заданной его части;

- поддержка различных форматов представления входных данных (*TXT*, *RTF*, *PDF*, *HTML*, *DOC*, *DOCX*).

Рекомендуется использовать функциональность стандартной, а также специализированных библиотек языка программирования *Python* для обработки естественного языка, например *nltk*.

Вариант задания выбирается студентом самостоятельно (табл. 2) и согласовывается с преподавателем. Средства разработки выбираются студентом самостоятельно. Защита лабораторной работы предполагает демонстрацию работоспособности всех реализованных функций в соответствии с требованиями.

Требования к отчету. В отчете представить, в том числе графически, используя такие программные средства, как *Microsoft Visio* или *Draw.io*:

- структурно-функциональную схему разработанного приложения;
- описание структур хранения данных, алгоритмов их обработки, необходимых для реализации базовых требований к разработанной программе;
- оценку быстродействия приложения;
- выводы по работе и по перспективам использования приложения.

Отчет представить для проверки в электронном виде.

Варианты заданий для выполнения приведены в табл. 2.

Таблица 2

Варианты индивидуальных заданий

Номер индивидуального задания	**Язык текста**	**Формат входного документа**
1	Русский	*TXT*
2	Русский	*RTF*
3	Русский	*PDF*
4	Русский	*DOC*
5	Русский	*DOCX*
6	Английский	*HTML*
7	Английский	*TXT*
8	Английский	*RTF*
9	Английский	*PDF*
10	Английский	*DOC*
11	Английский	*DOCX*
12	Английский	*HTML*

Лабораторная работа № 3. Семантико-синтаксический анализ текстов естественного языка

Цель работы: освоить принципы разработки прикладных сервисных программ для решения задачи автоматического семантико-синтаксического анализа текста естественного языка.

Задачи лабораторной работы:

- ознакомиться с назначением, структурой и функциональностью, предоставляемой базовым ЛП для решения задачи автоматического семантико-синтаксического анализа ТЕЯ;
- закрепить навыки программирования при решении задач автоматической обработки ТЕЯ.

Методические указания. Требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки, необходимые в рамках следующих базовых требований к разрабатываемому приложению:

- входные данные – текст заданного естественного языка;
- выходные данные – структуры, полученные при проведении автоматического семантико-синтаксического анализа предложений входного текста согласно варианту задания;
- взаимодействие с пользователем посредством графического интерфейса (интерфейс должен быть интуитивно понятным и дружественным пользователю);
- наличие системы средств помощи пользователю;
- обеспечение возможности построения, сохранения, просмотра, редактирования, документирования автоматически получаемого результата либо заданной его части;
- поддержка различных форматов представления входных данных (*TXT, RTF, PDF, HTML, DOC, DOCX*).

Рекомендуется использовать функциональность стандартной, а также специализированных библиотек языка программирования *Python* для обработки естественного языка, например *nltk, WordNet, EuroWordnet, ConceptNet, FrameNet*.

Вариант задания выбирается студентом самостоятельно (табл. 3) и согласовывается с преподавателем. Средства разработки выбираются студентом самостоятельно. Защита лабораторной работы предполагает демонстрацию работоспособности всех реализованных функций в соответствии с требованиями.

Требования к отчету. В отчете представить, в том числе графически, используя такие программные средства, как *Microsoft Visio* или *Draw.io*:

- структурно-функциональную схему разработанного приложения;
- описание структур хранения данных, алгоритмов их обработки, необходимых для реализации базовых требований к разработанной программе;
- оценку быстродействия приложения;
- выводы по работе и по перспективам использования приложения.

Отчет представить для проверки в электронном виде.
Варианты заданий для выполнения приведены в табл. 3.

Таблица 3

Варианты индивидуальных заданий

Номер индивидуального задания	**Язык текста**	**Формат входного документа**
1	Русский	*TXT*
2	Русский	*RTF*
3	Русский	*PDF*
4	Русский	*DOC*
5	Русский	*DOCX*
6	Английский	*HTML*
7	Английский	*TXT*
8	Английский	*RTF*
9	Английский	*PDF*
10	Английский	*DOC*
11	Английский	*DOCX*
12	Английский	*HTML*

Лабораторная работа № 4. Диалоговая система с поддержкой естественного языка

Цель работы: освоить принципы разработки диалоговых систем с поддержкой естественного языка.

Задачи лабораторной работы:

- изучить основы создания диалоговых систем с поддержкой естественного языка;
- закрепить навыки программирования при решении задач организации диалогового взаимодействия с поддержкой естественного языка.

Методические указания. Требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки, необходимые в рамках следующих базовых требований к разрабатываемому приложению:

- входные данные – текстовое сообщение на заданном естественном языке;
- выходные данные – автоматическая реакция системы на входное сообщение на естественном языке путем формирования ответного сообщения согласно варианту задания;
- взаимодействие с пользователем посредством графического интерфейса (интерфейс должен быть интуитивно понятным и дружественным пользователю);
- наличие системы средств помощи пользователю;

- обеспечение возможности ведения диалога с пользователем на естественном языке, сохранения, просмотра, редактирования истории диалога либо заданной его части.

Рекомендуется использовать результаты выполнения лабораторных работ № 1–3, функциональность стандартной, а также специализированных библиотек языка программирования *Python* для обработки естественного языка, например *nltk*.

Вариант задания выбирается студентом самостоятельно (табл. 4) и согласовывается с преподавателем. Средства разработки выбираются студентом самостоятельно. Защита лабораторной работы предполагает демонстрацию работоспособности всех реализованных функций в соответствии с требованиями.

Требования к отчету. В отчете представить, в том числе графически, используя такие программные средства, как *Microsoft Visio* или *Draw.io*:

- структурно-функциональную схему разработанного приложения;
- описание структур хранения данных, алгоритмов их обработки, необходимых для реализации базовых требований к разработанной программе;
- оценку быстродействия приложения;
- выводы по работе и по перспективам использования приложения.

Отчет представить для проверки в электронном виде.

Варианты заданий для выполнения приведены в табл. 4.

Таблица 4

Варианты индивидуальных заданий

Номер индивидуального задания	**Язык текста**	**Предметная область**
1	**2**	**3**
1	Русский	Кинематограф
2	Русский	Кулинария
3	Русский	Растения
4	Русский	Животные
5	Русский	Медицина
6	Русский	Литература
7	Русский	Транспорт
8	Русский	Спорт
9	Русский	Музыка
10	Русский	Досуг
11	Русский	Услуги
12	Русский	Недвижимость
13	Английский	Кинематограф
14	Английский	Кулинария
15	Английский	Растения
16	Английский	Животные
17	Английский	Медицина
18	Английский	Литература

Окончание табл. 4

1	2	3
19	Английский	Транспорт
20	Английский	Спорт
21	Английский	Музыка
22	Английский	Досуг
23	Английский	Услуги
24	Английский	Недвижимость

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Е. И. Большакова [и др.]. – М. : МИЭМ, 2011. – 272 с.

2. Елисеева, О. Е. Естественно-языковой интерфейс интеллектуальных систем : учеб. пособие / О. Е. Елисеева ; под науч. ред. В. В. Голенкова. – Минск : БГУИР, 2009. – 151 с.

3. Большой энциклопедический словарь. Языкознание / под ред. В. Н. Ярцевой. – М. : Большая Рос. энцикл., 1998. – 685 с.

4. Соснина, Е. П. Введение в прикладную лингвистику : учеб. пособие / Е. П. Соснина. – 2-е изд., испр. и доп. – Ульяновск : УлГТУ, 2012. – 110 с.

5. Чеусов, А. В. Разработка алгоритмов и технологии построения многоязычного базового лингвистического процессора : дис. … канд. техн. наук : 05.13.17 / А. В. Чеусов. – Минск, 2013. – 116 л.

6. Совпель, И. В. Автоматизированная переработка текста на основе воспроизводящего моделирования лингвистических объектов и процессов : дис. … д-ра техн. наук : 05.25.05 / И. В. Совпель. – Минск, 1991. – 360 л.

7. Jurafsky, D. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition / D. Jurafsky, J. H. Martin. – New Jersey : Prentice Hall, 2000. – 934 p.

8. Лингвистическое обеспечение системы ЭТАП-2 / Ю. Д. Апресян [и др.]. – М. : Наука, 1989. – 296 с.

9. Кулагина, О. С. Исследования по машинному переводу / О. С. Кулагина. – М. : Наука, 1979. – 324 с.

10. Жолковский, А. К. Модель «Смысл – Текст» / А. К. Жолковский // Энциклопедия кибернетики : в 2 т. / под ред. В. М. Глушкова. – К. : Укр. Совет. энцикл., 1974. – Т. 2. – С. 46–48.

11. Звегинцев, В. А. Мысли о лингвистике / В. А. Звегинцев. – М. : ЛКИ, 2008. – 336 с.

12. Chomsky, N. Syntactic Structures / N. Chomsky. – The Hague : Mouton & Co, 1957. – 117 p.

13. Manning, C. Foundations of Statistical Natural Language Processing / C. Manning, H. Schütze. – Cambridge : The MIT Press, 1999. – 860 p.

14. Люгер, Дж. Искусственный интеллект: стратегии и методы решения сложных проблем / Дж. Люгер. – М. : Изд. дом «Вильямс», 2005. – 864 с.

15. Воронков, Н. В. Методы, алгоритмы и модели систем автоматического реферирования текстовых документов : дис. … канд. техн. наук : 05.13.17 / Н. В. Воронков. – Минск, 2007. – 165 л.

16. Zhou, L. Mobile personal information management agent: supporting natural language interface and application integration / L. Zhou, A.S. Mohammed, D. Zhang // Information Processing and Management. – 2012. – V. 48, № 1. – P. 23–31.

17. Codd, E. Seven steps to rendezvous with the casual user / E. Codd // Data Base Management / eds.: J. Kimbie, K. Koffeman. – North-Holland, 1974. – P. 179–200.

18. Li, F. NaLIR: an interactive natural language interface for querying relational databases / F. Li, H.V. Jagadish // Proc. 2014 ACM SIGMOD Int. Conf. on Management of Data. – Snowbird, 2014. – P. 709–712.

19. Distributed representations of words and phrases and their compositionality / T. Mikolov [et al.] // Proc. 26th Int. Conf. on Neural Information Processing Systems. – USA, 2013. – P. 3111–3119.

20. Посевкин, Р. В. Применение семантической модели базы данных при реализации естественно-языкового пользовательского интерфейса / Р. В. Посевкин // Науч.-техн. вестн. информ. технологий, механики и оптики. – 2018. – Т. 18, № 2. – С. 262–267.

21. Евдокимова, И. С. Естественно-языковые системы : курс лекций / И. С. Евдокимова. – Улан-Удэ : Изд-во ВСГТУ, 2006. – 92 с.

22. Bengfort, B. Applied Text Analysis with Python / B. Bengfort, R. Bilbro, T. Ojeda. – Boston : O'Reilly Media, Inc., 2018. – 368 p.

23. Woudenberg, A. F. A Chatbot Dialogue Manager : Master Education in Computer Science : T76318 / A. F. Woudenberg. – Netherlands : Open University of the Netherlands, 2014. – 132 p.

24. Касевич, В. Б. Элементы общей лингвистики / В. Б. Касевич. – М. : Наука, 1977. – 183 с.

25. Лингвистический энциклопедический словарь / под ред. В. Н. Ярцевой. – М. : Совет. энцикл., 1990. – 685 с.

26. Попов, Э. В. Общение с ЭВМ на естественном языке / Э. В. Попов. – М. : Наука, 1982. – 360 с.

27. Валгина, Н. С. Современный русский язык. Синтаксис : учебник / Н. С. Валгина. – 4-е изд., испр. – М. : Высш. шк., 2003. – 416 с.

28. Копач, О. И. Общее языкознание : электрон. учеб.-метод. комплекс для специальности 1-21 06 01-01 «Соврем. иностр. языки (преподавание)» / О. И. Копач ; БГУ, фак. социокультур. коммуникаций, каф. компьютер. лингвистики и лингводидактики. – Минск : БГУ, 2017. – 239 с.

29. Искусственный интеллект. Справочник : в 3 т. – М. : Радио и связь, 1990. – Т. 2 : Модели и методы / под ред. Д. А. Поспелова. – 304 с.

30. Лингвистическое обеспечение системы ЭТАП-2 / Ю. Д. Апресян [и др.]. – М. : Наука, 1989. – 296 с.

31. Пиотровский, Р. Г. Текст, машина, человек / Р. Г. Пиотровский. – Л. : Наука, 1975. – 327 с.

32. Трауб, Д. Информация, неопределенность, сложность / Д. Трауб, Г. Васильковский, Х. Вожьяновский. – М. : Мир, 1988. – 183 с.

33. Крапивин, Ю. Б. Методы и алгоритмы автоматического распознавания воспроизведенных фрагментов текстовых документов : дис. … канд. техн. наук : 05.13.17 / Ю. Б. Крапивин. – Минск, 2019. – 129 л.

34. The LOB tag set [Electronic resource]. – 2021. – Mode of access: http://www.hit.uib.no/icame/lobman/lob3.html.

35. Совпель, И. В. Автоматическое распознавание основных типов знаний в текстовых документах / И. В. Совпель // Искусств. интеллект. – 2007. – № 3. – С. 328–332.

36. Connexor – Natural Knowledge [Electronic resource]. – 2021. – Mode of access: http://www.connexor.com.

37. About WordNet [Electronic resource]. – 2021. – Mode of access: http://wordnet.princeton.edu/.

38. Захаров, В. П. Корпусная лингвистика : учебник / В. П. Захаров, С. Ю. Богданова. – Иркутск : ИГЛУ, 2011. – 161 с.

39. Finegan, E. LANGUAGE: its structure and use / E. Finegan. – N. Y. : Harcourt Brace College Publishers, 2004. – 161 p.

40. Воронцов, А. В. Лингвистические аспекты разработки системы автоматического лексико-грамматического анализа англоязычных текстов : дис. … канд. филол. наук : 10.02.21 / А. В. Воронцов. – Минск, 2008. – 173 л.

41. Городецкий, В. Современное состояние технологии извлечения знаний из баз и хранилищ данных (часть 1) / В. Городецкий, В. Самойлов, А. Малов // AI News. – 2002. – № 3. – С. 3–12.

42. Брауер, В. Введение в теорию конечных автоматов / В. Брауер. – М. : Радио и связь, 1987. – 392 с.

43. Palmer, D. Adaptive sentence boundary disambiguation / D. Palmer, M. Hearst // Proceedings of the Conference on Applied Natural Language Processing (ANLP). – Stuttgart, 1994. – P. 78–83.

44. Reynar, J. A maximum entropy approach to identifying sentence boundaries / J. Reynar, A. Ratnaparkhi // Proceedings of the Conference on Applied Natural Language Processing (ANLP). – Washington, 1997. – P. 16–19.

45. Mikheev, A. Tagging sentence boundaries / A. Mikheev // Proceedings of the Conference on Applied Natural Language Processing (ANLP). – Washington, 2000. – P. 264–271.

46. Bolshakov, I. A. Computational linguistics: models, resources, applications / I. A. Bolshakov, A. Gelbukh. – Mexico, 2004. – 186 p.

47. Eeg-Olofsson, M. Word-class tagging: Some computational tools : doctoral thesis / M. Eeg-Olofsson. – Sweden, 1991. – 99 p.

48. Francis, W. Frequency analysis of English usage: Lexicon and grammar / W. Francis, H. Kucera. – Boston : Houghton Mifflin, 1982. – 561 p.

49. Leech, G. Claws4: The tagging of the British National Corpus / G. Leech, R. Garside, M. Bryant // Proceedings of the 15th International Conference on Computational Linguistics. – Kyoto, 1994. – P. 622–628.

50. Church, K. A stochastic parts program and noun phrase parser for unrestricted text / K. Church // Proceedings of the second conference on Applied natural language processing. – Austin, 1988. – P. 136–143.

51. A practical part-of-speech tagger / D. Cutting [et al.] // Proceedings of the third conference on Applied natural language processing. – Trento, 1992. – P. 133–140.

52. Brants, T. TnT – a statistical part-of-speech tagger / T. Brants // Proceedings of the 6th Applied NLP Conference. – Seattle, WA, 2000. – P. 224–231.

53. WordNet Documentation [Electronic resource]. – 2021. – Mode of access: http://wordnet.princeton.edu/documentation.

54. Davydenko, I. Semantic models, method and tools of knowledge bases coordinated development based on reusable components / I. Davydenko // Открытые технологии проектирования интеллектуальных систем = Open semantic technologies for intelligent systems (OSTIS-2018) : материалы Междунар. науч.-техн. конф. (Минск, 15–17 февр. 2018 г.) / БГУИР ; редкол.: В. В. Голенков (отв. ред.) [и др.]. – Минск : БГУИР, 2018. – С. 99–118.

55. Gorrell, P. Syntax and Parsing / P. Gorrell. – UK : Cambridge Univ. Press, 1995. – 161 p.

56. Chanod, J.-P. Tagging french: comparing a statistical and a constraint-based method / J.-P. Chanod, P. Tapanainen // Proceedings of the 7th Conference on European Chapter of the Association for Computational Linguistics. – Dublin, 1995. – P. 149–157.

57. Radford, A. Syntactic Theory and the Structure of English. A minimalist approach / A. Radford. – UK : Cambridge Univ. Press, 1997. – 558 p.

58. Ахо, А. Теория синтаксического анализа перевода и компиляции : в 2 т. / А. Ахо, Д. Ульман. – М. : Мир, 1978. – Т. 1 : Синтаксический анализ. – 613 с.

59. Fillmore, C. The Case for Case / C. Fillmore // Universals in Linguistic Theory. – Rinehart and Winston, 1968. – P. 1–88.

60. Бейлин, Д. Краткая история генеративной грамматики / Д. Бейлин // Современная американская лингвистика. Фундаментальные направления. – М. : МГУ, 1997. – С. 13–57.

61. Pollard, C. Head-Driven Phrase Structure Grammar / C. Pollard, I. A. Sag. – Chicago, 1994. – 454 p.

62. Halliday, M. A. An introduction to Functional Grammar / M. A. Halliday. – USA : A Hodder Arnold Publication, 1994. – 472 p.

63. Netter, K. Constraint-based grammar models / K. Netter // Proceedings of the European Summer School on Logic, Language and Information ESSLLI-96. – Prague, 1996. – P. 1–10.

64. Temperley, D. Link Grammar / D. Temperley, D. Sleator, J. Lafferty // Carnegie Mellon University [Electronic resource]. – 2021. – Mode of access: http://www.link.cs.cmu.edu/link/.

65. Home page [Electronic resource] // Connexor. – 2021. – Mode of access: http://www.connexor.com.

66. Home page [Electronic resource] // Ling Soft. – 2021. – Mode of access: http://www.lingsoft.fi.

67. Гаврилова, Т. А. Онтологический подход к управлению знаниями при разработке корпоративных информационных систем / Т. А. Гаврилова // Новости искусств. интеллекта. – 2003. – № 2. – С. 24–30.

68. Bach, E. An extension of classical transformational grammar / E. Bach // Problems of Linguistic Metatheory: proceedings of the 1976 Conference, Michigan, 1976. – Michigan, 1976. – P. 183–224.

69. Brown, J. S. Multiple representations of knowledge for tutorial reasoning / J. S. Brown, R. R. Burton // Representation and Understanding: Studies in Cognitive Science / D. G. Bobrow [et al.] ; ed.: D. G. Bobrow [et al.]. – N. Y., 1975. – P. 311–350.

70. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text / J. R. Hobbs [et al.] // Finite-State Devices for Natural Language Processing / E. Roche, Y. Schabes ; ed.: E. Roche, Y. Schabes. – USA, 1997. – P. 383–406.

71. Hearst, M. A. Noun homograph disambiguation / M. A. Hearst // The new OED and Text Research: proceedings of the 7th Annual Conference, Oxford, September 1991. – Oxford, 1991. – P. 1–19.

72. Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methids / D. Yarowsky // Association for Computational Linguistics: proceedings of the 33th Annual Meeting, Cambridge, 26–30 June 1995. – Cambridge, 1995. – P. 189–196.

73. Антонов, С. Г. К задаче разработки лингвистических процессоров / С. Г. Антонов, И. В. Совпель // Вестн. БрГТУ, сер. «Физика, математика, информатика». – 2013. – №. 5. – С. 36–38.

74. Постаногов, Д. Ю. Автоматическая обработка естественного языка в задаче инженерии знаний и доступа к ним : дис. … канд. техн. наук : 05.13.17 / Д. Ю. Постаногов. – Минск, 2012. – 134 л.

75. Goldfire // Invention Machine Corporation [Electronic resource]. – 2021. – Mode of access: https://gfi.goldfire.com.

76. Download Python | Python.org [Electronic resource]. – 2021. – Mode of access: https://www.python.org/downloads/.

77. Anaconda | Individual Edition [Electronic resource]. – 2021. – Mode of access: https://www.anaconda.com/products/individual.

Св. план 2022, поз. 3

Учебное издание

Крапивин Юрий Борисович

ЕСТЕСТВЕННО-ЯЗЫКОВОЙ ИНТЕРФЕЙС ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ПОСОБИЕ

Редактор *А. С. Мигно*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 08.02.2023. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 3,84. Уч.-изд. л. 4,0. Тираж 50 экз. Заказ 13.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск