Диплом.

Введение.

В современном мире управление программным обеспечением на естественном языке постепенно занимает место в одном ряду с привычными всем мышкой и клавиатурой. Начиная с 2005-2007 годов сенсорные экраны постепенно заменили клавиатуры на переносных устройствах. Уже 2 года как выпускаются телевизоры с голосовым управлением.

Задачей этого дипломного проекта будет создание простой системы, позволяющей работать с некоторым ограниченным набором программного обеспечения с помощью набора команд на естественном языке.

Определимся, что именно необходимо получить. Решение должно включать окно диалога между ПО и пользователем, для ввода команд на ЕЯ. Для тестирования решено создать программы для ведения заметок и для работы с электронной почтой, которые смогут интерпретировать команды на естественном языке.

Аналитический раздел.

Анализ существующих систем, с управлением на ЕЯ.

Поисковые системы.

Поисковую систему с той или иной долей вероятности можно отнести к системе, с управление на ЕЯ. Конечно, мы не формируем запрос распространённым предложением и стараемся вводить лишь ключевые слова. Как минимум поисковая система должна быть способна распознавать синонимы, и приводить слова к начальной форме.

Заметим, что попытки сделать поисковую систему более дружелюбной и близкой к привычному естественному языку предпринимаются довольно давно. Сейчас поисковая система Google способна отвечать на простые вопросы используя материалы из википедии (Рисунок 88). Сервис для решения математических задач также «учится» отвечать на вопросы, правда на данный момент, сформулированные только на английском языке (Рисунок 89).

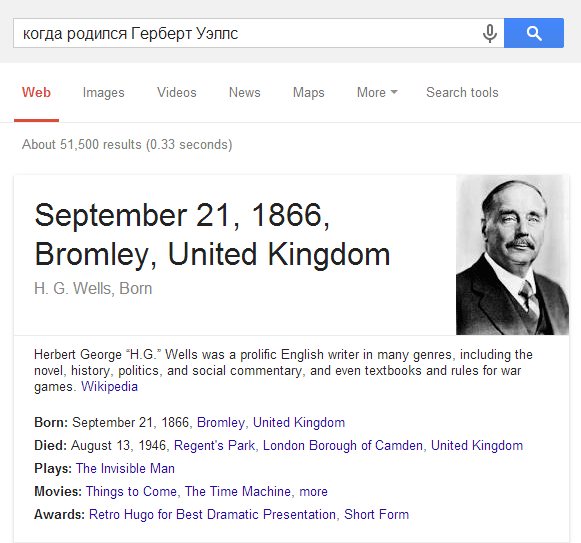


Рис 88.

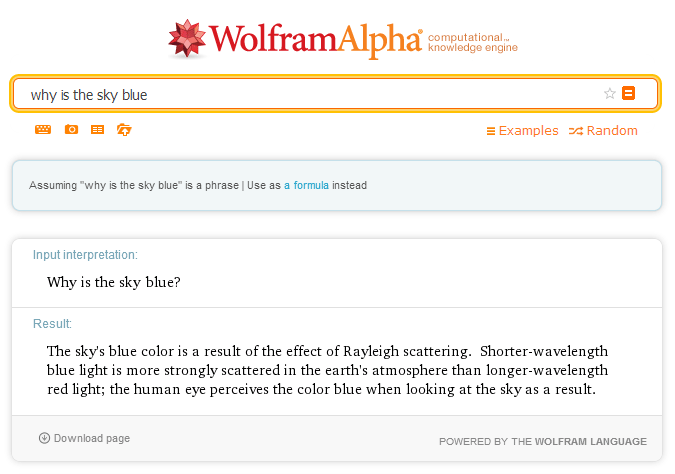


Рис 89.

Google now, Siri, Cortana.

Не так давно в мобильных операционных системах появилась новая функция «голосовой помошник». Первым подобную технологию представила компания Apple. Ниже рассмотрим трех помощников для трёх разных операционных систем.

Начнём с самой младшей – помошница из Windows Phone – Cortanа. Не так давно подразделение Microsoft представила нам свой собственный голосовой помощник, названный в честь ИИ из игры Halo. Его функции такие же, как и в более старших помощниках от гугл и эппл, но разработчики отмечают очень интересную особенность. Прежде чем пользователь начинает работать с Cortana , она задаёт ему несколько вопросов, к примеру его имя, также Кортана запоминает всё где пользователь был, что искал в поисковике bing, какие его ждут ближайшие рейсы или любые другие поездки. Всё это хранится в редактируемой записной книжке Кортаны. Так, например, вы сможете узнать сколько времени займёт поездка до ближайшего рейса, и вам не придётся указывать конечный пункт назначения и название аэропорта, а также время посадки, Кортана уже знает это из ваших электронных писем (конечно, это может не понравится персонам, которые желают оставаться "инкогнито"). Теперь, когда у Кортаны есть много информации о вас, она способна отфильтровывать новую информацию и помогать концентрироваться только на нужных вещях.

"И наконец, Кортана — не просто бесстрастный компьютер, показывающий вам результаты поиска. Точно так же, как в игре Halo, у Кортаны на Windows Phone есть свой характер. Но вам придется поболтать с ней самим, чтобы понять, что я имею ввиду",- говорит один из разработчиков голосового помощника.

Продолжим тему голосовых помощников рассказом о голосовом сервисе гугл. Так называемый Google now является голосовым помощником у смартфонов на базе Android. Суть его и назначение такие же, как и у других голосовых помощников: управляясь голосом пользователя Google now предоставляет требуемую информацию в виде "карточек", число которых в последней версии приложения достигло 10. На них сервис располагает информацию о погоде, загруженности дорог, последних новостей, не прочитанных письмах и о прочих вещах, которые могут быть полезны пользователю. На рисунке 99 представлены две копии экрана телефона с результатом выполнения команд на ЕЯ.

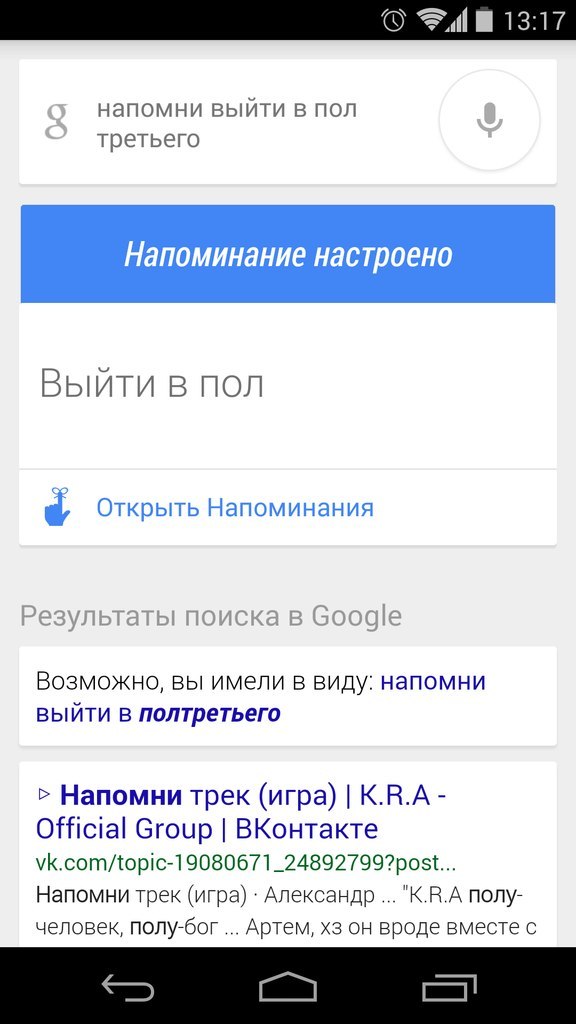
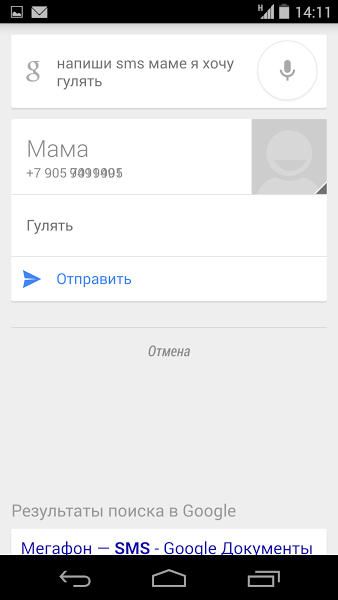
 

Рис 99.

Все эти сервисы объединяет одна особенность: они, если можно так выразиться, не многозадачные. И практически не заточены на работу в режиме диалога. Обычно взаимодействие происходит следующим образом: пользователь диктует команду, а затем производит действия не используя ЕЯ, набирая сообщение, или корректируя установки. Заметим и то, что не реализовано никакого API для реализации возможности голосового управления в сторонних приложениях.

Специфика ЕЯ.

Необходимо решить каким образом будут распознаваться команды на ЕЯ и каким образом результат распознавания будет сопоставляться с конечной целью – ответом. Попробуем проследить некоторые этапы общения на ЕЯ. Мозг решает сообщить или попросить что-либо кому-либо –> формируется предложение на ЕЯ –> передача предложения адресату –> адресат распознает начальный смысл (мотив), сформированный в мозгу автора предложения. Далее будут рассмотрены вопросы: что такое текст, что такое смысл текста, что он под собой подразумевает и каким образом на текст реагирует человек.

Смысл текста.

В [1] поднимается этот вопрос и ведутся интересные рассуждения. Упоминается факт о том, что само по себе предложение очень сильно сжато, относительно того количества информации, которое оно подразумевает под собой. Для разных собеседников, в разных контекстах одна и та же фраза может иметь совершенно разные смыслы.

Так, вопрос, адресованный курортной знакомой: «Ты помнишь июнь?», конечно же, имеет целью не напоминание о том, что между маем и июлем текущего года имелся июнь, а побуждение её к формированию в мозгу сложнейшего комплекса, включающего и зрительные образы моря, луны и т. д., и слуховые и обонятельные воспоминания, и определенное настроение, и многое другое. Разумеется, всё это ни в коей мере не содержится в словосочетании «Ты помнишь июнь?». Последнее, очевидно, является лишь кодом, запускающим у слушательницы механизм формирования указанного комплекса. В свою очередь, текст «Да» или «Нет», полученный в ответ автором вопроса, вызывает в его мозгу целый мир образов и переживаний, соответственно радостных или горестных, на что этот и рассчитан. Разумеется, смысл, вызываемый тем же текстом «Да», не имеет ничего общего со смыслом, вызываемым тем же текстом, например, при выяснении готовности яичницы.

Продолжая примеры. можно было бы напомнить, что фраза «Джон подбросил к Мэри книгу ногой» скорее всего предназначена для создания у читателя комплекса представлений об оскорбительном, унижающем отношении Джона к Мэри, о его пренебрежении ею и т.д., а не для сообщения того, что в этом действии участвовала именно книга и что она таким способом переменила владельца, как утверждается в [17]. Равным образом текст «Иванов шествовал по коридору с бутербродом в руке» почти наверняка предназначен для формирования у слушателя иронического отношения к Иванову вместе со зрительным образом надутого важничающего персонажа, скорее всего не слишком умного, а не для сообщения фактов о коридоре и бутерброде.

Изложенные соображения приводят нас к важному выводу. Текст не содержит и не передает смысла, он является лишь инструментом сообщения «понимателю» того, какую реакцию хочет от него получить автор текста.

Попытка охарактеризовать термин «смысл» (т.е. не смысл текста, а смысл в мозгу) в формальных терминах приводит к представлению о структуре, самое меньше чрезвычайно многомерной и многоуровневой, построенной на огромном количестве элементов-понятий и притом охваченной невообразимым числом внутренних связей по горизонтали и вертикали.

Как известно, структуры такого типа не допускают моделирования в виде замкнутой математической системы.

Мы приходим к тому, что *смысл* приходится считать имманентной, неотделимой и потому ненаблюдаемой принадлежностью мозга. Следовательно, и конструктивно использовать это понятие в разработке алгоритмов машинного понимания ЕЯ оказывается невозможно, что и наблюдалось при ознакомлении с существующими разработками.

Реакция на текст.

Исходя из выше описанных соображений мы можем отметить, что неправомерно рассматривать задачу понимания как «Текст на входе - смысл на выходе». Такое рассмотрение всегда будет неполным. Как мы видели тексту всегда предшествует некая цель автора, а формирование в мозгу «понимателя» смысла всегда является лишь средством выработать вслед за этим адекватную реакцию. Поэтому соответствие «текст – смысл» является внутренним по отношению к объемлющему «цель – реакция». Так мы можем прийти к интерпретации задачи понимания в более удобной для нас форме.

В самом деле, если смысл текста ненаблюдаем, то наблюдаема, по крайней мере в своей чувственно воспринимаемой части, реакция «понимателя» на этот текст. Поэтому, примирившись с фактом невозможности определить понятия «смысл» и «понимание, мы вместе с тем получаем возможность, введя в рассмотрение *цель* автора при формировании текста (всегда ему известную) и *реакцию* на этот текст «понимателя» (или хотя бы ее наблюдаемую часть), использовать критерий правильности понимания, основывающийся на степени совпадения полноценной реакции с ожидаемой. Напомним, что это есть вариант критерия Тьюринга для полноценности «машинного мышления». Важно, что такой критерии является по природе нелингвистический. Так в случае машинного понимания это есть просто правильность выполнения машиной делового нелингвистического задания, хотя и сообщенного ей на ЕЯ.

Вообще в прикладных случаях хорошее имитирование компьютерной системой одной лишь наблюдаемой (и потому, с определенной степенью полноты, формализуемой) части человеческой реакции на текст является достаточным для практики результатом.

Последовательность.

Итак, были сделаны выводы о том, что этапы, в которых фигурирует «смысл текста», очевидно, не стоит пытаться реализовать в рамках данной работы. Следовательно, в последовательности «цель – реакция» мы будем избегать всяческого «осознания» текста. В работе [1] предложено упрощение задачи вплоть до сопоставления состояния окружающей среды – одной из подпрограмм. Под состоянием окружающей среды имеется ввиду как некоторая информация о текущем моменте, так и тот текст или команда, которой человек хочет добиться ответной реакции от программы.

Далее проводится аналогия между поставленной нами задачей и задачей распознавания образов. И аналогия эта предлагает более чем полное решение поставленной задачи. Необходимо распознать к выполнению какой подпрограммы побуждает текст и выделить из текста параметры для выполнения задачи.

Распознавание образов.

Стоит определиться с тем, что именно будет выступать в роли материала для распознавания.

* Это может быть сам по себе текст. Например, «Закрой окно в через 2 часа». На мой взгляд, при таком подходе мы можем столкнуться со сложностями при обучении. Стоит не забывать, что русский язык – язык со свободным порядком слов в предложении. Также, русский отличается большим количеством синонимов и оборотов.
* Регулярные выражения, «Закрой \*obj\* через \*num\* часа».
* Вариант, более сложный, но более гибкий – это работа с деревом разбора предложения. Выделение главных и второстепенных членов предложения. Предложение из предыдущего пункта и предложение «Через 2 часа закрой окно» будут иметь одинаковые деревья разбора.

Дерево разбора предложений на ЕЯ.

Чаще, термин абстрактное синтаксическое дерево используется при работе с формальными языками, но по определению [3] синтакси́ческий ана́лиз (жарг. па́рсинг) в лингвистике и информатике — процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора (синтаксическое дерево).

Преимущество работы с деревом разбора очевидно, и я не буду на этом останавливаться. Построение синтаксического дерева для предложений на естественном языке – не новая задача, но её полное решение не найдено до сих пор. Не говоря уже о решении этой задачи для русского языка.

Построение синтаксического дерева для предложения на русском языке.

Введение.

В компьютерной лингвистике существует много прикладных задач, которые используют результаты работы специального класса программ – *лингвистических парсеров*. К таким задачам можно отнести: анализ/синтез предложения на естественном языке (ЕЯ) с целью формирования/распознавания запроса на искусственном языке (SQL, SPARQL, поисковые запросы и др.), преобразование предложения с одного ЕЯ на другой ЕЯ (системы машинного перевода), задачи семантического анализа текстов – TextMining, семантическая классификация текстовых ресурсов и др.

*Лингвистический парсер* – комплекс программных модулей, обеспечивающий разбор линейной последовательности лексем (слов) языка исходного текста во внутреннее представление смысла этого предложения. При этом используется многоуровневый анализ предложения на ЕЯ, реализующий морфологический, синтаксический, семантический (иногда и

прагматический) языковой уровень. На каждом языковом уровне используются свои структуры данных, которые обрабатываются и формируются соответствующими компонентами лингвистического парсера.

*Морфологический анализатор*, используя морфологические словари, строит разбор последовательности входящих в анализируемое предложение слов с указанием части речи и морфологических характеристик.

*Синтаксический анализатор* реализует построение дерева зависимостей, в узлах которого стоят слова данного предложения с указанием части речи и грамматических характеристик, а дуги соответствуют специфичным для данного естественного языка отношениям подчинения.

*Семантический анализатор* также часто использует дерево зависимостей, но в его узлах стоят либо предметные имена, либо слова универсального семантического языка (например, имена таблиц, в которых сосредоточены сведения о данной предметной области, атрибуты таблиц, операторные символы). Дуги соответствуют универсальным отношениям семантического подчинения: аргументное, атрибутивное, конъюнкция, дизъюнкция, равенство, неравенство, больше, меньше, принадлежит, не принадлежит и т. п. [3]

Анализ существующих решений.

Задача выбора лингвистических, алгоритмических и программных решений для построения парсера русского языка является сложной, в связи с особенностями структуры языка, трудностями в формализации языковых правил. В частности, русский язык является флективным, что подразумевает собой словообразование и словоизменение с использованием некоторых аффиксов, выражающих значения одной или чаще нескольких грамматических категорий. Кроме того, порядок слов в русском языке является свободным, что мешает выработать однозначные правила для синтаксического анализа.

Известным лингвистическим парсером для русского языка можно назвать систему «Диалинг» [5]. Эта система базируется на многоуровневом языковом представлении, заимствованном у системы ФРАП. (системы французско-русского машинного перевода, разработанная коллективом лаборатории машинного перевода Всесоюзного центра переводов совместно с коллективом лаборатории машинного перевода МГПИИЯ им М. Тореза).

Далее, можно упомянуть о парсере грамматики связей русского языка (Russian Link Grammar) [6]. Также интерес представляет лингвистический процессор ЭТАП-3 [7], который осуществляет перевод предложения на русском или английском языке на формальный семантический язык UNL. Пример работы ЭТАП-3 представлен на рисунке 1. Минусом этого анализатора является требовательность к правильности не только грамматики, но и пунктуации.



Рис1

Представляет интерес разработка компании Dictum [8] – синтаксический анализатор для русского языка, решающий задачу омонимии, собирая статистику из Интернета. Он также способен разбирать эллиптические конструкции. На рис.2 представлен пример дерева зависимостей, построенного этой программой для предложения «Ломбард несет ответственность за утрату вещей, если не докажет, что утрата произошла вследствие непреодолимой силы».



Рис 2

Компания PROMT, известная своими разработками в области машинного перевода, разработала PROMT Syntactic and Semantic Analyzer, выполняющий глубокий морфологи ческий, синтаксический и семантический анализ заданного текста на естественном языке. Лингвистическая база данных этого приложения содержит более 30 миллионов словоформ и позволяет вводить новые слова и модели, такие как новые семантические классы или определяемые пользователем синтаксические модели; кроме того, обеспечена поддержка 6 естественных языков. Результат выдается в формате XML, который легко может быть проанализирован. Комплектация продукта также содержит средство визуализации результатов анализа для экспертной оценки [9].

Существующие подходы к реализации морфологического анализа.

Морфологический компонент лингвистического анализа – это программный модуль, обеспечивающий морфологический анализ лексем исходного языка. Существующие в настоящее время морфологические модели различаются в основном по следующим параметрам.

Во-первых, по результатам работы основанных на них морфологических анализаторов. На вход морфологический анализатор получает словоформу некоторого естественного языка, а на выходе может выдавать все значения грамматических характеристик (род, число, падеж, вид, лицо и т.п.) заданной словоформы, а может просто отвечать на вопрос, принадлежит ли заданная словоформа некоторому естественному языку или нет (в этом случае морфологические анализаторы называют акцепторами).

Во-вторых, морфологические модели могут ориентироваться на полное покрытие лексики (т.е. все лексемы, которые могут обрабатывать программы морфологического уровня, находятся в базе данных) или частичное покрытие лексики (морфологическая модель учитывает возможность появления лексемы, не занесенной в базу данных).

В-третьих, морфологические модели различаются по способу представления и членения словоформ. Существует два основных способа представления лексем.

1. В базе данных хранятся все словоформы всех лексем (возможно, с набором их грамматических характеристик), и каким-то образом определяются словоформы, принадлежащие одной лексеме. Такой способ представления лексем удобен и эффективен для малофлективных языков, в которых различные грамматические категории реализуются, в основном, не с помощью вариации флексий, а некоторым грамматическим способом, например, с помощью предлогов. К малофлективным языкам относится, например, английский язык.

2) В базе данных хранятся основы лексем и списки флексий (возможно, с приписанными им значениями грамматических характеристик), которые присоединяются к основе для получения какой-либо словоформы. Такой способ представления лексем эффективен для флективных языков, в которых различные грамматические категории реализуются путем вариации флексий. Флективным является, например, русский язык.

Модели, в которых принят данный способ представления лексем подразделяются еще на две группы: в одной учитываются чисто орфографические основы и флексии, в другой – так называемые псевдоосновы (неизменяемая начальная часть слова) и псевдофлексии (варьируемая при словоизменении конечная часть слова). Выбор того или иного варианта определения основы связан, в основном, с эффективностью реализации и назначением морфологического компонента в целом [3].

Анализ подходов к реализации синтаксического анализа.

Синтаксический компонент производит синтаксический анализ предложения (parsing). На вход он получает данные морфологического анализа, а на выходе должен построить *дерево зависимостей*, отражающее структуру разобранного предложения. При этом предложение представляется как линейно упорядоченное множество элементов (словоформ), на котором можно задать ориентированное дерево (узлы – элементы множества). Каждая дуга, связывающая пару узлов, интерпретируется как подчинительная связь между двумя элементами,направление которой соответствует направлению данной дуги.Среди способов построения дерева зависимостей выделяют:

Модель разбора на составляющие *(phrase structure-based parsing*).

Модель отношений между словами *(dependency parsing)*

Суть dependency parsing состоит в том, что соединение зависимых слов происходит без создания дополнительных узлов.

Центром практически любой фразы является глагол (явный или подразумеваемый). Далее от глагола (действия) можно задавать вопросы: кто делает, что делает, где делает и так далее. Для присоединённых сущностей тоже можно задать вопросы (в первую очередь, вопрос «какой»). Этот метод подходит для языков со свободным порядком слов (например, русского языка), в отличии от *разбора на составляющие,* которая реализует разбор на основе грамматики Ноама Хомского и используется для языков с регулярной структурой.

Реализация семантического анализа

Семантический компонент может быть реализован на основе онтологий. В этом случае выделяется три компонента знаний: онтология, включающая в себя понятия и отношения; предметный словарь (тезаурус), содержащий термины, с помощью которых в тексте могут представляться понятия и отношения онтологии; информационное наполнение системы или база данных.

В рамках реализации такого подхода необходимо:

описать понятия (классы), которым соответствуют текстовые ресурсы;

определить формальную структуру содержания для каждого класса текстовых ресурсов;

задать схемы фактов, задающие правила извлечения содержательных объектов из текста [11].

Другим подходом к решению задачи семантического анализа является использования формальных семантик. В этом случае значение предложения представляется с помощью формулы лямбда-исчисления. Формальные семантики позволяют быстро перейти от языкового выражения к его логической структуре и референтам высказывания, но на очень ограниченном "фрагменте" языка [5].

Существует так же подход, называемый латентно-семантическим анализом (LSA, Latent Symantic Analysis). В этом подходе значение слов оценивается путем статистической обработки больших корпусов текстов, включающих в себя набор взаимных ограничений, благодаря которому можно определить семантическую идентичность двух и более слов. Для определения этих ограничений применяются методы линейной алгебры, в частности, сингулярное разложение [12].

Вывод аналита.

Проанализированы существующие систему, и средства, необходимы для их реализации. Важной частью этой работы будет обработка ЕЯ. Необходимо разработать морфологический, синтаксический и семантический анализатор. Разработать механизмы сопоставления команд на ЕЯ и подпрограммы приложения.

Скорее всего, конструкторский. Выбор методов. Реализация.

# Морфологический анализ.

Структура морфологического словаря.

Представление в памяти морфологического словаря схематически изображено на рисунке 67.

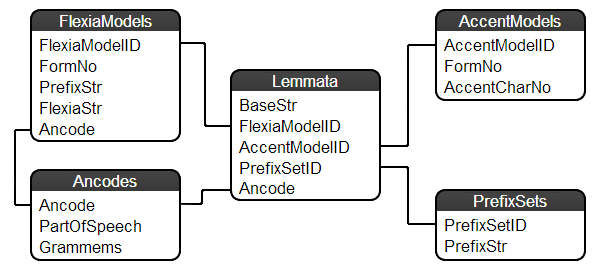


Рис 67.

Таблица Lemmata содержит перечень всех лемм данного словаря, для каждой леммы даны ее свойства:

псевдооснова слова (общая для всех словоформ данного слова подстрока),

(поле BaseStr);

ссылка на набор окончаний (поле FlexiaModelId);

ссылка на набор ударений (поле AccentModelId);

ссылка на набор приставок (поле PrefixSetId);

ссылка на общие граммемы данной леммы (поле Ancode) (может быть пустым).

Общие граммемы данной леммы, это те граммемы, которые должны быть приписаны всем словоформам данной леммы, например, граммема «фам» (фамилия), или граммема «лок» - локативность. Это часто уже семантизированные граммемы.

Набор приставок леммы – это те приставки, с которыми лемма образует полное слова языка. В набор приставок может входить пустая приставка, что означает, что лемма может быть использована сама по себе (без приставок).

Таблица FlexiaModels cодержит перечень возможных окончаний всех лемм. Уникальным ключом здесь являются поля FlexiaModelId и FormNo. Поле FormNo содержит порядковый номер окончания в данном наборе окончаний, соответственно, FormNo не превосходит максимальное кол-во словоформ в одно парадигме. Далее:

Поле PrefixStr содержит префикс данной словоформы (возможно, пустой)

Поле FlexiaStr содержит окончание данной словоформы (возможно, пустое)

Поле Ancode содержит морфологическую интерпретацию данной словоформы.

Пусть у нас есть запись Q из таблицы Lemmata. Пусть P один из ее возможных префиксов, взятых по полю Q.PrefixSetId. Для того, чтобы получить i-ю словоформу данной леммы, надо найти в таблице FlexiaModels запись R, такую, что Q.FlexiaModelId = R.FlexiaModelId и R.FormNo = i, тогда i-я словоформа будет равна:

P + R.PrefixStr + Q.BaseStr + R.FlexiaStr.

Таблица AccentModels содержит перечень возможных номеров ударных гласных для словоформ. Уникальным ключом являются поля AccentModelId и FormNo. Поле FormNo выполняет такую же роль, что и в таблице FlexiaModels. Поле AccentedCharNo содержит номер ударной гласной с конца слова. Для каждой словоформы в словаре должно быть указано ударение, если ударения нет, тогда используется специальная константа (255).

Таблица Ancodes содержит все возможные морфологические интерпретации. Ключом является поле Ancode. Поле PartOfSpeech содержит часть речи (C, Г, П, МС, …), а поле Grammems набор граммем, типа «мр, но, ед, им».

Вышеописанная схема показывает принципиальные возможности и ограничения структуры одного словаря. Видно, что словарь может хранить информацию о словах, возможных окончаниях, возможных приставках, которые могут присоединяться либо к отдельным словоформам, либо ко всем словоформам данной парадигмы. Словарь хранит еще информацию об ударениях. Однако очевидно, что данная схема не предназначена для хранения полного морфологического разбора слова, устройства компаундов и т.д.

Представление словаря и поиск.

Словарь в виде дерева предоставляет следующие функции:

Морфологический анализ: получение по словоформе леммы, ее свойств, уникального ID леммы, морфологических характеристик входной словоформы.

Морфологический синтез: получение по уникальному ID леммы всей парадигмы слова со всеми словоформами и их морф. Характеристиками.

Важно, что такое представление словаря оптимизировано прежде всего для проведения морфологического анализа. Основу этого представления составляет конечный автомат (аксептор) (см., например, [8]). Автомат детерминирован и не имеет циклов, что позволяет минимизировать его в процессе построения, как это предложено в [9].

Основной цикл построения автомата выглядит так:

foreach W in WordForms {

AddStringToAutomat(W + ‘|’ + Annot(W));

}

Символ ‘|’ (annotation char) – специальный разделительный символ, которого нет в алфавите словаря, т.е. он не может встречаться в словоформе W. Функция Annot(W) выдает строку аннотации словоформы W в любом текстовом виде, например, так: С жр, ед, им, од. Т.е., например, для словоформы «мама» в автомат может быть добавлена строка

мама|С жр,ед,им,од.

Функция AddStringToAutomat добавляет входную строку в автомат, сохраняя свойство минимальности и детерминированности автомата (см. [9]).

Поиск словоформы в таком автомате происходит за линейное от длины входной словоформы время: достаточно просто пройти все состояния автомата, которые соответствуют символам входной словоформы, далее пройти разделительный символ и получить обходом по графу все аннотации словоформы.

Для морфологического анализатора и построения бинарного дерева был выбран Русский морфологический словарь Диалинг. Он базируется на грамматическом словаре А.А.Зализняка[1987]. Включает на данный момент 161 тыс. лемм.

Неоднозначности.

Отдельный интерес представляют слова, для которых было найдено несколько парадигм и значений форм. В данном случае это слова «жил» и «времени». Ниже можно изучить отладочный вывод алгоритма морфологического анализа.

Paradigm: ЖИТЬ w=3011 hw=0 Founded: True

ParadigmID: 148035

AccentModelNo: 786

Grm = нс,нп

ЖИ’Л Г дст,прш,мр,ед

Paradigm: ЖИЛА w=39 hw=0 Founded: True

ParadigmID: 91144

AccentModelNo: 42

Grm = но,

ЖИ’Л С жр,мн,рд

Paradigm: ЖИЛА w=3 hw=0 Founded: True

ParadigmID: 131888

AccentModelNo: 16

Grm = од,

ЖИ’Л С мр-жр,мн,рд

ЖИ'Л С мр-жр,мн,вн

Paradigm: ВРЕМЯ w=6871 hw=10000 Founded: True

ParadigmID: 135563

AccentModelNo: 97

Grm = но,

ВРЕ’МЕНИ С ср,ед,рд

ВРЕ'МЕНИ С ср,ед,дт

ВРЕ'МЕНИ С ср,ед,пр

Paradigm: ВРЕМЕНИТЬ w=7 hw=1 Founded: True

ParadigmID: 132133

AccentModelNo: 214

Grm = нс,нп

ВРЕМЕНИ’ Г дст,пвл,2л,ед

Структура.

Структура класса, который будет хранить результаты морфологического анализа, и учитывать все варианты изображена на рисунке 44. WetCollocationPart хранит в себе список (Variants) экземпляров класса WetCollocationPartVariant. В свою очередь каждый вариант хранит список форм слова Forms, класса WordForm.

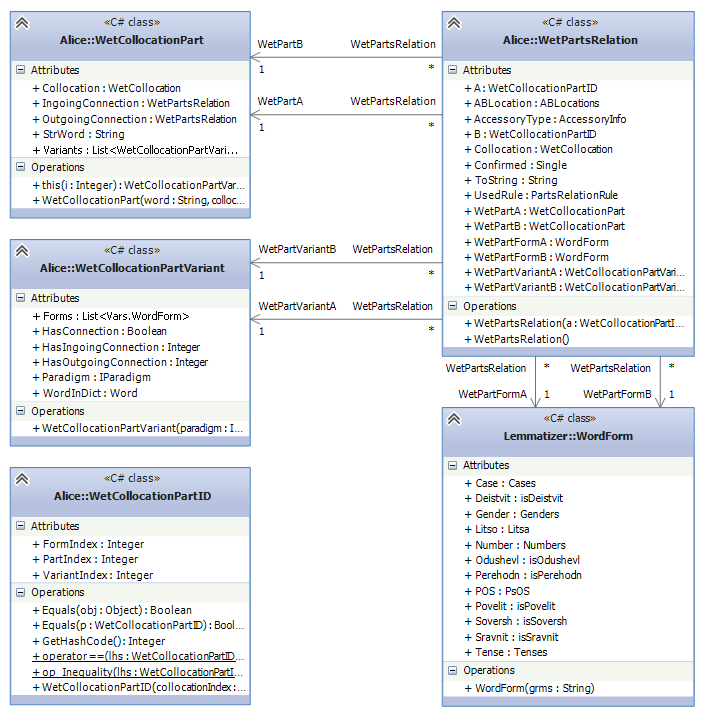


Рис 44.

# Синтаксический анализ. Построение дерева разбора.

Метод, который будет использоваться – комплексный. В нем будет и заданная система правил, и обучающаяся часть. Отметим, что в данной работе для построения синтаксического дерева знаки препинания не будут учитываться вообще. Это усложнит задачу, но сделает алгоритм многим устойчивее. Так алгоритм позволит анализировать надиктованные предложения, полученные с помощью систем распознавания голоса.

Далее описанный метод базируется на приеме анализа неограниченных или контекстно зависимых грамматик. Согласно этому методы за основу берется контекстно свободная грамматика, включающая в себя исходную, на которую после накладываются ограничения.

Как, пример, можно рассмотреть контекстно-зависимый язык {anbncn : n > 0}. Дальше утверждается следующее: данный язык является подмножеством языка {a+b+c+}. Начнем работать с более общим языком, а затем проверим совпадает ли количества букв a, b и c.

Первым этапом будет попытка разбора предложения посредством КС грамматики. Следующие этапы будут нацелены на устранение неопределенностей и в конечном счете должны привести к однозначному дереву разбора. Рассмотрим этапы.

Этап 1: «Полный перебор».

Первым этапом разбора предложения будет поиск всех возможных совпадений для связей следующего вида:

Главное слово – А. Часть речи, фиксированные граммемы.

Подчиненное слово – В. Часть речи, фиксированные граммемы.

Относительно расположение.

Правила согласования между словами.

Примером такого согласования может быть согласование между существительным и прилагательным.

А – Существительное

B – Прилагательное

Относительно расположение – любое

Согласование форм – род, число, падеж.

Или:

А – Существительное

B – Числительное

Относительно расположение – B перед А

Согласование форм – падеж, число.

Результатом работы первого этапа будет ориентированный граф, который обязательно содержит подграф, являющийся деревом разбора предложения. Каждая связь хранится в экземпляре класса WetPartsRelation (рис 44). Следующие этапы нацелены на удаление ложных связей между словами предложения.

Пример.

Рассмотрим пример «Он жил в покосившейся от времени хижине».

После морфологического анализа для каждого слова предложения были найдены парадигм и формы. И связи строятся для всех вариантов парадигм и форм.

В процессе работы было найдено 56 реляций.

Некоторые из них (первое слово – главное, второе – зависимое): «жил он», «жил в», «в хижине» являются верными и далее будут выделены. Некоторые удалены: «он в», «в времени», «от хижине». На этом этапе не учитывается согласование предлога и существительного по падежам.

Этап 2. «Обработка предлогов и послелогов».

Обзор моментов, которые возможно были упущены лингвистами не входит в тему данной работы. Скажем лишь то, что в список частей речи было решено добавить ещё одну: послелог. Ярким примером послелога является слово «назад» («2 недели назад»). Удаление неопределенностей для послелогов зеркально в сравнении с предлогами. Были учтены ситуации вложенных конструкций с предлогами. Рассмотрим то, как будет работать алгоритм.

Пример.

Продолжим рассматривать пример «Он жил в покосившейся от времени хижине».

Первый проход алгоритма пометит как верную реляцию «от времени» и пропустит

разбор неопределенности с предлогом «в». Т.к. результатом должно быть ориентированное дерево все остальные исходящие дуги от предлога «от» будут удалены, как и все входящие дуги в слово «времени». На этот момент количество Реляций в целом сократится с 56 до 32.

Второй проход теперь может разрешить неопределенность с предлогом «в» т.к. дуга «в времени» уже удалена и остается единственно верная дуга «в хижине». Опять удаляются лишние дуги. Заметим, что между предлогом и существительным стоит оборот «покосившейся от времени». Все дуги, которые выходят за пределы между «в» и «хижине» также будут удалены. Итого, после второй итерации, количество реляций сократилось с 32 до 17(2 из которых – подтвержденные).

После этого этапа возможно нарисовать граф, в котором не сложно заметить верный подграф – дерево разбора. На рисунке дуги со звездочками – подтвержденные связи, цифры на дугах показывают сколько дуг объединено в одной.

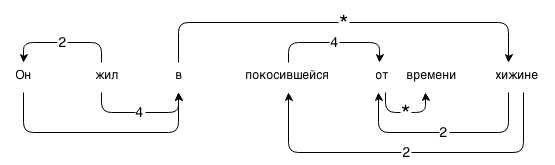


Рис 92.

Этап 3. Придаточные части в сложных предложениях.

Далее наступает момент, когда предложение разделяться на отдельные предикаты, центром каждого простого предложения является глагол, один из предикатов объявляется корневым, остальные предикаты - зависимые (из придаточных частей).

Придаточные делятся на 3 вида. Те, что зависят от глагола, причастия или деепричастия; от существительного и от предиката, то есть от всей главной части предложения.

В лингвистике групп придаточных 4:

СПП с придаточным изъяснительным (с союзами: что, как, чтобы, ли):

Ольга сказала, что вернётся из Пскова в понедельник.

СПП с придаточными определительными (с союзными словами: какой, который, чей, что; где, куда, откуда, как):

Вот дом, в котором я хотела бы жить.

СПП с придаточными присоединительными: (с союзными словами что (в любом падеже), отчего, почему, зачем):

По утрам он принимал душ, после чего жена кормила его завтраком.

СПП с придаточными обстоятельственными:

Обстоятельственное значение может быть разным: обстоятельство образа действия, времени, места и проч. Поэтому обстоятельственные СПП делят на виды по значению.

Обстоятельственные СПП делятся на 10 подгрупп. Предложения с придаточными:

места (союзные слова: где, куда, откуда):

Мы спустились к реке, где купались дети.

временными (союзы: когда, пока, лишь, только):

Когда ты позвонил, я спала.

условными (союзы: если, коли (устар.):

Если он пригласит меня в кино, я пойду.

И т.д. [http://russkiy-na-5.ru/articles/452]

Информация о видах придаточных, о том какие используются союзы и от каких частей речи они зависят заносится в специальных структуры.

Затем предложение анализируется на наличие подходящих конструкций. Если такая находится, то сначала анализируется придаточное, затем главное.

Рассмотрим пример: «Покажи письма которые пришли вчера».

После поиска было найдено 2 варианта. Из 3х вариантов форм для слова «письма» (ед, рд; мн, им; мн, вн) только 2 были согласованы с местоимением «которые» по роду и числу. Бывают случаи, когда возникает более сложная неопределенность, например, «ответь то что я занят». Разрешение неопределённости будет рассмотрено на этапе 5.

Этап 4. Удаление лишних дуг.

Без входящей дуги может быть только корень дерева – глагол. Опираясь на этот факт удалим все дуги, которые выходят из узлов (не глаголов), в которые не ведет ни одна дуга.

В примере «Он жил в покосившейся от времени хижине» алгоритм удалил все неопределенности согласования, выбрав конкретную парадигму и форму для каждого слова (заметим, что так бывает не всегда). Количество неподтвержденных реляций сократилось с 15 до 6.

Удалены были такие связи как «жил в», где «жил» - форма существительного «жила» (множественное число, пример: «в мясе много жил»), т.к. в эту парадигму не входило ни одной дуги. Продолжим пояснения. Среди всех вариантов для слова «хижине» только в один входит дуга от предлога «в», следовательно, были сохранены дуги, исходящие только из этого узла. Новый граф можно посмотреть на рисунке 12.

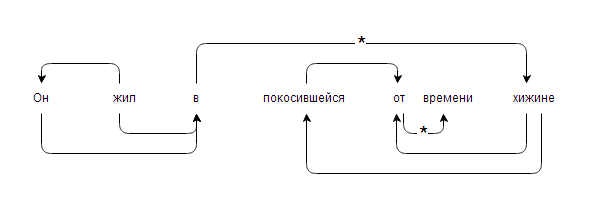


Рис12

Этап 5. Удаление неопределенностей.

Удаление неопределенностей происходит за счет словаря неопределенностей. В нем хранятся все встречающиеся неопределенности, контекст в которых они встречались и верное решение.

Поля структуры для хранения неопределенности:

Выбранная альтернатива

Альтернативы

Контекст

Флаг подтверждения правильности выбора

Рассмотрим неопределённость из этапа 4. Предлог «в» может зависеть как от глагола «жил», так и от местоимения «он». Контекстом в данном случае будет предлог. Альтернативами - 2 различные связи. На рисунке 71 показан разбор предложения с удаленными неопределенностями.

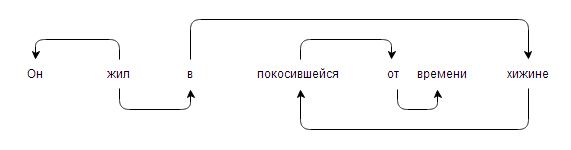


Рис 71.

Вообще говоря, подобного рода неопределенности должны решаться посредством базы данных пар слов, которые могут согласовываться. Но найти подобные данные с открытым доступом к ним не удалось.

Для разрешения неопределенности просматривается весь словарь неопределенностей, сначала подтвержденных, затем, если ничего не найдено – неподтвержденных. Сравнивается степень совпадения контекстов и альтернатив.

Важным вопросом является то, как много информации о контексте стоит хранить для анализа. Для каждого вида неопределенности решено хранить свой собственный словарь. Каждый словарь инициализируется некоторым, вручную составленным набором решений.

Этап 6. Обход дерева. Формирование предиката. В некоторой мере – семантический анализ.

На этом этапе производится обход предложения от корня (глагола) к листьям и формируется предикат предложения.

При формировании предиката основную роль играют классы Predicat, Object и Property. На рисунке 37 изображена диаграмма наиболее важных используемых классов.

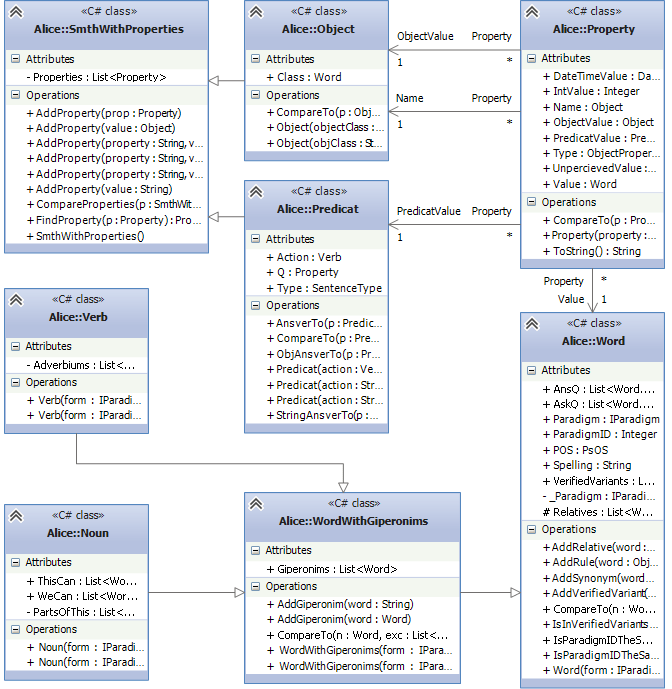


Рис 37.

И Object и Predicat имеют список Properties, в котором хранятся зависимые слова. Предложение из предыдущих примеров будет разобрано следующим образом:

*Predicat* Жил

*Properties* *Object* Он

*Properties* *null*

*Object (в)* Хижина

*Properties* *Predicat* Покоситься

*Properties* *Object (от)* Время

# Покончено с синт анализом. Довольно. Предикат-команда-выполнение.

В рамках этой работы решено реализовать работу с несколькими задачами на ЕЯ. У каждой задачи есть набор функций, которые она может выполнить. Каждой функции должен соответствовать обобщенный предикат. Рассмотрим процедуру написания письма кому-либо. Если обобщить – нам нужно запустить подпрограмму создания нового письма, указать текст, тему и получателя, после – отправить. Рассмотрим варианты команд на ЕЯ, которые могут запускать процесс создания письма: «напиши письмо», «создать новое письмо», «написать письмо Васе», «спроси Васю, когда у нас защита диплома».

Самый примитивный вариант: предикат от глагола «написать» с полем «что?» - «письмо». Но заметим, что привязка подобного предиката к открытию окна написания письма уже позволяет реагировать на команды «пожалуйста, черкани Васе сообщение». Разумеется, если мы занесли в словарь то, что парадигмы «черкани» и «напиши» синонимы на 80% а парадигмы «письмо» и «сообщение» - на 60%. В сумме вероятность принадлежности предиката к обобщенному предикату будет результатом произведения 0.8 и 0.6, то есть 0,48.

Мы можем сделать более совершенный обобщенный предикат, если зададим его так:

«спросить»

- кого? – «человек»

- что? – «»

Если для парадигмы «Вася» был задан гипероним «человек», то при сравнении с обобщенным предикатом вероятность совпадения не уменьшится.

«спросить»

- что? – «задание»

- какое? – «домашнее»

- у кого? – «Вася»

На рисунке 89 можно увидеть класс TaskAction. Одно из его полей – делегат delegate void TaskActionExecuter(Predicat predicat), тут будет храниться подпрограмма, связанная с обобщенным предикатом, о котором говорилось выше. Какой именно TaskAction будет выбран – решается на основе степени совпадения с обобщенным предикатом. Подпрограмма всегда принимает на вход исходный предикат и может более полно обрабатывать информацию, в нем хранящуюся. В контексте отправки сообщения - использовать данные об адресате или содержании письма.

Управляющий класс.

Все введенные команды отправляются в единственный обработчик, который будет анализировать какой из подзадач эта команда адресована. Это процедура NewText(String text) класса AliceGUIManager (рисунок 89). В этом классе хранится список всех возможных задач. При вводе новой команды она преобразовывается в предикат, затем каждая задача опрашивается на предмет возможности выполнения команды (результатом является число от 0 до 1). После, задаче, вернувшей наибольшую вероятность принадлежности ей команды, предоставляется возможность выполнить её. Также необходимо учитывать какая задача используется в данный момент, или использовалась последней.

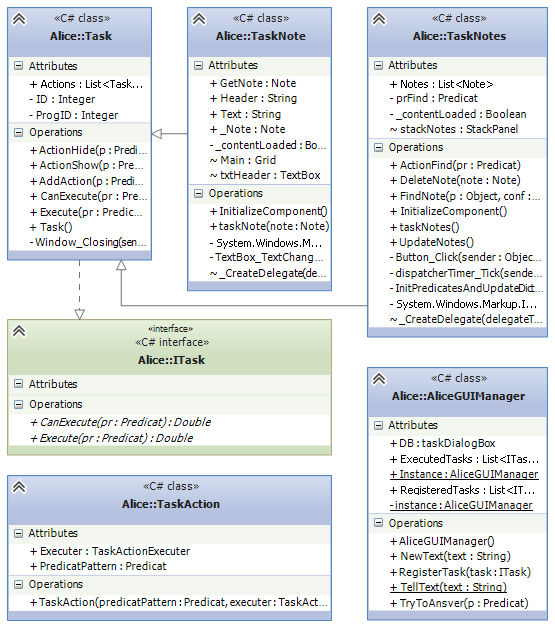


Рис 89.

«Текст на входе – выделение смысла текста – реакция, соответствующая смыслу».

[1] В. С. Файн. Распознавание образов и машинное понимание естественного языка.

[8]  Гладкий А.В. Формальные грамматики и языки. М.: Наука, 1973. 368 с.

[9]  Jan Daciuk, [Bruce Watson](http://www.win.tue.nl/fastar/main.php?button=supervisors), and Richard Watson, [*Incremental Construction of Minimal Acyclic Finite State Automata and Transducers*](http://www.eti.pg.gda.pl/~jandac/incr_fst.ps.gz), proceedings of [Finite State Methods in Natural Language Processing](http://www.nlp.cs.bilkent.edu.tr/fsmnlp98/), pp. 48-56, Bilkent University, Ankara, Turkey, June 29 - July 1, 1998.