Московский государственный университет имени М. В. Ломоносова
факультет вычислительной математики и кибернетики
кафедра алгоритмических языков

Курсовая работа

Библиотека для морфологического анализа свободнообразуемых слов русского языка

Выполнил: студент 324 группы
Гончаренко Евгений Евгеньевич

Научный руководитель: к.ф.-м.н., доцент
Головин Игорь Геннадьевич

Москва 2022

Содержание

Глава 1

Введение

Широкое внедрение вычислительных машин во многие сферы человеческой деятельности вызывает необходимость изучения человеком специальных языков общения с машиной. Естественный язык не требует специального изучения, он обладает огромной выразительной мощностью и, следовательно, пользователю любой специальности и уровня подготовки будет удобно общаться с машиной на естественном языке. Поэтому проблема общения человека с ЭВМ на естественном языке является важной практической задачей.

Достаточно вспомнить, как человек общался с первыми вычислительными машинами. А происходило это таким образом: оператор брал в руки провода с двумя разъемами на конце и соединял между собой триггеры, из которых собственно и состояла машина, таким образом, чтобы при запуске выполнялась нужная последовательность команд. Внешне это очень напоминало манипуляции телефонных барышень начала века. А по сути – это была очень квалифицированная работа. Программирование осуществлялось даже не в машинных командах, а на аппаратном уровне. Соответственно и квалификация тогдашних программистов была очень высокого уровня.

Потом задача упростилась, последовательность нужных команд стала записываться непосредственно в память машины, а для ее ввода стали применяться более скоростные и производительные устройства. Затем появились перфокарты, а чуть позже перфоленты. Скорость общения с машиной возросла. Количество ошибок при вводе резко уменьшилось, но сущность этого общения, его характер не изменились.

Сам процесс общения с машиной долгое время оставался уделом специалистов,

недоступным для понимания простыми пользователям. Теми самыми "простыми пользователями которые, собственно говоря, и являлись потребителями компьютерных услуг. Компьютерный интерфейс на первых этапах развития компьютерной техники в качестве обязательного элемента непременно включал в себя человека-специалиста. Вот если бы можно было бы пообщаться с компьютером напрямую, да при этом еще и не загружать свой багаж знаний всяческими техническими компьютерными сведениями...

Различные системы используют автоматическую обработку текстов. Интеллектуальные «вопрос - ответные» системы характеризуются конкретной предметной областью, разбор высказываний на естественном языке производится по некоторым правилам разбиения, а синтез ответа по некоторым правилам сборки. Поскольку в такой системе нет лингвистической обработки текста, ответы системы не всегда корректны. В системах общения с базами данных высказывания на естественном языке переводятся в запрос на формальном языке. В этих системах тоже нет лингвистического анализа текста. В диалоговых системах решения задач помимо перевода высказываний в формальное внутреннее представление решается некоторый достаточно узкий класс задач (например, планирование путешествий или составление контрактов). Так же автоматическая обработка текста используется в обучающих системах, системах распознавания речи, в системах машинного перевода, при создании редакторов текстов для выявления ошибок, связанных с порядком и сочетаемостью слов в предложении, в информационно-поисковых системах, а также для автоматизации лингвистических исследований.

Создавались системы обеспечивающие диалог с машиной на так называемом «ограниченном» естественном языке. Это системы ПОЭТ (программа обработки экономических текстов) Э. В. Попова, ДИЛОС В. М. Брябрина, ДИСПУТ Л. И. Микулича и А. Я. Червоненкиса. Но использование ограниченного естественного языка не так удобно для пользователя, как общение с ЭВМ на действительно естественном языке. Ограниченный естественный язык требует предварительного изучения принятых в системе ограничений на словарь и грамматику. Это затрудняет процесс общения пользователя с машиной. Часто пользователю проще выучить некоторый формальный язык и общаться с машиной на нем, чем постоянно следить за соблюдением правил наложенных на естественный язык. В этих системах также нет средств для

исследования незнакомых слов и структур предложений, запоминания новых фактов для последующего их использования.

Среди систем, имитирующих понимание естественного языка, известны APRIL, TULISP, TULISP-2 М. Г. Мальковского. Программа APRIL решает довольно узкий класс задач из школьного курса математики. Каждая задача решается независимо от других, никакой информации о решении программа не запоминает. В программе применяется достаточно полный и глубокий лексический и синтактико-семантический анализ фраз с использованием словаря, распознающей контекстно-свободной грамматики и описания, значимых рассматриваемой предметной области, семантических конструкций. Производится тщательный анализ предложений, необходимых для сведения задачи к одному из допустимых типов, и условий, определяющих существование решения и применимость допустимых типов, и условий, определяющих существование решения и применимость одного из стандартных способов решения. В то же время возможности учета в процессе анализа проблемно-ориентированной информации о характерных задачах были использованы далеко не в полной мере. Программа TULISP является реализованной на программном уровне диалоговой системой искусственного интеллекта широкой ориентации. Имеет три предметные области: решение арифметических задач в словесной формулировке (модернизированный вариант решателя APRIL), решение задач на планирование действий пользователя (на основе метода редукции задач), обучение языку (адаптация языковых знаний TULISP с привлечением данных из метаграмматики). Помимо этого имеется блок, осуществляющий синтез, адресуемых пользователю, сообщений, соответствующих запросам, комментариям, описаниям полученных результатов. Достаточно детально был разработан и внутренний язык программы - язык представления знаний. Развитие методов адаптации и обучения, более полное описание русского языка, разработанное на основе лингвистических данных, средства обнаружения и исправления речевых ошибок пользователя, появление режима, в котором с помощью специальных директив можно контролировать текущее состояние языковых знаний системы (словарей, грамматик) привело к появлению новой версии системы TULISP – системы TULISP-2.

Задача интеллектуальной обработки текстов на естественном языке впервые появилась на рубеже 60х—70х гг. С тех пор было предпринято множество различных по-

пыток ее решения [2-6], созданы десятки экспериментальных программ, способных вести диалог с пользователем на естественном языке. Однако широкого распространения такие системы пока не получили — как правило, из-за невысокого качества распознавания фраз, жестких требований к синтаксису “естественного языка”, а также больших затрат машинного времени и ресурсов, необходимых для их работы. Практически во всех системах машинного понимания текста используется ограниченный естественный язык, поскольку полной и строгой формальной модели ни для одного естественного языка пока не создано.

Продолжаются исследования в рамках программы создания информационных систем, в том числе и выполняющих обработку текстов на естественном языке. Один из классов таких систем образуют информационно-поисковые системы (ИПС), ориентированные на естественно-языковое общение с пользователем. Подобные ИПС могут использоваться в качестве консультанта — например, в области законодательства, медицины или любой другой предметной области, для которой характерно наличие большого количества информации, представленной документами на естественном языке.

Одним из ключевых элементов ИПС с естественно-языковой ориентацией также является лингвистический процессор, выполняющий роль посредника между пользователем и базой данных, в которой хранится интересующая его информация. Он включает в себя формальную модель естественного языка, базу данных модели (словарь) морфологический, синтаксический, семантический компоненты, кроме того может быть добавлена компонента, связанная с автоматической расстановкой ударений.

Задачей лингвистического процессора является преобразование естественно-языкового предложения (или даже целого текста) в некоторый набор семантических структур, являющихся формальным представлением “смысла” исходного предложения или текста. Цель такого преобразования — обеспечить исходные данные для работы поисковых механизмов СУБД.

Задачей лингвистического процессора является преобразование естественно-языкового предложения (или даже целого текста) в некоторый набор семантических структур, являющихся формальным представлением “смысла” исходного предложения или текста. Цель такого преобразования — обеспечить исходные данные для работы поисковых механизмов СУБД.

Вот список тех задач, в которых можно использовать лингвистический процессор:

1. написание переводчика;
2. задачи распознавания и синтеза речи;
3. распознавание текста;
4. проверка орфографии;
5. проверка синтаксиса;
6. информационно-поисковые системы;

Основным недостатком существующих лингвистических процессоров является чрезмерно большой объем словаря, порождающий ряд технических проблем:

- большие затраты труда на создание и поддержание словаря;
- невозможность полного размещения словаря в оперативной памяти компьютера при анализе;
- высокая избыточность информации, связанной с постоянными признаками каждой словоформы (морфологическими, синтаксическими, семантическими);

Современные компьютерные программы, анализирующие текст на естественном языке, как правило, используют словари. Цель словарей помочь распознать встреченную текстовую цепочку.

Целью данной работы является создание программы, которая, используя реально существующую лингвистическую базу данных, выдает морфологические характеристики некоторых слов, не содержащихся в этой базе данных. Программа основана на использовании морфологического анализа структуры незнакомого слова. Приблизительно анализ слова работает в такой последовательности. От предлагаемого слова отрезаются возможные префиксы, и оставшаяся часть проверяется на наличие в лингвистической базе данных. Если оставшаяся часть слова присутствует в базе данных, то в качестве информации об исходном слове, выдается полученная информация о части слова, с учетом всех префиксов.

Глава 2

Постановка задачи

Целью данной работы является создание программы, которая, используя реально существующую лингвистическую базу данных, выдает морфологические характеристики для следующих классов слов:

- свободнообразуемые слова;
- слова с дефисом;
- сложные слова.

Свободнообразуемые слова должны удовлетворять следующим условиям:

- Стандартность их соединения с существительными и прилагательными.
- Стандартность значения.
- Структурная самостоятельность.

Программа работает следующим образом:
От предлагаемого слова отрезаются возможные префиксы, и оставшаяся часть проверяется на наличие в лингвистической базе данных. Если оставшаяся часть слова присутствует в базе данных, то в качестве информации об исходном слове, выдается полученная информация (падеж, склонение и т.д.) о части слова, с учетом всех префиксов. Программа автоматически меняет основу, ударную букву, ставит второстепенное ударение.

Глава 3

Алгоритм анализа слова

Анализ слова сводится к следующей последовательности действий:

1. На вход процедуры подается слово X.

2. Пытаемся найти в слове X дефис. Если мы его нашли, то ту часть слова X, которая была до дефиса (включая дефис), мы сохраняем как X1; а оставшуюся часть слова X как X2 и переходим к шагу 5, иначе на шаг 3.

3. Если слово X начинается на префикс из списка префиксов (см. пункт: Словарная информация для базы данных), то мы сохраняем этот префикс как X1, а оставшуюся часть слова X как X2 и переходим к шагу 5, иначе на шаг 4.

4. Если слово X начинается на порядковое числительное (например: тысячетрехсотдвадцатичетырехдневный), то мы сохраняем это
числительное как X1, а оставшуюся часть слова X как X2 и переходим к шагу 5, иначе на шаг 8.

5. Обращаемся к морфологическому анализатору со словом X2. Если морфологический анализатор выдал морфологические характеристики слова X2, то перейти на шаг 6. Если же морфологический анализатор выдал, что слово не найдено, то перейти на шаг 7.

6. В качестве информации о слове X, выдается информация о слове X2, модифицированная следующим образом:

   - к основе слова X2 слева приписывается слово X1;

- меняется номер ударной буквы;
- ставится второстепенное ударение.

7. Рекурсивно вызываем данный алгоритм для слова X2. Если алгоритм выдал морфологические характеристики слова X2, то перейти на шаг 6. Если же алгоритм выдал, что слово не найдено, то перейти на шаг 8.

8. Слово не распознано. Стоп.

Глава 4

Реализация

Для реализации был выбран язык C++, в силу своего быстродействия и удобства использования различных библиотек на языке C.

4.1 Сборка проекта

Если проект собирается под Linux/MacOS и имеет все исходные файлы, включая rmu и bzip2, то достаточно просто запустить RunBuild.
Для других случаев ниже приведена подробная инструкция по сборке.

Для сборки проекта необходимо:

- rmu (надо использовать новую версию, взаимодействие с которой происходит в кодировке utf-8); rmu требует bzip2, собиралось на версии 1.0.6.
- Библиотеки по умолчанию должны находится в rmu/Libs.
- Исходники (anz.cpp, anz.hpp, makefile).
- Makefile копирует собранный файл в переменную PATH, поэтому конечный файл libanalize.a находится в rmu/Libs.

Для работы библиотеки так же требуется файл prefix.dat в рабочей директории, который содержит информацию о префиксах.
Для успешной инициализации rmu требуются файлы вида rm.*.bin.data,причем бинарные файлы должны быть собраны версией rmu после рефакторинга под utf-8,

если таких нет, их можно получить из json файлов (в кодировке utf-8) используя функцию int RMUEncode(const char * BaseName) расположенную в RMUtilites, подробнее смотреть документацию RMU.

4.2 Взаимодействие с библиотекой

Библиотека содержит только класс Word, взаимодействие с которым построено следующим образом:

- Для начала необходимо создать экземпляр класса:
  Word word (”черно-белое”, ”rm”);
- После необходимо проанализировать слово:
  word.analize();
- Если word.error != true, то в word.result находится начальная форма слова,
  в word.stress - номер ударной буквы, а в word.RMUAnswer - исходное слово и вся информация о нем.
- Результатом работы программы будет:
  Result: черно-белый;
  Stressed letter: 8;
  Успешный результат работы RMU.

4.3 Описание класса Word

- Конструктор:
  Word(const std::string word, const char* rm = "rm");
  word - обязательное поле, слово которое необходимо проанализировать;
  rm – опциональный параметр, по умолчанию это рабочая директория, который указывает путь к базе данных вида rm.*.bin.data, собранных под utf-8 из кодировки utf-8.
- RMUAnswer:
  Содержит всю информацию об исходном слове, поле типа std::string.

- result:

  Простая форма слова, результат работы алгоритма, имеет тип std::string.

- stress:

  Номер ударной буквы в слове, типа int.

- error :

  Содержит bool значение, в случае если error = true, в процессе работы произошла ошибка и слово не может быть распознано.

- step1 – step7:

  Шаги в алгоритме соответсвенно.

- AnalyzeBefore:

  Префикс добавляемый в начале запроса к rmu.

- AnalyzeAfter:

  Постфикс добавляемый в конце запрооса к rmu.

- IRMUnit:

  Экземпляр RMU.

- X:

  Исходное слово.

- X1:

  Префикс.

- X2:

  Постфикс.

- Rec:

  Позволяет понять вызывался ли алгоритм рекурсивно до этого.

Глава 5

Результаты

Основными результатами работы являются:

1. Разработаны алгоритмы анализа.

2. Реализована программа на языке C++ в виде библиотеки.

Созданная морфологическая компонента компактна, удобна, легко встраиваема в любые системы автоматической обработки текстов.

Глава 6

Заключение

Основными результатами работы являются:

1. Адаптация библиотеки Rmu для использования, а также переформатирование в кодировку utf-8.

2. Предложенный алгоритм был реализован на языке C++ в виде библиотеки.

Доработанный алгоритм был реализован в виде библиотеки на языке C++, допускающей повторное использование как компонент более крупной системы обработки текста на естественном языке.

Список литературы

1. Зализняк А.А. «Грамматический словарь русского языка».
2. Волкова И.А. «Адаптация и обучение системы общения с ЭВМ на естественном языке», Москва, 1982.
3. Потиха З. А. «Строение русского слова». М., 1981г.
4. Цыганенко Г. П. «Словарь служебных морфем русского языка». Киев, 1982г.
5. Кузнецова А. И., Ефремова Т. Ф. «Словарь морфем русского языка». М., 1986.
6. Тихонов А. Н. «Словообразовательный словарь русского языка: в 2 т». М., 1985г.