|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»** |

**Институт кибернетики**

**Кафедра проблем управления**

**ОТЧЁТ ПО ПРАКТИКЕ по ППУ и ОПД**

**Тема практики:** **Алгоритмы нейросетевой обработки голосовых команд в задаче управления мобильным роботом**

|  |  |  |
| --- | --- | --- |
| Отчет представлен к рассмотрению: |  |  |
| Студент группы | \_\_\_\_\_\_\_\_\_\_\_\_ | П.Р. Кабанов  «\_\_» \_\_\_\_\_\_ 2020 г. |
|  |  |  |
| Отчет утвержден |  |  |
| Допущен к защите: |  |  |
|  |  |  |
| Руководитель практики от кафедры | \_\_\_\_\_\_\_\_\_\_\_\_ | С.А.К. Диане  «\_\_» \_\_\_\_\_\_ 2020 г. |
|  |  |  |
| Руководитель практики от университета | \_\_\_\_\_\_\_\_\_\_\_\_ | А. А. Сухоленцева  «\_\_» \_\_\_\_\_\_ 2020 г. |

**ОТЧЕТ**

**по практике по ППУ и ОПД**

**студента 3 курса учебной группы КРБО-01-17**

**института кибернетики**

**Кабанова Павла Романовича**

Практику проходил с 10.02.2020 по 25.05.2020 в лаборатории для самостоятельных занятий кафедры проблем управления

1. Задание на практику выполнил в полном объеме.
2. Подробное содержание выполненной на практике работы и достигнутые результаты: изучены нейросетевые методы обработки звука; рассмотрены существующие программные средства для обучения нейронных сетей; разработано тестовое программное обеспечение, демонстрирующее возможности голосового управления виртуальной моделью мобильного робота.
3. Предложения по совершенствованию организации и прохождения практики: предложений нет.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_ (Кабанов П.Р.) «\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

1. Заключение руководителя от кафедры проблем управления по практике студента Кабанова Павла Романовича:

В ходе практики Кабанов П.Р. приобрел следующие профессиональные компетенции:

- способность разрабатывать программное обеспечение, необходимое для обработки информации и управления в мехатронных и робототехнических системах, а также для их проектирования;

- готовность участвовать в составлении аналитических обзоров и научно-технических отчетов по результатам выполненной работы, в подготовке публикаций по результатам исследований и разработок;

Кабанов П.Р. проявил себя как организованный, целеустремленный специалист, способный самостоятельно решать поставленные перед ним инженерно-технические задачи и представить отчет по практике в полном объеме и в установленный срок и заслуживает оценки «отлично».

Руководитель практики от кафедры \_\_\_\_\_\_\_\_\_\_\_\_ С.А.К. Диане

«\_\_» \_\_\_\_\_\_ 2020 г.

Руководитель практики от университета \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Сухоленцева

«\_\_» \_\_\_\_\_\_ 2020 г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт кибернетики

*(наименование института, филиала)*

Кафедра проблем управления

*(наименование кафедры)*

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ ПО ППУ И ОПД**

*(указать вид практики*)

**Студенту 3 курса учебной группы КРБО-01-17**

**Кабанову Павлу Романовичу**

**Место и время практики: лаборатория для самостоятельных занятий кафедры проблем управления**

**Должность на практике: -**

**1. ЦЕЛЕВАЯ УСТАНОВКА:** **Алгоритмы нейросетевой обработки голосовых команд в задаче управления мобильным роботом**

**2. СОДЕРЖАНИЕ ПРАКТИКИ:**

2.1 Изучить: нейросетевые методы обработки звука, алгоритмы коррекции текста

2.2 Практически выполнить: разработать тестовое программное обеспечение, демонстрирующее возможности голосового управления виртуальной моделью мобильного робота

2.3 Ознакомиться: с существующими программными средствами для обучения нейронных сетей

**3.ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** -

**4. ОГРАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Заведующий кафедрой:

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.П. Романов

СОГЛАСОВАНО:

Руководитель практики от кафедры:

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А.К. Диане

Руководитель практики от Университета:

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Сухоленцева

Задание получил:

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Р. Кабанов

**Проведенные инструктажи:**

Охрана труда: «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

Инструктирующий \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент С.А.К. Диане

*Подпись Расшифровка, должность*

Инструктируемый \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Р. Кабанов

*Подпись Расшифровка*

Техника безопасности: «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

Инструктирующий \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент С.А.К. Диане

*Подпись Расшифровка, должность*

Инструктируемый \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Р. Кабанов

*Подпись Расшифровка*

Пожарная безопасность: «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

Инструктирующий \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент С.А.К. Диане

*Подпись Расшифровка, должность*

Инструктируемый \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Р. Кабанов

*Подпись Расшифровка*

С правилами внутреннего распорядка ознакомлен: «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Р. Кабанов

*Подпись Расшифровка*

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ**

**ПРАКТИКИ ПО ППУ И ОПД**

студента 3 курса группы **КРБО-01-17** очной формы обучения, обучающегося по направлению подготовки «Мехатроника и робототехника», профиль «Автономные роботы»

|  |  |  |  |
| --- | --- | --- | --- |
| **Неделя** | **Сроки выполнения** | **Этап** | **Отметка о выполнении** |
| **1-5** | **10.02.2020 – 10.03.2020** | **Изучение нейросетевых методов обработки звука, алгоритмов коррекции текста** |  |
| **5-10** | **10.03.2020 – 20.04.2020** | **Ознакомление с существующими программными средствами для обучения нейронных сетей** |  |
| **10-16** | **20.04.2020 – 30.05.2020** | **Разработка тестового программного обеспечения, демонстрирующего возможности голосового управления виртуальной моделью мобильного робота** |  |

**Согласовано:**

|  |  |  |
| --- | --- | --- |
| Заведующий кафедрой | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | М.П. Романов |
| Руководитель практики от кафедры | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | С.А.К. Диане |
| Руководитель практики от Университета | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | А.А. Сухоленцева |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | П.Р. Кабанов |

Оглавление

[**1 Введение** 7](#_Toc41704032)

[**2 Нейросетевые методы распознавания речи** 8](#_Toc41704033)

[2.1 Принцип работы 8](#_Toc41704034)

[2.2 Основные проблемы 17](#_Toc41704035)

[2.3 Метрики оценки 18](#_Toc41704036)

[**3 Алгоритмы коррекции ошибок распознавания** 21](#_Toc41704037)

[**4 Управление мобильным роботом с помощью голосовых команд** 22](#_Toc41704038)

[4.1 Обработка естественного языка 22](#_Toc41704039)

[4.2 Алгоритм поиска по словарю 25](#_Toc41704040)

[4.3 Нейросетевые алгоритмы обработки команд 26](#_Toc41704041)

[**5 Средства реализации** 28](#_Toc41704042)

[**6 Разработка тестового ПО для управления роботом** 28](#_Toc41704043)

[**7 Вывод** 31](#_Toc41704044)

[**8 Список литературы** 32](#_Toc41704045)

[**9 Приложение** 34](#_Toc41704046)

# 1 Введение

В последние годы распознавание речи набирает все больше популярности и используется во многих сферах жизни человека.

Целью данной работы является изучение принципов работы технологий распознавания речи для исследования возможности улучшения качества распознавания и методов обработки текста. Подобное исследование обеспечит возможность создания доступного человеко-машинного интерфейса для управления мобильным роботом.

В работе приведен результат создания и тестирования симулятора робота, управляемого с помощью голосовых команд.

# 2 Нейросетевые методы распознавания речи

## 2.1 Принцип работы

Распознавание речи – одна из задач, в которой активно используются нейронные сети и машинное обучение. Рассмотрим основные принципы работы и архитектуры систем распознавания.

В большинстве случаев нейросетевое распознавание речи работает по принципу, изображенному на Рис.1.



Рисунок 1 – Процесс распознавания речи

Первый шаг подразумевает под собой акустическое окружение и оборудование для съема сигнала (микрофон, предусилитель и АЦП). Это немаловажные факторы, которые повлияют на дальнейшее качество распознавания речи. Также сюда можно включить посторонний шум и реверберацию помещения.

Второй шаг – подготовка полученного сигнала к дальнейшей обработке путем избавления от помех, полученных при его записи, а также преобразование звуковой волны в доступный для нейросетевой обработки формат.

Третьим шагом является извлечение признаков из обработанной речи.

Четвертый шаг – классификация полученных признаков из входных данных по известному словарному/фонемному запасу, после чего осуществляется вывод текста.

Остановимся немного подробнее на втором шаге. В первую очередь, нужно преобразовать аналоговый сигнал в цифровой. Для этого используется техника, называемая сэмплированием. *Сэмплирование* представляет собой дискретизацию исходной волны с определенной частотой. Как правило, для распознавания речи достаточно частоты 16 кГц.

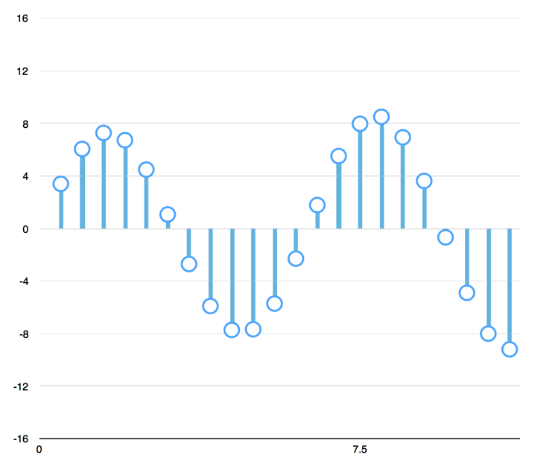
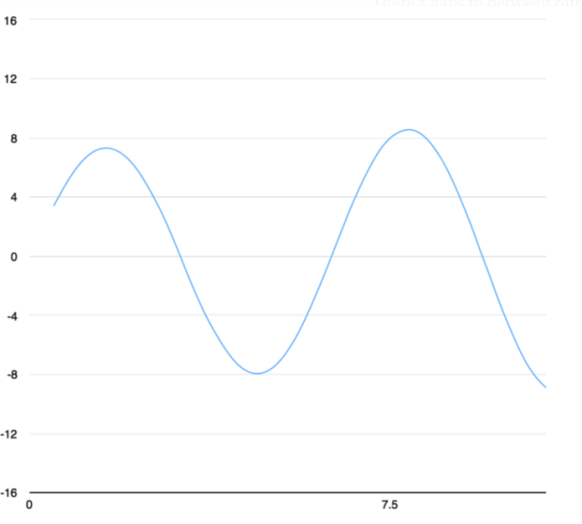


Рисунок 2 – Результат сэмплирования

По теореме Котельникова, гласящей, что любую функцию{\displaystyle F(t)}, состоящую из частот от 0 до f1{\displaystyle f\_{1}}, можно непрерывно передавать с любой точностью при помощи чисел, следующих друг за другом через{\displaystyle 1/(2f\_{1})} 1/(2f1) секунд, можно утверждать, что данное преобразование справедливо и никаких потерь данных не будет.

Прежде чем обрабатывать преобразованный сигнал, нужно определить в какой форме его лучше всего представить для последующего анализа. Речь можно представить несколькими способам, например, в виде *волны*, *спектрограммы* или *разложения по коэффициентам мел-кепстрограммы* (Mel Frequency Cepstrum Coefficients (MFCCs)).

Главным недостатком представления звука в виде волны является невозможность извлечения какой-либо информации о речи, и даже, наоборот, наличие слишком большого количества ненужных данных. На Рис. 3 изображено сравнение волновых изображений слов «Left» и «One».

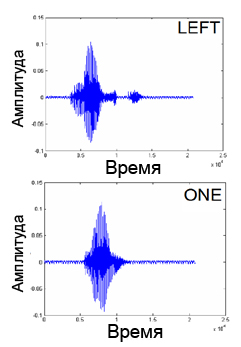


Рисунок 3 – Сравнение волновых диаграмм

Как видно на приведенном примере, по этим данным очень сложно различить эти слова, не говоря уже об извлечении каких-либо признаков из них.

В случае спектрограммы данные будут представляться следующим образом:

-Ось X: Время (мс)

-Ось Y: Частота

-Ось Z: Значение частоты

Сэмпл разбивается на несколько фреймов, как правило, перекрывающих 50% сэмпла. Для каждого такого фрейма рассчитывается частотный спектр.

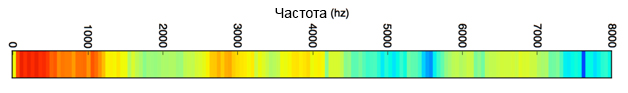


Рисунок 4 – Пример частотного спектра

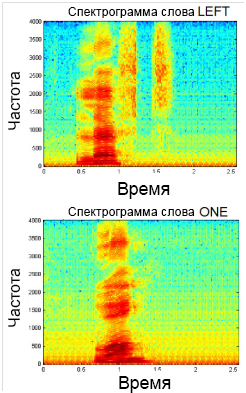


Рисунок 5 – Пример спектрограмм

Как можно заметить, в отличие от волнового представления, спектрограмма дает гораздо больше информации о сигнале, что позволяет четко отличить два разных слова друг от друга.

Несмотря на качественную визуализацию сигнала, существует ряд проблем, которые возникают при обработке речи. Так как скорость речи может варьироваться, сэмплы не могут каждый раз показывать одно и то же место, временной параметр спектрограммы будет отличаться, соответственно значения частот на одинаковых временных отметках также будут различаться. На Рис. 5 приведен пример такого «смещения» слова «Left». Обе спектрограммы имеют одинаковый рисунок, но на практике их не удастся качественно сравнить, пока не будет произведена нормализация по времени.

Для определения начала слова используют детекцию на основе *энтропии*. Энтропия - мера неопределенности некоторой случайной величины. Любую информацию можно охарактеризовать как уменьшение энтропии системы, которое она производит.

Вначале нужно вычислить энтропию речевого сигнала, затем установить критерий, по которому будет определяться нужного слова:

где Hk – энтропия, pk – плотность вероятностей.

Началом слова будет точка, где кривая энтропии пересечет прямую

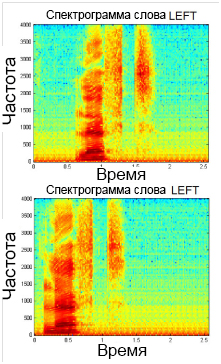


Рисунок 6 – Спектрограммы двух разных сэмплов

Рассчитав энтропию слова «left» получим график, изображенный на Рис.7. Горизонтальная линия является критерием lambda, вертикальная – началом слова.

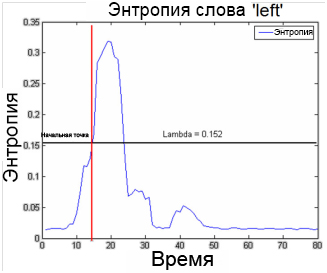


Рисунок 7 – Энтропия слова «Left»

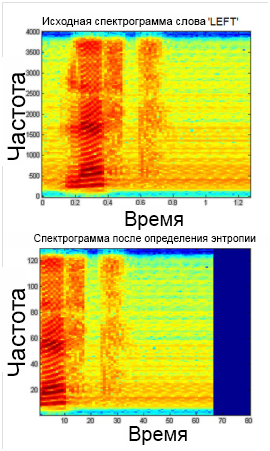


Рисунок 8 – Смещение фрейма после обнаружения начала слова

Теперь, когда фрейм был нормализован по времени, можно приступать к распознаванию самого слова. Существует несколько методов классификации, используемых в распознавании речи.

*Dynamic Time Warping* (DTW) работает по принципу сравнения спектрограмм двух слов. Так как слова могут быть произнесены с разной скоростью, необходимо провести нормализацию спектрограммы распознаваемого слова по времени, растягивая или сжимая ее, пока не будет совпадения**.**

*Hidden Markov Modelling* (HMM) - статистическая модель, имитирующая марковский процесс. Слово разбивается на составляющие, которые представляются как состояния в Марковской модели. Обрабатываемое слово сравнивается с наиболее подходящей моделью, и с различной вероятностью будет предсказываться переход из одного состояния к другому. Например, вероятность того, что слово может начинаться с “xq”практически равна нулю.

При использовании *нейросетей* основной задачей становится установление весовых коэффициентов между нейронами, в то время как при использовании HMM главной задачей является поиск подходящих вероятностей и переход в нужное состояние. В нашей работе мы будем рассматривать *нейросетевой подход* к распознаванию речи.

В системах распознавания речи используются несколько архитектур нейронных сетей: рекуррентные (RNN), свёрточные (CNN), Multilayer Feedforward и Radial basis networks. В настоящее время для распознавания чаще всего используются рекуррентные нейронные сети, которые и будут рассмотрены далее.

*Рекуррентные нейронные сети* - вид [нейронных сетей](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных [перцептронов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD" \o "Перцептрон), рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети *RNN* применимы в таких задачах, где нечто целостное разбито на части, что как раз подходит для распознавания речи. Схематично описать рекуррентную нейронную сеть можно следующим образом: слой RNN использует цикл for для итерации по упорядоченной по времени последовательности, храня при этом во внутреннем состоянии, закодированную информацию о шагах, которые он уже видел.

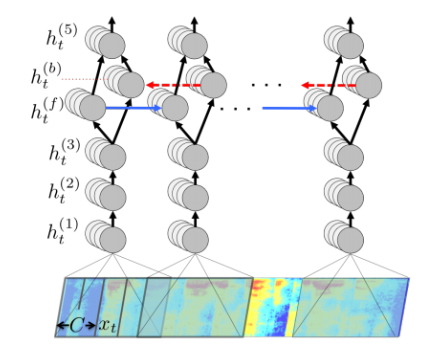


Рисунок 9 – Пример RNN-модели

Разделим фрейм со словом на куски по 20 мс и будем определять буквы, соответствующие произнесенным звукам.

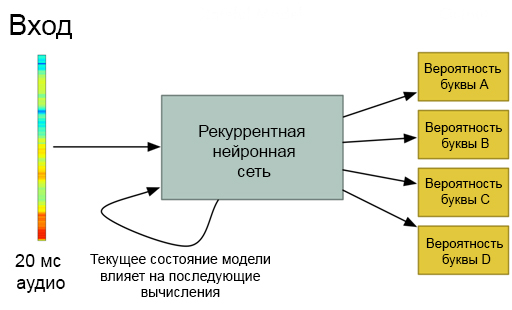


Рисунок 10 – Схема использования RNN

Благодаря использованию RNN, каждая новая распознанная буква будет влиять на вероятность последующей буквы в слове. Например, если сказать «HEL», вероятность того что за этим последует «LO» гораздо больше, чем вероятность, например, «XYZ». Таким образом, наличие памяти о предыдущих предсказаниях помогает нейронной сети делать более точные прогнозы в будущем.

После того, как все части слова пройдут через нейросеть, на выходе получится сопоставление каждого аудиофрагмента с соответствующими буквами. Рассмотрим на примере слова «Hello», Рис. 11. Красным цветом отмечены наиболее вероятные варианты распознавания.

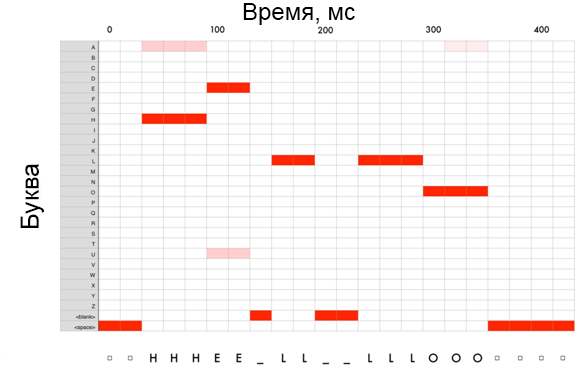


Рисунок 11 – Результат распознавания

Нейросеть выдала следующие вероятные результаты:

* HHHEE\_LL\_LLLOOO
* HHHUU\_LL\_LLLOOO
* AAAUU\_LL\_LLLOOO

Повторяющиеся буквы говорят о длительности произнесения определенного звука, так как система может распознавать только одну букву за раз. Произведя замену повторяющихся букв и пробелов, можно получить следующее

* HHHEE\_LL\_LLLOOO -> HELLO
* HHHUU\_LL\_LLLOOO -> HULLO
* AAAUU\_LL\_LLLOOO -> AULLO

Все три слова созвучны, и немаловажной задачей является определение верного варианта. Для этого используется база каких-либо текстов, в которых это слово может встречаться чаще всего. В целом, работу систем распознавания речи можно изобразить на Рис.12.

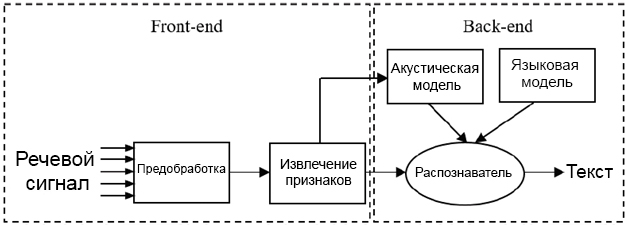


Рисунок 12 – Схема работы систем распознавания речи

## 2.2 Основные проблемы

Как уже было ранее упомянуто, при распознавании речи может возникнуть множество проблем в связи с тем, что человеческая речь – очень субъективный феномен. Рассмотрим одни из основных проблем.

* Особенности произношения: одно и то же слово может звучать по-разному у разных людей в зависимости от возраста, пола, анатомических особенностей, скорости речи, эмоционального состояния и особенностей диалекта.
* Посторонний шум: шумовое окружение может очень сильно засорить полезный сигнал. Даже сам говорящий может создавать шум при произнесении фразы.
* Супрасегментные единицы языка: влияние интонации, тона, мелодики, ритма, интенсивности, долготы и ударения на гласные. Эти аспекты сильно влияют на произношение каждого отдельного слова.
* Непрерывность речи: во время произнесений фразы мы редко делаем длинные паузы между словами. Речь это в основном непрерывный поток звуков, что делает обнаружение отдельных слов достаточно трудной задачей.
* Прочие факторы: позиция микрофона по отношению к говорящему, направление микрофона и т.д.

Для решения некоторых проблем можно применить как аппаратные решения для шумоподавления и улучшения качества входного полезного сигнала (полосовые фильтры, активное шумоподавление), так и программные, такие как фильтры частот в диапазоне человеческой речи.

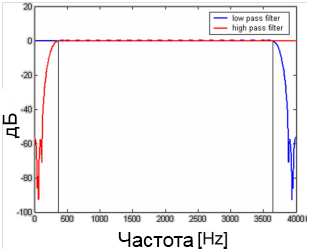


Рисунок 13 – Использование фильтров высоких и низких частот для очистки сигнала от шумов

Многие факторы также должны быть учтены в используемом для обучения наборе нейросети данных (далее датасет), так чтобы будущая модель была готова с ними справиться в реальных условиях. Для создания качественной модели распознавания могут понадобиться тысячи часов размеченной записи речи разных людей. Несмотря на большие объемы данных для обучения, современные системы распознавания хоть и активно совершенствуются, но все же имеют ряд недостатков и неточности распознавания.

## 2.3 Метрики оценки

Для оценки работы систем распознавания речи и машинного перевода принято использовать метрику *Word Error Rate* (WER).

Основная сложность при оценке эффективности заключается в том, что распознанная последовательность слов может иметь длину, отличную от исходной последовательности (предположительно, правильной). WER основана на метрике Расстояния Левенштейна.

Расстояние Левенштейна -  [метрика](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0_(%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)), измеряющая разность между двумя последовательностями символов. Она определяется как минимальное количество односимвольных операций (а именно вставки, удаления, замены), необходимых для превращения одной последовательности символов в другую. В общем случае, операциям, используемым в этом преобразовании, можно назначить разные цены.

Основное отличие метрики WER от Расстояния Левенштейна заключается в том, что WER работает на уровне цельных слов, а не отдельных фонем. WER является ценным инструментом для сравнения качества различных систем, а также для оценки улучшений в рамках одной системы. Однако, данная метрика не дает подробных сведений о природе ошибок распознавания, и поэтому требуется дальнейшая работа для определения источника ошибок.

WER рассчитывается по следующей формуле:

где

* S – число замен (Substitutions). Правильное слово распознано и заменено неправильным;
* D – число удалений (Deletions). Исходное слово отсутствует в результате распознавания;
* I – число вставок (Insertions). В последовательность слов вставляется не относящееся к оригиналу слово;
* C – число верных слов (Correct words);
* N – общее число слов примера (N=S+D+C).

В частности для систем распознавания речи также иногда используют метрику *Word Accuracy* (WAcc), которая рассчитывается по формуле

Так как обе метрики являются отношением к количеству слов в исходном (правильном) примере, значение WER не может быть больше 1, а значение WAcc , таким образом, может быть меньше 0.0.

Главным недостатком метрики WER является то, что она не показывает абсолютный процент качества системы, а лишь позволяет сравнить одну систему с другой.

Помимо WER существуют и другие показатели качества модели. Relative Information Loss (RIL), основанная на взаимной информации (MI), измеряет статистическую зависимость между входом X и выходом Y, и вычисляется с помощью энтропии (H)

где

следовательно

Однако, RIL тоже все еще далека от идеала из-за сложности применения. Также RIL измеряет нулевую ошибку для любого однозначного соответствия между словами ввода/вывода, что не соответствует критериям идеального показателя качества модели распознавания речи.

Существует метрика Word Information Loss (WIL), являющаяся аппроксимацией RIL, но в отличие от нее проще в использовании, так как основывается на вышеуказанных величинах и рассчитывается по формуле

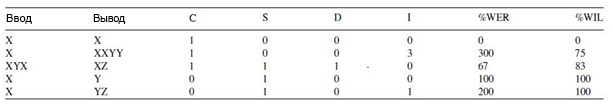


Рисунок 14 – Сравнение по метрикам WER и WIL

# 3 Алгоритмы коррекции ошибок распознавания

Как было описано выше, распознавание речи не обходится без ошибок. Их минимизация является не менее сложной задачей, чем само распознавание слов. Автоматическая коррекция может стать единственным способом улучшения производительности системы распознавания, в случае если она является «Черным ящиком» и у нас нет доступа к ее внутренностям или в случае если исправление вручную не имеет места.

Коррекция результатов распознавания состоит из двух этапов – детектирования ошибок и поиска оптимального варианта их исправления.

Основной задачей при детектировании ошибок является создание транскрипций с использованием признаков, полученных по результатам работы системы (Decoder-based features), к чему можно отнести определение степени уверенности в правильности слова, использование языковой модели и сети ошибок, представляющей собой ориентированный граф без циклов с возможными вариантами слов (confusion network density). В дальнейшем это поможет классифицировать слово как верное или неверное.

Существует два направления исследований в области коррекции результатов работы системы распознавания: извлечение признаков из decoder-based features и поиск дополнительных признаков (non-decoder-based features) из гипотетической последовательности слов в предложении, такие как N-граммы, части речи, синтаксические и семантические признаки. Говоря просnо, второй подход представляет собой некий дополнительный модуль, который добавляется к любой готовой системе распознавания речи без необходимости глубокого внедрения в алгоритмы обработки и распознавания звука, работающий с уже готовым текстом. Такой подход можно назвать Post-Editing correction (PEC), и благодаря своему принципу работы он имеет множество перспектив в области OCR, NLP и машинного перевода.

В настоящее время хороших показателей добилась компания Google, опубликовав статью «A spelling correction model for end-to-end speech recognition», в которой подробно описывается их принципа Post-Editing коррекции, в котором анализируется конечный текст путем его соотношения с примерами звучания на text to speech модели. Этот метод позволил уменьшить количество ошибок на 29%.

Протестировав различные системы распознавания речи на русском языке, удалось заметить, что в конечном тексте иногда появляются ошибки следующего характера:

* Неверно распознана форма слова, что влияет на согласование слов в предложении;
* Из-за некачественного распознавания слово было заменено/удалено, что препятствует пониманию смысла предложения;
* В некоторых системах отсутствует пунктуация.

Если ошибки первого типа не так сильно влияют на понимание текста, то во втором случае это становится серьезной проблемой. Восстановить неверное или пропущенное слово можно только по контексту предложения или текста.

Для решения этой проблемы можно предложить следующее решение: начать работать с текстом по направлению NLP, что также позволит использовать это в мобильной робототехнике.

# 4 Управление мобильным роботом с помощью голосовых команд

Технологии распознавания речи можно использовать для управления мобильными роботами. Одним из преимуществ такого подхода является его простота использования, так как для управления роботом не потребуется специальной подготовки и оборудования, что позволит создать максимально простой человеко-машинный интерфейс. С другой стороны, для создания максимально эффективного интерфейса требуется тщательный и грамотный подход к разработке алгоритма распознавания и анализа команд.

## 4.1 Обработка естественного языка

Прежде чем начать описание алгоритмов, стоит ввести понятие обработки естественного языка (Natural Language Processing, далее – NLP). *Обработка естественного языка* - подраздел информатики, посвященный тому, как компьютеры анализируют естественные (человеческие) языки. NLP позволяет применять алгоритмы машинного обучения для текста и речи. В задачи NLP также входит анализ синтаксиса, семантики и морфологии текста.

Для анализа текста его необходимо разбить на предложения-компоненты. Этот процесс называется *токенизацией по предложениям*. В английском и некоторых других языках мы можем вычленять предложение каждый раз, когда находим определенный знак пунктуации – точку.

Но даже в английском эта задача нетривиальна, так как точка используется и в сокращениях. Таблица сокращений может сильно помочь во время обработки текста, чтобы избежать неверной расстановки границ предложений. В большинстве случаев для этого используются библиотеки, которые будут рассмотрены позже.

Приведем пример из английского языка (в дальнейшем также будем приводить примеры на нем, так как английский язык чаще всего применяется в NLP, и на данном этапе работы на нем имеется больше всего примеров и исследований). Возьмем небольшой текст про настольную игру нарды:

*«Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East. It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.»*

После токенизации получим следующий результат:

*«Backgammon is one of the oldest known board games.»*

*«Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.»*

*«It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.»*

Следующим шагом станет *токенизация по словам*. В английском и многих других языках, использующих ту или иную версию латинского алфавита, пробел – это разделитель слов. Тем не менее, могут возникнуть проблемы, если мы будем использовать только пробел – в английском составные существительные пишутся по-разному и иногда через пробел.

Пример:

*['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']*

*['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']*

*['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']*

Обычно тексты содержат разные грамматические формы одного и того же слова, а также могут встречаться однокоренные слова. *Нормализация* преследует цель привести все встречающиеся словоформы к одной, нормальной словарной форме. Приведение разных словоформ к одной:

*dog, dogs, dog’s, dogs’ => dog*

То же самое, но уже применительно к целому предложению:

*the boy’s dogs are different sizes => the boy dog be differ size*

Два основных способа нормализации – *лемматизация* и *стемминг*.

*Стемминг* – это грубый процесс, который отрезает «лишнее» от корня слов, часто это приводит к потере словообразовательных суффиксов.

*Лемматизация* – это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме.

Отличие в том, что стеммер (конкретная реализация алгоритма) действует без знания контекста и, соответственно, не понимает разницу между словами, которые имеют разный смысл в зависимости от части речи. Однако у стеммеров есть и свои преимущества: их проще внедрить, и они работают быстрее. Плюс, более низкая точность может не иметь значения в некоторых случаях. Приведем примеры:

1. Слово good – это лемма для слова better. Стеммер не увидит эту связь, так как здесь нужно сверяться со словарем.
2. Слово play – это базовая форма слова playing. Тут справятся и стемминг, и лемматизация.
3. Слово meeting может быть как нормальной формой существительного, так и формой глагола to meet, в зависимости от контекста. В отличие от стемминга, лемматизация попробует выбрать правильную лемму, опираясь на контекст.

*Стоп-слова* – это слова, которые выкидываются из текста до/после его обработки. Когда мы применяем машинное обучение к текстам, такие слова могут внести лишний шум, поэтому необходимо избавиться от нерелевантных слов. Под стоп-словами обычно подразумевают артикли, междометия, союзы и т.д., которые не несут смысловой нагрузки. При этом надо понимать, что не существует универсального списка стоп-слов, все зависит от конкретного случая.

Возьмем предыдущий пример с нардами:

*['Backgammon', 'one', 'oldest', 'known', 'board', 'games', '.']*

Алгоритмы машинного обучения не могут напрямую работать с сырым текстом, поэтому необходимо конвертировать текст в векторы. *Мешок слов (Bag-of-words)* – это популярная и простая техника извлечения признаков, используемая при работе с текстом. Она описывает вхождения каждого слова в текст. Чтобы использовать модель, нам нужно:

1. Определить словарь известных слов (токенов).
2. Выбрать степень присутствия известных слов.

Любая информация о порядке или структуре слов игнорируется. Отсюда и происходит название «мешок слов». Эта модель пытается понять, встречается ли знакомое слово в документе, но не знает, где именно оно встречается.

## 4.2 Алгоритм поиска по словарю

Первый и самый простой вариант алгоритма распознавания команд для управления роботом – создание словаря с заранее прописанными ключевыми словами. При вводе голосовой команды в распознанном тексте осуществляется поиск определенных ключевых слов, и при их наличии запускается последующий алгоритм действий робота. Это самый простой в реализации алгоритм, имеющий ряд недостатков. Во-первых, необходимо вручную создать словарь, с которым будут сравниваться команды. Во-вторых, для работы необходимо точно называть нужные слова в определенном порядке. Также желательно знать заранее на какие команды робот среагирует, а на какие нет. В целом такой подход в обработке команд является крайне примитивным и малоэффективным, что было продемонстрировано в описанной далее программе.

## 4.3 Нейросетевые алгоритмы обработки команд

Для обработки введенных команд можно использовать нейросети. Рассмотрим алгоритмы, с помощью которых возможно решить эту задачу.

После нормализации текста нужно определить, к каким частям речи относятся слова в предложении. Так как для обозначения каких-либо действий используются глаголы, будем определять их. Но в предложении нам понадобятся не все глаголы, а какой-то один, определяющий действие робота. Здесь можно выделить два подхода.

Первый подход – определять какой из глаголов в предложении является синонимом нужной нам команды. Для решения этой задачи можно предложить алгоритмы word2vec. В общих чертах смысл алгоритма заключается в преобразовании слов в вектора и поиске ближайших к ним соседей. В нашем случае мы будем искать синоним к заранее заданному слову-команде в предложении. После нахождения синонима нужно будет извлечь дополнительную информацию, например, при команде переместиться на определенное расстояние, нужно найти само расстояние, на которое должен переместиться робот.

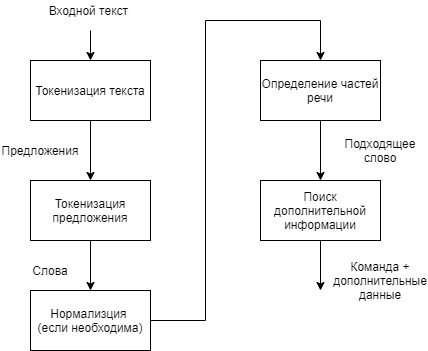


Рисунок 15 – Алгоритм работы обработчика

Результатом работы такого алгоритма должен стать список, состоящий из команды (того слова, синоним которого мы искали в предложении) и дополнительных данных, если они требуются. После того как нужные данные были определены, их можно передать в систему управления роботам для исполнения.

В сравнении с поиском по словарю, данный алгоритм имеет следующие преимущества:

* Не требуется вручную заполнять словарь для каждой отдельно взятой команды;
* Гибкость в использовании. Формы слова не должны влиять на результат работы, команда будет распознана;
* Нет привязки к конкретным ключевым словам.

Второй вариант исполнения нейросетевого метода обработки команд почти идентичен методу с определением синонимов, с единственным отличием в определении глагола действия, относящегося к команде, которую нужно выполнить роботу. Вместо поиска синонима, можно провести классификацию глаголов в предложении по значению, отнеся их таким образом к глаголом определенного действия (глагол движения, перемещения какого-либо объекта, применения силы и т.д.). Это позволит работать без привязки к какому-либо конкретному слову-синониму, а опираться лишь на семантическое значение слова. Проблема данного метода заключается в сложности семантической классификации слов, и требуется дополнительно изучить методы подобной классификации. Пример подобной работы приведен в статье [Assigning verbs to semantic classes via wordnet].

# 5 Средства реализации

Одним из самых популярных и эффективных средств для работы с естественным языком является библиотека для Python «Natural Language Toolkit». Она включает в себя графические представления деревьев решений, примеры данных, а также функционал, позволяющий сосредоточиться на применении алгоритмов и обучении нейронных сетей. Работу можно начать с использованием английского языка, так как в сравнении с русским он может быть проще с точки зрения анализа и имеет больше данных для обучения и исследования. В качестве лексической и языковой базы предлагается использовать платформу wordnet.

Для обучения нейронных сетей и решения некоторых задач машинного обучения можно использовать библиотеки Python tensorflow, scikitlearn, для подготовки данных и визуализации – pandas и matplotlib.

# 6 Разработка тестового ПО для управления роботом

В ходе практики была написана программа на языке python, демонстрирующая возможность управления виртуальной моделью робота с помощью голосового интерфейса. На данном этапе работы был использован метод определения команд по поиску в заранее созданном словаре.

Для демонстрации было проработано движение робота на количество пикселей в определенном направлении. Для работы нужно произнести фразу примерно следующего содержания: «\*кол-во пикселей\* pixels \*направление движения\*». В данный момент используется модуль распознавания англоязычной речи из библиотеки speechrecognition.

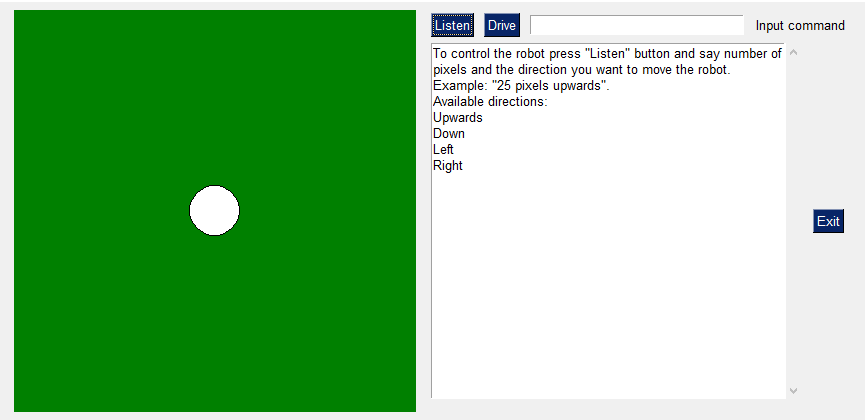


Рисунок 16 – Стартовый экран симулятора

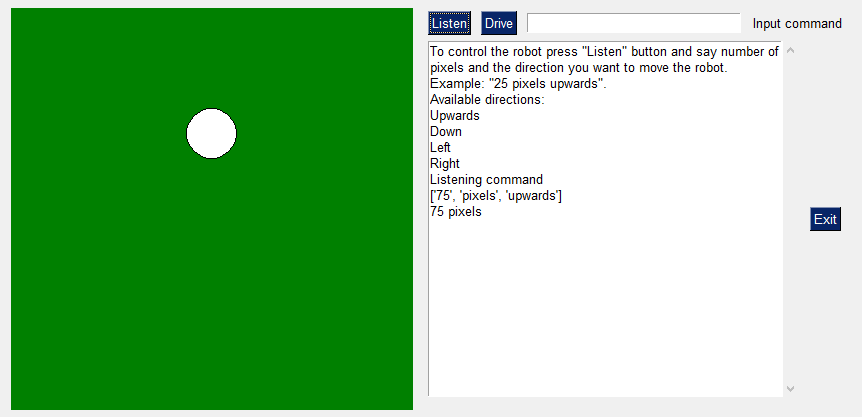


Рисунок 17 – Результат выполнения команды перемещения на 75 пикселей вверх

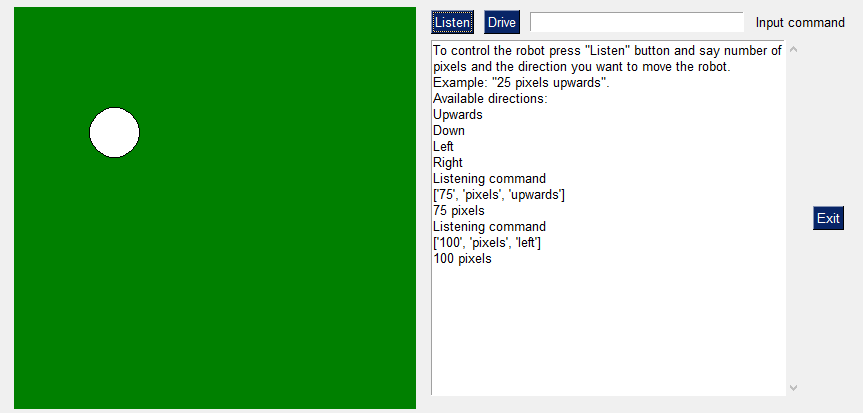


Рисунок 18 – Результат выполнения команды перемещения на 100 пикселей влево

# 7 Вывод

В результате изучения материалов по теме были разобраны принципы и особенности распознавания речи и возможности коррекции полученных результатов.

Также в рамках практики были изучены методы обработки естественного языка для управления роботом. Были предложены три метода обработки команд, по одному из них написана программная реализация.

По результатам тестирования программы, можно сказать, что метод поиска команд по словарю – довольно неоптимален и имеет малую эффективность при работе. Дополнительно стоит отметить, что при использовании API для распознавания речи возникали достаточно серьезные ошибки распознавания. Даже при достаточно сильном акценте, английская речь достаточно точно распознается при хорошем качестве входного сигнала. Из этого можно сделать вывод, что проблема состоит в низком качестве микрофона, с использованием которого проводились тесты. Также при выборе движка для распознавания необходимо учитывать метрики оценки качества. Для решения данной проблемы можно предложить реализовать аппаратный/программный фильтр, настроенный в диапазоне человеческой речи и изучить дополнительные возможные средства коррекции ошибок распознавания в форме текста с использованием нейросетей.

В целом, задача разработки и улучшения производительности системы распознавания речи очень масштабна и может оказаться нецелесообразной в связи с недостатком ресурсов и возможностей. Вместо этого можно сосредоточиться на реализации и проверке предложенных алгоритмов обработки команд, а также в целом подробнее изучить методы обработки естественного языка.

# 8 Список литературы

* + 1. Д.П. Ким Теория автоматического управления. Том 1. – М.: «Физматлит», 2007. – 310с.
    2. S. Bird, E. Klein, E. Loper Natural Language Processing with Python – «O’Reilly Media», 2009 – 504с.
    3. Gevaert, G. Tsenov, V. Mladenov Neural Networks used for Speech Recognition // Journal of Automatic Control – 2010. – № 20
    4. J. Guo, T.N. Sainath, R.J. Weiss A Spelling Correction model for End-to-End Speech Secognition // University of California – 2019.
    5. R.Errattahi, A.E.Hannani, H.Ouahmane Automatic Speech Recognition Errors Detection and Correction: A Review // Procedia Computer Science – 2018г. – №128
    6. A. Korhonen Assigning Verbs to Semantic Classes via Wordnet //University of Cambridge – 2002г.
    7. V.Yordanov Introduction to Natural Language Processing for Text [Электронный ресурс]: - Электрон. текстовые дан. – 2018г. –Режим доступа: <https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63>
    8. T. Ganegedara Intuitive Guide to Understanding Word2vec [Электронный ресурс]: - Электрон. текстовые дан. – 2018г. –Режим доступа: <https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-word2vec-e0128a460f0f>
    9. A. Radzinski Introduction Into Semantic Modelling for Natural Language Processing [Электронный ресурс]: – Электрон. текстовые дан. – 2018г. –Режим доступа: <https://chatbotslife.com/introduction-into-semantic-modelling-for-natural-language-processing-c4c885175ca7>
    10. Z. Li A Beginner’s Guide to Word Embedding with Gensim Word2Vec Model [Электронный ресурс]: – Электрон. текстовые дан. – 2019г. –Режим доступа: <https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92>
    11. A. Getitgey Machine Learning is Fun Part 6: How to do Speech Recognition with Deep Learning [Электронный ресурс]: – Электрон. текстовые дан. – 2016г. – Режим доступа: <https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a>
    12. D. Mwiti A 2019 Guide for Automatic Speech Recognition [Электронный ресурс]: – Электрон. текстовые дан. – 2019г. – Режим доступа: <https://heartbeat.fritz.ai/a-2019-guide-for-automatic-speech-recognition-f1e1129a141c>

# 9 Приложение

import PySimpleGUI as sg

import time

import speech\_recognition as sr

sg.theme('SystemDefault')

column1 = [[sg.Button("Listen", key = 'listen'),sg.Button("Drive", key = 'drive'), sg.InputText(size=(30,5)),

           sg.T(text="Input command")],

            [sg.Output(size = (50,22)),sg.Exit()]]

layout = [[sg.Graph(background\_color = "green", canvas\_size=(400, 400), graph\_bottom\_left = (0,0),graph\_top\_right = (400,400), key='graph'),sg.Column(column1)]]

class Robot:

    def \_\_init\_\_(self, xcoord, ycoord, grObj):

        self.xcoord = xcoord

        self.ycoord = ycoord

        self.grObj = grObj

        self.model = self.grObj.DrawCircle((xcoord, ycoord), 25, fill\_color="white")

    def move\_robot(self,dx,dy):

        self.grObj.MoveFigure(self.model, dx, dy)

def recognition(recognizer,src, wndw):

    recognizer.adjust\_for\_ambient\_noise(src)

    print("Listening command")

    wndw.Refresh()

    audio = recognizer.listen(src)

    try:

        result = recognizer.recognize\_google(audio)

        return result

    except Exception as e:

        print("Error " + str(e))

        destination = [0, 0]

        return destination

def parse\_command(cmd):

    try:

        words = cmd.split(' ')

        print (words)

        if 3 == len(words):

            try:

                words[0] = int(words[0])

                if words[1]=='pixels':

                    if words[2] == 'upwards':

                        print (words[0], 'pixels')

                        destination = [0,words[0]]

                    elif words[2] == 'down':

                        print(words[0], 'pixels')

                        destination = [0, -words[0]]

                    elif words[2] == 'left':

                        print(words[0], 'pixels')

                        destination = [-words[0],0 ]

                    elif words[2] == 'right':

                        print(words[0], 'pixels')

                        destination = [words[0],0 ]

                    else:

                        print('Try again')

                        destination = [0,0]

                else:

                    print('Try again')

                    destination = [0, 0]

            except:

                print ('error converting str to int')

                destination = [0, 0]

        else:

            print('Try again')

            destination = [0, 0]

        return destination

    except:

        print('Try again')

        destination = [0, 0]

def main():

    r = sr.Recognizer()

    with sr.Microphone() as source:

        r.adjust\_for\_ambient\_noise(source)

        window = sg.Window('Canvas test', layout)

        window.Finalize()

        graph = window['graph']

        robot = Robot(200,200,graph)

        print(

            'To control the robot press "Listen" button and say number of pixels and the direction you want to move the robot.\nExample: "25 pixels upwards". \nAvailable directions: \nUpwards\nDown\nLeft\nRight')

        while True:

            event, values = window.Read(timeout=100)

            if event == 'drive':

                print('You entered ', int(values[0]))

                robot.move\_robot(0,int(values[0]))

            if event == 'listen':

                command = recognition(r,source ,window)

                dest = parse\_command(command)

                if dest[1]==0:

                    for i in range(abs(dest[0])):

                        robot.move\_robot(dest[0]/abs(dest[0]), 0)

                        window.refresh()

                        time.sleep(0.01)

                else:

                    for i in range(abs(dest[1])):

                        robot.move\_robot(0, dest[1]/abs(dest[1]))

                        window.refresh()

                        time.sleep(0.01)

            if event == 'Exit':

                break

        window.close()

if \_\_name\_\_ == '\_\_main\_\_':

    main()