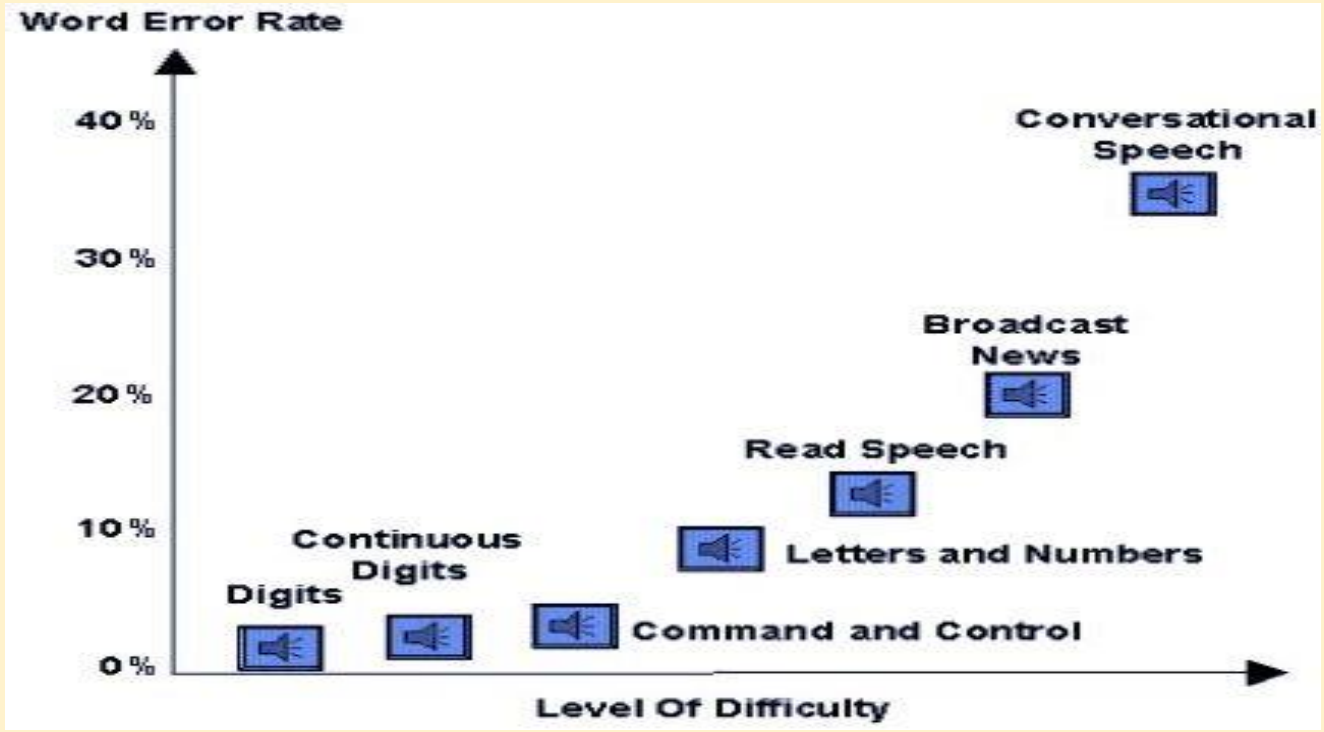
Interpretador de comandos de voz para controlo de veículos motorizado

- Evolução
- Reconhecimento de voz
- Ferramentas utilizadas
- Elementos da ferramenta
- Idiomas
- Cliente-Servidor
- Simulação / Robot

Evolução

- Grandes desenvolvimentos nesta área ao longo das últimas décadas



Reconhecimento de voz

Conceitos básicos

- Processo dinâmico
- Impossibilidade de distinguir as suas componentes de forma sistemática
- Conceito de probabilidade em vez de o determinístico

Estrutura

- Ideia de "stream" de áudio
- Fatores que afetam as propriedades acústicas
- Difonemas, Trifonemas ou até mesmo Quinfonemas

Seleção da ferramenta

- Cloud Speech API (Google)
- Bing Speech API (Microsoft)
- Kaldi
- CMU Sphinx

Funcionamento da ferramenta

- Palavras são constituídas por fonemas, porém estes nunca são iguais

- Nas ondas sonoras são identificadas 3 partes, designadas trifonemas
  - 1ª parte: depende do fonema precedente
  - 2ª parte: parte estável
  - 3ª parte: depende do fonema subsequente
- Onda de som dividida em *utterances*

Elementos da ferramenta

- O correto funcionamento da ferramenta necessita de vários elementos:
  - Modelo acústico
  - Dicionário
  - *Language Model*

Elementos da ferramenta

- Dicionário - palavras a ser traduzidas e a sua respetiva sequência de fonemas

```
AND       AH N D
AND(2)    AE N D
FORWARD   F AO R W ER D
LEFT      L EH F T
LISTENING         L IH S AH N IH NG
LISTENING(2)      L IH S N IH NG
MOVE      M UW V
```

Elementos da ferramenta

- Modelo de linguagem – Fornece informação da sequência de palavras que é possível reconhecer

- Tipos de modelo de linguagem:
  - *Keyword lists*
  - *Grammars*
  - *Statistical Language Model*

Keyword lists

- Possibilidade de especificar um *threshold* para cada palavra-chave presente na *keyword list*
- Apenas deteta as palavras/frases que lá estão presentes

```
oh mighty computer /1e-40/
hello world /1e-30/
other phrase /1e-20/
```

Grammar

- Modelo de linguagem mais restrito
- Sem necessidade de uma ferramenta para a sua construção
- Gramáticas muito complexas atrasam o reconhecimento

```
#JSGF V1.0;

grammar hello;
public <greet> = (good morning | hello) ( bhiksha | evandro | rita | will );
```

Statistical Language Model

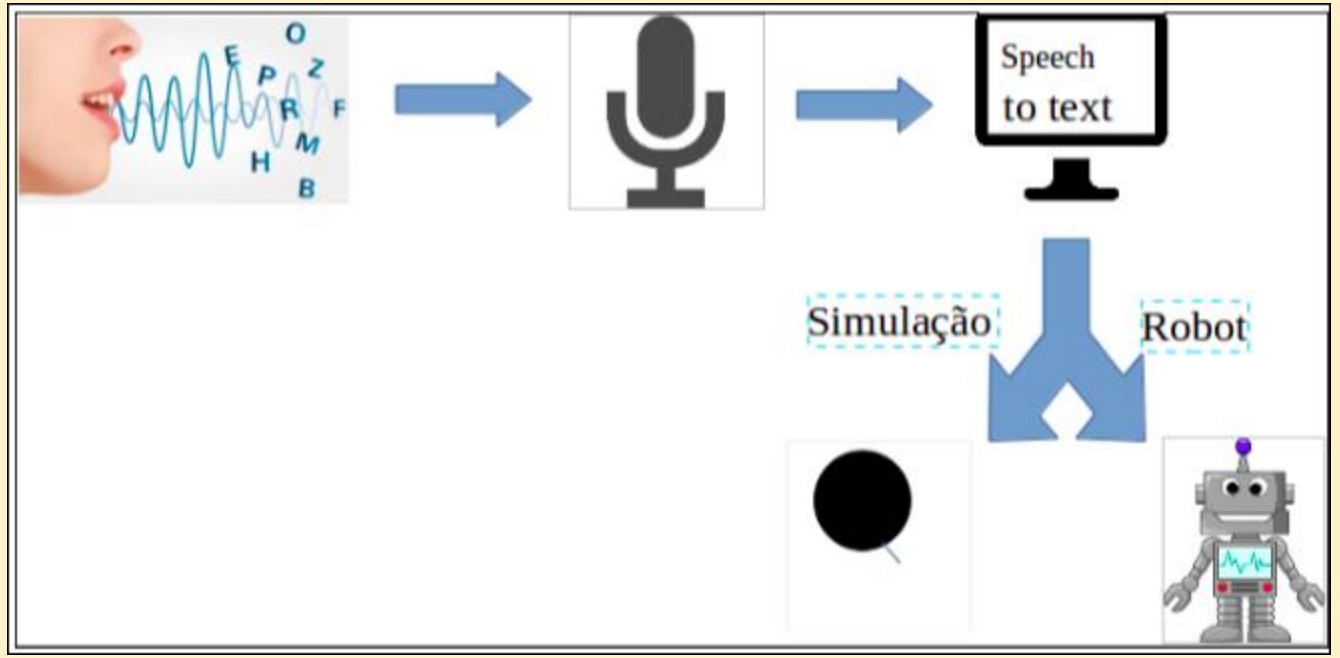
- Contém probabilidades das palavras
- A probabilidade da próxima palavra varia de acordo com a palavra dita
- Reconhece palavras OOV (out of vocabulary)

```
\1-grams:
-0.9638 </s> -0.3010
-0.9638 <s> -0.2511
-1.6628 BROWSER -0.2511
-1.3617 COMPUTER -0.2511
-1.6628 HELLO -0.2817
-1.3617 MUSIC -0.2511
-1.3617 OFF -0.2615
-1.6628 ON -0.2817
-1.6628 OPEN -0.2915
-1.1856 TURN -0.2717
```

Idioma

- O idioma utilizado neste projeto foi o Inglês
- Para adicionar um novo idioma é necessário seguir os seguintes passos:
  - Dados de áudio com algumas horas de duração
  - Treinar o modelo acústico, apenas recomendado caso se tenha 1 mês ou mais

Cliente - Servidor



Cliente-Servidor

- O projeto baseia-se numa arquitetura cliente-servidor
- O Cliente é o detetor de voz
- O Servidor é o objeto a ser controlado

Cliente

- Inicialmente começou-se por utilizar o *statistical language model*
- Numa fase posterior o mesmo foi substituído pela *keyword list*

Cliente

- De forma a generalizar o cliente foi desenvolvido um ficheiro de configuração com o seguinte formato:

  [Files]

  Dictionary: path

  Kws: path

  Logfn: path

  [Commands]

  comando = mensagem a enviar

Cliente - Algoritmo

- O programa começa por ler um ficheiro com o nome "config.txt", carrega os ficheiros necessários para o *Pocketsphinx* funcionar e de seguida carrega todos os comandos e as respetivas mensagens para um dicionário com o comando como chave e a mensagem como valor.
- Espera por comandos e envia a mensagem correspondente para o servidor.

Servidor

- O servidor deve começar por esperar por uma mensagem vinda do cliente com o conteúdo "talker"
- Neste projeto temos dois servidores o que controla o Robot e o que controla a simulação.

Servidor - Simulação

- Comandos
  - Move Forward - Esta mensagem ordena que a bola se mova em frente e envia a mensagem ”f 0.7”
  - Rotate Right - Ordena que a bola rode sobre si no sentido negativo, a mensageḿ e ”r -0.01”
  - Rotate Left - Ordena que a bola rode sobre si no sentido positivo, a mensageḿ e ”r 0.01”

Servidor - Simulação

- Comandos
  - Stop Moving - Ao receber esta mensagem a bola para completamente qualquer tipo de movimento, a mensagem é ”0”
  - Save Point - Este comando guarda o local atual da bola, a mensagem é "save point"
  - Go To Location - É o comando que ordena a bola a ir ao encontro do ponto guardado com o comando ”Save Point”, a mensagem enviada é "go to"

Servidor - Simulação

- Ficheiro de configuração da simulação

```
[Files]
Dictionary: /home/paulo/PycharmProjects/VoiceRecognition/Data/Simulacao/Simulacao.dict
Kws: /home/paulo/PycharmProjects/VoiceRecognition/Data/Simulacao/Simulacao_Keyphrase.list
Logfn: /home/paulo/PycharmProjects/VoiceRecognition/Data/Simulacao/logfn.log

[Commands]
ROTATE RIGHT = r -0.01
ROTATE LEFT = r 0.01
MOVE FORWARD = f 0.7
STOP MOVING = 0
SAVE POINT = save point
GO TO LOCATION = go to point
```

Servidor - Simulação

- Dicionário da simulação

```
FORWARD F AO R W ER D
GO      G OW
LEFT    L EH F T
LISTENING       L IH S AH N IH NG
LISTENING(2)    L IH S N IH NG
LOCATION        L OW K EY SH AH N
MOVE    M UW V
MOVING  M UW V IH NG
POINT   P OY N T
RIGHT   R AY T
ROTATE  R OW T EY T
SAVE    S EY V
STOP    S T AA P
TO      T UW
TO(2)   T IH
TO(3)   T AH
```

Servidor - Simulação

- Keyword List da simulação

```
ROTATE RIGHT/1e-40/
ROTATE LEFT/1e-40/
MOVE FORWARD/1e-50/
STOP MOVING/1e-10/
STOP LISTENING/1e-20/
SAVE POINT/1e-20/
GO TO LOCATION/1e-25/
```

Servidor - Simulação

- Os thresholds da keyword list foram definidos utilizando o comando:
- pocketsphinx_continuous -infile <your_file.wav> -keyphrase <your keyphrase> -kws_threshold <your_threshold> -time yes

Servidor - Simulação

- Foi desenvolvida em Processing
- É constituída por 3 classes, Servidor, Ball e Simulação.

Servidor - Robot

- É um servidor que recebe mensagens do tipo "vel_linear vel_angular" e de seguida controla o Robot através de ROS (Robot Operating System).

Servidor - Robot

- Comandos
  - Move Forward - Esta mensagem ordena que o robot se mova em frente e envia a mensagem ”0.1 0”
  - Rotate Right - Ordena que o robot rode sobre si no sentido negativo, a mensageḿ e ”0 -0.7”
  - Rotate Left - Ordena que o robot rode sobre si no sentido positivo, a mensageḿ e ”0 0.7”

Servidor - Robot

- Comandos
  - Stop Moving - Ao receber esta mensagem o Robot para completamente qualquer tipo de movimento, a mensagem é ”0 0”.
  - Walk Left - Este comando faz com que o robot se mova em frente e rode sobre si ao mesmo tempo, a mensagem é ”0.1 0.7”.
  - Walk Right - Tal como o comando anterior, este faz o mesmo, porém o robot roda sobre si noutro sentido, logo a mensagem é ”0.1".

Servidor - Robot

- Ficheiro de configuração do robot

```
[Files]
Dictionary: /home/paulo/PycharmProjects/VoiceRecognition/Data/Robot/Robot.dic
Kws: /home/paulo/PycharmProjects/VoiceRecognition/Data/Robot/Robot_Keyphrase.list
Logfn: /home/paulo/PycharmProjects/VoiceRecognition/Data/Robot/logfn.log

[Commands]
ROTATE RIGHT = 0 -0.7
ROTATE LEFT = 0 0.7
MOVE FORWARD = 0.1 0
STOP MOVING = 0 0
WALK LEFT = 0.1 0.7
WALK RIGHT = 0.1 -0.7
```

Servidor - Robot

- Dicionário do robot

```
AND AH N D
AND(2)  AE N D
FORWARD F AO R W ER D
LEFT    L EH F T
LISTENING   L IH S AH N IH NG
LISTENING(2)    L IH S N IH NG
MOVE    M UW V
MOVING  M UW V IH NG
RIGHT   R AY T
ROTATE  R OW T EY T
STOP    S T AA P
WALK    W AO K
WALK(2) W AA K
```

Servidor - Robot

- Keyword List do Robot

```
ROTATE RIGHT/1e-40/
ROTATE LEFT/1e-40/
MOVE FORWARD/1e-50/
STOP MOVING/1e-10/
STOP LISTENING/1e-20/
WALK LEFT/1e-15/
WALK RIGHT/1e-15/
```

DEMONSTRAÇÕES