Dissertação

Mestrado em Engenharia Informática, Computação Móvel

Conversação Homem-máquina. Caracterização e Avaliação do Estado Actual das Soluções de Speech Recognition, Speech Synthesis e Sistemas de conversação Homem-máquina

João Pedro Cordeiro Rato

Leiria, *Setembro* de 2016

Dissertação

Mestrado em Engenharia Informática, Computação Móvel

Conversação Homem-máquina. Caracterização e Avaliação do Estado Actual das Soluções de Speech Recognition, Speech Synthesis e Sistemas de conversação Homem-máquina

João Pedro Cordeiro Rato

Dissertação de Mestrado realizada sob a orientação do Doutor Nuno Costa, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, *Setembro* de 2016

Agradecimentos

Com o término desta dissertação, quero expressar o meu profundo agradecimento a todos os professores e à escola, que me ajudaram a aumentar o nível de conhecimento e experiência ao longo de todo o mestrado de engenharia informática - Computação Móvel, com especial relevo para o professor Doutor Nuno Costa por toda a disponibilidade e apoio prestado ao longo do processo de desenvolvimento deste trabalho.

Agradeço aos meus pais que, como sempre, me apoiaram ao longo deste grande desafio. Foram e são sempre um grande suporte em todos os momentos difíceis.

À minha filha Camila que, mesmo em fase de gestação, me acompanhou durante todo este trabalho na barriga da mãe.

À minha namorada Marisa o meu agradecimento pelo apoio, compreensão e paciência, sobretudo nesta bonita fase da nossa vida.

Aos meus amigos, um enorme obrigado pela presença e apoio nos momentos mais complicados ao longo deste desafio.

A todos, muito obrigado.

Resumo

A comunicação verbal humana é realizada em dois sentidos, existindo uma compreensão de ambas as partes que resulta em determinadas considerações. Este tipo de comunicação, também chamada de diálogo, para além de agentes humanos pode ser constituído por agentes humanos e máquinas. A interação entre o Homem e máquinas, através de linguagem natural, desempenha um papel importante na melhoria da comunicação entre ambos.

Com o objetivo de perceber melhor a comunicação entre Homem e máquina este documento apresenta vários conhecimentos sobre sistemas de conversação Homem-máquina, entre os quais, os seus módulos e funcionamento, estratégias de diálogo e desafios a ter em conta na sua implementação.

Para além disso, são ainda apresentados vários sistemas de *Speech Recognition*, *Speech Synthesis* e sistemas que usam conversação Homem-máquina.

Por último são feitos testes de performance sobre alguns sistemas de *Speech Recognition* e de forma a colocar em prática alguns conceitos apresentados neste trabalho, é apresentado a implementação de um sistema de conversação Homem-máquina.

Sobre este trabalho várias ilações foram obtidas, entre as quais, a alta complexidade dos sistemas de conversação Homem-máquina, a baixa performance no reconhecimento de voz em ambientes com ruído e as barreiras que se podem encontrar na implementação destes sistemas.

Palavras-chave: Speech Recognition, Text To Speech, Automatic Speech Recognition, Spoken Dialogue Systems, Conversação Homem-máquina, Word Error Rate.

Abstract

The human verbal communication is determinated in two ways, existing an understanding between both parties, that results in certain considerations. This type of communication, also known as dialogue, beyond, human agents resources, could also be constituted by human agents and machines. The interaction between Human and machines through natural language, has an important role in the improvement of their communications.

With the target to understand better the communication between Human and machines this document represents various knowledges of the Human-machine conversion, including modules and behaviour, dialogue strategies and challenges to take into account in its implementation.

Furthermore there is also featured several speech recognition, speech synthesis and systems that use Human-machine conversion.

Lastly there are performance tests about some systems of speech recognition and the way to put in practice some of the concepts presented in this work, it's shown a system of implementation of a Human-machine conversion. About this work a lot of conclusion were gained including the high complexity of the Human-machine conversion systems, the low performance of voice recognition in noisey environments and the barriers you could find in the implementation of these systems.

Key-Words: *Speech Recognition, Text To Speech, Automatic Speech Recognition, Spoken Dialogue Systems, Man Machine Conversation, Word Error Rate*.

Índice de Figuras

Índice de Quadros

Lista de Siglas

AMI	Amazon Machine Image
ANN	Artificial Neural Networks
API	Application Programming Interface
ASK	Alexa Skills Kit
ASR	Automatic Speech Recognition
AVI	Audio Video Interleave
AVS	Alexa Voice Service
BBM	BlackBerry Messenger
BSD	Berkeley Software Distribution
CALO	Cognitive Assistant that Learns and Organizes
CSR	Command Success Rate
DAE	Deep Auto Encoders
DARPA	Defense Advanced Research Projects Agency
DBN	Deep Belief Networks
DL	Deep Learning
DM	Dialogue Management
DNN	Deep Neural Networks
GMM	Gaussian Mixture Models
GPS	Global Positioning System
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
HNM	Harmonic Plus Noise Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IBM	International Business Machines

IDE	Integrated Development Environment
IoT	Internet of Things
IPA	Intelligent Personal Assistant
ISCA	International Speech Communication Association
JSGF	Java Speech Grammar Format
JSON	JavaScript Object Notation
LG	Language Generation
LP	Linear Prediction
LU	Language Understanding
LVCSR	Large Vocabulary Continuous Speech Recognition
MB	Megabyte
MDN	Mozilla Developer Network
MDP	Markov Decision Processes
MFCC	Mel Frequency Cepstral Coefficients
ML	Machine Learning
MLP	Multilayer Perceptron
MP3	MPEG-1/2 Audio Layer 3
MP4	MPEG-4 Part 14
NN	Neural Networks
PAL	Personalized Assistant that Learns
PC	Personal Computer
PDA	Personal Digital Assistants
PDF	Portable Document Format
PLS	Pronunciation Lexicon Specification
POMDP	Partially Observable Markov Decision Process
REST	Representational State Transfer
RL	Reinforcement Learning
RTP	Rádio Televisão Portuguesa
SaaS	Software as a service
SDK	Software Development Kit
SDS	Spoken Dialogue Systems
Siri	Speech Interpretation and Recognition Interface
SLG	Spoken Language Generation

SLU	Spoken Language Understanding
SOAP	Simple Object Access Protocol
SR	Speech Recognition
SRE	Speech Recognition Engine
SRGS	Speech Recognition Grammar Specification
SRS	Speech Recognition System
SSML	Speech Synthesis Markup Language
SWER	Single Word Error Rate
TAP	Transportes Aéreos Portugueses
TTS	Text To Speech
W3C	World Wide Web Consortium
WAV	WAVEform audio format
WAVE	WAVEform audio format
WCF	Windows Communication Foundation
WER	Word Error Rate
WRR	Word Recognition Error
WSDL	Web Service Definition Language
WWW	World Wide Web
XML	eXtensible Markup Language

Índice

Introdução

Neste capítulo é apresentado o enquadramento e introdução ao tema da dissertação, os objetivos propostos e a organização do documento.

1.1 Enquadramento do tema

Durante décadas um dos desafios que intrigou a ciência e a engenharia foi a busca de formas que permitissem ao Homem interagir com máquinas de forma natural. Até recentemente, algumas das interfaces com mais sucesso e mais usadas pelo Homem para interagir com máquinas, foram o teclado e o rato [1].

A interação entre o Homem e a máquina ocorre de diversas formas, podendo ser feito, por exemplo, através de teclados, ratos, ecrãs [2] e manípulos. De forma a facilitar esta interação, o uso da fala tem vindo a ser cada vez mais adotado, sendo a linguagem um processo natural de comunicação, este tipo de interação revela-se bastante atrativo [3].

O interesse por este tipo de interfaces não é de agora e já tem alguns anos de pesquisa. O seu estudo tem sido facilitado à medida que as tecnologias vêm evoluindo [4]. A cada dia que passa a tecnologia faz cada vez mais parte das nossas vidas. O seu custo é cada vez menor e a sua performance cada vez maior [3], tornando-se cada vez mais acessível à maioria das pessoas.

Esta interação por voz é possível graças a sistemas de reconhecimento de voz (*Speech Recognition System, SRS*), sintetizadores de voz (*Text To Speech, TTS*) e sistemas conversação Homem-máquina (*Spoken Dialogue Systems, SDS)*, entre outros. Estes sistemas vêm ajudar os Humanos em vários cenários e essas ajudas resultam em melhorias da nossa qualidade de vida. À medida que a tecnologia vai avançando, este tipo de interfaces vão aparecendo com mais frequência nas nossas rotinas diárias. São introduzidas de forma tão natural que, muitas das vezes, já as achamos tão normais que estranhamos quando, por alguma razão, não estão disponíveis para nos servirem.

Um sistema de conversação por voz Homem-máquina é uma interface entre o utilizador e uma aplicação instalada numa máquina que permite uma interação falada de uma forma

relativamente natural [5]. Este sistema permite aos utilizadores interagirem com aplicações computacionais usando uma linguagem natural falada.

A abrangência destes sistemas tem vindo a aumentar e são usados em diferentes áreas, tanto a nível profissional como pessoal. Estas interfaces são sem dúvida importantes para pessoas que tenham problemas físicos e limitações ao nível de coordenação motora. Paralisias, membros amputados, cegueira, deficiências genéticas entre outros motivos, são alguns exemplos de situações onde este tipo de interfaces pode ajudar e dar mais autonomia e qualidade de vida [1].

Em relação à sua complexidade, ela pode variar entre sistemas de pergunta-resposta, em que é respondida uma questão de cada vez, e sistemas que suportam uma conversa estendida com o utilizador [5], como se fossem duas pessoas a conversar. O modo de comunicação pode variar de uma linguagem natural minimalista, constituída por palavras como "sim" ou "não" e números 0 a 9, a uma linguagem mais completa sem restrição de palavras [5].

O desenvolvimento de *software* para um sistema de conversação Homem-máquina não é fácil, pois exige vários tipos de conhecimento, como por exemplo, *APIs* para o tratamento da fala, gestão do diálogo, gramática do diálogo recebido, semântica, aplicações *web*, etc [6].

Para além da comunicação por voz, existem outras formas de comunicação entre o Homem e a máquina, tais como, movimentos de membros do corpo humano, como por exemplo, acenar as mãos, expressões faciais e movimentos dos olhos. Têm sido desenvolvidos também métodos de comunicação com base em sinais eletrobiológicos, como a leitura de sinais cerebrais, sinais musculares e sinais da retina [1]. A máquina pode responder também por voz, imagens, toques, entre outras formas. Todas estas formas de comunicação formam um sistema do tipo multímodo [1].

1.2 Introdução ao tema

Para que seja possível haver uma comunicação verbal entre Homem e máquina, muita investigação tem sido realizada e vários modelos de processamento de voz e linguagem foram criados.

Basicamente, um sistema de comunicação por voz Homem-máquina necessita de receber a voz humana, interpretá-la e devolver uma resposta ou executar uma tarefa. Todavia, as dificuldades são várias, tais como, a utilização em ambientes com ruído e/ou o treino dos sistemas.

Como os utilizadores destes sistemas são, no fundo, os humanos, convém ter em atenção as suas necessidades para que o sistema seja bem-sucedido.

Um dos sistemas mais conhecidos, mais completos e complexos de interação são os *Spoken Dialogue Systems (SDS)*. É muito usado como interface por voz em *smartphones*, devido à evolução de sistemas como o *Siri* da *Apple* ou o *Dragon Go* da *Nuance*, entre outros, que demonstraram o potencial da integração da conversação por voz natural em dispositivos móveis, aplicações e serviços [7].

Contudo e apesar do progresso constante ao longo das últimas décadas na tecnologia de reconhecimento de voz, o processo de conversão da fala em palavras ainda incorre em taxas de erro entre 15% a 30% em muitos ambientes operacionais do mundo real, como espaços públicos ou automóveis [2].

1.3 Objetivos

Pretende-se com este trabalho, de uma forma geral, caracterizar um conjunto variado de sistemas de conversação Homem-máquina e posterior avaliação. Desta forma, este trabalho pretende explicar a estrutura genérica de um sistema de conversação Homem-máquina, juntamente com vários conceitos, dificuldades e metodologias.

Para além disso, pretende-se também expor vários sistemas de reconhecimento de voz, de síntese e sistemas que usam conversação Homem-máquina, com a discrição de algumas das suas características.

Pretende-se ainda apresentar os resultados de testes efetuados sobre certos sistemas de reconhecimento de voz, para melhor perceber a sua performance.

Por último, implementar um pequeno sistema de comunicação Homem-máquina e registar toda a evolução e desafios.

1.4 Organização do documento

Este documento, para além do capítulo "Introdução", encontra-se organizado em mais seis capítulos que são apresentados de seguida:

- Capítulo "Estado da Arte" – Neste capítulo são apresentados vários módulos que compõem um sistema de comunicação Homem-máquina, as estratégias de diálogo do módulo *Dialogue Management* onde se incluem técnicas de controlo de diálogo e métodos de aprendizagem, os desafios que podem surgir com estes tipo de sistemas, sistemas de conversação Multímodo, diversos sistemas de reconhecimento e sintetizadores de voz e, por último, várias aplicações presentes no mercado de conversação Homem-máquina;
- Capítulo "Avaliação de performance de ASR com base no método WER" – É feita uma avaliação de performance em alguns sistemas de reconhecimento de voz, através de testes que permitem a medição de erros pelo método *Word Error Rate*;
- Capítulo "Implementação e teste de um sistema SDS" – Neste capítulo é apresentado e caraterizado o processo de implementação de um sistema de conversação Homem-máquina e são expostas várias dificuldades encontradas durante o processo;
- Capítulo "Discussão de Resultados" – É feita uma análise com base nos resultados obtidos;
- Capítulo "Conclusão" – Este capítulo apresenta as conclusões do trabalho efetuado, as principais contribuições e algumas propostas para trabalhos futuros.

Estado da Arte

Neste capítulo pretende-se apresentar um grande conjunto de informações para que se perceba o funcionamento de um sistema básico de conversação Homem-máquina, as estratégias de diálogo, os vários desafios que se podem enfrentar neste tipo de sistemas e o funcionamento de sistemas multímodo. São também apresentados, neste capítulo, vários sistemas de reconhecimento e sintetização de voz e diversas aplicações completas de conversação Homem-máquina.

2.1 Componentes básicos de um sistema de conversação por voz entre Homem-máquina

De forma a perceber o funcionamento de um sistema de conversação Homem-máquina, são apresentados, neste capítulo, os vários módulos que compõem um sistema básico.

Num sistema normal de comunicação por voz existem vários componentes, em que, juntos permitem a correta comunicação Homem-máquina. A imagem apresentada a baixo representa um desses sistemas que funciona da seguinte forma [2, 3, 8, 9]:

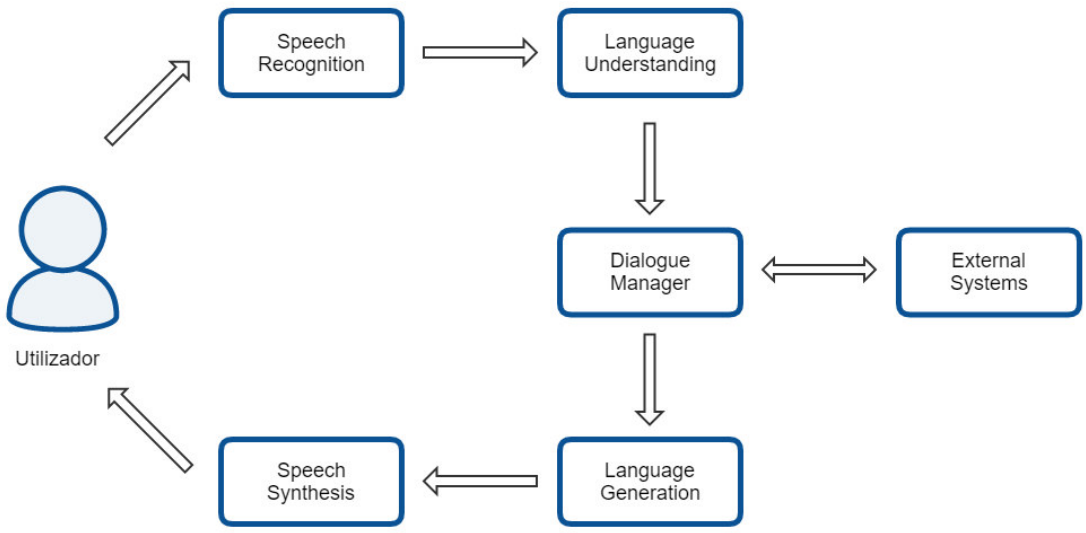

Figura 1. Sistema típico de conversação Homem-máquina.

O utilizador é a parte humana que interage com a máquina.

O *Speech Recognition (SR),* também conhecido por *Automatic Speech Recognition (ASR),* permite fazer a captação da voz do utilizador e transforma a voz em formato texto de forma a poder ser interpretada pela componente *Language Understanding (LU)*. O reconhecimento da fala pode ser feito pela seguinte forma: a fala é captada e de seguida é dividida em partes com cortes feitos nos intervalos de silêncio. Depois, é feito o reconhecimento do significado de cada parte obtida com base em várias combinações de palavras que são combinadas com o áudio de cada parte. A melhor combinação é a escolhida.

A componente *Dialogue Management (DM)* gere a interação entre o Homem e a máquina com base nos conteúdos recebidos e também, por vezes, com base em conteúdos armazenados em sistemas externos como bases de dados ou outro tipo de repositório. A partir desta componente, as respostas podem ser apresentadas por texto, gráficos, imagens, tabelas, entre outras formas [3, 10]. Nos casos da informação ser insuficiente ou confusa, o sistema pode consultar o utilizador para ser mais explícito e claro [3].

Por fim, o sistema gera a resposta pelo componente *Language Generation (LG)* e disponibiliza-a através do componente *Speech Synthesis*.

Para além da voz, a comunicação pode ser complementada com outras formas de interface. Para que a máquina possa perceber melhor as nossas intensões durante a comunicação, outras formas de interface para além da voz, como por exemplo, gestos, escrita, reconhecimento facial, imagens, entre outros, podem ser uma mais-valia [1, 3]. Para isso são necessários vários sensores e metodologias de tratamento de sinais.

De seguida são descritas, de forma mais detalhada as características de cada componente.

2.1.1. Utilizador

O utilizador representa a parte humana do processo de comunicação por voz. Ele pode ter a iniciativa do diálogo ou pode esperar pela iniciativa da máquina. Tem o papel de fornecer, pedir informações ou pedir a execução de tarefas ao sistema de conversação.

2.1.2. Speech Recognition

O módulo *Speech Recognition (SR)* ou *Automatic Speech Recognition (ASR)* [11] é responsável pela conversão da expressão oral que dá entrada no sistema, constituída por uma sequência de parâmetros acústicos e fonéticos, numa sequência de palavras [5].

O processo básico de reconhecimento de voz envolve a descoberta de uma sequência de

palavras ou outras entidades linguísticas correspondentes [11], utilizando um conjunto de modelos adquiridos numa fase de treino prévio, combinando-os com o sinal de voz de entrada do discurso do utilizador. Os modelos podem ser modelos de palavras, no caso de sistemas com pouco vocabulário. Mas o mais comum é os modelos serem constituídos por unidades de som como fonemas, que modelam um som, bem como o seu contexto em termos dos sons anteriores e posteriores [5].

A interpretação da voz captada por parte da máquina pode ser uma tarefa complicada devido a alguns aspetos que a voz possa conter [12]. A origem de alguns problemas pode estar relacionada com a presença de interjeições, pausas e palavras incompletas. Outro problema pode ser originado pela tentativa de interpretação de palavras que não façam parte do sistema. O reconhecimento da fala é muita das vezes usado em sistemas acessíveis por telefone e a qualidade da fala transmitida por telefone é muita das vezes mais difícil de reconhecer devido a ruídos de linha, falhas de som e de sinal. Para que estes tipos de dificuldades sejam ultrapassados, várias melhorias têm sido desenvolvidas [3]. Foram introduzidos modelos acústicos para o tratamento de pausas, para a deteção de palavras incompletas ou desconhecidas. Alguns modelos permitem aprender as novas palavras uma vez que forem detetadas.

As aplicações que usam *Speech Recognition* podem ser usadas em sistemas de marcação de chamadas telefónicas, encaminhamento de chamadas, respostas por voz interativa, entrada de voz e ditados, criação de documento estruturados, controlo de aparelhos por voz, aprendizagem de línguas por computador, busca de conteúdos e robótica [11]. Contudo, é necessário que este módulo seja robusto e que esteja preparado para os vários ruídos existentes e distorções acústicas [11].

A tarefa do componente *Speech Recognition* num sistema de conversação Homem-máquina, como já foi referido, é converter a fala do discurso emitido pelo utilizador numa sequência de palavras. Um dos maiores obstáculos é o grau de variação do sinal da fala [5]. Esta variação aumenta com os seguintes fatores:

- *Linguistic Variability* – São efeitos sobre o sinal de voz causados por vários fenómenos linguísticos. Como por exemplo, o mesmo fonema pode ter diferentes compreensões acústicas em diferentes contextos, determinados pelos fonemas anteriores e seguintes do som em questão;
- *Speaker Variability* – São as diferenças entre oradores, relativas a fatores físicos tais como, a forma do trato vocal, a idade do orador, género, origem regional e variações com o próprio orador, porque as mesmas palavras faladas em diferentes ocasiões pelo

mesmo orador podem ter diferenças em termo de propriedades acústicas. Fatores físicos como cansaço, nariz entupido devido a uma constipação, mudanças de humor, a localização de uma palavra dentro de uma frase e do grau de ênfase que é dada à palavra, têm influência sobre a forma como as palavras são pronunciadas;

- *Channel Variability* – São os efeitos do ruído de fundo, que pode ser constante ou transitório, e do canal de transmissão, tais como a rede telefónica ou um microfone.

O componente *Speech Recognition* de um sistema típico de conversação por voz tem de ser capaz de lidar com os seguintes fatores adicionais [5]:

- *Speaker Independence* – Como a aplicação será normalmente usada para uma ampla variedade de utilizadores casuais, o reconhecedor de voz do sistema não pode ser treinado com um só orador ou alguns oradores. Tem de ser treinado com vários oradores em que os respetivos padrões de fala sejam representativos dos potenciais utilizadores do sistema;
- *Vocabulary Size* – O tamanho do vocabulário varia com a aplicação e com a conceção particular do sistema de diálogo. Assim, um diálogo cuidadosamente controlado pode restringir o utilizador a um vocabulário limitado a algumas palavras que expressam as opções que estão disponíveis no sistema, enquanto num sistema mais flexível do vocabulário pode aceitar muitas palavras;
- *Continuous Speech* – Os utilizadores de sistemas de conversação por voz esperam ser capazes de falar normalmente com o sistema. No entanto, é difícil de determinar um limite de palavras durante o decorrer da fala, uma vez que não existe uma separação física no tempo contínuo do sinal de voz;
- *Spontaneous Conversational Speech* – Uma vez que a fala do utilizador é a entrada para o sistema de conversação por voz, ela, normalmente, é espontânea e não planeada. O que significa que o discurso pode conter difluências, como hesitações e enchimentos, como por exemplo “hummm” ou “ahhh”, falsos inícios do discurso em que o orador começa o discurso e em seguida faz pausas e depois começa de novo, e fenômenos linguísticos naturais, tais como tosse. O *Speech Recogniser* tem de ser capaz de extrair do discurso uma sequência de palavras para que a partir das quais, o seu significado possa ser tratado.

A capacidade de robustez contra os ruídos é um foco importante dos sistemas de *Speech Recognition*, porque cada vez mais eles trabalham em ambientes muito difíceis ao nível de

ruído [11]. No entanto existem formas de analisar e caracterizar a robustez contra o ruído [11]:

- Compensação *Feature-Domain vs. Model-Domain*;
- Compensação com modelagem de distorção implícita *versus* explícita;
- Compensação usando conhecimento prévio sobre a distorção acústica;
- Compensação com uma *deterministic strategy versus* processamento de incerteza;
- Modelo de treino em conjunto *versus* modelo de treino em separado.

No processo de reconhecimento da fala, o sistema WAXHOLM [10] utiliza redes neurais como ferramenta para combinar as abordagens *Artificial Neural Networks (ANN)* e *Speech Production Oriented Approach*. A abordagem *Speech Production Approach* gera protótipos espectrais de palavras a partir de um determinado vocabulário. Neste sistema os modelos de todas as palavras são usados para descrever os fonemas internos da palavra, enquanto três grupos de fonemas são usados para modelar os limites das palavras. Tem ainda um método dinâmico que se adapta à fonte de voz. A abordagem *Artificial Neural Networks* utilizada em WAXHOLM [10], consistiu na utilização de redes auto-organizadas e redes que utilizam técnicas de propagação de erros. A utilização de redes de propagação de erros melhora substancialmente o desempenho. As redes de auto-organização aprendem mais rápido que as redes de propagação de erros, mas não são tão facilmente transformadas em estruturas recorrentes.

Com o aumento da informação e do poder computacional, o *Speech Recognition* foi crescendo e novos sistemas foram aparecendo. Por exemplo, pesquisas por voz e interação com dispositivos móveis, como o *Siri* no *iPhone*, *Bing Voice Search* no *Windows Phone*, e *Google Now* no *Android*, controlo por voz em sistemas de entretenimento doméstico como o *Kinect* da *xBox*, entre outros sistemas [11]. Mas estes sistemas continuam a ter problemas por resolver nos dias de hoje, mesmo que eles estejam presentes em muitas soluções comerciais e sejam vistos como solucionados [13].

Alguns sistemas de reconhecimento da voz, para interpretarem a voz recebida, usam algoritmos de restrições linguísticas como é o caso do algoritmo *n-Gram* [3]. Este algoritmo permite reduzir o espaço de pesquisa à medida que vai interpretando o significado de cada palavra, permitindo um aumento de performance na interpretação da voz, sendo mais eficiente quando as mensagens têm mais palavras do que poucas palavras [14]. No entanto não abordam o problema da compreensão da voz [3].

O reconhecimento de voz pode ser abordado de forma semântica e sintática. A abordagem

semântica tenta decifrar o significado das palavras. Contudo, esta abordagem pode ter problemas na interpretação adequada de construções linguísticas complexas, mas a necessidade de se adaptar a conteúdos espontâneos de fala tem anulado esses potenciais problemas. A abordagem sintática consiste em recolher partes de frases, para que depois sejam juntas e se possa obter uma análise completa do significado. O analisador inclui um método probabilístico para a análise de fragmentos quando a análise linguística total não é alcançável [3].

Uma das estratégias mais usadas para os componentes de reconhecimento de voz interagirem com linguagens naturais é a interface *N-best* [3, 10], que representa uma lista de hipóteses, juntamente com dezenas de possibilidades associadas e onde o *N* representa o número de hipóteses [8]. Neste tipo de interfaces, o sistema de reconhecimento propõem as suas melhores *N* frases, que depois são analisadas uma de cada vez pelo componente de *Language Understanding* até que seja analisado com sucesso a primeira frase. O componente de *Language Understanding* funciona como um filtro sobre todo o conjunto de hipóteses. Uma alternativa ao *N-best* consiste em representar as hipóteses recolhidas pelo sistema de reconhecimento sobre a forma de um gráfico de palavras. Esta representação é mais compacta que a lista usada na *N-best* e permite uma pesquisa mais completa. Muitas das hipóteses presentes numa lista *N-best* podem variar devido à informação acústica não ser muito robusta. Existem palavras acusticamente idênticas como por exemplo "*an*" e "*and*" e por vezes podem levar a que toda a frase seja eliminada por razões linguísticas, mesmo antes de chegar ao fim da análise sintática e semântica. Uma solução para o referido problema consiste em que os componentes, *Speech Recognition* e *Language Understanding*, trabalhem juntos para que as hipóteses com mais probabilidade acústicas de sucesso possam ser selecionadas [14].

2.1.3. Language Understanding

Também conhecido como *Spoken Language Understanding (SLU)*, o papel deste módulo é analisar uma sequência de palavras com o objetivo de produzir uma representação semântica do significado da expressão oral reconhecida pelo componente de reconhecimento de voz, de forma a ser usado na componente *Dialogue Management* [5].

A compreensão da linguagem envolve a análise sintática, para determinar a estrutura constitutiva da cadeia reconhecida composta por um grupo de palavras juntas, e análise semântica, para determinar os significados dos elementos. Outras abordagens para a compreensão da linguagem podem envolver pouco ou nenhum processamento sintático e gerar uma representação semântica diretamente a partir da cadeia reconhecida [5].

Os fundamentos teóricos para processamento de linguagem podem ser encontrados na psicologia linguística e linguística computacional. Formalismos gramaticais vigentes em linguística computacional partilham uma série de características chave, das quais o principal ingrediente é uma descrição baseada em funcionalidades de unidades gramaticais, tais como, palavras e frases. Estes formalismos baseados em recurso são semelhantes aos usados em pesquisas de representação do conhecimento e teoria do tipo de dados [5]. Termos de recursos são conjuntos de pares atributo-valor, em que os valores podem ser *atomic symbols* ou então termos de recursos, que podem ser organizadas numa hierarquia de tipo ou como termos disjuntivos, limitações funcionais, ou conjuntos [5].

Para um sistema de diálogo, o significado exato das palavras não é importante. O que realmente importa é o que o utilizador estava a tentar alcançar através das palavras [8]. Por exemplo o utilizador pode dizer: "*I'd like the phone number*" ou "*What is the phone number?*". O resultado desejado é o mesmo, o utilizador está a pedir o número de telefone. Esta distinção entre a semântica exata de um enunciado pode ser caracterizada por *speech act* ou *dialogue act* que, no fundo, é uma representação do discurso subjacente que identifica a função de um enunciado como um pedido ou afirmação [2]. No exemplo anterior o *speech act* para ambos os discursos é "*request*".

2.1.4. Dialogue Management

O componente *Dialogue Management (DM)* de um sistema de conversação gere a interação entre o utilizador e a máquina. Ele planeia e resolve problemas das iterações, coordena a interação com outros componentes do sistema [5], gera as respostas ao utilizador para serem apresentadas de forma verbal mas também se pode apresentar noutras formas como tabelas de dados gráficos, entre outras [3].

O *Dialogue Management* gere todos os aspetos do diálogo. Ele necessita de uma representação semântica do texto do utilizador, descobre como se encaixa no contexto global e cria uma representação semântica da resposta do sistema. Durante todo este processo ele pode executar outras tarefas como manter um histórico do diálogo, adotar certas estratégias de diálogo, lidar com texto mal formado e não reconhecido, recuperar o conteúdo armazenado em bases de dados ou outros sistemas externos, decidir a melhor resposta para o utilizador, gerir a iniciativa e a resposta, resolver questões de pragmatismo e fazer análise do discurso.

Este componente pode ser capaz de clarificar informação ambígua [3], por exemplo "*Did you say Boston or Austin?*" ou informação incompleta, por exemplo "*On what day would you like to travel?*". Este componente deve ser capaz de ajudar o utilizador em situações em que as

consultas à base de dados devolvem mais que uma resposta, fazendo perguntas que ajudam a filtrar as respostas obtidas, por exemplo "*I found ten flights, do you have a preferred airline or connecting city?*". Este componente deve também sugerir alternativas para o caso das pesquisas não devolverem a informação pretendida [3], por exemplo "*don't have sunrise information for Oakland, but in San Francisco…*". Deve também sugerir novas consultas, por exemplo "*Would you like me to price your itinerary?*". No início do diálogo, um dos papéis deste componente é recolher informações do utilizador.

O *Dialogue Management* pode ser liderado pelo sistema, liderado pelo utilizador ou por iniciativa mista [5]. Se o controlo do diálogo for liderado pelo sistema, o sistema faz uma ou várias questões para obter os parâmetros necessários à tarefa do utilizador. Se o sistema for liderado pelo utilizador, o utilizador faz as perguntas ao sistema, com o objetivo de obter informação. Se o sistema for de iniciativa mista, o controlo do diálogo é partilhado pelo sistema e pelo utilizador.

Escolher a melhor ação requer mais conhecimento do que simplesmente a última observação do *input* recebido. O histórico completo do diálogo e o contexto também desempenham um papel importante. Tendo isso em consideração, o *DM* tenta manter uma representação integral de todas as sequências observadas, ao qual é chamado de *dialogue state*, *system state* ou *belief state* [8]. O *belief state* atual depende de uma função de transição de *belief state* que recolhe um determinado *belief state* e atualiza-o para cada nova ação e observação do sistema. O componente DM define as políticas de diálogo que determinam o que o sistema deve fazer em cada *dialogue state* [8].

O *DM* pode recorrer a várias fontes de conhecimento. Estas fontes de conhecimento são por vezes chamadas de *Dialogue Model* [5]. Um *Dialogue Model* pode incluir os seguintes tipos de fontes de conhecimento:

- *Dialogue History* - São os registos do diálogo até ao momento. Este histórico fornece uma base para a coerência conceptual;
- *Task Record* - Representa a informação a recolher no âmbito do diálogo. Pode ser visto como um modelo ou *template* e é usado para determinar que informação ainda não foi recolhida durante o diálogo. Este registo pode também ser usado para memórias de tarefa, para casos em que os utilizadores queiram alterar os valores de alguns parâmetros;
- *World Knowledge Model* – Este modelo contém, geralmente, informação de senso comum, por exemplo a data do dia de natal (25 de Dezembro);

- *Domain Model* – Modelo com informação específica sobre uma área, por exemplo informação sobre voos;
- *Generic Model of Conversational Competence* – Este modelo inclui conhecimento sobre os princípios da comunicação, regras do discurso e controlo de vez (*turn-taking*) da comunicação;
- *User Model* – Este modelo pode conter informação sobre o utilizador que pode ser relevante para o diálogo. Como por exemplo as preferências, objetivos e intenções do utilizador.

Estas fontes de conhecimento são usadas em diferentes formas, de acordo com a estratégia de controlo de diálogo escolhida. Segundo Michael F Mctear [5] o controlo do diálogo nos sistemas de conversação por voz entre Homem-máquina pode ser classificado em três métodos, *State-based* systems, *Frame-based* systems e *Agent-based* systems e permitem conhecer o contexto do diálogo, ou seja, as abordagens podem permitir tomar uma decisão clara sobre as intenções e crenças do utilizador [2].

Num sistema com controlo de diálogo *State-based* o controlo é feito pelo sistema e todas as perguntas que o sistema faz, estão pré-configuradas.

Se o sistema usar um controlo de diálogo *Frame-based*, o controlo do diálogo é principalmente feito pelo sistema, contudo permite de forma limitada, algumas iniciativas do utilizador. Permitem um maior grau de flexibilidade, uma vez que, o utilizador pode fornecer mais informação do que o sistema requer.

Num controlo de diálogo *Agent-based* ou *Plan-based* [2], o controlo do diálogo tanto pode ser feito pelo utilizador como pelo sistema. Este controlo contém um grande número de abordagens que derivam de diferentes teorias de diálogo. Mais à frente neste trabalho, estas estratégias de diálogo serão apresentadas com mais detalhe.

Por exemplo, no diálogo baseado no sistema *State-based*, se estas fontes existirem, estão representados implicitamente no sistema. A informação e a sequência em que é adquirida é pré-configurada [5].

A função principal do componente de gestão de diálogo é de controlar o fluxo de diálogo. Isso envolve as seguintes tarefas [5]:

- Determinar se existe informação suficiente extraída ao utilizador, a fim de permitir a comunicação com a aplicação externa;
- Comunicar com aplicação externa;
- Comunicar informações de volta para o utilizador.

Numa arquitetura simples, estas tarefas podem ser um conjunto de processos em série. Por exemplo, o sistema consegue perceber o que o utilizador pretende, pode consultar aplicações externas e devolver o resultado ao utilizador. Contudo, o processo de determinar o que o utilizador pretende é normalmente complicado.

A informação fornecida pelo utilizador pode não ser suficiente para que o sistema possa obter informação de uma aplicação externa [5], pelas seguintes razões:

- Os *inputs* do utilizador podem estar mal formatados devido a uma má interpretação dos componentes *Speech Recognition* e *Language Understanding*;
- Os *inputs* do utilizador podem estar incompletos e errados, o que resulta em pouca informação para consultar em aplicações externas.

A maneira mais simples de lidar com os *inputs* mal formados ou incompletos é simplesmente reportar o problema de volta ao utilizador para solicitar uma reformulação do *input*. No entanto, este método é inadequado, porque não distingue as diferentes formas em que a entrada está mal formada ou incompleta e não consegue auxiliar o utilizador na reformulação do *input* [5].

As fontes de conhecimento podem ser utilizadas para ajudar na interpretação dos *inputs* mal formados. Ou seja, permite que o sistema de diálogo compense os *inputs* mal formados ou incompletos sem ter que consultar o utilizador com pedidos de repetição e esclarecimentos [5]. No entanto, esta consulta às fontes de conhecimento podem não ser solução para todos os problemas. O sistema pode entender que fez uma má interpretação dos *inputs* fornecidos pelo utilizador. Dessa forma, é necessário fazer uma verificação. Esta técnica é muito utilizada entre pessoas de modo a confirmar que a informação foi percebida por ambos.

A confirmação de que o sistema entendeu a intenção do utilizador é ainda mais necessária em diálogos falados com computadores, dada a possibilidade de reconhecimento e compreensão de erros [5]. Por isso, existem várias maneiras diferentes de um gestor de diálogo poder verificar ou confirmar que as declarações do utilizador foram corretamente entendidas. Alguns exemplos são:

- *Explicit Verification* - Verificação explícita, assume a forma de uma pergunta que questiona explicitamente a confirmação da entrada. Isto pode ser acompanhado de um pedido de responder com sim ou não;
- *Implicit Verification* - O sistema incorpora na sua próxima pergunta uma repetição da sua compreensão do que o utilizador disse. O utilizador pode ainda corrigir o valor repetido, mas se o utilizador responder à pergunta sem corrigir o valor, em seguida,

esse valor foi implicitamente confirmado.

Existem várias abordagens para controlos de diálogo. Contudo, pode-se concluir que os sistemas que incorporam princípios de raciocínio e de cooperação sejam os mais apropriados, uma vez que se aproximam mais da capacidade humana de comunicação [5]. Para aplicações que envolvam tarefas que exijam mais cooperação, interação entre o utilizador e o sistema de conversação, os sistemas mais simples de diálogo podem não ser a melhor opção. Contudo, para aplicações mais simples que envolvam tarefas limitadas, um sistema de controlo de diálogo como o *State-based* é o suficiente [5].

Quando o discurso é descodificado pelo *DM*, o sistema deve encontrar uma resposta apropriada. Essa resposta escolhida pelo sistema é selecionada a partir de um conjunto de ações possíveis e é codificado como um ato de diálogo, chamado de *system action* ou *system act* [8].

O domínio e a classificação das intenções são uma tarefa crítica para muitos sistemas de comunicação Homem-máquina, pois permite para certos tipos de discurso, ser encaminhado para subsistemas particulares [15]. Por exemplo se um utilizador perguntar a um sistema *Spoken Dialogue Systems (SDS)* "*tell me about the weather*", o sistema deve classificar o domínio como "*weatherquery*", para que a pesquisa seja encaminhada para o subsistema correto que trate das *querys* desse tipo.

De forma geral, o objetivo deste componente é gerir a conversa entre o Homem e a máquina, para que o utilizador obtenha com o sucesso que pretende [3].

2.1.5. Language Generation

De forma a dar uma resposta ao utilizador, o sistema deve ter a capacidade de transmitir a informação por voz ao utilizador. Este processo é normalmente realizado em duas fases. Numa, a informação é convertida em frases completas e depois numa outra fase, as frases são transmitidas através de um sistema que converte o texto para voz [3].

O módulo *Language Generation (LG)* também conhecido por *Spoken Language Generation (SLG)* constrói a mensagem em formato texto que depois é enviada para o componente *Speech Synthesis*, para ser reproduzida ao utilizador.

A geração de linguagem de um sistema convencional normalmente produz a resposta numa frase de cada vez. Em alguns casos as respostas são geradas de forma concatenada baseadas em modelos que respeitam determinadas regras [3]. Noutras situações as respostas são geradas usando simples métodos como a inserção de dados pré configurados em vários *slots*

de um *template*. Outros métodos mais complexos para a geração de respostas podem ser ainda usados [5].

A geração de linguagem falada é importante porque fornece um conteúdo que pode ser representado verbalmente ao utilizador. A construção desse conteúdo pode envolver os seguintes aspetos: Que informação deve ser incluída; Como é que a informação deve ser estruturada; A forma da mensagem, como por exemplo, a escolha das palavras e estrutura sintática.

2.1.6. Speech Synthesis

Este módulo consiste na utilização de um sistema *Text To Speech (TTS)* para a tradução da mensagem construída pelo componente *Language Generation* em forma de voz. Em casos mais simples as vozes estão pré-gravadas e são depois reproduzidas [5].

Text To Speech pode ser visto como tendo duas fazes [5], são elas:

1. *Text Analysis* – envolve a análise do texto de entrada que resulta na representação linguística e numa normalização [2]. *Text Analysis* é muitas das vezes referido como *text-to-phoneme conversion*.
2. *Speech Generation* – O *Speech Generation* é onde o sinal de voz é gerado [2], produzindo voz sintética a partir da representação linguística. É muitas vezes referido como *phoneme to speech conversion*. Esta fase envolve a geração da prenúncia, ritmo da fala e gerar o discurso em formato voz.

A fase *Text Analysis* pode ser constituída por quatro tarefas [5]:

1. *Text Segmentation and Normalisation* - Segmentação do texto e normalização, a análise morfológica trata da separação do texto em unidades mais pequenas, como parágrafos e frases. A normalização envolve a interpretação de abreviaturas e outras formas padrão, tais como datas, horas e moedas, e a sua conversão numa forma que possa ser falada;
2. *Morphological Analysis* - Análise morfológica, serve por um lado para lidar com o problema do armazenamento de pronúncias de um grande número de palavras que são morfologicamente variáveis umas das outras, e por outro lado, para ajudar com a pronúncia;
3. *Syntactic Tagging and Parsing* - Marcação e análise sintática, a marcação é necessária para determinar as partes do discurso das palavras no texto e para permitir uma análise sintática limitada;
4. *The Modelling of Continuous Speech Effects* - A modelagem dos efeitos contínuos da voz

está preocupada com que o discurso seja o mais natural em situações em que as palavras são faladas em uma sequência contínua;

Durante o passo da normalização, o texto em formato bruto contém números e abreviações e outros símbolos que são substituídos pelo texto correspondente [2].

Existem diferentes esquemas de síntese, tais como: *Concatenation Synthesis; Formant Synthesis*; *Articulatory Synthesis*; e *Statistical Parametric Synthesis*. Esta última utiliza modelos de sintetização baseados em *Hidden Markov Model (HMM)* e mais recentemente baseados em *Gaussian Mixture Models (GMM)* [2].

O esquema *Concatenation Synthesis* concatena unidades de discurso pré-gravado que estão armazenados num sistema externo como numa base de dados.

Os primeiros módulos *TTS* transformavam o texto em voz com a ajuda de dicionários que continham regras de pronunciação. Contudo, existiam alguns problemas de ambiguidades relativos à transcrição *TTS*, devido à falta de informação semântica e sintática [4].

Na geração da voz sintética é necessário ter em atenção à prosódia. A Prosódia está relacionada com a construção das frases, tons, intensidade, tempos e ritmo, e é utilizada para transmitir diferenças de significado, bem como para transmitir atitudes [5].

Waxholm [10] considera que os módulos de reconhecimento da fala e de reprodução de fala tem as mesmas necessidades de informação semântica, sintática e pragmática, e por isso, a informação lexical deve ser compartilhada.

2.1.7. External System

Geralmente os sistemas de conversação Homem-máquina necessitam de aceder a fontes externas de conhecimento, de forma a obter informação necessária e solicitada durante o curso do diálogo. Esse acesso a sistemas externos é gerido pelo componente *Dialogue Management* e pode ser dos seguintes tipos [5]:

- *Communicating with a database* - As bases de dados contém toda a informação relativa ao âmbito do sistema. Contêm um conjunto de dados que são disponibilizados conforme as consultas feitas de forma a satisfazer as necessidades da comunicação. O uso de *databases* tem uma utilização ampla nos sistemas de reconhecimento de fala assim como nos sistemas sintetizadores de voz;
- *Communicating with a knowledge base* - A comunicação com uma base de conhecimento é necessário para sistemas que suportam a resolução de problemas, em

vez de recuperação de informação;

- *Communication with a planning system* - A solução de problemas também pode ser alcançado através da utilização de um sistema de planeamento que suporta o raciocínio sobre metas, planos e ações.

Obter as informações necessárias a partir da fonte externa não é necessariamente uma tarefa simples. Complexas interações podem ser necessárias envolvendo mediações entre o gestor de diálogo e o utilizador.

Neste subcapítulo foram descritos os componentes básicos de um sistema de conversação por voz entre Homem-máquina. Estes componentes, que representam várias tecnologias diferentes, devem funcionar de forma integrada. Um dos grandes desafios destes sistemas é integrar todos os componentes de forma a obter um sistema robusto e funcional.

2.2 Estratégias de diálogo

A performance de um sistema de conversação Homem-máquina depende muito das regras e da qualidade da estratégia do *DM*. A configuração da estratégia de diálogo define o comportamento de conversação do sistema e é uma tarefa que não é fácil, uma vez que não existe uma definição clara do que constitui uma boa estratégia [9].

Sistemas que interpretam e respondem a comandos de voz devem portanto, implementar estratégias de diálogo que consigam lidar com a falta de fiabilidade dos dados de entrada e fornecer uma verificação de erros e mecanismos de recuperação [7].

2.2.1. Controlo do diálogo

Segundo Michael F Mctear [5] o controlo do diálogo nos sistemas de conversação por voz entre Homem-máquina pode ser classificado em três métodos, *State-based systems*, *Frame-based systems* e *Agent-based systems*. A seleção de uma estratégia de controlo de diálogo determina a flexibilidade que o sistema de conversação Homem-máquina possa ter.

No método *State-based systems* ou *System-initiative* [2, 3], a iniciativa da comunicação é feita por parte da máquina, onde ela toma o controlo total da interação, fazendo perguntas pré-configuradas ao utilizador, que exigem respostas simples [3]. Como exemplo destes sistemas pode-se referir os sistemas de telefone mãos-livres e os sistemas de navegação usados, por exemplo, nos carros. Uma das desvantagens deste método está relacionado com o uso de palavras e frases pré-determinadas o que dificulta a correção de conteúdos e não permite que tenha a iniciativa na conversa [5].

Exemplo de um diálogo com o método *State-based*:

S1: What company would you like to pay?
U1: Abbey National
S2: How much would you like to pay?
U2: One hundred pounds next Monday
S3: What date would you like the payment to be made on?
U3: Next Monday
S4: You want to pay Abbey National one hundred pounds and sixty-one pence on April 5th? Is this correct?
U4: No
S5: Would you like to change the company, amount, or date?

U5: Amount
S6: How much would you like to pay?
U6: One hundred pounds
S7: You want to pay Abbey National one hundred pounds on April 5th? Is this correct?
U7: Yes

"S" corresponde ao sistema e "U" corresponde ao utilizador.
Avaliando as vantagens deste método, pode-se dizer que a estrutura do diálogo é simples de construir e o vocabulário e a gramática necessários para cada estado poder ser determinado com antecedência. Em contrapartida, este método também tem desvantagens. Os diálogos não são naturais, não são permitidas respostas mais completas e não permite que o utilizador tome a iniciativa e possa fazer perguntas.

O método *Frame-based systems* não tem um fluxo pré-configurado de perguntas e respostas. A comunicação depende do conteúdo dos *inputs* fornecidos pelo utilizador e da informação que o sistema tem para extrair. Este método tem alguma capacidade de tratamento de linguagem natural de forma a permitir que o utilizador possa responder de forma mais flexível. A capacidade de linguagem natural é também necessária para corrigir erros de reconhecimento e compreensão, para que o âmbito da comunicação do utilizador seja reconhecida pelo sistema [5].
Este método permite um fluxo de diálogo mais natural e um diálogo potencialmente mais curto, devido ao sistema ser capaz de processar até os *inputs* mais complexos do utilizador, que podem surgir quando o utilizador toma a iniciativa de corrigir os falsos reconhecimentos e mal-entendidos do sistema [5].
Exemplo de um diálogo com o método *Frame-based*:

S1: Good morning. This is the automatic timetable information service.
How can I help you?
U1: I need a connection from Hamburg to Munich.
S2: When would you like to go from Hamburg to Munich?
U2: Next Friday at eight o'clock.
S3: So you want to travel on Sunday the 20th of November at eight o'clock a.m.?
U3: No, on Friday.
S4: So you want to travel on Friday the 18th of November at eight o'clock a.m.?

U4: Yes.
S5: There is the following connection: With Intercity Express 793, departure from Hamburg at 8:02, arrival in Munich at 14:05. Do you want me to repeat the connection?
U5: No, but I need a train for the return trip on Sunday.
S6: At what time would you like to go from Munich to Hamburg on Sunday the 20th of November?
U6: I have to arrive in Hamburg late in the evening.
S7: So you want to arrive in Hamburg between nine p.m. and eleven p.m.?
U7: Yes.
S8: There are three connections. This is the first one: With Intercity Express 880, departure from Munich at 15:54, arrival in Hamburg at 21:58. Do you want the next connection?
U8: No, thanks, that one is fine.
S9: Do you need yet another connection?
U9: No, thank you.
S10: We wish you a pleasant journey. Thank you for calling. Good bye.

"S" corresponde ao sistema e "U" corresponde ao utilizador.
Como vantagens este método permite diálogos mais naturais e o utilizador pode fornecer respostas mais completas. Contudo, os sistemas que utilizem este método não conseguem tratar de diálogos complexos.

O método *Agent-based systems* foi desenhado para permitir uma comunicação complexa entre o sistema, utilizador e a aplicação subjacente de forma a resolver problemas ou tarefas. Existem várias variantes deste método que dependem de aspetos do comportamento de inteligência presente no sistema. Neste método, cada agente da comunicação entre Homem-máquina, tem a capacidade de raciocinar sobre as suas próprias ações e em algumas situações, sobre as ações do outro agente. O sistema tem a capacidade de prever e interpretar as próximas declarações do utilizador. Este método tem normalmente mecanismos para a deteção e correção de erros. Este método é também conhecido como *Mixed-initiative* [3, 5] ou *Plan-based* [2], porque tanto o utilizador como o sistema podem controlar a conversação.
Exemplo de um diálogo com o método *Agent-based*:

U: I'm looking for a job in the Calais area. Are there any servers?
S: No, there aren't any employment servers for Calais. However, there is an employment

server for Pas-de Calais and an employment sever for Lille. Are you interested in one of these?

"S" corresponde ao sistema e "U" corresponde ao utilizador.

Este método permite a utilização de uma linguagem mais natural e mais idêntica a uma linguagem entre humanos e permite uma maior complexidade do âmbito do diálogo. Contudo, estes sistemas são mais difíceis de contruir.

2.2.2. Métodos de aprendizagem

Um sistema de comunicação Homem-máquina, para além da interação, pode fornecer também inteligência de forma a fornecer a melhor resposta ao utilizador e perceber as suas necessidades. O interesse em tecnologias que incorporem a capacidade de diálogo com especial relevo dos assistentes pessoais virtuais tem vindo a aumentar. Segundo o estudo da *MIC* (*Market Intelligence & Consulting Institute*), está previsto que o mercado de inteligência artificial tenha um crescimento exponencial ao longo dos próximo 10 anos e que os assistentes pessoais virtuais inteligentes sejam um dos principais responsáveis do crescimento. À medida que os sistemas se tornam mais inteligentes, também os diálogos se tornam mais complexos e o número de estados, transições e as políticas de decisão, tornam-se muito maiores. Desta forma, foram desenvolvidas várias técnicas para facilitar o processo do *DM*.

Relativamente à modelagem do diálogo, ela pode ser distinguida em dois tipos, *Rule-based* ou *Stochastic* [2].

Rule-based é uma das primeiras abordagens e a mais típica para desenvolver sistemas de conversação Homem-máquina [2]. Nesta abordagem o programador tenta antecipar a comunicação, configurando manualmente regras (*hand crafted*) e as respostas do sistema [2], os componentes de conceitos de estado de crença, transições de estado e as políticas de diálogo [8]. Os sistemas que seguem esta abordagem oferecem um desempenho de confiança, mas apenas para eventos de diálogo que foram antecipadas pelo programador e codificadas nas normas e nas bases de conhecimento. No entanto, devido à grande variabilidade de fatores, para conversações que fujam a diálogos limiares e previsíveis, as regras definidas manualmente não são normalmente muito robustas [2], tornando-se uma tarefa intratável quando se pretende um sistema mais realista, porque é necessário identificar todas as situações possíveis que podem ser encontradas durante o curso de um diálogo [16]. A fim de abordar as limitações da gestão de diálogos baseadas em regras, foram criadas as abordagens *Stochastic* [2]. Estas abordagens usam dados da interação para aprender uma política de

diálogo correta, associando cada estado do diálogo com uma ação do sistema. Definem uma medida objetiva de desempenho e permitem uma otimização de forma automática. São também conhecidas por abordagens *data-driven* e permitem que os sistemas aprendam automaticamente regras generalizáveis, e podem ser categorizadas em *supervised approaches* e *reinforcement learning* [2]. Os modelos treinados oferecem robustez a erros, cobertura a tarefas mais amplas, e podem ser rapidamente treinados para um novo domínio.

Supervised Learning e *Reinforcement Learning* são técnicas de *Machine Learning (ML)*, às quais se pode juntar também a técnica *Unsupervised Learning*.

A tarefa do Gestor Diálogo (*DM*), num sistema de conversação por voz, é controlar o fluxo do diálogo e decidir quais as ações a tomar em resposta ao *input* do utilizador. A fim de concluir essas tarefas, o *DM* precisa de uma estratégia de diálogo que defina quando tomar a iniciativa de um diálogo, quando confirmar o recebimento de uma determinada informação, como identificar e recuperar do reconhecimento e compreensão de erros, entre outras coisas [9]. *ML* dá aos computadores a capacidade de aprender a executar tarefas importantes a partir de vários dados [2].

2.2.2.1. Supervised Learning

Na abordagem *Supervised Learning*, a interação de dados entre Homem e máquina é alimentada por algoritmos de aprendizagem para uma política de diálogo que imita a política de diálogo de um operador humano, como por exemplo, um funcionário de determinado serviço. Nesta abordagem a tarefa de aprender uma política de diálogo é modelado como um problema de classificação com o estado do diálogo como entrada e a ação como saída [2]. Um inconveniente comum das abordagens de *Supervised Learning* é que a seleção da ação é vista como uma sequência de decisões de problemas isolados [2].

2.2.2.2. Unsupervised Learning

Esta aprendizagem distingue-se da *Supervised Learning* pelo facto de que não há um conhecimento anterior dos objetivos de *output*. As observações não são rotuladas com as suas classes correspondentes e é deixado ao algoritmo de aprendizagem que construa uma representação da entrada que é útil para, por exemplo, tomada de decisões, previsões futuras de *inputs* ou deteção de situações externas [9]. Em suma, a *Unsupervised Learning* pode ser descrita como "encontrar padrões nos dados acima e por baixo do que seria considerado ruído puro não estruturado" [9].

2.2.2.3. Reinforcement Learning

Reinforcement learning é um método de aprendizagem que tenta otimizar uma recompensa que é obtida a partir do ambiente e que tem sido aplicado com sucesso na otimização de sistemas SDS [16].

Esta forma de *Machine Learning* pode ser vista como uma aprendizagem por tentativa e erro [9].

O agente de aprendizagem do RL interage com o seu ambiente dinâmico, recebe *feedback* na forma de recompensas positivas ou negativas, de acordo com uma função de recompensa, e tenta otimizar as suas ações de forma a maximizar a recompensa total [9].

Ao atribuir pequenas recompensas negativas para cada ação e uma grande recompensa positiva para a ação bem-sucedida, um agente pode ser treinado para agir de forma *goal-oriented* e eficiente, sem fornecer exemplos explícitos de comportamento ideal ou especificando a forma como uma determinada tarefa deve ser concluída [9].

RL é uma boa opção para usar quando o âmbito do sistema é complexo e incerto, sendo o objetivo de RL maximizar a recompensa [9]. Sistemas que tentam otimizar a recompensa devem usar RL, uma vez que, as boas políticas são reforçadas por boas recompensas [8].

A principal desvantagem de métodos RL padrão usados até agora na área de otimização de SDS prende-se com a elevada exigência ao nível de dados [16].

A aprendizagem da estratégia de diálogo por RL pode ser abordada usando duas formas diferentes. A *model-based learning* e *simulation-based learning* [9].

Model-based

A abordagem *model-based* usa os dados do treino para construir um modelo completo do ambiente, ou seja, estima as probabilidades de transição de um conjunto de dados em que as transições de estado foram registados [9].

Esta abordagem negligencia as respostas dos utilizadores reais para ações do sistema e apenas considera as transições de estado do sistema e os benefícios (*rewards*) associados com eles. Uma vez que as probabilidades de transição são estimadas e a função recompensa é definida pelo programador do sistema, a melhor estratégia pode ser encontrada *offline*, ou seja, sem qualquer interação entre o *DM* e o seu ambiente. É uma forma simples do reforço da aprendizagem e não requer uma ferramenta de simulação de utilizador. No entanto, não tem sido usada em estudos mais recentes, devido a um número de deficiências significativas [9].

Simulation-based

Na abordagem *simulation-based*, uma ferramenta de simulação do utilizador é treinada com um conjunto de dados fornecidos, usando aprendizagem supervisionada [9]. Para isso é utilizado um modelo estatístico para simular o comportamento do utilizador que é primeiro treinado com uma coleção de diálogos entre Homem e máquina. Depois, utilizando *RL*, uma estratégia de diálogo ótima pode ser encontrada por meio da interação de tentativa e erro entre o utilizador simulado e o *DM* de aprendizagem [9]. O sistema de aprendizagem interage com o utilizador simulado e otimiza a sua estratégia de diálogo com base no *feedback* dado pelo utilizador simulado.

Esta forma de *RL*, com uma ferramenta automática de simulação de utilizador, permite que qualquer número de diálogos de treino seja gerado. Além disso, permite também estratégias de diálogo que não estejam presentes na coleção de diálogos existente entre computador e Humanos, para serem exploradas [9]. Possibilita o DM desviar-se das estratégias conhecidas, aprender novas e potencialmente melhores.

O modelo do utilizador precisa de ser capaz de produzir respostas que um utilizador real poderia ter dado numa determinada situação de diálogo, tendo em conta aspetos observáveis, como a história do diálogo e aspetos não observáveis tais como o objetivo do utilizador, memória e preferências [9].

O modelo de utilizador também deve cobrir uma ampla variedade de comportamentos naturais e refletir a diversidade de uma população de utilizadores reais para garantir que as estratégias aprendidas funcionam bem para todos os tipos de comportamento de utilizadores humanos [9].

Esta abordagem é mais complexa mas oferece algumas vantagens. A simulação do utilizador permite que qualquer número de episódios de formação seja gerado de modo que o gerente de diálogo de aprendizagem possa exaustivamente explorar o espaço de possíveis estratégias [9].

Uma forma comum de modular os problemas de RL é usar o modelo *Markov Decision Processes (MDP)*. Este modelo ajuda a encontrar estratégias de diálogo ideais que fazem uma suposição sobre a observação dos dados de eventos gerais [9]. Serve também como uma representação formal do diálogo Homem-máquina e fornece a base para a formulação de estratégias de aprendizagem de problemas. A definição de recompensas é feita em função dos estados de crença [8].

Noutra forma de modulação idêntica ao modelo MDP, o agente não pode ver o estado atual do meio ambiente, mas em vez disso deve inferir o estado a partir de observações que são estatisticamente emitidas a partir do estado. O modelo *Partially Observable Markov Decision*

Process (POMDP) consegue isso através do alargamento do modelo de MDP com um conjunto de variáveis que representam as observações do diálogo e do mapeamento das observações com os estados de diálogo [9]. O modelo POMDP combina a abordagem estatística para a aprendizagem e para tratar de resultados incertos [8]. As recompensas são definidas em função do estado do ambiente verdadeiro em vez do estado de crença.

Por este motivo, os métodos orientados a dados para *Machine Learning* têm, cada vez mais, ganho popularidade na investigação de sistemas SDS. Métodos orientados a dados permitem aos sistemas aprender automaticamente regras generalizáveis a partir de um grande conjunto de dados de exemplos [2].

2.3 Desafios de um sistema de conversação por voz

Como qualquer outro sistema, existem problemas e desafios que têm de ser tidos em conta num sistema de conversação Homem-máquina. Um sistema de conversação ideal é aquele que permite o utilizador realizar os seus objetivos com o mínimo de esforço e problemas. O sucesso da conversação entre Homem e máquina pode depender do nível de maturidade do sistema, dando origem, ou não, nalguma desilusão por parte do utilizador quando o sistema não entende o que o utilizador pretende [3].

Para comunicar com eficiência deve ser capaz de resolver casos dícticos, como por exemplo, "eu vou escolher a segunda opção", "qual é o seu número de telefone". Outra situação que deve ser tida em conta é a capacidade de lidar com partes da fala que podem ser importantes para efetuar uma consulta. Por exemplo, "Eu quero ir de Lisboa para o Porto, mostra-me apenas os voos da TAP". O objetivo desta frase é saber todos os voos da TAP que liguem Lisboa ao Porto de uma só vez, evitando que sejam feitas várias consultas para obter o mesmo resultado.

A capacidade de herdar informações das comunicações anteriores pode ser muito útil em relação aos erros de interpretação. O utilizador pode ter uma pergunta muito complexa onde uma palavra pode ser mal interpretada, como por exemplo, um número. Se o sistema conhecer bem o contexto, basta o utilizador dizer uma frase simples de forma a corrigir o erro, evitando que se repita toda a frase inicial e que se originem novos erros [3].

Sistemas de conversação por voz entre Homem e máquina exigem componentes de tecnologia de alto desempenho. Para além disso, o desenvolvimento destes sistemas exige também que se tenha em atenção outros aspetos. Nesta seção, serão apresentados alguns desses aspetos a ter em conta.

Várias companhias procuram e têm integrado a tecnologia de conversação por voz Homem-máquina nos seus produtos, contudo é importante ter conhecimento das suas limitações, desafios e problemas.

2.3.1. Output inadequado ao tipo de informação

Existem situações onde este tipo de interação pode não ser o ideal no processo de comunicação. Principalmente em situações em que a resposta não é facilmente explicada de forma verbal, como por exemplo, quando a informação contém a apresentação de mapas, imagens e largas tabelas de informação [3].

2.3.2. Trabalhar em ambientes reais

O objetivo de um sistema de conversação Homem-máquina é proporcionar uma forma natural de o utilizador obter a informação que pretende, daí ser importante ter em atenção as suas reais necessidades.

Para desenvolver este tipo de interface é necessário perceber bem como é feita a conversação entre pessoas, uma vez que, se os mesmos métodos forem usados na conversação Homem-máquina, os utilizadores podem sentir-se mais confortáveis com a interação [3].

Uma estratégia para um maior e mais rápido sucesso destes sistemas consiste em realizar testes em ambientes reais, em vez de cenários artificiais. Neste tipo de estratégia, os testes são efetuados por utilizadores reais e permitem que sejam encontrados problemas e soluções de forma mais rápida.

Para Victor W. Zue e James R. Glass [3] uma estratégia que incorpore o mais cedo possível contacto com ambientes de testes reais durante o desenvolvimento, é mais vantajoso do que os sistemas que são desenvolvidos em cenários de testes encenados.

2.3.3. Obter dados

De forma a desenvolver as capacidades do sistema de conversação, é necessário ter um grande conjunto de dados para o desenvolvimento, formação e avaliação dos sistemas.

Para cada ambiente ou linguagem nova, é necessário desenvolver capacidades de linguagem natural. Para isso, vários testes são desenvolvidos com utilizadores reais a fim de treinar o reconhecimento de voz e tornar o sistema mais maduro.

Se o sistema poder fornecer informações úteis e verdadeiras, os utilizadores vão usar o sistema e vão abastecendo-o com dados úteis [3].

2.3.4. Avaliação e treino

Um dos maiores problemas que existem no desenvolvimento de sistemas de comunicação por voz é saber como avaliar o progresso, de forma a saber se foi criado um bom sistema [12]. Normalmente o objetivo de um sistema de conversação Homem-máquina é proporcionar a satisfação do utilizador, tarefa que, no entanto, é difícil de avaliar. Para Steve Young e colegas, [7] não existem medidas padrão para a avaliação do sistema de diálogo.

Os programadores devem definir métricas para avaliar os sistemas e garantir que está a haver evolução [3]. Por exemplo, avaliar se um utilizador completou ou não uma tarefa, o tempo que demorou a concluir a tarefa e número de tentativas. Contudo, estas métricas podem não

ser importantes para a satisfação dos utilizadores, porque os utilizadores podem querer fazer a pesquisa por outras formas e demorar mais tempo e até mesmo em mais que uma tentativa. Uma métrica importante poderá passar por perceber se os utilizadores gostaram do sistema, pedir-lhes para avaliar o sistema e saber se o vão utilizar novamente.

Outras técnicas de avaliação consistem em recolher as respostas a determinadas perguntas [17]. As respostas dadas pelo sistema são comparadas com um conjunto de respostas certas. A avaliação é feita pelo número de respostas certas que o sistema responde. Contudo, esta abordagem tem várias limitações porque podem existir várias estratégias para obter as respostas. Noutras técnicas usadas, a avaliação corresponde à contagem da quantidade de discurso inadequado, precisão do conceito, sucesso da transação, *delay* do sistema, entre outros [17]. Deve ter-se em conta que, a avaliação feita por utilizadores pode ser muito subjetiva, dependendo do diálogo que tem com o sistema.

Testar um sistema de SDS com os utilizadores humanos pode ser uma tarefa muito demorada. Simuladores de utilizadores fornecem uma forma de acelerar o processo de desenvolvimento, fornecendo um mecanismo de teste mais eficiente [8]. Os simuladores conseguem simular erros de forma a gerar confusões apropriadas e pontuações de confiança e podem ser usados para treino ou avaliação [18].

Os resultados de Tatjana Scheffler e colegas [18] mostram que uma boa configuração do sintetizador TTS rivaliza com a voz humana, tanto nos índices de reconhecimento, bem como variabilidade. No entanto existem dúvidas em relação à utilização de voz sintetizada na simulação de utilizadores na obtenção de diálogos realistas, já quê a voz sintetizada em combinação com ASR pode ter sempre o mesmo resultado, uma vez que o reconhecimento pelo ASR será sempre da mesma maneira. Pelo contrário, a simulação com um utilizador real, em que a voz pode variar em vários aspetos, pode resultar em resultados aleatórios do ASR.

Desenvolver um sistema de conversação Homem-máquina, envolve decisões que dependem do tipo de tarefas que o sistema vai executar [5] e da quantidade de dados de treino no domínio [8]. É importante ter uma estrutura que vá de encontro às necessidades. Vários métodos são utilizados de forma a ir ao encontro dos requisitos pretendidos. Alguns deles são:

- Pesquisa bibliográfica;
- Entrevistas aos utilizadores para extrair a informação requerida de forma a construir o domínio e modelos das tarefas;
- Observações de casos de estudo e gravações de tarefas executadas por seres humanos;
- Experiências reais no campo, onde alguns parâmetros das tarefas são simulados, simulações em grande escala e rápida prototipagem;

- Análise de conversação entre humanos.

O treino dos sistemas SDS, em teoria é feito usando os utilizadores humanos ou através do diálogo entre humano e computador, mas na prática o vasto espaço de estados possíveis do diálogo e estratégias não podem ser exploradas sem o uso de ferramentas de simulação de utilizadores automáticas [9]. Os modelos treinados oferecem robustez a erros, uma cobertura mais ampla sobre tarefas, e podem ser rapidamente treinados para um novo domínio [2].
Como em qualquer *software*, o sucesso de um sistema de conversação Homem-máquina não depende apenas das funcionalidades e performance do software, mas também da sua usabilidade e aceitação pelos utilizadores [5].

2.3.5. Incapacidade humana de comunicar

Outro problema está relacionado com a versatilidade em relação às linguagens aceites pelos sistemas. Existem vários sistemas de conversação Homem-máquina, contudo muitos não incluem a capacidade de multilinguagem, considerado um dos requisitos dos sistemas de comunicação dos dias de hoje [6]. Isso impede, por exemplo, que um utilizador que só saiba falar português possa usar um sistema de conversação em que a linguagem do sistema seja o inglês.
Os sistemas de conversação Homem-máquina podem ser muito importantes para as pessoas que tenham limitações físicas e motoras. Contudo, se a limitação física estiver relacionada com audição ou incapacidade de falar, estes sistemas podem não ser uma mais-valia.

2.3.6. Compreensão da linguagem falada

A conversação Homem-máquina, de forma a ser cada vez mais funcional, exige a utilização de componentes de alta performance, tais como, o reconhecimento de voz e de compreensão da linguagem [3].
O desenvolvimento de sistemas de conversação partilha muitos dos desafios de investigação com outros sistemas que usam o reconhecimento de voz para outras aplicações, como por exemplo, a criação de textos a partir de voz.
Os telefones móveis são, cada vez mais, meios de comunicação usados para a conversação por voz Homem-máquina e exigem que os sistemas estejam preparados para problemas na qualidade da voz, relacionados como baixas larguras de banda, baixa relação sinal ruído, entre outros.

Nos carros, na utilização destes sistemas, também podem trazer alguns problemas, nomeadamente com ruídos de fundo gerados pelo motor ou pelo contacto dos pneus com a estrada.

Outra situação pode estar relacionada simplesmente com o utilizador, como por exemplo, situações de stresse. Este tipo de situações pode influenciar a dicção e o timbre.

A prenuncia pode ser também um problema para o sistema de reconhecimento de fala, uma vez que a dicção e a acentuação das palavras pode variar de pessoa para pessoa. Para este tipo de problemas, uma solução pode ser a criação de perfis que identifiquem as características de cada utilizador.

O sistema deve ser capaz de lidar com palavras desconhecidas, ou seja, deve detetar e aprender novas palavras. Ignorar esta funcionalidade pode resultar no descontentamento do utilizador. Para que estes sistemas possam superar estas dificuldades, estes tem de ser exercitados e testados, de forma a ficar mais robustos.

Para aplicações simples, como por exemplo, assistentes automáticos, é possível utilizar técnicas de análise de voz não tão sofisticadas. Mas à medida que as iterações se tornam mais complexas o sistema pode necessitar de um sistema de análise de linguagem mais sofisticado de forma a entender o contexto.

Algumas estratégias têm sido estudadas relativamente ao reconhecimento da linguagem falada. Numa delas, em primeiro lugar, deve reconhecer as palavras e frases, depois quando necessário, recorre a uma análise linguística mais completa. Numa estratégia alternativa é executado uma análise linguística mais completa a fim de descobrir a estrutura linguística da expressão, quando a análise linguística falha, usa então o reconhecimento das palavras e frases [3].

A capacidade do sistema perceber as intensões do utilizador durante a comunicação, pode ser melhorada com outros tipos de interfaces, como a leitura de expressões humanas [1].

2.3.7. Geração de linguagem falada

Os componentes de síntese de voz são normalmente os que mais impressionam os utilizadores. A geração de linguagem falada é uma componente importante na conversação Homem-máquina. Modelos e métodos devem ser desenvolvidos de forma a gerar frases para que pareçam o mais natural possível aos utilizadores, como por exemplo, aspetos como a pronúncia e a sensação de emoção [3]. O objetivo de ter um sistema que permita uma comunicação o mais natural possível é uma preocupação que vem desde os primeiros sistemas implementados [4].

2.3.8. Gestor do diálogo

Em alguns sistemas de conversação, a estratégia dos esquemas de diálogo é feito de forma manual pelos programadores dos sistemas e por isso, dependem muito da intuição dos programadores, podendo ser uma tarefa demorada. Uma das alternativas é o uso de técnicas de aprendizagem para detetar automaticamente a estratégia do diálogo. Contudo, independentemente da abordagem, é necessário recolher dados do diálogo entre os humanos e os humanos e a máquina.

Como ainda não existem sistemas de diálogo capazes de dialogar sem restrições, é importante que os utilizadores tenham conhecimento das limitações dos sistemas, tanto a nível de conhecimento como a nível de consultas que o sistema possa entender. De forma a auxiliar os utilizadores devido às limitações do sistema, uma técnica de ajuda pode ser a solução [3].

2.3.9. Portabilidade

Criar um sistema de conversação por voz com iniciativa mista e robusto, exige uma grande quantidade de trabalho e de esforço. Para que estes sejam mais flexíveis, eles devem ser também portáteis, ou seja, devem ter a capacidade de ser usados em diferentes domínios e com diferentes línguas [4]. Isto permite, por exemplo, que sistemas já existentes e preparados para a portabilidade sejam usados por estudantes para criarem os seus próprios sistemas com o âmbito e linguagens que necessitarem.

Tem havido também o esforço para que o *Voice eXtensible Markup Language (VoiceXML)* seja um modelo *standard* para permitir o acesso a informação disponível na internet através da voz [3].

Para que um sistema seja portátil, deve separar corretamente os algorítmicos do sistema dos algoritmos da aplicação. Deve ter métodos para adquirir os modelos acústicos, modelos de linguagem, gramáticas, estruturas semânticas para compreensão da linguagem e modelos de diálogo exigidos por uma nova aplicação [3].

2.3.10. Tomada da vez (Turn-Taking)

Um diálogo segue um ciclo, conhecido como o ciclo de diálogo, em que quando um dos participantes diz alguma coisa, o conteúdo é interpretado pelo ouvinte, que toma uma decisão sobre como responder e a resposta é transmitida de volta para o participante original [8]. Esse

mesmo processo de ouvir, compreender, decidir e responder, ocorre de seguida com o outro interveniente e o ciclo repete-se quantas vezes for necessário.

Num diálogo entre humanos a comunicação é feita pelos intervenientes envolvidos, durante a comunicação, cada um dos intervenientes pode interromper o discurso do outro de forma a corrigir, acrescentar aspetos, etc. A maioria dos sistemas de diálogo funcionam num modelo sequencial em que cada interveniente fala de cada vez. O utilizador fala e quando termina o seu discurso, o sistema deteta um silêncio seguinte à fala do utilizador e então percebe que é a sua vez de falar [19].

Pode-se definir "vez" de um diálogo como um período em que um dos participantes tem para dizer alguma coisa ao outro participante [8].

De forma a conseguir ultrapassar esta limitação, os sistemas com diálogo incremental são capazes de compreender o discurso do utilizador em tempo real permitindo, assim, um conjunto mais rico de comportamentos que refletem na tomada de posse para falar [19]. Eles podem interromper o discurso do utilizador e rapidamente reportar um problema e podem também ser interrompidos. Desta forma, o diálogo torna-se mais eficiente e mais curto, e assim, obter o objetivo mais rapidamente.

Os sistemas de diálogo incrementais têm a capacidade de indicar um erro ao utilizador de uma forma mais reativa e impedi-lo de falar por um longo tempo sem ser compreendido pelo sistema. Em ambientes ruidosos, estes problemas são ainda mais prováveis de acontecer. Como Hatim Khouzaimi, Romain Laroche e Fabrice Lefèvre [19] demostram, os sistemas de diálogo incremental são mais eficazes que os sistemas não incrementais onde os utilizadores ou os sistemas não se podem interromper um ao outro.

É importante ultrapassar o modelo de diálogo *turn-by-turn* onde o diálogo é gerido à vez, falando um elemento de cada vez, e oferecer a cada um a possibilidade de intervir quando for necessário, como por exemplo, para corrigir um erro, esclarecer um mal-entendido, fazer referência a algo específico, etc.

2.3.11. Identificação e verificação do locutor

Na conversação Homem-máquina, o reconhecimento de voz e interpretação semântica têm sido os principais domínios de investigação, contudo, novas funcionalidades sobre a interação entre humanos e máquinas tem sido estudadas de forma a tornar os sistemas *SDS* mais inteligentes [20]. A identificação e verificação do locutor são umas das técnicas que trazem mais-valias para um sistema *SDS* e que tornará o sistema *SDS* mais natural e fácil de usar no futuro.

O objetivo de um procedimento de identificação do locutor é descobrir qual a pessoa que está a falar nesse momento. Para isso, é necessário treinar o sistema com amostras dos sinais de fala de todas as pessoas cujo acesso seja permitido.

Pode-se distinguir dois tipos diferentes de sistemas de reconhecimento de voz em termos de orador. São eles o *open-set* e o *closed-set*. Na identificação pelo sistema *closed-set,* os dados da pessoa a ser identificada devem estar presentes na base de dados de voz existente. A identificação pelo sistema *open-set* permite a entrada de voz de uma pessoa ausente da base de dados. Neste caso, o sistema deve identificar que o locutor é desconhecido.

2.4 Soluções Multímodo

Os sistemas de conversação Homem-máquina podem funcionar bem em muitos casos. No entanto, existem situações em que uma simples comunicação por voz pode não funcionar. Por isso, é importante juntar várias formas de interação, que em conjunto, originam um sistema multímodo que permite resolver muitas das falhas que um sistema *unimodo* possa ter [1].

As interfaces multímodo mais comuns combinam vários modos de *inputs* e *outputs*, como por exemplo, voz, imagens, gestos e expressões faciais, como a leitura de lábios, tato, entre outros. Desta forma, é mais fácil a comunicação entre o Homem e a máquina, aumentando a precisão do reconhecimento da informação [1].

Estudos sobre a fala mostram que ter informações visuais e de áudio aumentam a taxa de sucesso da transferência de informações, especialmente quando a mensagem é complexa ou quando a comunicação ocorre num ambiente ruidoso [12].

A imagem seguinte [1] representa uma estrutura genérica de um sistema multímodo. O sistema, para além da voz, capta vários sinais emitidos pelo utilizador. No fim de processar, o sistema pode responder ao utilizador em várias formas para além da voz.

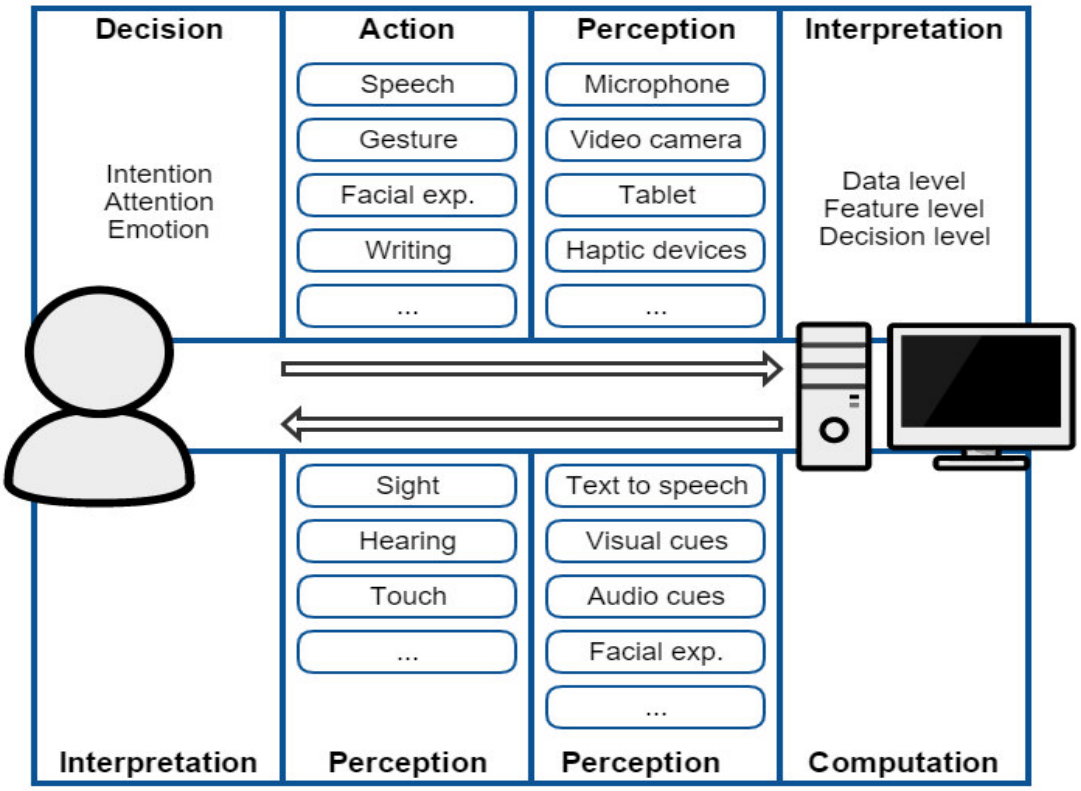

Figura 2.Exemplo de um sistema de conversação Homem-máquina multímodo.

Um sistema multímodo tem vantagens sobre um sistema *unimodo*. A utilização de vários sensores na tarefa de reconhecimento da voz aumenta a precisão do reconhecimento. Numa situação em que é feita a leitura da voz juntamente com a leitura facial, pode resultar num aumento de precisão da fala, até em ambientes com muito ruído, uma vez que, os sensores de leitura facial ajudam a resolver as falhas que os sensores de voz podem originar [1]. A leitura de expressões faciais, juntamente com a voz, ajudam a máquina a perceber melhor as emoções humanas, como por exemplo, raiva, alegria, tristeza, prazer.

2.5 Sistemas de reconhecimento de voz

O desejo de implementar sistemas que tenham a capacidade de emular uma comunicação verbal humana é antigo. A fala é a forma mais natural de comunicação humana e o processamento de voz tem sido um desafio já com algumas décadas.

Speech Recognition tornou possível que um computador entenda a linguagem humana e que siga instruções comunicadas por voz, tornando-se num passo importante do processo conversação Homem-máquina, podendo-se definir da seguinte forma:

> "Speech Recognition (is also known as Automatic Speech Recognition (ASR), or computer speech recognition) is the process of converting a speech signal to a sequence of words, by means of an algorithm implemented as a computer program" *[12]*.

Ou seja, *Speech Recognition*, também conhecido por *Automatic Speech Recognition (ASR)* ou *Computer Speech Recognition*, é o processo de converter a voz numa sequência de palavras, através de um algoritmo implementado num programa de computador.

Atualmente existe uma grande variedade de sistemas onde é aplicado uma interface de conversação Homem-máquina. *Speech Recognition* pode ser usado em sistemas de inserção de dados, transcrição de voz, comandos de tarefas e controlo de sistemas (por exemplo casas inteligentes, domótica), entre outros, trazendo comodidade e rapidez na execução das tarefas.

Se olharmos para o *Speech Recognition* como uma área de investigação, o objetivo é de desenvolver técnicas e sistemas para o reconhecimento de voz pelas máquinas. Se olharmos num ponto de vista de produto, atualmente existem várias companhias, empresas e comunidades que tem disponíveis várias formas de integrar a capacidade de reconhecimento de voz em aplicações e *hardware*.

Existem vários componentes que podem fazer parte dos sistemas de reconhecimento de voz, como por exemplo, modelos de linguagem ou gramática, modelos acústicos e um descodificador.

Um modelo de linguagem ou gramática são modelos compostos por idiomas que contêm uma lista bastante grande de palavras e a sua probabilidade de ocorrência numa determinada sequência. São frequentemente usadas em aplicações de *call centers* e controlo de aplicações.

Cada palavra tem uma lista associada de fonemas, que correspondem aos sons distintos que formam uma palavra.

O modelo acústico contém uma representação estatística dos sons distintos que compõem

cada palavra no modelo de linguagem ou gramática. Cada som distinto corresponde a um fonema.

O descodificador ou *Decoder* é um *software* que usa no áudio da fala do utilizador e procura o modelo acústico para os sons equivalentes. Quando é encontrada uma correspondência, o descodificador determina o fonema correspondente ao som. Ele continua a pesquisa dos fonemas correspondentes até que o utilizador faça uma pausa no discurso. Depois ele procura o modelo de linguagem ou o ficheiro da gramática para a série equivalente de fonemas. Se houver correspondência ele retorna o texto da palavra correspondente ou frase.

Os sistemas de reconhecimento de voz têm inúmeras características. Eles podem ser usados ou distribuídos a partir de várias tecnologias diferentes, como por exemplo por *SDKs*, *APIs*, *toolkits*, *framework*s e *libraries*, podem ter objetivos diferentes, disponibilizar várias formas de integração e de usabilidade, depender de uma ligação à internet ou não para funcionar, funcionar em vários ambientes como *web*, *mobile*, *desktop*, funcionar em várias plataformas como *Android*, *iOS*, *Windows*, *Linux*, entre outros.

Para além das características, todos estes mecanismos tem outros aspetos que podem ser relevantes para a satisfação do utilizador. Aspetos como funcionalidades, velocidade, performance, usabilidade ou esforço relativo à facilidade de trabalho com o produto e precisão.

O aspeto velocidade mede o tempo em que o mecanismo de reconhecimento de voz leva para fazer o reconhecimento e é medido pelo fator tempo real.

A usabilidade de um mecanismo de reconhecimento de voz é importante na medida em que se for mais intuitivo de usar, mais fácil de configurar, com mais recursos *out of the box* e mais percetível, proporciona um esforço menor de aprendizagem, refletindo-se num período de tempo menor despendido para pôr um mecanismo de *Speech Recognition* a funcionar.

Uma das funcionalidades importantes está relacionada com o tipo de expressões orais que consegue reconhecer, podendo estas ser do tipo:

- Palavras isoladas, onde o mecanismo apenas aceita uma palavra ou expressão de cada vez;
- Palavras ligadas, idêntico ao tipo de palavras isoladas, mas permite que várias palavras sejam tratadas ao mesmo tempo com o mínimo de pausa entre elas;
- Discurso contínuo, permite aos utilizadores falar de uma forma próxima do natural, enquanto o computador determina o conteúdo. Os sistemas de reconhecimento de voz com este tipo de capacidades são os mais difíceis de implementar, porque necessitam

de métodos especiais para identificar corretamente a sintaxe;

- Discurso espontâneo, deve aceitar qualquer tipo de discurso, com capacidade para lidar com interjeições tipo “ahh”, “humm” gaguejos, entre outros.

A precisão dos mecanismos de reconhecimento de voz é muito importante para a satisfação dos utilizadores e essa precisão pode variar conforme vários fatores [12]. Um deles é o meio envolvente que corresponde ao tipo e frequência de ruídos e às condições de trabalho. O tipo de transdutor, microfone, telefone entre outros, também pode afetar, assim como o canal de comunicação relativamente à largura de banda, distorção e eco. Outros fatores têm a ver com o orador, onde o sexo, a idade, o estado físico e psíquico, a pronúncia, o tom de voz (tranquila, normal, tons altos), tipo de expressões (palavras isoladas, discurso contínuo, discurso espontâneo) e velocidade da fala, podem influenciar a precisão. Outro fator importante tem a ver com o vocabulário do mecanismo, ou seja, com as características da informação treinada disponíveis, vocabulário específico ou genérico. A precisão pode ser medida em termos de precisão de desempenho, que é normalmente classificado pela taxa de erro por palavra *Word Error Rate (WER)*. Pode ser também avaliada pela *Single Word Error Rate (SWER)* ou por *Command Success Rate (CSR)* [12].

Os mecanismos de reconhecimento de voz são mecanismos que necessitam de algum poder computacional e, portanto, o aspeto performance é relevante principalmente em sistemas que funcionem na própria máquina. É importante perceber a quantidade de recursos que cada mecanismo consome para que seja possível não desperdiçar o uso de muita memória ou processador e até mesmo bateria em alguns casos.

O desempenho dos mecanismos de reconhecimento de voz pode se degradar se não forem treinados e testados em condições reais de usabilidade [21]. Como foi descrito anteriormente são vários os fatores que podem influenciar o desempenho do sistema de reconhecimento de voz. O processo de treino consiste em treinar o sistema, de forma a aprender sobre as inflexões únicas e variações da voz do utilizador, em condições reais ou simuladas.

A quantidade de tempo que é despendida no treino do *software* de reconhecimento de voz, mesmo sendo elevada, no final vai compensar porque depois o *software* não vai apresentar tantos erros. Para aqueles que apenas usam o *software* de reconhecimento de voz para tomar notas ou para escrever documentos que não requerem um alto nível de precisão, a formação básica oferecida dentro dos principais recursos do software deve ser suficiente.

Um mecanismo de *Speech Recognition* pode ser também caracterizado relativamente à sua abordagem [12]. *Acoustic Phonetic*, *Pattern Recognition*, *Artificial Intelligence* e *Connectionist Approaches* são algumas abordagens utilizadas [12].

Na abordagem *Acoustic Phonetic* o reconhecimento de voz baseia-se na pesquisa de sons da fala e no seu relacionamento com etiquetas. Isto significa que existem finitas unidades fonéticos distintas (fonemas) em linguagem falada e que estas unidades se caracterizam em geral por um conjunto de propriedades acústicas que se manifestam no sinal de fala ao longo do tempo. Num primeiro passo, desta abordagem, é realizada uma análise espectral do discurso combinada com uma característica de deteção que converte as medições espectrais para um conjunto de características que descrevem as amplas propriedades acústicas de diferentes unidades fonéticas. Depois é feita uma segmentação e rotulagem em que o sinal do discurso é segmentado em regiões acústicas estáveis. É anexada uma ou mais etiquetas fonéticas para cada região segmentada, resultando numa caracterização fonética da estrutura do discurso. Por último esta abordagem tenta determinar uma palavra válida ou série de palavras a partir das sequências de etiqueta fonéticas produzidas pela segmentação para a rotulagem.

A abordagem *Pattern Recognition* envolve essencialmente dois passos, treino de padrões e comparação de padrões [12]. A característica essencial desta abordagem é que ela utiliza estruturas matemáticas e estabelece representações consistentes do padrão da fala, para comparação de padrões de confiança a partir de um conjunto de amostras de treino marcados através de um algoritmo de treino. Uma representação padrão da fala pode ser na forma de um modelo de fala ou de um modelo estatístico, como por exemplo, o modelo *Hidden Markov Model (HMM)*, e pode ser aplicado sobre um som, palavra ou frase. Na comparação de padrões é feita uma comparação direta entre o discurso desconhecido (a fala a ser identificada) com os possíveis padrões aprendidos durante o treino. Esta abordagem pode ser executada pelos seguintes métodos [12]:

- *Template approach* – consiste numa coleção de protótipos de padrões de fala que são armazenados como padrões de referência que representam o dicionário de palavras candidatas. O reconhecimento é então realizado, combinando um enunciado falado desconhecido com cada um desses modelos de referência e selecionando a categoria do melhor padrão correspondente;
- *Stochastic approach* – consiste na utilização de modelos de probabilidade para tratar informações incertas ou incompletas. No reconhecimento de fala podem surgir dúvidas a partir de muitas fontes, como por exemplo, sons confusos, dicção do orador,

efeitos contextuais, e palavras homófonas.

A abordagem *Artificial Intelligent* pode caracterizar-se como uma tentativa de mecanizar o processo de reconhecimento de acordo com a maneira como uma pessoa aplica a inteligência em visualizar, analisar e caracterizar o discurso baseado num conjunto de características acústicas avaliadas. Aplicando o acesso a bases de dados, esta abordagem utiliza informação sobre linguística, fonética e espectrograma. Estas bases de conhecimento normalmente resultam de um cuidadoso estudo de espectrogramas e é incorporada usando regras ou procedimentos.

Connectionist Approaches está relacionada com a abordagem *Artificial Intelligence*, e permite a utilização de técnicas especiais, como redes neurais. São técnicas que permitem integrar fonemas, léxicos, sintaxe, semântica e até mesmo conhecimento pragmático. As redes neurais permitem aprender as relações entre eventos fonéticos.

A maioria dos sistemas de reconhecimento de voz utilizam *Hidden Markov Models* para lidar com a variação temporal da fala e *Gaussian Mixture Models (GMMs)* para determinar a melhor forma de cada estado de cada *HMM* se encaixar numa *frame* ou numa pequena janela de *frames* de coeficientes que representam o *input* acústico [22]. *GMMs* podem ser substituídos por modelos *Deep Learning (DL)* como são os casos de *Deep Neural Networks (DNNs)*, *Deep Belief Networks (DBNs)*, ou *Deep Auto Encoders (DAEs)* [23].

Neste capítulo são apresentados vários sistemas de *Speech Recognition* juntamente com algumas das suas principais características. Uns são distribuídos de forma comercial, enquanto outros, são mantidos por comunidades ativas para estudo, investigação e utilização geral. Devido à existência de sistemas que não fornecem grande controlo sobre algumas propriedades essenciais para o reconhecimento de voz, têm sido lançados alguns sistemas *open source* de *ASR* [24]. Existem muitos outros sistemas disponíveis, mas devido a terem uma comunidade de apoio pequena ou por não terem um conjunto de informação razoável disponível, não aparecem nesta listagem.

Segundo o estudo de Christian Gaida e os seus colegas [24], acerca da comparação dos mecanismos *open source* de *Speech Recognition Kaldi*, *Sphinx toolkit* e *HTK*, chegou-se à conclusão que o *HTK toolkit* necessita de muito mais tempo, conhecimento e esforço para obter resultados em relação as outras duas *toolkits*, é mais complicada e os resultados obtidos são idênticos aos resultados obtidos com a *Sphinx toolkit*, mas o esforço é muito maior. O *Sphinx toolkit* tem boas instruções para a preparação do treino e permite reconhecimento de

voz com bons resultados num curto espaço de tempo, contudo não tem tantos recursos como o *Kaldi*. *Kaldi* fornece instruções para a preparação do treino mais avançadas e permite obter muito bons resultados. Comparando *Kaldi* com *Sphinx toolkit* e *HTK*, Christian Gaida e os seus colegas [24], consideram que *Kaldi* pode ser visto como uma revolução nas tecnologias de reconhecimento de voz.

2.5.1. CMUSphinx

CMUSphinx [25] é um conjunto de ferramentas *open source* dedicadas ao reconhecimento de voz e é o resultado de mais de 20 anos de investigação pela *Carnegie Mellon University*. O grupo é composto essencialmente por quatro ferramentas. *Pocketsphinx*, que é uma *library* de reconhecimento de voz que pode ser integrada em sistemas como *Linux*, *Windows*, *MacOS*, *iOS** e *Android*; *Sphinxtrain* é uma ferramenta de treino do modelo acústico; *Sphinxbase* é uma *library* de suporte necessária a *Pocketsphinx* e *Sphinxtrain*; *Sphinx4* é uma *library* de reconhecimento de voz que pode ser ajustada e modificada, vocacionada para funcionar em servidores e aplicações de *desktop*. O *CMUSphinx* tem vários modelos que representam várias linguagens e permite ainda a criação de novos modelos para outras linguagens. Estas ferramentas foram desenvolvidas essencialmente para plataformas com poucos recursos e podem ser distribuídas comercialmente devido a licença de código aberto *BSD licence*. Tem frequentes *releases* e tem uma comunidade forte e ativa. Existe também disponível muita documentação sobre estas ferramentas.

Na utilização das ferramentas disponíveis, todos os parâmetros podem ser facilmente configurados num ficheiro de configuração, contudo é necessário alguma experiência, uma vez que a documentação é suficiente, mas não extensa [24].

A sua licença é baseada no modelo BSD, o que significa que pode ser distribuído de forma livre e até mesmo de forma comercial e, portanto, permite que investigadores e programadores possam desenvolver sistemas de reconhecimento de voz.

* De referir que é possível utilizar a *Pocketsphinx library* para reconhecimento de voz em sistemas *iOS* através da *framework OpenEars*.

2.5.2. Nuance

Nuance [26] dispõe de vários produtos comerciais de reconhecimento de voz e *TTS* dedicados a várias áreas de negócio. Para além disso tem disponíveis alguns *SDKs* para integrar com outros sistemas, são eles: *Dragon NaturallySpeaking SDK Client Edition*; *Dragon*

NaturallySpeaking SDK Server Edition e *Nuance Mobile Developer SDK*.

O *Dragon NaturallySpeaking SDK Client Edition* pode ser integrado com qualquer aplicação *Windows* de forma a obter as capacidades de reconhecimento de voz. É bastante completo e, para além do reconhecimento de voz, permite configurar comandos por voz para responder quando determinada sequência de palavras for reconhecida, *TTS*, entre outras características.

O *Dragon NaturallySpeaking SDK Server Edition* é especificamente concebido para ser integrado em soluções que requerem o reconhecimento de voz para ser executado em segundo plano, sem qualquer interação com os utilizadores em sistemas *Windows*. Permite o reconhecimento de voz a partir de gravações de voz.

Nuance tem também disponíveis *SDKs* para integrar com aplicações móveis que funcionem sobre plataformas *Android*, *iOS* e *Windows Phone*. Estes *SDKs* pertencem a um grupo chamado de *Nuance Mobile Developer SDK* e permitem integrar os serviços de reconhecimento de voz que estão disponíveis através da sua *cloud*.

Disponibiliza também *webservices* que permitem o acesso aos seus serviços de reconhecimento de voz.

A *Nuance* comprou a *IBM Desktop ViaVoice* em 2003 e tem disponível uma boa documentação para quem quiser utilizar os produtos da *Nuance*.

2.5.3. Kaldi

Kaldi [27] é um *toolkit* destinado à investigação para o reconhecimento de voz. Os seus algoritmos usados são principalmente algoritmos genéricos que possam ser executados com diferentes conjuntos de dados e foram projetados de forma a não terem falhas, exceto em situações muito raras. *Kaldi* tem a capacidade de fazer conversões por voz em tempo real, sem que primeiro seja necessário capturar o áudio todo para depois obter o texto. *Kaldi* dispõem de variantes que permitem o reconhecimento da fala através de redes neurais. Em relação ao código desta *toolkit*, ele é exaustivamente testado, estruturado e fácil de entender, tornando-o fácil de utilizar e de modificar.

Uma vez que o treino de modelos acústicos é um processo computacional exigente. *Kaldi* dispõem de meios que permite o uso de *GPUs* para acelerar o processo [24].

Não é necessário ter muita experiencia para poder utilizar as instruções e scripts disponíveis nesta *toolkit* e a aprendizagem é feita num curto espaço de tempo [24].

Tem uma boa documentação, traz muitas funcionalidades *out-of-the-box* e é *open soure*, pode ser usada e distribuída de forma livre e comercial, segundo o licenciamento *Apache 2.0* que permite entre outras coisas, que o código seja alterado, distribuído e usado de forma

comercial.

Kaldi pode ser usado em plataformas *Windows*, *Darwin*, *Cygwin* e *Linux*.

2.5.4. OpenEars

OpenEars [28] é um *Speech Recognition* e *speech synthesis* para *iPhones*, *iPads* e *iPods*, ou seja sistemas *iOS*, que permite o reconhecimento e sintetização de voz em modo *offline* de forma rápida e simples. Permite também obter grandes resultados como modelos de linguagem estatísticos e gramáticas, sem haver a necessidade de mais esforço do que criar uma estrutura *NSArray* ou *NSDictionary*. Não usa ligações à internet e por isso não tem custos extras e contas para criar. Esta *framework* usa a *framework open source CMU Pocketsphinx*.

OpenEars usa cinco bibliotecas, *OpenEars*, *CMU Pocketsphinx*, *CMU Sphinxbase*, *CMU Flite* e *CMUCLMTK*. A biblioteca *OpenEars* é licenciada pela *Politepix Public License version 1.0* e todas as restantes bibliotecas têm a sua própria licença. De forma resumida, o licenciamento é livre e esta *framework* pode ser usada em aplicações de forma comercial.

No *Speech Recognition* do *OpenEars, o* reconhecimento de voz é apenas suportado para as linguagens inglês e espanhol. O seu funcionamento exige poucos recursos no processo de *Speech Recognition*. Consegue detetar se os *headphones* estão ligados, continuando o *Speech Recognition* durante o *TTS* apenas quando eles estão ligados e suporta dispositivos de áudio por *bluetooth*. *OpenEars* encaminha a informação resultante do *Speech Recognition* para qualquer parte da sua aplicação, suporta gramáticas e consegue obter a lista *n-best* com pontuação.

Caso a sua aplicação necessite de funcionalidades mais específicas e melhoradas, o *OpenEars* tem disponível vários *plugins* para adicionar. De referir que estes *plugins,* que permitem mais funcionalidades e otimização de desempenho, são pagos. O *plugin RapidEars* acrescenta a funcionalidade de reconhecimento de voz em tempo real de um discurso que esteja a acontecer no momento. Não necessita de esperar por pausas para o processamento da voz e não é necessário uma conexão de rede. A sensibilidade deste *plugin* é a suficiente para ser usada em jogos e aplicações que necessitem de reconhecimento de voz em tempo real, *feedback* rápido e vocabulário de tamanho médio.

O *plugin NeatSpeech* é para *speech synthesis (TTS)*. O *NeatSpeech* trata de todas as filas de vozes e do *multithreading*, permitindo que o programador se concentre mais no desenvolvimento da sua aplicação.

RuleORama cria gramáticas baseadas em regras usando a mesma *API* de geração de gramáticas dinâmicas do *OpenEars*. Este *plugin* permite adicionar gramáticas de

reconhecimento baseadas em regras para aplicações de reconhecimento em tempo real.

O *plugin Rejecto* funciona como um filtro de sons e vocabulário não desejados. Um dos problemas mais comuns nos sistemas de reconhecimento de voz em modo *offline* são os sons e palavras captados que não estão no vocabulário das aplicações e as aplicações tentam interpretar como se fossem palavras conhecidas. Este *plugin* permite reconhecer vários sons e palavras que não conhece, rejeita-os e assim melhora-se a precisão do sistema.

SaveThatWave permite a gravação de ficheiros de áudio *WAV* sem bloquear o discurso obtido, pelo controlo *Pocketsphinx*. Permite também a receção de notificações de escrita do ficheiro de áudio, incluindo o caminho para o ficheiro. Possui ainda as funcionalidades para apagar suspender e retomar uma gravação. Este *plugin* é bom para quem também pretende fazer *upload* de vozes para um serviço *cloud*.

Tem disponível uma boa documentação.

2.5.5. HTK

Hidden Markov Model Toolkit (HTK) [29] é um conjunto de ferramentas *open source* portáteis, usadas principalmente para a investigação de *Speech Recognition* mas também é usado em muitas outras tarefas como *Speech Synthesis*, treino de modelos acústicos, reconhecimento de caracteres e sequências de *ADN*. É composto por um conjunto de bibliotecas, modelos e ferramentas. As ferramentas fornecem funcionalidades sofisticadas para a análise da fala e podem ser usadas em sistemas como *Linux*, *Solaris*, *IRIX*, *HPUX*, *MacOSX*, *FreeBSD* e *Windows*.

HTK foi originalmente criado no *Machine Intelligence Laboratory*, do departamento de engenharia da universidade de *Cambridge* no ano de 1989. Em 1995 a *Entropic Research Laboratory Inc* adquiriu os direitos do *HTK*. Depois em 1999 a *Microsoft* comprou a *Entropic* e decidiu entregar a licença do *HTK toolkit* ao departamento de engenharia da universidade de *Cambridge* para que esta possa desenvolver e distribuir o *software*.

A ferramenta disponibilizada pela *toolkit HTK* para o reconhecimento de voz tem o nome de *HVite*. Esta ferramenta permite receber como *input* uma sequência de palavras admissíveis, um dicionário que define como cada palavra é pronunciada e conjuntos de *Hidden Markov Models*.

A *toollkit* pode ser obtida de forma gratuita bastando aceitar os termos da licença *HTK Licence*. Podem-se construir aplicações com a *HTK toolkit* mas não se podem distribuir. A documentação disponível é boa e são fornecidos alguns exemplos.

O desenvolvimento de treinos de modelos acústicos de forma mais avançada é possível mas

são necessários mais conhecimentos e tempo, comparando com outras *toolkits*.

De forma a facilitar o desenvolvimento de aplicações sobre esta *toolkit*, a *ATK API* [1005], que consiste numa camada situada sobre a biblioteca *HTK*, foi desenhada para permitir que novos programadores possam implementar usando versões customizadas do *HTK* com *ATK* e as possam testar em sistemas de trabalho. Como a *HTK*, a *ATK* é compatível com a maioria dos sistemas *Unix* e *Windows*.

2.5.6. Wit.ai

Wit.ai [30] é uma plataforma *online* que permite aos programadores desenvolverem aplicações e incorporar *Speech Recognition* e *interfaces* de linguagem natural em qualquer aplicação ou componente de *hardware*, sobre vários sistemas como *iOS*, *Android*, *Web*, *Node.js*, *Raspberry Pi*, *Ruby*, *Python*, *c*, *Rust*, *Windows*. Desta forma, os programadores não precisam de se preocupar com os algoritmos do processamento da linguagem natural, configurações, performance e *tuning*. *Wit.ai* foca-se nessas funções e permite que os programadores se concentrem nas suas próprias aplicações. *Wit.ai* permite converter o discurso do utilizador em dados estruturados. A *API Wit* permite chamadas *HTTP Get*, retornando o significado extraído de uma determinada frase com base em exemplos. A autenticação com a *API* é feita com *OAuth 2.0* e devolve as respostas em formato *json*.

Wit.ai está focado na próxima geração de dispositivos, onde cada vez menos existe a utilização de teclados, como por exemplo, *Nest*, *Google Glass*, entre outros dispositivos.

Esta plataforma foi criada em Outubro de 2013. Incorpora vários mecanismos de processamento a correr em paralelo, incluindo o *CMU Sphinx*. Usando um método de aprendizagem, esta plataforma consegue combinar os resultados, aceitar a linguagem em geral e vocabulário especializado. Wit.ai é uma camada virtual sobre todos estes mecanismos que agrupam vários serviços no mesmo sítio. Em Janeiro de 2015 o *Facebook* adquiriu a Wit.ai para ajudar os seus programadores com *Speech Recognition* e interfaces por voz.

A plataforma *Wit* inclui um repositório de projetos desenhado para ser acedido, introduzir e guardar dados relacionados com a interface do seu projeto. Os dados de cada aplicação *Wit* podem incluir "*Expressions*", "*Intents*" e "*Entities*". Uma "*Expression*" é um discurso ou uma frase dita por uma pessoa. Um "*Intent*" é o comando ou a ação desejada que o orador da expressão quer que o dispositivo ou aplicação realize. "*Entity*" é um valor ou uma parte de dados associados com o "*Intent*".

Os termos do serviço desta plataforma estão bem explicados na sua página web, inclusive os termos de licença que é livre mas limitada.

Esta plataforma está bem documentada e contém alguns exemplos de implementação.

2.5.7. Speechmatics

Speechmatics [31] fornece um serviço de *Speech Recognition* baseado em *cloud* e contém uma *interface web* que permite gerir a interação com o sistema. Esta ferramenta suporta várias linguagens que são baseadas nos seus modelos de linguagem inovadores e modelos acústicos.
O reconhecimento de voz tem a capacidade de obter para além das palavras e frases, a pontuação e detetar as palavras que começam com letra maiúscula. O *output* deste conteúdo pode ser feito para ficheiros *HTML, XML* e *PDF*. Esta ferramenta permite também extrair texto a partir de vários formatos áudio como *mp3*, *mp4*, *avi, wav*, entre outros.
Devido ao sistema ser baseado em *cloud*, é possível ter algumas vantagens como baixos custos de instalação, atualizações automáticas e melhorias regulares e também acesso a uma gama cada vez maior de ferramentas e produtos de linguagem. Disponibiliza também um sistema de *pay-as-you-go* altamente flexível que permitirá que se use *Speechmatics* sempre que se precisar e a preços competitivos. *Speechmatics* oferece bom desempenho sobre os seus produtos de linguagem que se estão expandindo continuamente devido ao desenvolvimento de novos recursos.

2.5.8. VoxSigma

VoxSigma [32] é um conjunto de produtos comerciais de *Speech Recognition* em modo *standalone* e *web service*, disponibilizados pela *Vocapia Research* para plataformas *Linux x86* e *x86-64* (*OpenSuse*, *Debian*, *Fedora*, *CentOS*, *Ubuntu*, *SuSE*, *Red Hat*, entre outras).
O *VoxSigma* permite a capacidade de transcrever voz para texto em várias linguagens (incluindo português), mesmo em situações com ruído de fundo. Foi desenvolvido para dar resposta às necessidades profissionais dos utilizadores de forma a transcrever grandes quantidades de informação, tanto em tempo real como em lotes de informação. O processo de conversão de voz para texto é feito em três passos. No primeiro, o *software* identifica os segmentos de áudio que contém vozes, depois é identificado a linguagem falada e por último converte os segmentos de voz para texto num formato *XML*.
O *VoxSigma* permite também *Speech Recognition* através de um *web service* via *REST* sobre *HTTPS* e desta forma permite que os utilizadores facilmente integrem estes serviços nas suas aplicações e que acedam sempre às suas últimas versões.

2.5.9. AUDIMUS

AUDIMUS [33] é uma plataforma para investigação de novas técnicas e desenvolvimento de novas aplicações para *automatic Speech Recognition* na língua portuguesa. Esta plataforma é usada como base na investigação de novas técnicas de diferentes componentes de sistemas de *Speech Recognition*. As melhorias resultantes do trabalho de investigação são integradas na plataforma com o objetivo de ter disponível novas e melhores aplicações.

AUDIMUS foi criado pelo L²F (Laboratório de Sistemas de Língua Falada), que é um laboratório dedicado à área de processamento computacional de linguagem falada para português europeu, e pertence ao *Institute for Systems and Computer Engineering: Research and Development, INESC-ID*, que é uma instituição sem fins lucrativos dedicada à investigação no domínio das tecnologias de informação e comunicação.

Esta plataforma combina as capacidades temporais de modelagem do *Hidden Markov Models* com as capacidades de classificação padrão da *multilayer perceptrons* (*MLPs*). Cada modelo acústico depende das características de *input*, porque para as vozes de telefone, vozes de microfone ou vozes provenientes de notícias, são usados modelos diferentes. O mesmo acontece com os modelos de vocabulário e de linguagem, em que variam conforme o domínio da aplicação.

L²F é um membro institucional da ISCA (*International Speech Communication Association*). O trabalho deste laboratório tem sido reconhecido internacionalmente através de uma cooperação com outros centros de pesquisa na Europa e Estados Unidos. Para além disso, a L²F tem cooperado ativamente com instituições nacionais, tais como Vodafone, Portugal Telecom, Microsoft Portugal, Rádio Televisão Portuguesa (RTP), Porto Editora e Texto Editora.

De forma a disponibilizar um produto acabado a L2F/INESC-ID criou a empresa *VoiceInteraction S.A.* em que o principal objetivo é potenciar os produtos das empresas, integrando as tecnologias de processamento de fala. Esta empresa, através da sua própria capacidade, bem como da forte ligação ao L2F/INESC-ID, consegue realizar projetos de inovação tecnológica que permitem aos seus parceiros e clientes apresentar soluções capazes de assegurar uma maior rentabilidade dos seus próprios produtos. *VoiceInteraction* tem então disponível um sistema de *Speech Recognition* chamado de *AUDIMUS* com capacidade de transformar o som produzido pelo orador numa sequência de palavras e tem como principais características os seguintes pontos:

- Processa áudio proveniente de fontes pré-existentes (ficheiros) ou produzidas em tempo real (*streaming*);

- É capaz de combinar dinamicamente gramáticas livres de contexto (SRGS) com gramáticas estatísticas;
- Suporta adaptação ao orador;
- Tem módulos de pré-processamento acústico que conferem maior robustez ao reconhecimento em condições adversas;
- O sistema de reconhecimento foi portado com sucesso para *PDAs*, sendo capaz de suportar dicionários com mais de 10000 palavras, mantendo o processamento em tempo real;
- Existem versões adaptadas para áreas específicas como várias especialidades de medicina, justiça, media;
- Os resultados têm medidas de confiança associadas para permitir a recuperação de erros;
- É possível ajustar as transcrições fonéticas das palavras do dicionário.

Outro serviço disponibilizado pela *VoiceInteraction* é o *AUDIMUS.SERVER* [34]. Este serviço permite efetuar a transcrição de ficheiros de média de uma forma automática, ou seja permite transformar os seus arquivos audiovisuais em documentos textuais. A sua estrutura possui uma interface de administração e de utilização, podendo ser facilmente integrado com aplicações através de interface *Webservices* (SOAP/WSDL). Este sistema recebe um ficheiro áudio/vídeo, coloca-o numa lista para processamento e notifica o utilizador após a sua conclusão, para que este o possa consultar e utilizar o resultado do reconhecimento. Trabalha com vocabulários superiores a 100.000 palavras e modelos de linguagem treinados com textos de jornais. Nas zonas de fala, faz uma transcrição completa do que foi dito, com a indicação de níveis de confiança no reconhecimento.

2.5.10. Bing Speech API

A *Microsoft* em 2015 lançou o *Project Oxford* que era constituído por um conjunto de *SDKs* e *APIs*. Entre eles o *Emotion API* para o reconhecimento de emoções, *Face API* para a deteção e reconhecimento de faces, o *Speaker Recognition API* para a verificação do locutor e identificação do locutor e o *Speech API* para o processamento de voz.

Mais recentemente o *Project Oxford* foi inserido num novo projeto chamado de *Congnitive Services*, que incorpora entre outros serviços o *Bing Speech API* [35], que permite *Speech Recognition* e *Text to Speech*.

A *Speech API* é uma *cloud-based API* (*Azure*) que permite aos programadores incluírem

facilmente comandos por voz e iterações com as suas aplicações. Esta API é multiplataforma e os seus *web-services* podem ser acedidos através de uma *interface* REST, permitindo que seja utilizado por qualquer dispositivo ligado à internet. Este projeto disponibiliza também *SDKs* que contêm as mesmas chamadas aos *web-services* e possibilita a integração em plataformas *Android*, *iOS*, *Windows* e plataformas ligadas ao *IoT*.

A API de reconhecimento de voz oferece a capacidade de converter voz para texto, enviando o áudio para os servidores da *Microsoft* na *cloud*. É possível aceder a esta *API* por *REST* ou por uma *cliente library* que permite *streaming* por *socket* em tempo real e também permite que, ao mesmo tempo que a voz está a ser enviada para o servidor, uma parte está a ser recebida.

A *Speech API* não tem custos até um certo limite de utilização, reconhece a língua portuguesa, e a sua documentação é boa. Tem também disponível muitos exemplos criados pela *Microsoft* e pela comunidade.

2.5.11. Annyang

Annyang [36] é uma ferramenta para *Speech Recognition* composta por uma biblioteca *javascript* que pode ser integrada em aplicações *web* para que os utilizadores as possam controlar com comandos por voz. Esta ferramenta suporta várias linguagens (incluindo português), não tem dependências, é livre de usar e funciona bem com todos os *browsers*. Tem disponível alguma documentação com instruções necessárias para a integração numa aplicação *web*.

2.5.12. MDN Web Speech API

MDN (*Mozilla Developer Network*) [37] tem disponível a *Web Speech API* que permite integrar dados de voz nas aplicações *web*. Esta *API* tem duas *components*, a *SpeechSynthesis* para *Text To Speech* e a *SpeechRecognition* para *Asynchronous Speech Recognition*.

Speech Recogition está acessível através de uma interface, que oferece a capacidade de reconhecimento de voz, a partir de uma entrada de áudio, e depois devolve uma resposta apropriada. A gramática é configurada a partir da interface *SpeechGrammar* que contém um conjunto de gramática para a aplicação reconhecer. A gramática é definida usando o formato *JSpeech Grammar Format (JSGF)*.

Esta *API* está em fase de testes, não tendo ainda numa versão estabilizada e não está disponível para todos os *browsers* estando sujeita a várias alterações em futuras versões.

Dispõem de boa documentação.

2.5.13. Python SpeechRecognition

Python SpeechRecognition [38] é uma biblioteca que permite o reconhecimento de voz com suporte para *Google Speech Recognition*, *Wit.ai* e *IBM Speech to Text*. Esta biblioteca pode ser usada em sistemas *Windows*, *Linux* e *OS X*. Pode ser necessário instalar alguns componentes que dependem do que se pretende instalar e do sistema onde se vai instalar.

Esta biblioteca está disponível segundo a licença *3-clause BSD* e para alguns componentes usados, devem ser respeitadas apenas algumas notas de *copyright*. A sua documentação é razoável.

2.5.14. IBM Watson Speech to Text

O serviço *IBM Watson Speech to Text* [39] pode ser utilizado em qualquer sistema onde seja necessário a interação por voz. Ele disponibiliza as capacidades de reconhecimento de voz da IBM para a transcrição de voz em texto. Este serviço utiliza inteligência computacional para combinar informação sobre a gramática e a estrutura da linguagem com conhecimento da composição do sinal de áudio de forma a gerar uma transcrição precisa. Permite aos programadores, através das suas *APIs*, implementarem aplicações com a funcionalidade de *Speech Recognition*.

A transcrição do discurso é contínua e é devolvida ao cliente com o mínimo atraso, para além disso tem a capacidade de reconhecer *keywords* durante o *streaming* do áudio.

Este serviço pode ser acedido por *WebSocket* ou por *REST API*, pode fazer a transcrição a partir de vários formatos de ficheiros áudio e suporta várias línguas, incluindo português do Brasil.

O uso deste serviço não tem custos durante os primeiros 1000 minutos de cada mês, depois é acrescentado um valor taxado ao minuto.

A página *IBM Watson Developer Cloud* tem disponível documentação completa, acompanhada de demos, um fórum, entre outros conteúdos.

2.5.15. iSpeech API

iSpeech API [40] permite aos programadores implementarem um sistema automatizado de reconhecimento de voz em qualquer aplicação com acesso à internet. Esta *API* é independente

de qualquer plataforma, o que significa que, qualquer dispositivo que possa captar ou reproduzir áudio e esteja ligado à internet a possa usar.

O protocolo usado segue o *HTTP standard* usando *GET* e *POST* e os pedidos podem ser do tipo *URL-Encoding*, *JSON* ou *XML*.

Com o *ASR* desta *API* pode-se converter áudio em texto usando uma de várias linguagens (incluindo português) e modelos de reconhecimento. É possível também criar modelos de reconhecimento para melhorar a qualidade do reconhecimento.

iSpeech API está disponível para os sistemas *iPhone*, *Android*, *BlackBerry*, *.NET*, *Java* (*Server*), *PHP*, *Flash*, *Javascript/Flash*, *Ruby*, *Python* e *Perl*. Para os sistemas *mobile* o *iSpeech* tem disponíveis vários *SDKs*.

Esta *API* tem custos de utilização. Apenas para as aplicações móveis que sejam distribuídas de forma sem custos é que a utilização é gratuita. De referir também a boa documentação desta *API*.

2.5.16. API.AI

O objetivo da *Api.ai* [41] é fazer com que o processo de criar e integrar interfaces de voz seja o mais simples possível. Esta plataforma permite que os programadores possam integrar sistemas inteligentes de comandos de voz nos seus produtos de forma a criar interfaces por voz fáceis de usar.

Esta plataforma tem três componentes disponíveis, *Speech Recognition*, *natural language understanding* e *user-fulfillment*. Estes três componentes permitem que, para além do reconhecimento de voz, seja feito uma gestão da comunicação com perguntas e respostas por ambas as partes (utilizador/aplicação ou dispositivos).

Esta plataforma disponibiliza uma ferramenta que funciona como uma espécie de *backoffice*. Nessa ferramenta é possível criar um agente que vai estar relacionado com a aplicação ou dispositivo a que se pretende integrar uma interface por voz. É nesse agente que se seleciona a linguagem de comunicação. Depois é possível criar *Entities*, que são objetos muitas vezes específicos a um domínio como um meio de mapear frases de linguagem natural para frases canónicas que capturam o seu significado. Podem ser criados também *Intents* em que o objetivo é mapear todos os tipos de solicitações dos utilizadores para uma ação. É nesta ferramenta que se testa e treina o Agente, que depois de criado se pode integrar com uma aplicação ou dispositivo usando um dos vários *SDKs* disponíveis para as mais populares plataformas e tecnologias, tais como, *Android*, *HTML*, *C#.NET*, *C++, iOS, Mac OSX* e *Apple Watch, JavaScript, Unity, Python, Cordova, Node.js, Xamarin* e *Ruby*.

De referir que esta *Api.ai* disponibiliza uma boa e organizada documentação. Em relação aos preços, esta plataforma apresenta várias modalidades em que a mais básica é grátis.

2.5.17. LumenVox ASR

A *LumenVox* [42] já ganhou vários prémios de inovação e excelência técnica. A sua tecnologia em sistemas de voz é certificada como uma das mais precisas, naturais e confiáveis. Dispõem de soluções de voz com bastante sucesso para *Speech Recognizing*, *Text To Speech Server*, *Speech Tuner* e, para além disso, dispõem de *SDKs* que permitem ser integradas noutras aplicações.

O *Automatic Speech Recognizer (ASR) SDK* contém um conjunto de pacotes de *software*, testes, suporte e licenças que permitem a qualquer programador criar aplicações de forma rápida e eficiente.

A *LumenVox Speech Engine* permite o reconhecimento de voz por *streaming* de áudio, pode realizar reconhecimento de voz a partir de dados em formato áudio, proveniente de qualquer fonte, suporta várias línguas, incluindo português do Brasil, tem uma arquitetura cliente/servidor com o carregamento do processamento de voz distribuído, permite definir gramáticas em *runtime* inseridas como simples texto, e funciona em plataformas *Linux* e *Windows*.

A *LumenVox ASR SDK* tem disponíveis três licenças para o *LumenVox Speech Engine (ASR)*. São licenças permanentes com determinado custo e podem ser usadas para desenvolvimento, testes e produção. Outros pacotes de serviços são vendidos juntamente com as licenças.

2.5.18. EduSpeak Speech Recognition Toolkit

EduSpeak é uma *toolkit* [43] para reconhecimento de voz, produzido pela *SRI International*, e especialmente desenhada para programadores de aplicações de aprendizagem de línguas ou outras aplicações educacionais.

Cada vez mais se usa material interativo, através de um computador para ensinar e treinar os alunos de todas as idades. Com reconhecimento de voz, é possível melhorar essa interação entre o aluno e o computador e, dessa forma, obter bons resultados na aprendizagem. Contudo os sistemas *Speech Recognition* de uso geral não têm uma ampla gama de *interfaces* para desenvolvimentos multimédia essenciais para o uso em diversas plataformas educacionais. Ao contrário desses sistemas de uso geral, a *EduSpeak* trabalha com *Flash*, *Java*, *ActiveX*, *C/C++* e *Java Script* e comtempla a patente de *human-calibrated pronunciation-scoring technology*.

Reconhece vozes de crianças e adultos e suporta algumas línguas, mas não suporta português. Pode ser instalado localmente ou pode funcionar em modo cliente/servidor ou através de serviços *cloud*.

As pesquisas e desenvolvimentos realizados pela *SRI International* deram origem a algumas oportunidades comerciais entre as quais a *Nuance Communications* e o *Siri*.

2.5.19. Julius

Julius [44] é um *software* descodificador *Large Vocabulary Continuous Speech Recognition (LVCSR)* para investigadores e programadores que trabalhem neste âmbito.

Baseado no *word N-gram* e dependente do contexto *HMM*, *Julius* pode descodificar em tempo real em vários computadores ou dispositivos desde microcomputadores até *cloud servers*.

Julius pode executar reconhecimento multi-instância, executando discurso, reconhecimento baseado em gramática ou reconhecimento de palavras isoladas em simultâneo numa *thread* isolada. Foram adotados formatos *standard* para os modelos de forma a lidar com outras *toolkits* de modelagem de linguagens tais como *HTK, SRILM (SRI Language Modeling)*. As versões recentes já suportam *Deep Neural Network (DNN)*, baseadas em descodificação em tempo real.

Julius funciona nas plataformas *Linux* e outras plataformas *Unix*, *Windows*, *Mac*, *Android*, entre outras. É *open source*, tem tido *releases* recentes e a sua documentação é boa.

2.5.20. Pocketsphinx.js

Pocketsphinx.js [45] é uma *library open source* de reconhecimento de voz escrita em *javascript* e que funciona simplesmente nos *webs browsers*. Não necessita de *flash* ou de qualquer tipo de *plug-in* no *browser* e não necessita de qualquer tipo de processamento do lado de um servidor. Esta *library* faz uso do *Emscripten* (é um compilador de *LLVM* para *javascript*), para converter *PocketSphinx* em *javascript*. O áudio é gravado com o *getUserMedia JavaScript API* e processado através do *Web Audio API*.

As características do *pocketsphinx.js* são semelhantes às do *Pocketsphinx,* no entanto, existem algumas especificações relacionadas com o ambiente *browser*. Funciona apenas nos *browsers Chrome* e *Firefox.*

2.5.21. Ceedvocal

CeedVocal [46] é um *SDK* de *Speech Recognition* para sistemas *iOS* que foi implementado com base em bibliotecas *open source* como a *Julios* e o *FLite*. *CeedVocal* é compatível com as versões *iOS 5* e superiores e em alguns casos específicos funciona também em versões inferiores à 5, tanto para arquiteturas *32-bit* como *64-bit*.

Funciona diretamente no dispositivo sem necessidade a ligação à internet com seis linguagens disponíveis e treinadas. Consegue fazer o reconhecimento de palavras isoladas a partir de uma lista pré-definida pelo utilizador, mas não consegue tratar de ditados contínuos como o *Siri*. Trabalha com qualquer palavra, mas funciona melhor com palavras maiores ou frases mais longas.

De forma a integrar o *CeedVocal* numa aplicação deve ser feito, antes, um licenciamento. Existe uma versão trial de avaliação e testes que é livre. Para uso comercial existem outras licenças mas com custos associados.

2.5.22. BitVoicer Server

BitVoicer Server [47] é um servidor de reconhecimento e síntese de fala para automação por voz. O seu desenvolvimento teve em conta a capacidade de permitir que dispositivos simples, com pequeno poder de processamento, possam ser operados através de comandos de voz.

Na generalidade, os microcontroladores não possuem poder de processamento e memória suficientes para realizar reconhecimento e síntese de fala avançados. Com o *BitVoicer Server* essas limitações são postas de parte porque este sistema realiza o trabalho pesado. Dessa forma, o microcontrolador pode destinar a maioria de seus recursos à sua funcionalidade principal.

Numa solução de automação do *BitVoicer Server*, os microcontroladores podem funcionar basicamente de três formas: como dispositivos de entrada, como dispositivos de saída, ou como dispositivos mistos. Dispositivos de entrada são aqueles com a capacidade de capturar, digitalizar e enviar fluxos de áudio para o servidor. Quando o *BitVoicer Server* identifica um dispositivo de entrada, ele aloca um Motor de Reconhecimento de Fala (*Speech Recognition Engine - SRE*), exclusivo para este dispositivo. Os *SREs* analisam constantemente os fluxos de áudio enviados ao servidor e quando uma frase, previamente definida, é identificada, o *BitVoicer Server* realiza as ações definidas pelo utilizador. Estas ações são chamadas de comandos no *BitVoicer Server* e podem ser usadas para executar outro programa no servidor, sintetizar fala, reproduzir um ficheiro de áudio ou enviar dados a dispositivos de saída ou

mistos. O utilizador pode definir um ou vários comandos para cada frase. Também é possível definir a ordem em que os comandos são executados, o intervalo de tempo entre eles e quais os dispositivos de saída ou mistos serão alvos dos comandos, ou seja, com apenas uma licença de dispositivo de entrada é possível controlar diversos dispositivos de saída. E por fim, há os dispositivos mistos que são aqueles capazes de atuar como dispositivo de entrada e saída ao mesmo tempo.

Embora o *BitVoicer Server* tenha sido desenvolvido principalmente para o controlo por voz de dispositivos eletrónicos, também é possível utilizá-lo para o controlo de aplicações. Isso é possível porque o *BitVoicer Server* pode utilizar o adaptador de áudio do servidor para capturar e reproduzir áudio. Além disso, o *BitVoicer Server* expõe serviços *Windows Communication Foundation (WCF)* para comunicação com aplicativos externos. Os serviços *WCF* expostos pelo *BitVoicer Server* utilizam especificações *Web Service* que permitem a integração com diversas plataformas de desenvolvimento, como por exemplo, *Java*, *Delphi*, entre outros. No caso de desenvolvimento *.NET* há ainda a opção de utilizar a biblioteca de integração descrita na documentação do produto.

Dispositivos clientes podem comunicar com o *BitVoicer Server* através de duas interfaces de comunicação, a *serial* e *TCP/IP*. Também são suportadas portas virtuais idênticas às criadas por adaptadores *Bluetooth* ou *USB/Serial*. Isso permite que o *BitVoicer Server* consiga enviar comandos para dispositivos que utilizam outros protocolos de comunicação.

Este serviço suporta 17 idiomas de 26 países ou regiões, tanto para reconhecimento como para sintetização de voz, inclui português de Portugal, tem a capacidade de gerir *inputs* e *outputs* de vários dispositivos em simultâneo, pode ser integrado em aplicações por *webservice* ou por biblioteca de integração para *.Net* e os motores de reconhecimento não necessitam de treino.

O uso do *BitVoicer Server* tem custos e são contabilizados pelo número de *Input Devices*, dispositivos que capturam e enviam áudio para o *software*. De referir a boa documentação e boa organização da página *web*.

2.5.23. Google Cloud Speech API

A *Google Cloud Speech API* [48] permite aos programadores integrarem nas suas aplicações a capacidade de converter áudio em texto, através de modelos de redes neurais. Reconhece mais de 80 línguas e variações diferentes, de forma a dar resposta ao máximo de utilizadores possível. As suas funções permitem transformar voz em texto e operações de controlo e comando através da voz.

O texto é transcrito em tempo real, os resultados do reconhecimento de voz são devolvidos de

forma parcial assim que estiverem disponíveis. O reconhecimento de texto pode também ser devolvido a partir de um ficheiro de áudio.

Esta *API* tem a capacidade de transcrever a voz em texto, mesmo em ambientes com ruído e aplica algoritmos de aprendizagem neuronal de forma a obter uma alta taxa de precisão, que vai aumentado à medida que novos termos vão sendo introduzidos.

A *Google Cloud Speech API* disponibiliza vários exemplos e documentação e, o seu uso é grátis durante os primeiros 60 minutos de cada mês.

A seguinte tabela apresenta um resumo com algumas das características dos sistemas *ASR* descritos neste subcapítulo.

	Nome	Open Source	Acesso	Plataforma	Tipo	Custos	Português
CMUSphinx	Pocketsphinx	Sim	Library	Linux; Windows; Mac OS; Android	Standalone	Não	Não
	Sphinx4	Sim	Library	Aplicações java	Standalone	Não	Não
Nuance	Dragon NaturallySpeaking SDK Client Edition	Não	SDK	Windows	-	-	Não
	Dragon NaturallySpeaking SDK Server Edition	Não	SDK	Windows	-	-	Não
	Nuance Mobile Developer SDK	Não	SDK; HTTP REST Interface	Android; iOS; Windows Phone; HTTP interfaces	Cloud-based	Sim	Sim
Kaldi	Kaldi	Sim	Toolkit	Windows; Darwin; Cygwin; Linux	Standalone	Não	-
OpenEars	OpenEars	Não	Framework	Sistemas iOS	Standalone	Não	Não
HTK	HTK	Sim	Toolkit	Linux; Solaris; IRIX; HPUX; MacOSX; FreeBSD; Windows	Standalone	Não	-
Wit.ai	HTTP API	Não	REST API	Várias	Cloud-based	Sim	Sim
Speechmatics	Speechmatics	Não	REST API	Várias	Cloud-based	Não	Sim
L²F	AUDIMUS	Não	-	-	-	-	-
	AUDIMUS.SERVER	Não	Webservices	Várias	Cloud-based	-	-
Microsoft	Bing Speech API	Não	Web services; SDKs	Várias	Cloud-based	Sim	Sim
Annyang	Annyang	-	Library	Web applications	Cloud-based	Não	Sim
Mozilla Developer Network	SpeechRecognition	-	Library	Web applications	Cloud-based	-	-
IBM	Watson Speech to Text	Não	WebSocket; REST API	Várias	Cloud-based	Sim	do Brasil
iSpeech API	iSpeech API	Não	HTTP Get/Post	Várias	Cloud-based	Sim (excepto aplicações móveis)	Sim
API.AI	API.AI	Não	SDKs; REST	Várias	Cloud-based	Sim	Sim
Mindmeld	Mindmeld	Não	SDKs	Web applications; iOS; Android	Cloud-based; Standalone	-	Sim
LumenVox	Automated Speech Recognizer (ASR)	Não	SDKs	Linux; Windows	-	Sim	do Brasil
EduSpeak	EduSpeak Speech Recognition Toolkit	Não	Toolkit	Windows XP or later; Mac OS X 10.4 or later; Linux; Android	Cloud-based; Standalone; Client-server	Sim	Não
Pocketsphinx.js	Pocketsphinx.js	Sim	Library	Chrome; Firefox	-	Não	-
BitVoicer Server	BitVoicer Server	Não	Package	Microsoft	Standalone; Client-server	sim	sim
Google	Google Cloud Speech API	Não	Rest	Várias	Cloud-based	Sim	Sim

Tabela 1. Resumo dos sistemas ASR.

Sistemas de sintetização

A área da geração de voz tem sido uma das áreas com maiores dificuldades em obter um grau viável de sucesso. Em mais de 50 anos de investigação os investigadores têm tentado imitar os processos físicos de geração da fala, através de modelos articulatórios do trato vocal humano, ou através de modelos de síntese analógica, terminais das propriedades espectrais e propriedades temporais da fala [49].

Nos dias de hoje, os sistemas de sintetização de voz mais evoluídos já não têm aquele som mecânico e robótico, mas ainda falta algum trabalho para que a voz sintetizada seja indistinguível da voz humana. Uma das necessidades existentes tem a ver com a expressividade da voz [49], em que o sintetizador, para além de sintetizar a voz, deve também transmitir a emoção associada ao contexto.

Na síntese de voz, o objetivo é obter voz sintetizada para que seja o mais natural possível e idêntica à voz humana. Duas características são requeridas, a qualidade e a inteligibilidade [50].

A tecnologia *TTS* é amplamente aplicada em vários sistemas e soluções. Alguns exemplos são os sistemas de navegação ou orientação, como por exemplo sistemas *GPS* aplicados nas viaturas, para que, as orientações visuais sejam acompanhadas por indicações por voz, resultando em maior segurança na condução; Sistemas *e-learning*, onde a informação visual é acompanhada por voz, podendo ser personalizada com parâmetros como a língua, locutor, entre outros, tornando a aprendizagem mais agradável, mais eficaz e que resulta num aumento de motivação; sistemas de assistência virtual; sistemas de conversação Homem-máquina; sistemas de leitura de mensagens, *emails*, *sms*, páginas *web*, listas telefónicas; quiosques digitais com informações turísticas ou de localização; comunicações em estações de metro, comboio; entre muitos outros.

A sua aplicação é variada e é também muito útil, podendo ajudar pessoas com incapacidades visuais e que tenham a impossibilidade de ler. Pode também ajudar pessoas que padeçam de algum tipo de paralisia que as impeça de falar e comunicar. Um caso muito famoso de utilização de um sistema *TTS* é o do físico *Stephen Hawking*. *Stephen Hawking* tem uma rara doença degenerativa que paralisa os músculos do corpo sem, no entanto, atingir as funções cerebrais, desde então, utiliza um sintetizador de voz para comunicar.

Um sistema de *TTS* pode ser dividido em duas fases principais, a fase de análise do texto e prosódica e a fase de sintetização [49]. Na fase de análise do texto e prosódica, um texto de qualquer tamanho dá entrada e é segmentado em partes usando um algoritmo de forma a tornar o processo mais fácil. Para cada parte o conteúdo é dividido numa sequência de *tokens*, com base na presença de espaços, pontuação e outros fatores. Esses *tokens* podem ser constituídos apenas por uma palavra, números, datas, ou outros tipos. Depois é encontrada uma classe semiótica [49] para cada *token*. Para *tokens* de linguagem não-natural, é usado um sistema diferente para cada tipo, em que o texto descodificado é posto num *form*. Em seguida, usando regras, o *form* é traduzido num *form* de linguagem natural com palavras. Para *tokens* de linguagem natural tenta-se resolver alguma ambiguidade e encontrar as palavras, depois tenta-se fazer uma análise prosódica básica do texto. Como muita da informação pode não estar no texto, são usados algoritmos para determinar frases, padrões relevantes e entoação do discurso.

Depois, na fase de sintetização, as palavras encontradas na fase anterior são convertidas em fonemas, de forma a criar uma representação mais compacta para futuros processos de síntese. Palavras, fonemas e frases formam especificações de entrada para o módulo de seleção de unidades. A sintetização é realizada através do acesso a vozes guardadas em base de dados, de forma a encontrar unidades que sejam o mais possível compatíveis com as especificações de entrada. As vozes guardadas em base de dados, são no fundo, fragmentos de *waveforms* e quando uma sequência é escolhida o processamento de sinal é usado para as unir de forma a gerar um conteúdo de saída contínua de *waveforms* em forma de voz.

Para a fase de sintetização existem várias técnicas ou abordagens. Youcef Tabet e Mohamed Boughazi [50] identificam três abordagens principais para a síntese de voz, são elas a *Formant Synthesis, Articulatory Synthesis*, e *Concatenative Synthesis*.

Para Paul Taylor [49] existem três gerações de técnicas de sintetização. A primeira geração é menos utilizada nos dias de hoje e utiliza regras explícitas escritas à mão, fazendo parte desta geração as abordagens *Formant Synthesis*, *Classical Linear Prediction Synthesis* e *Articulatory Synthesis*. A segunda geração utiliza um método orientado a dados para gerar o conteúdo verbal do sinal, e dela fazem parte as abordagens *Concatenative Speech Synthesis*. A terceira abordagem é estatística e dedicada a dados, é composta por abordagens como *Hidden Markov Models* e *Unit Selection Synthesis*.

A abordagem *Formant Synthesis* [50] é baseada em regras e modela a frequência do sinal de

voz.

A *Classical Linear Prediction Synthesis* adota o modelo de trato vocal *all-pole* e a *source* é modelada por dois sistemas, um para os sons da voz e outro para os sons obstruentes [49].

Articulatory Synthesis [50] é baseada em regras e permite a geração de voz através da modelagem direta do comportamento articulador humano que é definido pelos seguintes parâmetros: abertura labial, protrusão labial, a posição da ponta da língua, altura da ponta da língua, posição da língua e altura da língua.

A primeira geração de técnicas de sintetização são menos usadas nos dias de hoje, contudo podem ser vantajosas em aplicações que necessitem de menos memória e pouco processamento. As vozes criadas tem uma pequena pegada do sintetizador [51].

A limitação principal das abordagens anteriores não incide na geração da fala a partir de uma representação paramétrica, mas sim em encontrar esses parâmetros a partir das especificações de entrada que foram criadas pelo processo de análise de texto. Para superar essa limitação, a síntese *Concatenative Speech* segue uma abordagem orientada a dados.

Concatenative Speech Synthesis [50] produz voz, através da concatenação de pequenas unidades de voz pré-gravadas, como palavras, sílabas, meias-sílabas, fonemas, *diphones* (par de sons fonéticos que são adjacentes uns aos outros), *triphones* (três sons fonéticos que são adjacentes uns aos outros).

A *Unit Selection Synthesis* combinada com *Harmonic Plus Noise Model (HNM)* permite modificações mais naturais do sinal [50]. A representação paramétrica da voz usando *HNM* fornece uma forma fácil de suavizar descontinuidades de unidades acústicas em redor de pontos de concatenação. As principais limitações da sintetização de seleção de unidades combinada com *HNM* são os custos de processamento elevados e são as poucas permissões para variações nos dados gravados.

Na *Unit Selection Synthesis*, múltiplas instâncias de cada fonema para cada contexto são guardadas em base de dados. Construir essa base de dados é uma tarefa demorada e o tamanho da base de dados aumenta exageradamente. Uma alternativa é a utilização de técnicas de síntese de estatísticas paramétricas para inferir a especificação do mapeamento de dados paramétricos. Estas técnicas tem duas vantagens, na primeira o uso de memória é menor para guardar os parâmetros dos modelos do que para armazenar os próprios dados. Na segunda vantagem são permitidas mais variações, por exemplo, a voz original pode ser convertida numa outra voz. Uma das técnicas de síntese de estatísticas paramétricas mais

usadas é a *Hidden Markov Model (HMM) Synthesis* [50].

A qualidade da voz gerada por *HMM* não é tão boa como a qualidade da voz gerada pelo *Unit Selection Synthesis*, mas a integração do *Harmonic plus Noise Model (HNM)* num sistema baseado no *Hidden Markov Model Speech Synthesis System (HTS)*, leva a um sistema de *TTS* com necessidade de menor desenvolvimento, tempo e custos, comparando com alguns sistemas típicos, como os baseados em seleção automática [50].

Os sistemas *TTS* modernos chegaram a um nível em que é possível a sua utilização de voz sintetizada em várias aplicações do dia-a-dia. Contudo, ainda continuam a existir algumas deficiências que prejudicam a qualidade da voz gerada [52].
De seguida são apresentados vários sistemas *TTS* que podem ser usados para estudo, investigação ou até mesmo em projetos finais. Existem algumas características que os destingem como por exemplo o uso de *Speech Synthesis Markup Language (SSML)*. O *SSML* [53] é uma linguagem de marcação, recomendada pela *W3C*, baseada em *XML* para auxiliar a geração de voz sintética em páginas *web* e noutras aplicações. A sua versão atual é a 1.1. *W3C* também recomenda a utilização do *Pronunciation Lexicon Specification (PLS)* versão 1.0. *PLS* [54] permite definir uma sintaxe de forma a especificar a pronúncia dos léxicos a serem usados por sistemas *TTS* e também por *ASR* em aplicações *web*. Outras características são utilizadas, como, o tipo de sintetização, as linguagens suportadas, as plataformas suportadas, as formas de integração disponíveis, entre outras.
A comunidade *W3C* introduziu em 2012 o *Web Speech API Specification* com o objetivo dos *browsers* mais modernos terem a capacidade de reconhecer e sintetizar voz, contudo, até à data de Julho de 2015, apenas no *Google Chrome* era possível funcionar o sintetizador de voz desta *API* [55]. Esta *API,* desenvolvida em *javascript,* permite que os programadores *web* possam integrar nas suas páginas *web scripts*, de forma a poderem sintetizar voz. O sintetizador de voz pode ser *server-based* ou *cliente-based*. Com esta funcionalidade é possível a reprodução de texto presente na página, efetuar questões ou, simplesmente, apresentar mensagens.

2.6.1. Nuance

Nuance [26] dispõe de vários produtos comerciais de reconhecimento de voz e *TTS* dedicados a várias áreas de negócio. Para além disso tem disponíveis alguns *SDKs* para integrar com outros sistemas. São eles o *Dragon NaturallySpeaking SDK Client Edition* e o *Nuance Mobile*

Developer SDK. O *SDK Dragon NaturallySpeaking SDK Client Edition* pode ser integrado com qualquer aplicação *Windows* de forma a obter as capacidades de *TTS*.

Nuance tem também disponíveis *SDKs* para integrar com aplicações móveis que funcionem sobre plataformas *Android*, *iOS* e *Windows Phone*. Estes *SDKs* pertencem a um grupo chamado de *Nuance Mobile Developer SDK* e permitem integrar os serviços de *TTS* que estão disponíveis através da sua *cloud*.

Disponibiliza também *webservices* que permitem o acesso aos seus serviços de *TTS*.

2.6.2. OpenEars

OpenEars [28] é um S*peech Recognition* e *Speech Synthesis open source* para *iPhones*, *iPads* e *iPods*, (sistemas *iOS*), que permite o reconhecimento e sintetização de voz em modo *offline* de forma rápida e simples.

OpenEars tem disponíveis vários *plugins*, um dos quais é o *plugin NeatSpeech* dedicado ao *Speech Synthesis*. Permite a reprodução de voz de forma contínua e praticamente sem atrasos, mesmo no caso de os discursos serem extremamente longos. Também não é necessária qualquer ligação à rede.

Algumas características mais específicas do *OpenEars* em modo *Speech Synthesis* são, a linguagem utilizada para *TTS* é apenas o inglês, mas caso se use o *plugin NeatSpeech* pode também ser feito para espanhol; Usa poucos recursos no processo de *Speech Synthesis*; Tem nove vozes masculinas e femininas de boa qualidade para *Speech Synthesis* e permite a mudança do orador na hora; Permite alterar o tom e velocidade de qualquer voz produzida pelo *TTS*; Consegue detetar se os *headphones* estão ligados e continuar o *Speech Recognition* durante o *TTS* apenas quando eles estão ligados; Suporta dispositivos de áudio por *bluetooth*;

Descritos no capítulo anterior, esta *framework* tem disponíveis vários *plugins* que permitem mais funcionalidades e otimização de desempenho para além disso, dispõem de uma boa documentação;

2.6.3. DIXI

O sistema *DIXI* [56] é um sintetizador para português Europeu desenvolvido pelo *L²F* no âmbito da cooperação entre a *Speech Processing Group of INESC* e a *Phonetic and Phonology Group of CLUL*. A versão atual segue o modelo de *Klatt's* e tem um modelo de regras linguísticas multilinguagem. O objetivo deste projeto é o desenvolvimento de um sintetizador de voz a fim de melhorar a sua qualidade e de ser usado em diversas aplicações.

De forma a disponibilizar este produto comercialmente a *L2F/INESC-ID* criou a empresa *VoiceInteraction S.A*. que apresenta o sistema de síntese de fala *DIXI*. Este sistema permite a utilização de sistemas de síntese de fala para reproduzir um texto escrito através de fala para várias línguas.

Algumas características deste sintetizador: Usa técnicas de concatenação de unidades de comprimento variável; É compatível com o *standard SSML* (*Speech Synthesis Markup Language*); É capaz de personalizar a voz gerada consoante as necessidades específicas (*branding*); É capaz de reaproveitar material previamente gravado para gerar novas vozes; Tem a possibilidade de adicionar novas pronúncias para todas as palavras; Permite a normalização de palavras não *standard* como números, siglas, acrónimos, expressões matemáticas, grandezas físicas entre outras, que podem também ser especificadas pelos utilizadores.

Outro serviço disponibilizado é o *DIXI.SERVER* [57]. Este serviço permite através de uma *interface web* de utilização/administração ou através de integração com *webservices*, disponibilizar todas as características do *DIXI*, mas de uma forma cliente-servidor. O modo de operação do *DIXI.SERVER* permite uma otimização de recursos, na medida em que apenas são usados a memória e processador do servidor e não dos restantes computadores, evitando assim a replicação de dados e facilitando o seu uso. O sistema recebe o texto ou ficheiro de texto a sintetizar, efetua essa operação e, em seguida, notifica o utilizador do término da tarefa. O *DIXI.SERVER* pode ser aplicado na leitura de páginas *web*, na criação de áudio-livros, na criação de interações para diálogos específicos, nas áreas da medicina nos blocos de atendimento e filas de espera, serviços judiciais, entre outras e variadas possibilidades.

2.6.4. Bing Speech API

Como já foi referido no capítulo anterior, o *Bing Speech API* [35] é constituído por uma *API* para *Speech Recognition* e outra para *Text To Speech*. A *API Text To Speech* tem a capacidade de converter texto em voz. Quando as aplicações precisam de "falar" para os seus utilizadores, esta *API* pode ser usada para converter texto gerado pela aplicação em áudio que, depois, é reproduzido ao utilizador. O acesso ao serviço de *TTS* é feito através de uma *API REST*.

A *Speech API* não tem custos até um certo limite de utilização, a sua documentação é boa e a *Microsoft* disponibiliza alguns exemplos práticos.

2.6.5. MDN Web Speech API

MDN (*Mozilla Developer Network*) [37] tem disponível a *Web Speech API* que permite integrar dados de voz nas aplicações *web*. Esta *API* tem duas *components*, a *SpeechSynthesis* para *TTS*, e *SpeechRecognition* para ASR.

Em relação ao sintetizador de voz, esta *API* tem disponível a interface *Speechsynthesis* que permite as aplicações reproduzirem conteúdos por voz. Através da interface *SpeechSynthesisVoice* é possível representar a voz que o sistema suporta e a interface *SpeechSynthesisUtterance* permite representar as diferentes partes do texto que se pretendem ser reproduzidas.

2.6.6. IBM Watson Text to Speech

Watson Text to Speech é um serviço da *IBM* [58] que fornece uma interface *REST API* que permite sintetizar voz a partir de texto. A sintetização pode ser feita por vozes femininas e masculinas e para diferentes línguas (incluindo português do Brasil). Quando a sintetização é feita em tempo real, o áudio é devolvido com o mínimo de atraso e é possível controlar a pronúncia para palavras específicas.

Este serviço é gratuito para o primeiro milhão de caracteres em cada mês, depois é adicionada uma taxa por cada mil caracteres.

2.6.7. iSpeech API

Dando seguimento à informação descrita no capítulo anterior, esta *API* [40] permite aos programadores implementarem um sistema sintetizador de voz *TTS* em qualquer aplicação com acesso à internet.

Com esta *API* é possível sintetizar voz em várias linguagens (incluindo português), e géneros (feminino e masculino), formatos, *bitrates*, frequências e velocidade de reprodução. Também são suportadas as linguagens *Math markup language* (*MathML*) e *Speech Synthesis Sarkup Language* (*SSML*).

2.6.8. API.AI

Com base no que já foi escrito no capítulo anterior sobre esta plataforma, para além de todas as características apresentadas a *Api.ai* [41] tem também uma componente de *Text To Speech* que funciona por *HTTP REST*. Esta componente permite gerar voz a partir de texto.

2.6.9. Festival Speech Synthesis System

Festival Speech Synthesis System [59] dispõe de uma *framework* que possibilita a implementação de sistemas com capacidade de sintetização de voz através de algumas *APIs*. Esta *framework* inclui exemplos de vários módulos e suporta várias línguas.

Festival Speech Synthesis System foi implementada a pensar em, pelo menos, três tipos de utilizadores. Para utilizadores que simplesmente pretendem vozes de alta qualidade a partir de vários textos com o mínimo de esforço, para os utilizadores que estão a desenvolver sistemas de conversação e pretendem adicionar um sintetizador de voz. Por último, para utilizadores que pretendem desenvolver e testar novos métodos de sintetização.

Este sistema foi desenvolvido em *C++*, tem tido atualizações recentes, a sua licença é livre e permite ser usada para fins comerciais.

2.6.10. Flite (festival-lite)

Flite (*festival-lite*) [60] é um pequeno e rápido sintetizador *Text To Speech* desenvolvido pela *Carnegie Mellon University* e foi principalmente desenhado para sistemas dedicados e/ou grandes servidores. *Flite* suporta multi-línguas e faz parte de um conjunto de ferramentas sintetizadoras livres onde estão incluídas *Edinburgh University's Festival Speech Synthesis System* e *Carnegie Mellon University's FestVox project*, que fornece ferramentas, *scripts* e documentação para a construção de novos sintetizadores.

Flite foi criado para resolver alguns problemas do *Festival Speech Synthesis System*. Uma das primeiras queixas tinha a ver com o facto de a *framework* ser grande e lenta. Para além disso, consome muitos recursos e pode ter problemas de gestão de *threads*. Devido a ser desenvolvida em *C++* esta *framework* não é tão portátil como a *Flite*.

Flite é uma biblioteca composta por um conjunto de ficheiros implementados em *ANSI C* e foi desenhada para ser suportada por quase todas as plataformas, incluindo *hardware* de tamanho reduzido e pode ser integrada noutras aplicações. *Flite* não pretende ser uma plataforma de pesquisa e desenvolvimento para a síntese de voz. *Festival* é e continuará a ser a melhor plataforma para isso.

Este *software* é livre e é distribuído sobre a licença *X11-like*. A sua documentação não é muito completa.

2.6.11. LumenVox TTS

Dando seguimento ao que já foi referido sobre a *Lumenvox*, esta companhia tem também disponível um *SDK* para *Text To Speech*.

A *LumenVox Text To Speech SDK* [61] consiste num conjunto de *software*, treinos, suporte e licenças que permite qualquer programador, independentemente da sua experiência, implementar aplicações de forma rápida e eficiente.

O *LumenVox TTS Server* permite a sintetização de voz, transformando texto escrito em voz falada. Esta tecnologia permite que qualquer aplicação computacional transmita informação para um utilizador e tem disponíveis vozes femininas e masculinas para várias línguas, incluindo português. Suporta *Speech Synthesis Markup Language* (*SSML*) e está preparado para ser controlado por uma *API C/C++* ou *MRCP* versão 1 ou 2.

Este *SDK* inclui cinco licenças do *LumenVox Text To Speech Server*. Estas licenças são permanentes e podem ser usadas para desenvolvimento, testes e produção, e podem também ser vendidas juntamente com outros pacotes de serviços.

2.6.12. BitVoicer Server

Dando seguimento ao que foi escrito no capítulo anterior, O *BitVoicer Server* [47] pode ser usado para a sintetização de voz em autómatos. Ele é capaz de sintetizar voz a pedido de aplicações externas ou em resposta a um reconhecimento de fala válido. A voz sintetizada pode ser reproduzida nos dispositivos de som do servidor com o auxílio da aplicação *BitVoicer Server UI Link* ou enviada como fluxo de áudio para dispositivos clientes.

Permite a sintetização de voz para 17 idiomas, incluindo português de Portugal.

2.6.13. MaryTTS

MARY [62] é uma plataforma *open source* de *TTS* multilinguagem escrita em *JAVA*, atualmente mantida pelo *Multimodal Speech Processing Group*. Suporta atualmente várias línguas e estão mais em desenvolvimento.

Esta plataforma é composta por um servidor principal ou um programa que faz de gestor, e um cliente que trata de fazer os pedidos e receção de dados resultantes do processamento. O servidor é *multi-thread*, o que permite que o servidor trate de múltiplos pedidos. É flexível devido aos módulos suportados, que podem ser facilmente integrados num sistema. É também *XML-based*.

MARY HTTP Interface permite que programadores possam integrar nas suas aplicações *web*

um cliente do *MARY HTTP server*. Cada *request* ao servidor pode ser feito por *GET* ou por *POST* e pode ser configurado com parâmetros, como por exemplo, estilos da fala, efeitos do áudio, vocalizações e tipos de texto a reproduzir.
Esta plataforma como já foi referido, é *open source* e pode ser usada para uso comercial, pode ser modificada e redistribuída. A sua documentação não é muito esclarecedora.

2.6.14. VocalWare

Vocalware [63] disponibiliza um conjunto de *cloud-based APIs* que possibilitam os programadores integrarem a capacidade de *Text To Speech* com uma grande variedade de vozes de alta qualidade em aplicações *web* e *mobile*. Tem disponíveis mais de 100 vozes diferentes distribuídas por mais de 20 línguas (incluindo português), e a possibilidade de adicionar efeitos à reprodução de voz.
Disponibiliza três tipos de *APIs* de forma a cobrirem todos os tipos de plataforma. São compatíveis com muitas linguagens de programação e *frameworks*. As *APIs JavaScript/HTML5* e *ActionScript* (*Flash*) são *cliente side* desenhadas de forma a funcionarem em *web browser*s. A *API HTTP* (*REST*) é apropriada para aplicações *mobile* ou outras aplicações *standalone*.
O serviço da *VocalWare* não é gratuito, com exceção para um período *trial* de 15 dias com 1000 *streams* de limite. O serviço é taxado em *streams* que correspondem a 60 segundos de reprodução de áudio. O serviço é *self-service* e *pay-as-you-go*, não são necessários contratos ou acordos.

2.6.15. SpeakRight

SpeakRight [64] é uma *framework open source* em *Java* que permite a criação de aplicações de reconhecimento de voz em *VoiceXML*. A geração dinâmica de *VoiceXML* é feita através da *framework StringTemplate*. *VoiceXML* é um formato derivado a partir do *XML* e foi criado pelo *W3C*, cujo propósito é facultar diálogos de voz interativos entre Homem e máquina.
Como o *SpeakRight* é *code-based* a partir de um *IDE*, como por exemplo o Eclipse, podem-se desenvolver aplicações em *Java* e usar as suas classes. Dessa forma é possível um melhor *debug*, melhores testes, e ter mais ferramentas disponíveis, como geradores de documentação e de diagramas, entre outras vantagens.
Com esta *framework* é possível criar um sistema de conversação Homem-máquina devido à sua capacidade de reconhecimento e de sintetização de voz. Contudo, o *Speech Recognition* é

ainda um pouco limitado e por esse motivo não é feita referência no capítulo anterior.
O áudio reproduzido pode ser dos tipos, ficheiros de áudio, por *TTS* ou por valores gerados, por exemplo a partir de uma variável da aplicação. Não suporta iniciativas mistas, ou seja, apenas pode ser o Homem ou a máquina a iniciar a interação.
Esta *framework* permite que seja possível a definição de frases que são usadas para os mais variados objetivos, como por exemplo, perguntas ou mensagens de erro. Estas frases podem ser definidas em ficheiros *XML* para aplicações multilinguagem, não havendo a necessidade de compilar a aplicação.
SpeakRight está licenciado através da *OSS-approved Eclipse Public License* (*EPL*). Com esta licença qualquer pessoa pode fazer *download* e usar a *framework SpeakRight*. Não existem taxas de licença para uso comercial.

2.6.16. FreeTTS

FreeTTS [65] é um sistema sintetizador de voz escrito em *Java* baseado no *Flite*, desenvolvido pela *Sun Microsystems Laboratories*.
Não tem disponíveis muitas vozes nem muitas linguagens, contudo permite que sejam criados mais vozes e criar suporte para outras línguas. Permite também a criação de ficheiros áudio.
FreeTTS pode ser usado com o *Java Speech API* (*JSAPI*) 1.0, com *Java Web Start*, pode fazer de servidor *TTS* remoto ou como motor *TTS* a funcionar em *desktops*, entre outras integrações.
Foi testado em sistemas operativos como *Solaris TM Operating Environment*, *Mac OS X*, *Linux* e *Win32* e para o executar e testar é necessário o *Java 2 SDK Standard edition 1.4*.
A sua licença é livre e a sua página *web* contém documentação aparentemente suficiente para por o *FreeTTS* a funcionar.

2.6.17. Ivona

IVONA Software [66] é uma companhia da *Amazon,* que tem o objetivo de produzir tecnologia *TTS* de alta qualidade, que suporta as normas *SSML 1.0* e *SSML 1.1*. A sua tecnologia permite a geração de vozes naturais que proporcionam uma melhor comunicação e experiência do utilizador e é usada em milhões de dispositivos móveis, computadores, sistemas de comunicação e serviços. Algumas das suas características incluem: pedidos de autenticação únicos; consegue produzir *streaming*; é desenhada para minimizar o tempo de latência devolvendo sempre que possível as respostas; segurança através de comunicação

encriptada; é sempre monitorizado para identificação de anomalias e pronúncia personalizável devido à *API* suportar o *Pronunciation Lexicon Specification* (*PLS*).

Ivona tem disponíveis 51 vozes distribuídas por 23 línguas (incluindo português) e apresenta um serviço chamado de *Ivona Speech Cloud* que permite de uma maneira fácil adicionar um sintetizador de voz às aplicações. Permite definir o texto e a voz, usando chamadas *HTTP* e devolve a fala correspondente ao texto e à voz selecionada.

Para além da *Ivona Speech Cloud* existem outras formas de integração. São elas o *SDK open source* em *Java* e uma *API* que no entanto ainda está numa fase beta e pode haver alterações a qualquer momento.

Relativamente às licenças, existem dois tipos. Uma, a *Development Account*, é livre e tem um limite de 50000 *units* (cada *unit* são cerca de 200 caracteres) por mês. A outra, *Commercial Account* são $1000 por mês com pagamento pré-pago + $0.003 por *unit* para utilização acima de 250.000 *units* por mês.

2.6.18. eSpeak

eSpeak [67] é um software compacto *open source* para *TTS*. Tem disponíveis várias línguas, incluindo português e está disponível para ser usado em *Windows* e *Linux*. A fala produzida é clara mas não é muito natural e suave.

Este *software* está disponível também por outras formas. Por linha de comandos em *Windows* ou *Linux* a partir de ficheiros ou *stdin*, por *libraries*, para serem integradas em outras aplicações, por uma versão para *Windows SAPI5*, de forma a ser usado por leitores de ecrã e outros programas que suportem a interface *Windows SAPI5*. Este *software* foi também portado para outras plataformas como por exemplo *Android, Mac OSX* e *Solaris*.

eSpeak apresenta várias funcionalidades e características, como por exemplo, inclui diversas vozes sobre as quais várias características podem ser configuradas; Pode criar ficheiros áudio em formato *WAV*; Suporta o *SSML* e também *HTML*; Ocupa pouco espaço, cerca de 2 *Mb*; Pode traduzir o texto em códigos de fonemas, por isso pode ser adaptado como *front-end* para outros sintetizadores de voz; Escrito em *C*. Inclui ainda boa documentação disponível na sua página e um fórum de discussão com alguma atividade.

2.6.19. ReadSpeaker

ReadSpeaker [68] disponibiliza várias soluções de *TTS* para dar voz às páginas *web*, aplicações móveis, livros digitais, material de *e-learning*, documentos, entre outras coisas.

Entre as suas soluções, estão duas *APIs*, a *ReadSpeaker TTS Production API* e a *ReadSpeaker speechCloud API*, que podem ser integradas em aplicações *desktop*, *web* e móveis.

A *ReadSpeaker TTS Production API* disponibiliza um método automatizado para gerar ficheiros áudio, usando uma inovadora tecnologia *TTS*, através de uma *API* simples de usar com base em *HTTP*. Apta para o desenvolvimento e de alta capacidade, a *ReadSpeaker TTS Production API* permite o acesso a vozes de alta qualidade em muitos idiomas, incluindo português, e permite que a sua aplicação ou *software* envie texto e receba os correspondentes ficheiros áudio gerados pelo serviço de texto-voz. Esta *API* apresenta as seguintes características: Conversão texto-voz inteiramente automatizada; Dicionário personalizado de pronunciação, onde pode controlar como certas palavras são lidas; Diversos formatos de ficheiros áudio (*A-law, u-law, PCM, WAV, Ogg, MP3*); Vários idiomas e vozes à escolha, utilizando as melhores vozes de síntese; Exemplos de código para diferentes linguagens de programação: *PHP Java, Objective C* e *C++*; Interface de estatísticas (via *web* e / ou *API*); Ambiente de alojamento de alta segurança com alto nível de redundância e transferência em caso de falha.

A *ReadSpeaker speechCloud API* é uma *API* de *TTS online* para dar voz a aplicações para *desktop*, *web* ou móveis e a dispositivos com ligação à internet. Esta *API* é fácil de integrar, de usar e de alta capacidade. Disponibiliza vozes de alta qualidade em diversos idiomas para lerem o conteúdo das suas aplicações e dispositivos. Permite que uma aplicação ou *software* envie texto e receba o correspondente ficheiro áudio gerado por *TTS* para ser ouvido em aplicações e dispositivos com ligação à internet. Algumas das suas características são: Dicionário personalizado de pronunciação, onde os clientes podem controlar como certas palavras são lidas; Diversos formatos de ficheiros áudio (*A-law, u-law, PCM, WAV, Ogg, MP3*); Vários idiomas e vozes à escolha, utilizando as melhores vozes de síntese; Exemplos de código para diferentes linguagens de programação: *Java* (*Android*), *Objective C* (*iOS*), *PHP, ASP, Flash* / *ActionScript*; Estatísticas (via *web* e *API*); *Interface web* simples de usar, contendo tudo o que precisa para começar; Sistema de pagamento *online*; Pode ser usada em aplicações e dispositivos que pertençam à família da *Internet of Things*.

ReadSpeaker cria dicionários de pronunciação personalizados que tenham em conta palavras e termos específicos relacionados com atividades específicas que não estão a ser lidos de forma correta. É comum haver esse tipo de necessidade, por exemplo, para siglas e abreviaturas.

O *ReadSpeaker* é vendido como *Software as a Service* (*SaaS*) que está hospedado numa *cloud*. Para todos estes serviços existem vários modelos de preços que podem ser personalizados conforme as necessidades do cliente.

A seguinte tabela apresenta um resumo com algumas das características dos sistemas *TTS* descritos neste subcapítulo.

	Nome	Open Source	Acesso	Plataforma	Tipo	Custos	Português
Nuance	Dragon NaturallySpeaking SDK Client Edition	Não	SDK	Windows	-	-	Não
	Nuance Mobile Developer SDK	Não	SDK; HTTP REST Interface	Android; iOS; Windows Phone; HTTP interfaces	Cloud-based	Sim	Sim
OpenEars	OpenEars	Não	Framework	Sistemas iOS	Standalone	Não	Não
L²F	DIXI	Não	-	-	-	-	Sim
	DIXI.SERVER	Não	webservices	-	-	-	Sim
Microsoft	Speech API	Não	API REST	Vários	Cloud-based	Sim	Sim
Mozilla Developer Network	SpeechSynthesis	-	Library	Web applications	Cloud-based	-	-
IBM	Watson Text to Speech	Não	REST API	Várias	Cloud-based	Sim	do Brasil
iSpeech API	iSpeech API	Não	HTTP Get/Post	Várias	Cloud-based	Sim (excepto aplicações móveis)	Sim
API.AI	API.AI	Não	HTTP REST	Várias	Cloud-based	Sim	Sim
Festival Speech Synthesis System	Festival Speech Synthesis System	-	Framework	Unix	Standalone; Client/Server	Não	Não
Flite	Flite	Sim	Tool	Windows; PalmOS	Standalone	Não	Não
LumenVox	Text-to-Speech Server	Não	SDKs	Linux; Windows	-	Sim	Sim
BitVoicer Server	BitVoicer Server	Não	Package	Microsoft	Client/Server	Sim	Sim
MARY	MaryTTS	Sim	Framework	Linux; Windows	Local server	Não	Não
Vocalware	Vocalware	Não	HTTP REST	Web; Mobile	Cloud-based	Sim	Sim
SpeakRight	SpeakRight	Sim	Framework	Vários	Code-based	Não	-
FreeTTS	FreeTTS	Sim	Packages	Solaris TM Operating Environment; Mac OS X; Linux; Win32	Standalone	Não	Não
IVONA	IVONA	Não	SDK; APIs	Várias	Cloud-based	Sim	Sim
eSpeak	eSpeak	Sim	libraries	Windows; Linux; Android; Mac OSX; Solaris	Code-based	Não	Sim
ReadSpeaker	ReadSpeaker TTS Production API	Não	HTTP API	Várias	Code-based	Sim	Sim
	ReadSpeaker speechCloud API	Não	-	Desktop; Web; Mobile; Internet-connected devices	Code-based	Sim	Sim

Tabela 2. Resumo dos sistemas TTS.

2.6 Sistemas de conversação Homem-máquina

Na comunicação entre Homem e computador o meio mais convencional para interação entre Humanos e computadores tem sido a interface gráfica [2]. Os utilizadores interagem com os computadores, escrevendo textos, navegando por menus gráficos e clicando em itens como menu, botões e imagens. Os computadores respondem apresentando elementos gráficos representando a informação de forma visual.

A origem dos sistemas de conversação por voz coincide com o estudo inicial da inteligência artificial na década de 50 [5]. Nas últimas décadas estas interfaces tiveram um grande desenvolvimento devido, por um lado, aos avanços efetuados no processamento da linguagem e modelagem do diálogo, e por outro, à maior capacidade de processamento dos computadores. Todos estes avanços resultaram em soluções que já se encontram no meio comercial [5]. Muitas das empresas de *software* e telecomunicações perceberam o potencial deste tipo de interfaces e começaram a criar soluções que permitam a comunicação por voz [5].

Em vários filmes de ficção científica vem-se pessoas a falar com máquinas, robôs etc. Algo que era visto como ficção há anos atrás, é nos dias de hoje uma realidade. A tecnologia avança a passos largos, o que ontem era visto como algo impossível ou difícil de realizar, hoje já existe e pode ser visto como banal.

Os sistemas de conversação Homem-máquina, onde se incluem os sistemas *SDS*, são sistemas que permitem aos utilizadores interagirem com máquinas através da fala, de forma a obter informação, pedir a execução de tarefas, realizar transações, estudar, etc.

Alguns dos sistemas mais comuns dos *SDSs* são os assistentes virtuais inteligentes ou *intelligent personal assistants* (*IPAs*). Os *IPAs* são mais que uma ferramenta para realizar pesquisas e devolver informação, apesar de isso ser uma das suas funções. As suas funções principais consistem em fornecer respostas e executar ações de forma personalizada, contextual e consciente [69]. No fundo, os assistentes virtuais são uma interface para um conjunto de funcionalidades. Pretende-se que os *IPAs* tenham a capacidade de antecipar o que os utilizadores pretendem, mesmo antes do utilizador pensar nisso. Com todos os avanços é possível que brevemente estes sistemas revolucionem e venham substituir as pesquisas de informação como as conhecemos hoje [69].

Os sistemas *SDS* foram muito implantados em *call-centers*, onde o sistema pode reduzir a necessidade de operadores humanos e assim reduzir os custos [7].

Analisando o nome, temos então três palavras-chave [69]. *Intelligent*, devido à interface de comunicação entre Homem e máquina, o sistema entende o nosso pedido e responde ao

utilizador com respostas ou ações; *Personal*, ao contrário de uma simples pesquisa, estes sistemas adaptam-se ao utilizador de uma forma mais personalizada; *Assistant*, ajuda em tarefas de rotina, como marcar uma reunião, fazer uma reserva etc.

Uma das funcionalidades destas interfaces é canalizar o pedido que o utilizador faz para um serviço que lida apenas com os pedidos desse tipo, como por exemplo modificar as configurações do telefone, encontrar informações de filmes ou fazer uma reserva de restaurante. Caso não haja uma aplicação para o que foi pedido, o *IPA* faz uma pesquisa num motor de busca [70].

A inovação vai avançando e com várias *APIs* de terceiros, novas funcionalidades vão sendo adicionadas. Contudo, quando se constrói um sistema *SDS*, deve-se ter em conta a eficiência e naturalidade, uma vez que os seres humanos podem ficar rapidamente irritados quando falam com uma máquina.

Não há dúvida que todas estas funcionalidades são úteis, mas isso não torna um assistente virtual verdadeiramente inteligente [70]. O ideal será ter um sistema que responda a todas as questões que lhe são pedidas sem depender das aplicações instaladas ou *APIs* disponibilizadas. Contudo os sistemas existentes são ainda muito virados para determinados domínios e ainda não existe uma interface verdadeiramente inteligente.

Atualmente uma das queixas mais comuns sobre os *IPAs* tem a ver com o seu uso em veículos automóveis, devido ao reconhecimento de voz não funcionar da melhor maneira. Mas para além do uso nos veículos automóveis, a tecnologia de reconhecimento de voz está a ser testada em aviões [70].

O facto de os *IPAs* terem evoluído juntamente com a computação móvel, não é uma coincidência. Os *smartphones* são os dispositivos perfeitos para suportar sistemas inteligentes, devido a estarem quase sempre ligados à internet, tem bastante poder computacional, e podem sentir de forma virtual através dos seus sensores como a camara (visão), microfone (audição), entre outros sensores. Os *smartphones,* com todas estas capacidades, imitam os sentidos da perceção humana [69]. Para além disso os dispositivos móveis juntamente com os *SDSs* são cada vez mais fundamentais na aprendizagem *on-line* [71] e interativa.

No futuro a capacidade de perceber e sentir será combinada com o raciocínio lógico de forma a fornecer um *IPA* completamente pessoal, contextual e que irá ajustar as pesquisas em tempo real baseadas na localização, nas pessoas com quem fala, quem está à sua volta, expressões faciais [71]. Não se sabe se todas estas funções algumas vez vão ser realizadas no futuro, mas sabe-se que as empresas estão a trabalhar nisso [69].

Antes de existirem os *IPAs*, existiam os *Personal Digital Assistants* (*PDAs*), também

conhecidos como *handheld PCs*, ou *personal data assistants*, foram lançados por volta de 1980, com a finalidade de facilitar a vida diária dos utilizadores e permitir a gestão de alguma informação. Algumas das suas funcionalidades permitiam gerir uma agenda, gravar números de telefone, endereços, tarefas, etc. Com todo o avanço tecnológico presente nos dias de hoje pode-se considerar os *PDAs* como uma tecnologia ascendente dos *IPAs* que temos hoje [71].

A inteligência artificial (*AI*), como um subcampo da informática, pode desempenhar um papel importante neste tipo de sistemas e combinar a aprendizagem humana com raciocínio computacional [71].

Os *SDSs* podem ser utilizados em vários tipos de dispositivos, como diversos tipos de computadores, *Pcs*, dispositivos móveis, *pdas*, *IoT*, etc. Tem um papel importante no melhoramento da usabilidade do uso nos diversos dispositivos, contudo o sucesso desta tecnologia depende muito da performance do sistema. Entre algumas das suas características estão os seguintes pontos:

- Permitem interação de pessoas que tenham limitações, como cegueira e problemas motores;
- Utilizam uma interação intuitiva, porque a fala é natural para a maioria das pessoas e requer pouco ou nenhum treino especial;
- Permitem uma melhor mobilidade, os utilizadores podem mover-se à vontade nas suas casas, escritórios, na rua, em vários sítios, e ainda interagir com os computadores sem ter que se sentar e usar um rato ou teclado;
- Permitem a interação com as mãos-livres com a realidade virtual e aplicações de realidade aumentada.

Um dos maiores desafios para ter um *SDS* robusto é o de lidar com os ruídos de entrada [72]. Os *Speech Recognitions* atualmente ainda têm dificuldade em entender fala espontânea em ambientes com ruído. Outra dificuldade está relacionada com a fala dos utilizadores. Os utilizadores humanos quando falam para um *SDS* também cometem erros, verbalizam as palavras de forma incorreta, má dicção, entre outros problemas, que podem resultar numa má interpretação do *SDS*.

Atualmente existem várias comparações entre *SDSs*, especialmente entre *Google Now, Siri* e *Cortana*, contudo os seus resultados são muito relativos e não existe um consenso que indique o melhor sistema [71].

De seguida são apresentados alguns *SDSs* com uma breve descrição das suas capacidades e características.

2.7.1. Siri

Siri [73] foi o primeiro grande *IPA* a ser lançado e fornece uma grande variedade de serviços em que muitos deles dependem de sistemas de terceiros, como *Wikipedia*, *Bing*, etc. Essa informação é combinada com informação do perfil pessoal obtida dos contactos, calendário, *email*, lista músical, etc, juntamente com informação temporal, espaço, localização, entre outras informações de outros sensores. Tudo isto combinado com tecnologia *ASR* e *TTS*, para permitir que o utilizador interaja com o sistema por conversação [69].

Siri é uma das aplicações com mais relevo da *Apple*, foi o primeiro assistente virtual pessoal, resultado de décadas de investigação em inteligência artificial (*IA*) pela *SRI*. A tecnologia foi desenvolvida pelo projeto *SRI Cognitive Assistant that Learns and Organizes* (*CALO*), com o programa *Personalized Assistant that Learns* (*PAL*) da *Defense Advanced Research Projects Agency* (*DARPA*), o maior projeto de inteligência artificial conhecido na história dos Estados Unidos, e ainda trabalhando em conjunto com o Instituto de Tecnologia da Suíça *EPFL*. Em 2007 foi criada uma entidade separada com o nome de *Siri Inc*. para trazer a tecnologia aos consumidores. Em Abril de 2010 a *Apple* comprou a *Siri* e em Outubro de 2011 foi apresentado [74].

Esta aplicação necessita de uma ligação à Internet por *Wi-Fi* ou dados móveis. Atualmente não disponibiliza a língua português de Portugal, apenas português do Brasil.

Siri permite que o utilizador, através da sua voz, consiga enviar mensagens, agendar reuniões, fazer chamadas de telefone e muito mais. *Siri* consegue entender a voz humana, perceber o que se pretende e dar respostas.

Pode-se falar com *Siri* como se estivéssemos a falar com uma pessoa. Pode-se dizer algo como "Diga à minha esposa que eu estou atrasado" ou "Lembre-me de chamar o veterinário." Siri não só entende o que é dito, como é inteligente o suficiente para saber o que se quer dizer. Então, quando se pergunta "Existe algum bom sushi por aqui?" *Siri* vai responder "Eu encontrei alguns restaurantes de sushi perto." Então o utilizador pode dizer "Hmm. E quanto a pizza?" *Siri* lembra-se que o que foi perguntado anteriormente foi sobre restaurantes, por isso vai procurar restaurantes italianos na área. *Siri* é pró-ativo e então vai questionar o utilizador até encontrar o que ele está a procurar.

Siri ajuda os utilizadores a executar as tarefas do dia-a-dia. Consegue perceber quais são as aplicações a utilizar para o que foi solicitado, e encontra respostas a consultas por meio de fontes como o *Yelp* e *WolframAlpha*. Põe a tocar as músicas que o utilizador quer ouvir, dá-lhe instruções, acorda-o, diz os resultados desportivos dos jogos anteriores, etc.

A *Apple* está a trabalhar com fabricantes de automóveis para integrar a *Siri* em sistemas de

controlo de voz selecionados. Através de um botão de comando de voz no volante, o utilizador/condutor será capaz de fazer perguntas à *Siri* sem tirar os olhos da estrada. Para minimizar ainda mais as distrações, o ecrã do dispositivo não apresenta qualquer imagem ou luz. Com a funcionalidade *Eyes Free*, pode-se pedir à *Siri* para fazer uma chamada, selecionar e reproduzir música, ouvir e compor mensagens de texto, usar mapas e obter instruções, ler as notificações do utilizador, encontrar informações no calendário, adicionar lembretes e muito mais. *Eyes Free* é o suporte perfeito para a *Siri* ajudar o utilizador a fazer as tarefas, mesmo quando está do volante.

2.7.2. Cortana

Cortana [75] permite através da voz, interagir com dispositivos, como computadores, *smartphones* e *tablets*, facultando várias funcionalidades como abrir uma aplicação, tocar uma música, definir um lembrete baseado na localização, por exemplo, um utilizador tiver marcado um lembrete para comprar flores, uma notificação aparece no dispositivo quando o utilizador vai a passar perto de uma loja. Permite marcar um lembrete em determinada data no dispositivo e esse lembrete é partilhado com todos os dispositivos *Windows* do utilizador. *Cortana* permite lembrar o utilizador de perguntar ou dizer algo na próxima vez que ele for telefonar a alguém. Ajuda os utilizadores a fazer chamadas, enviar mensagens, registar lembretes, obter notas, reconhecer música, encontrar restaurantes nas proximidades, verificar o calendário e muito mais [71].

Se for preciso estar nalgum lugar num determinado momento, *Cortana* pode definir um lembrete com base no tempo que aparece em todos os seus dispositivos com o *Windows 10*. E com lembretes baseados nas pessoas, *Cortana* pode lembrar o utilizador de pedir ao seu amigo algo da próxima vez que lhe ligar.

Criado pela *Microsoft* para o *Windows Phone 8.1* e alimentado pelo motor de busca *Bing*, *Cortana* foi oficialmente apresentado na conferência *Microsoft BUILD developer* realizada entre os dias 2 e 4 de Abril de 2014 [71].

Tal como o *Google Now, Cortana* guarda informações anteriores, compostas por dados da localização, comportamentos, informações pessoais, lembretes e informações dos contactos [71].

Cortana pode ser o assistente ideal para alunos desorganizados. Tem a capacidade de fazer questões sobre as próximas aulas, notificações sobre tarefas a fazer, entre outros [71].

2.7.3. Google Now

O *Google Now* [76] permite a um utilizador estar a par do que se passa na sua vida todos os dias, incluindo o que necessita de fazer, onde precisa de ir e como lá chegar. Permite o utilizador estar informado dos seus interesses, notícias e informações importantes quando está em viagem. Obter as melhores informações sobre o trânsito, a moeda, os locais de interesse e o que está a acontecer, mesmo quando está longe de casa.

Google Now é um *IPA* com base no reconhecimento de voz, que investiga as questões que lhes são colocadas e marca eventos importantes para o utilizador. Ao contrário do *Siri*, o *Google Now* é um motor de busca desenhado para ouvir e compreender os comandos e consultas dos utilizadores. O que torna o *Google Now* único é a sua capacidade de utilizar *IA* mais ativamente [71].

Sempre que os utilizadores visitam uma página *web*, a aplicação regista os interesses e preferências dos utilizadores automaticamente [71]. Desta forma, a aplicação pode usar ativamente *AI* com base nos interesses registados combinados com os dados dos sensores dos dispositivos, de forma a fornecer sugestões.

Google Now permite antecipar os pedidos dos utilizadores e fornecer informações usando uma linguagem natural e é capaz de prever quais as informações que o utilizador precisa com base nas pesquisas anteriores e dados do contexto atual [71].

Google Now não é considerado um sistema de diálogo completo [71]. Pode ser utilizado no *Android*, *iPhone* ou *iPad*, *Chrome & Chromebooks*.

2.7.4. S Voice

S Voice [77] é um assistente de voz da *Samsung* que permite a interação entre o utilizador e o dispositivo através da voz. Está disponível em alguns dos seus dispositivos e pode ajudar em algumas tarefas.

S Voice pode ser ativado por voz e pode ajudar em tarefas como chamadas telefónicas, envio de mensagens, registar lembretes, controlar definições do dispositivo, pesquisas, tirar fotografias, colocar comentários nas redes sociais, entre outras tarefas. É possível também pesquisar direções no *Google Maps* enquanto o condutor se concentra na condução sem tirar as mãos do volante.

2.7.5. Nina

Nina [78] é um assistente virtual inteligente da Nuance, que permite aos utilizadores obterem

o que realmente procuram. É mais vocacionada para atendimentos a clientes.

Nina devolve resultados imediatos, precisos e bem-sucedidos de uma forma natural idêntica à humana. *Nina* comunica com os utilizadores numa forma natural usando voz e texto através da utilização de tecnologia de compreensão de linguagem natural da *Nuance*.

Nina tem duas soluções disponíveis, a *Nina Web* e a *Nina Mobile*. *Nina Web* permite adicionar um guia virtual a conteúdos *web*, podendo fazer o papel de assistente de compras, guia da página, esclarecedor de dúvidas, etc. *Nina Mobile* permite o uso de comandos por voz em aplicações móveis.

2.7.6. Vlingo

Vlingo [79] é um assistente virtual fundado em 2006 que pertence atualmente à *Nuance*. Permite transformar as palavras do utilizador em ações através da combinação de tecnologias de voz para texto, processamento de linguagem natural, e a tecnologia *Vlingo* para entender a intenção do utilizador e tomar as medidas adequadas.

Basta falar com o seu telefone ou digitar um comando através do *ActionBar* para fazer praticamente qualquer coisa.

Entre as suas funcionalidades estão tarefas como, procurar direções para um restaurante, realizar pesquisas sobre determinado tema, atualizar informações nas redes sociais, enviar mensagens, entre muitas outras coisas. Tudo isto pode ser feito por exemplo, enquanto o utilizador conduz ou vai a correr.

Vlingo tem uma tecnologia precisa de reconhecimento de voz, velocidade e inteligência para ajudar o utilizador a ficar ligado com as pessoas, empresas e atividades que são importantes para ele.

Vlingo está disponível para a maioria dos telefones *BlackBerry, iPhones* e *iPod Touch* da *Apple*, telefones *Android* com versão 2.0 ou superior, telefones *Nokia* e dispositivos *Windows Mobile*.

2.7.7. Viv

Viv [80] é um produto ainda em desenvolvimento pela equipa do *Viv Labs*, que incorpora na sua equipa alguns elementos fundadores do *Siri* da *Apple* e do *Watson* da *IBM*.

De forma geral, *Viv* é uma plataforma global que permite aos programadores criar uma interface inteligente de conversação para qualquer situação. É a maneira mais simples para o mundo interagir com dispositivos, serviços e objetos em todo o lado. "*Viv* é ensinado pelo

mundo, sabe mais do que é ensinado, e aprende todos os dias".

Desde o lançamento do *Siri*, várias limitações foram detetadas na realização das tarefas que lhe eram pedidas. *Viv* afirma estar à beira de realizar uma forma avançada de inteligência artificial que remove essas limitações. Será capaz de ensinar a si mesmo, dando-lhe capacidades quase ilimitadas. Ao longo do tempo, *Viv* será capaz de usar as preferências pessoais do utilizador e terá infinitas conexões para responder a quase qualquer consulta e executar quase todas as funções.

O objetivo da *Viv* é a construção de uma nova geração de inteligência artificial que possa processar grandes quantidades de dados para prever e satisfazer os desejos dos utilizadores [81]. *Viv* está a ser desenvolvido de forma a ser o primeiro assistente amigo do consumidor que realmente atinge essa promessa.

Os criadores da *Viv* esperam que em breve seja incorporado numa infinidade de objetos do quotidiano, conectados à internet. Eles dizem que os utilizadores podem aceder à inteligência artificial de *Viv* como um serviço. O acesso é feito ao que eles chamam "cérebro global" e esse cérebro pode fornecer a capacidade de inteligência a diferentes aplicações e dispositivos.

Para além da sua potencialidade, não é certo que *Viv* venha a ser um sucesso. Um dos grandes desafios será colocar todos os componentes de terceiros a trabalhar em conjunto.

Em relação ao *Siri*, *Cortana* e *Google Now*, existem algumas diferenças, a mais óbvia e de maior realce é a diferença em termos de personificação. *Siri* e *Cortana* representam melhor o papel de humano virtual, enquanto que o *Google Now* não [70]. Os inventores do *Siri*, agora em *Viv Labs*, parecem reconhecer o problema, e afirmam que a solução que eles estão a desenvolver vai simplificar radicalmente o mundo, fornecendo uma interface inteligente para tudo [70].

2.7.8. Alexa Amazon

Alexa [82] é um serviço de voz que fornece capacidades e competências que permitem aos utilizadores interagir com os dispositivos de uma forma mais intuitiva usando a voz. Exemplos de funcionalidades são a capacidade de reproduzir música, responder a perguntas gerais, definir um alarme ou horas, responder a perguntas de conhecimentos gerais, fornecer previsões meteorológicas, consultar a *Wikipedia* e muito mais.

Alexa tem disponível o *Alexa Skills Kit*, que é uma coleção de *self-service APIs*, ferramentas, documentação e exemplos de código, que torna a adição de competências à *Alexa*, por parte dos programadores, num processo rápido e fácil. Uma das competências deste *kit* é o *Smart*

Home Skill API. *Smart Home Skill API* ensina facilmente *Alexa* a controlar os seus dispositivos *cloud-controlled* de iluminação e termóstato. Todo o código é executado na *cloud*.

Para dispositivos que estejam conectados e que possuam um microfone e um altifalante, o *Alexa Voice Service* (*AVS*) permite adicionar uma interface comunicativa por voz aos dispositivos conectados. Os utilizadores podem simplesmente falar com *Alexa* através do microfone no seu dispositivo e *Alexa* irá responder através dos altifalantes do seu dispositivo. Isto fornece aos utilizadores o acesso às capacidades de *Alexa*, incluindo capacidades nativas e capacidades criadas pelos programadores através do *Alexa Skills Kit* (*ASK*). Adicionar *Alexa* ao seu dispositivo através do *AVS* é simples e não tem custos. Permite estender as capacidades do seu *hardware* para que a interação com os seus utilizadores seja feita de novas maneiras.

As vantagens deste serviço são várias. Permite que os seus utilizadores possam controlar dispositivos de maneiras mais intuitivas através de uma interface por voz no idioma natural; Como *AVS* é *cloud-based* o utilizador não precisa de se preocupar com o *hardware* e gestão de infraestruturas para tratar da comunicação verbal. A sua integração é fácil porque não é necessário ter experiência com reconhecimento de voz e compreensão de linguagem natural, devido ao *Alexa* tratar desses aspetos e usar *AVS* nos dispositivos é totalmente gratuito.

Um dos dispositivos que usa *AVS* é o *Amazon Echo* [83]. Este dispositivo é um altifalante mãos-livres que se pode controlar através da voz. *Echo* conecta-se ao *AVS* para reproduzir música, fornecer informações, notícias, resultados desportivos, tempo, entre outras funcionalidades. Para isso basta apenas pedir ao *Echo* para fazer.

2.7.9. BlackBerry Assistant

Com o *BlackBerry Assistant* [84] é possível executar várias tarefas pessoais e de trabalho, e interagir com o dispositivo usando linguagem natural. Este assistente é multitarefa, em vez de navegar para uma aplicação específica, pode-se simplesmente abrir o assistente e comunicar por voz os pedidos. *BlackBerry Assistant* responde e ajuda a completar a tarefa ou a interação rapidamente. Os pedidos, para além de linguagem natural, podem ser feitos de forma escrita e a resposta do *BlackBerry Assistant* pode ser feita por voz ou por texto apresentado no ecrã.

Quando os pedidos são feitos através de um dispositivo *bluetooth*, como por exemplo um fone de ouvido, o *BlackBerry Assistant* responde com informação mais detalhada e instruções para ajudar o utilizador a completar as tarefas sem olhar para o ecrã do dispositivo.

BlackBerry Assistant permite executar várias tarefas, tais como, enviar ou pesquisar *emails*; enviar, responder e procurar mensagens; fazer chamadas e verificar chamadas não atendidas;

criar, editar e pesquisar eventos num calendário; obter respostas a várias perguntas; enviar e procurar mensagens de texto; alterar definições do dispositivo; obter direções; procurar e ouvir música entre muitos outros.
Dependendo do idioma do dispositivo, das configurações deste assistente ou da rede *wireless*, pode não ser possível realizar algumas dessas tarefas.

2.7.10. Braina

Braina (*Brain Artificial*) [85] é um *Intelligent Personal Assistant*, interface de linguagem humana e *software* de automação para computadores com o sistema operativo *Windows*, que permite que um utilizador interaja com o computador. Não é um clone do *Siri* ou um *Cortana* para computadores, mas sim um poderoso *software* de produtividade pessoal e escritório, que não funciona apenas como um *chatbot*. A sua prioridade é ser bastante funcional e ajudar a realizar as tarefas. O utilizador pode escrever ou transmitir por voz os comandos e *Briana* vai entender o que o utilizador quer fazer.
Ao usar o reconhecimento de voz implementado no *Braina*, o utilizador pode interagir com o seu computador em qualquer lugar de sua casa através de uma rede *Wi-Fi* e pode ajudar o utilizador a realizar tarefas do dia-a-dia. É um *software* multifuncional que fornece um único ambiente de janelas para controlar o computador e executar uma ampla gama de tarefas usando comandos de voz. Pode registar um ditado através do reconhecimento de voz, pesquisar informação na internet, pode tocar as músicas que o utilizador quer ouvir, pode abrir ou pesquisar ficheiros no computador, pode definir alarmes e lembretes, pode fazer cálculos matemáticos, lembrar de notas, automatizar várias tarefas no computador, ler *ebooks* e muito mais.
Braina é o resultado do trabalho de investigação feito no campo da inteligência artificial, que tem o objetivo de o tornar num assistente digital que pode entender, pensar e até mesmo aprender com a experiência, como um cérebro humano. Faz compreensão da linguagem e aprende com o diálogo.
É também um *software* de S*peech Recognition* que converte voz em texto em qualquer página *web* e *software*, como por exemplo, *MS Word*, *Notepad*, pesquisas na *web*, etc. Pode também ler texto em voz alta de forma natural. O utilizador pode ouvir *ebooks*, *emails*, conteúdo *web*, etc. É possível escolher diferentes vozes e ajustar a velocidade de leitura.
Suporta várias línguas como, inglês, alemão, espanhol, francês, italiano, português, russo, Japonês entre outras. É rápido, fácil e preciso, ajudando a ter mais produtividade.

2.7.11. HTC Hidi

Hidi [86] é um assistente de voz idêntico ao *Google Now*, que funciona através de comandos por voz. Foi lançado em 2013 pela *HTC* apenas na versão chinesa e em alguns *smartphones* da marca. Ainda não existe informação oficial a indicar se esta aplicação será lançada em outros países, ou regiões com disponibilidade para outros idiomas.

Dispõe de funcionalidades semelhantes ao *Google Now*, como consultar o *email*, enviar mensagens, obter informação de filmes e hotéis, ouvir música, entre outras funcionalidades, através de comandos por voz.

A informação existente sobre este sistema é escassa.

2.7.12. Maluuba Inc's, Maluuba

Maluuba [87] foi fundado por vários alunos da Universidade de *Waterloo* no Canada em 2010, que começaram por desenvolver um assistente pessoal inteligente para *smartphones*. Mais à frente a *Maluuba* centrou as suas investigações sobre o processamento de linguagem natural [88].

A visão desta empresa consiste num mundo em que máquinas inteligentes trabalhem lado a lado com os humanos de forma a desenvolver a inteligência coletiva da espécie humana.

Maluuba utilizada uma abordagem de aprendizagem por redes neurais conhecida como *Deep Learning* para treinar o sistema. Os seus investigadores implementaram a rede de forma a considerar texto em diferentes níveis de abstração.

A tecnologia de *Maluuba* pode ser encontrada em milhões de dispositivos, sobre vários produtos e indústrias, incluindo *smartphones*, sistemas *IoT* e outras plataformas e está atualmente a construir as instalações de pesquisa mais avançadas do mundo em inteligência artificial com o objetivo de ensinar máquinas a pensar, raciocinar e a comunicar. A sua tecnologia de linguagem natural é usada em *smartphones* e *TVs* inteligentes para marcas como *Samsung* e *LG* e oferece uma experiência idêntica ao *Siri*, para dispositivos *Android* [89].

A investigação da *Maluuba* em *Spoken Dialogue Systems* tem-se focado na *Machine Learning Technology*, ferramentas de resolução de problemas e na funcionalidade de processamento da linguagem, que podem ser aplicados sobre *smart machines*. *Maluuba* tem-se desviado das técnicas tradicionais para obter novas técnicas baseadas em aprendizagem, para otimizar a satisfação dos utilizadores e a aquisição de conhecimento simultaneamente.

A próxima fase para *Maluuba* vai consistir em aplicar a aprendizagem por linguagem natural à sua tecnologia de pesquisa por voz para tornar a experiência entre as pessoas e os seus

dispositivos, *smartphones*, automóveis, mais suave e mais fácil do que nunca.

A leitura e compreensão de texto é uma tarefa computacional difícil, mas *Maluuba* tem feito progressos com um algoritmo que pode ler o texto e responder a perguntas sobre determinado assunto com uma precisão impressionante. Mais importante ainda, ao contrário de outras abordagens, ele funciona apenas com pequenas quantidades de texto. Esta tecnologia pode eventualmente ajudar computadores a "compreender" documentos.

2.7.13. Cognitive Code's 'SILVIA'

Cognitive Code Corporation [90] é uma empresa que tem vindo, desde 2007, a especializar-se no desenvolvimento e implantação de sistemas de inteligência artificial para conversação, baseados na tecnologia *SILVIA*.

SILVIA é um sistema completo para o desenvolvimento e implementação de aplicações inteligentes para praticamente qualquer plataforma computacional ou sistema operativo, com um núcleo tecnológico que permite que os seres humanos interajam com os computadores de formas completamente naturais e intuitivas. A Plataforma *SILVIA* é otimizada para desenvolvimento rápido e as suas aplicações podem ser instaladas em servidores *web*, *desktops* e dispositivos móveis. *SILVIA* pode até ser executado nativamente em *tablets*, *smartphones* e outros sistemas incorporados.

Cognitive Code uniu-se com algumas das maiores marcas globais em áreas como *big data*, saúde, finanças, jogos de vídeo, fabrico industrial e tecnologia da informação, bem como com agências governamentais abrangendo um amplo espectro, incluindo a defesa, inteligência e educação, todos usando soluções baseadas no conhecimento *SILVIA*.

SILVIA é uma plataforma de conversação que interpreta voz, texto ou outro tipo de *input*. Para além disso, interage com aplicações apropriadas ou sistemas operativos e com o utilizador. Não está vinculado a qualquer língua porque o “cérebro” de *SILVIA* pensa em termos de unidades matemáticas, o que faz de *SILVIA* poliglota, que pode facilmente passar de uma língua para outra sem a necessidade de programação especial.

SILVIA não depende exclusivamente de correspondências padrão simplistas, ela é capaz de avaliar o significado de entrada e gerar dinamicamente uma nova saída. Com a capacidade de ligar conceitos e ideias, *SILVIA* permite algo semelhante a uma conversa real. Ela é também fácil de treinar, porque não é necessário preencher previamente a sua base de dados com todas as perguntas e respostas possíveis. Não é necessário explicar erros ortográficos ou diferentes formas de fazer a mesma pergunta. O core de inteligência da *SILVIA* é flexível e pode classificar essas situações automaticamente.

A tecnologia *SILVIA* não tem de ser apenas *server-base*, porque ela tem a capacidade de ser executada nativamente no dispositivo e pode operar dentro do contexto da aplicação *desktop* de um cliente, dispositivo móvel, como parte do nosso servidor *SILVIA*, ou como um nó de uma rede *peer-to-peer* e também na *cloud*.

SILVIA tem disponíveis os seguintes componentes: *SILVIA Core,* um motor de tempo de execução de alto desempenho, configurável para o cliente, servidor ou dispositivo móvel; *SILVIA Server,* um sistema de configuração para uma gestão automatizada de *n* núcleos *SILVIA* em um ou mais servidores de rede; *SILVIA Voice*, um componente *add-on* modular para aceitar a entrada de voz e saída de voz dentro de uma aplicação cliente, uma página *web*, ou como parte de *SILVIA Server* para *streaming* otimizado de conteúdos; *SILVIA API* permite aos programadores criarem novas aplicações e funcionalidade baseadas nos *plug-ins*; *SILVIA Studio,* um gráfico de sistemas para o desenvolvimento de um comportamento específico do aplicativo;

2.7.14. IBM's Watson

IBM Watson [91] é uma plataforma de tecnologia que utiliza processamento de linguagem natural e aprendizagem de máquina para distinguir grandes quantidades de dados não estruturados.

De uma forma geral esta plataforma pode responder às perguntas dos seus utilizadores, extrair de forma rápida as principais informações de vários documentos, padrões e relacionamentos entre os dados.

Watson consegue ter o discernimento de informação, analisando dados não estruturados utilizando processamento de linguagem natural para entender a gramática e contexto. Compreende questões complexas avaliando todos os significados possíveis e determinar o que está a ser pedido, para depois apresentar respostas e soluções, com base em provas e qualidade da informação encontrada.

Watson para responder as perguntas que lhe são feitas tem de, em primeiro lugar, aprender sobre o assunto antes que possa responder a perguntas sobre esse assunto. Para aprender um novo assunto, *Watson* através da sua capacidade de analisar informação não estruturada, que inclui entre outros, artigos de notícias, relatórios de pesquisa, mensagens da comunicação social e dados do sistema empresarial, recebe vários conteúdos relacionados com o tema, podendo esses conteúdos ser do tipo documentos do *Word*, *PDFs* e páginas da *web*. Depois são adicionados combinações de perguntas e respostas sobre o assunto para treinar *Watson*. À medida que *Watson* recebe novas informações, este é atualizado.

Para responder às questões colocadas, *Watson* procura em milhões de documentos para encontrar milhares de possíveis respostas. Depois recolhe evidências e através de um algoritmo de pontuação, avalia a qualidade dessas evidências, classificando todas as respostas possíveis com base na pontuação das evidências.

A *IBM* disponibiliza, através da *Watson Developer Cloud*, várias *APIs* para suportar linguagem, voz, visão e dados, de forma a trazer recursos de computação cognitiva para as aplicações. Do ponto de vista da conversação Homem-máquina, estes serviços permitem que sejam integrados facilmente os recursos de processamento de linguagem natural em aplicações, independentemente da origem do seu código fonte.

2.7.15. Facebook M

M é o *IPA* do *Facebook* [92], disponível através da aplicação *Messenger*, onde o utilizador pode interagir com ele como se fosse um dos seus amigos verdadeiros.

Pretende-se que ele execute as tarefas pretendidas pelo utilizador, mesmo que ele esteja ausente. Ao contrário de outros serviços baseados em *IA* no mercado, *M* pode efetivamente completar as tarefas em vez do seu utilizador. Este sistema pode comprar produtos, entregar presentes a alguém, reservar restaurantes, organizar viagens, agendar compromissos e muito mais.

M é um sistema híbrido que combina tecnologia de inteligência artificial com uma equipa de ajudantes humanos, de forma a garantir que todos os pedidos são respondidos.

Facebook M está atualmente em fase de testes internamente e com alguns utilizadores do *Facebook* e ainda não tem uma data de lançamento confirmada.

M também faz parte da investigação do *Facebook* na tecnologia de *IA*. A empresa comprou a *Wit.ai* em janeiro de 2015 de forma a obter a sua tecnologia de processamento de linguagem natural para transformar a fala e texto em dados que podem ser usados.

2.7.16. MindMeld

MindMeld [93], pioneira no desenvolvimento de tecnologia para aplicações orientadas a voz, utiliza uma metodologia um pouco diferente das metodologias do *Siri*, *Cortana*, e *Google Now*, que são aplicações que tentam fornecer uma experiência mais abrangente de assistente inteligente aos utilizadores. Eles tentam encontrar respostas para vários temas de conhecimento geral através de pesquisas sobre toda a *web* e conteúdos, ajudando também com a realização de tarefas específicas.

Suporta várias linguagens, incluindo português e os seus utilizadores podem encontrar a informação que pretendem, usando linguagem natural simples em vez de palavras-chave ou comandos, enquanto conduzem, andam de bicicleta, caminham, cozinham, etc.

MindMeld, por outro lado, é um serviço *cloud-based* que pode ser utilizado em milhares de outras aplicações que podem beneficiar com a interação por voz. *MindMeld* não foi projetado para ser um assistente inteligente de âmbito geral. Em vez disso, ele permite que milhares de empresas de diferentes indústrias criem o seu próprio assistente inteligente dedicado ao domínio ou área do seu negócio. Por exemplo, utilizando *MindMeld*, os programadores podem criar uma aplicação baseada em voz para encontrar produtos num catálogo de produtos específicos ou vídeos numa lista de vídeos.

MindMeld para entender a linguagem humana e responder às questões que lhe são colocadas, usa um avançado algoritmo *NLP*. Mas em vez de aplicar esse algoritmo sobre os padrões gerais do uso da linguagem humana, aplica-o sobre um conhecimento personalizado criado para cada aplicação. Pesquisando num domínio mais específico, *MindMeld* consegue obter melhores resultados.

MindMeld trabalha com o *Speech Recognition* presente em variados sistemas operativos e *browsers*. Os seus *SDKs* integram, de forma automática, com as bibliotecas de *Speech Recognition* presentes no dispositivo ou plataforma.

2.7.17. api.ai

O objetivo da *Api.ai* [41] é fazer com que o processo de criar e integrar sofisticadas interfaces de voz seja o mais simples possível. Esta plataforma permite que programadores possam integrar sistemas inteligentes de comandos de voz nos seus produtos de forma a criar interfaces por voz fáceis de usar.

Esta plataforma disponibiliza uma ferramenta que funciona como uma espécie de *backoffice* e permite a criação de um agente que vai estar relacionado com a aplicação ou dispositivo a que se pretende integrar uma interface por voz. É nesse agente que se seleciona a linguagem de comunicação. Depois é possível criar *Entities*, que são objetos muitas vezes específicos a um domínio como um meio de mapear frases de linguagem natural para frases canónicas que capturam o seu significado. Podem ser criadas também *Intents* em que o objetivo é mapear todos os tipos de solicitações dos utilizadores para uma ação. É nesta ferramenta que se testa e treina o agente.

Depois do Agente criado esta interface pode ser distribuída sobre aplicações móveis, por *web services*, dispositivos *IoT*, Jogos, plataformas de comunicação, usando um dos vários *SDKs*

disponíveis para as mais populares plataformas. Os *SDKs* estão disponíveis para as plataformas *Android, HTML, C#.NET, C++, iOS, Mac OSX* e *Apple Watch, JavaScript, Unity, Python, Cordova, Node.js, Xamarin* e *Ruby*.

A *Api.ai* tem disponível uma boa e organizada documentação. Existem várias modalidades de utilização, a mais básica é grátis. A *Api.ai* juntou-se recentemente à *Google*.

2.7.18. Duolingo

Duolingo [94], através da comunicação por voz entre Homem-máquina, disponibiliza uma ferramenta para a aprendizagem de línguas, que possibilita a cada utilizador aprender ao seu ritmo e de uma forma divertida.

Duolingo foi fundada em 2011 e como é uma empresa jovem, tem que se concentrar em maximizar o crescimento e trazer novos recursos para o mercado o mais rápido possível. Por isso, em vez de terem os seus próprios servidores, optaram por adquirir um serviço *cloud-based*, e assim não necessitam de se preocupar com infraestruturas. Os serviços de *cloud* são prestados pela *Amazon*. Serviços como *Amazon CloudFront* e *Amazon Simple Storage Service* (*Amazon S3*) servem mais de três mil milhões de pedidos por mês para *avatares* e tratam de transformar *Text To Speech*. O *software* que trata do *Speech Recognition* corre em instâncias especializadas do *Amazon EC2* para serviços específicos.

Duolingo está disponível para *iOS*, *Android* e através do *browser Chrome*.

2.7.19. Outros sistemas

Existem muitos outros projetos de comunicação Homem-máquina para vários objetivos. De seguida são referenciados mais alguns: *Dragon Assistant da Nuance* que ainda está numa versão beta [95]; *Indigo* [96]; *Voice Actions* [97]; *Voice Search da Soundhound* [98]; AVX, *EVA, EVAN Voice assistente* [99]; *Evi* [100]; *Andy X* [101]; *Memrise* [102].

Em relação a projetos *open source* existem também várias soluções. Algumas delas são a *Ariadne Spoken Dialogue System* [103]; *Olympus* [104]; *SEMAINE* [105]; *TRINDI* [106]; *PED* [107]; *HALEF* [108]; *HOWe* [109]; *Regulus* [110]*; Perlbox.org* [111]; *Alex* [112]; *Jindingo* [113].

A seguinte tabela apresenta um resumo com algumas das características dos sistemas de conversação Homem-máquina descritos neste subcapítulo.

Origem	Nome	Tipo	Plataforma	Custos	Português
Apple	Siri	IPA/SDS	iOS	Não	Do Brasil
Windows	Cortana	IPA/SDS	Windows, Android, iOS	Não	Do Brasil
Google	Google Now	IPA	Android, iOS, Chrome OS, Google Chrome	Não	Sim
Samsung	S Voice	IPA	Android	Não	Sim
Nuance	Nina	IPA	Web application; Mobile	-	SIM
Vlingo	Vlingo	IPA	BlackBerry; iOS; Android; Windows Mobile; Nokia system	Sim	Só para Nokia
Viv Labs	Viv	IPA	-	-	-
Amazon	Amazon Alexa	IPA	Varias	-	Não
BlackBerry	BlackBerry Assistant	IPA/SDS	BlackBerry	-	-
Braina	Braina	IPA	Windows	-	Sim
HTC	Hidi	IPA	Android	-	Não
Maluuba Inc's, Maluuba	Maluuba Inc's, Maluuba	IPA	TVs; smartphones; automóveis	-	-
Cognitive Code Corporation	SILVIA	IPA/SDS	Varias	-	-
IBM	IBM Watson	IPA/SDS	Varias por APIs	Sim	-
Facebook	M	IPA hibrido		-	-
MindMeld	MindMeld	Conversational Interfaces	Varias por APIs e SDKs	-	Sim
Api.ai	Api.ai	Conversational Interfaces	Varias	Sim	Sim
Duolingo	Duolingo	Learning Interface	iOS; Android; Chrome, Windows Phone	Não	Do Brasil

Tabela 3. Resumo dos sistemas de conversação Homem-máquina.

Avaliação de performance de ASR com base no método WER

Considera-se que, para a comunidade de pessoas que trabalha e estuda sistemas de reconhecimento e interpretação da voz, ter uma boa precisão num sistema de reconhecimento de voz, é um bom indicador para obter bons resultados no *SLU*. Na verdade, isto nem sempre é a realidade como é demonstrado por Ye-Yi Wang, Alex Acero e Ciprian Chelba [114].
A performance dos sistemas de reconhecimento de voz é muitas das vezes medida através dos parâmetros de precisão e velocidade [12]. A precisão pode ser medida pela precisão do desempenho, que é muitas das vezes avaliada pelo método *Word Error Rate* (*WER*), enquanto que, a velocidade é medida pelo tempo real do reconhecimento. Existem outras formas para medir a precisão, tais como *Word Recognition Error* (*WRR*), *Single Word Error Rate* (*SWER*), *Command Success Rate* (*CSR*), entre outras.

Word Error Rate (*WER*) é uma medida de avaliação quantitativa [2] muito usada na avaliação do desempenho de sistemas de reconhecimento de voz, muitas das vezes para determinar o desempenho e o progresso no desenvolvimento de diferentes modelos acústicos e linguísticos. Também é usado em sistemas de tradução.
O *WER* é derivado da distância *Levenshtein*, trabalha ao nível das palavras em vez de ser ao nível do fonema [12]. Consiste em fazer um alinhamento entre o output do sistema de *ASR*, composto pelo texto reconhecido pelo sistema e a transcrição do *input*, composto pelo texto referência que foi transmitido ao sistema de reconhecimento de voz.

A sua fórmula de cálculo é a seguinte [12]:

$$WER = \frac{S + D + I}{\mathrm{N}}$$

Em que:

S: Corresponde ao número de substituições;

D: Corresponde ao número de eliminações;

I: Corresponde ao número de inserções;

N: Corresponde ao número de palavras do texto de referência;

O seu resultado ideal é zero (*WER = 0*) o que significa que o texto da referência coincide na totalidade com a melhor hipótese obtida pelo *ASR*.

Para perceber o funcionamento deste método de avaliação, pode-se seguir o exemplo de Gabriel Skantze [115]. Supondo que a frase de referência (*REF*) "*I have a large building on my left.*" é transmitida por voz para um sistema de *ASR* e o resultado obtido é a seguinte hipótese (*HYP*) "*I have large blue building on my right*".

			D		I				S
REF	I	have	a	large		building	on	my	left
HYP (ASR)	I	have		large	blue	building	on	my	right

Tabela 4. Exemplo de um alinhamento segundo o método WER.

A tabela representada em cima apresenta o alinhamento da frase de referência e da frase obtida pelo *ASR*. Desse alinhamento pode-se retirar a seguinte informação: O número de palavras do texto de referência (N) é 9; O número de substituições (S) é 1; O número de eliminações (D) é 1; O número de inserções (I) é 1.

Aplicando a fórmula temos:

$$WER = \frac{1+1+1}{9}$$

Em que o resultado é igual a:

$$WER = 0.33$$

Para obter a percentagem multiplicamos por 100:

$$WER = 33.33\%$$

A taxa de *WER* deste exemplo é de 33.33%.

Outra forma de avaliação da performance de um sistema *ASR* é o uso do *Word Recognition Rate* (*WRR*) [12]. A sua fórmula de cálculo é a seguinte:

$$WRR = 1 - WER = \frac{N - S - D - I}{\mathrm{N}}$$

Para o exemplo anterior o resultado é:

$$WRR = 1 - 0.33 = \frac{9 - 1 - 1 - 1}{9}$$

$$WRR = 0.67$$

Para perceber a capacidade de reconhecimento de voz por parte de alguns sistemas de *ASR* apresentados neste trabalho, neste capítulo são mostrados os resultados de alguns testes efetuados com base no método de avaliação *WER*, devido a ser uma medida de avaliação de performance muito comum para a avaliação de *ASRs*.

Para isso foram usados quatro ficheiros com perto de trinta segundos de áudio, em que a linguagem da fala é o inglês. Cada ficheiro foi obtido a partir da página YouTube e foi editado de forma a poder ser testado. Cada um tem características diferentes de forma a submeter os sistemas *ASR* a níveis de dificuldade diferentes. Os temas dos textos são variados e não seguem qualquer lógica.

O texto contido em cada um dos ficheiros é o seguinte:

"AudioEnglishExample_1"

"*Many people are worried about learning English. They think English is difficult and it's hard to memorize new words and grammatical rules. In fact, learning English can be a piece of cake. Don't worry about pronunciation. Don't worry about grammar."*

"AudioEnglishExample_2"

"*I am so angry with my boss. I finished the project for her more than four weeks ago and she is only looking at it now. She asks me questions I have already answered. She phones me all the time to ask about it and to talk about it. She is driving me crazy! I need the money from the project but I don't need all the problems! Auuugh! She is also telling me every small error*

I have made. I am human! I make errors! It's normal! It is so annoying, I want to scream."

"AudioEnglishExample_3"
"*My name is Shauna, and I'm a caterer actually, I've got a lot of work to do because I catered a wedding last night this wedding had probably 300 people and it was a lovely wedding the bride was just beautiful she had an ivory dress, ruffles all the way down the back. Probably 20 people in the wedding party. Umm… her colors were brown and light green, it was a lovely wedding. Well I was in charge of the food of course.*"

"AudioEnglishExample_4"
"*Introducing new Wondersponge comfort and hygiene in a sponge! Wondersponge is designed to fit comfortably in the average sized hand making it easy to hold, and it's almost twice as thick as most other sponges so it will last longer. Its unique design uses more air making it lighter and faster drying. What's more it has an inbuilt antiseptic which makes it the cleanest sponge available. Don't just use a sponge, use Wondersponge.*"

Cada um dos ficheiros tem características que os tornam diferentes um dos outros, com o objetivo de submeter os sistemas *ASR* testados neste capítulo a algumas condições variadas. No primeiro ficheiro de áudio a pronunciação da voz é feita de forma pausada e clara, tornando-o à partida num teste simples. No segundo ficheiro a pronunciação é feita de forma mais rápida e com várias entoações, tornando-o num teste mais difícil que o ficheiro anterior. O terceiro ficheiro submete os *ASR* a um teste mais difícil, uma vez que apresenta uma pronunciação fluida, em que o áudio não está nas melhores condições, num ambiente em que se ouvem vários ruídos de fundo. No quarto exemplo, para além de a pronunciação ser também rápida e de haver entoações em algumas palavras, existe também uma música de fundo que acompanha todo o discurso.

Os sistemas *ASR* testados são, o *CMUSphinx, Wit.ai, Bing Speech API, Speechmatics, iSpeech API* e o *IBM Watson Speech to Text*. Para cada um foi feito um teste com cada um dos ficheiros áudio apresentados em cima e os resultados foram registados. Cada sistema apenas realizou um teste com cada um dos ficheiros, porque os resultados são sempre iguais.

Para o sistema *CMUSphinx,* o reconhecimento foi realizado diretamente com ficheiros áudio, com base numa aplicação *java,* que utiliza a *librariy Sphinx4* do *CMUSphinx,* criada para

efetuar os testes. Os testes foram efetuados para cada um dos ficheiros e os dicionários e modelos linguísticos e acústicos correspondem à língua Inglesa (*en-us*). Cada ficheiro testado com esta *ASR* tinha as seguintes características, formato *WAVE*, *bit depth* de 16 *bit*, canal áudio mono e frequência de 16000 *Hz*.

Os testes efetuados com o sistema de reconhecimento *Wit.ai* foram também realizados a partir de ficheiros áudio. Esses ficheiros foram editados de forma a cumprirem alguns requisitos, ficando com as seguintes características: formato *WAVE*, *bit depth* de 8 *bit* e frequência de 8000 *Hz*. Os testes foram efetuados com base numa aplicação java implementada para o efeito onde no código o ficheiro áudio é enviado por *POST* através de uma *API*. Foi necessário o registo na plataforma web do *Wit.ai*, de forma a puder utilizar os seus serviços. A linguagem usada para a conversão é o inglês.

A realização dos testes no *ASR Bing Speech API* foi feita a partir de uma aplicação desenvolvida em *C#*, num projeto do tipo *Windows Presentation Foundation* (*WPF*). A base desta aplicação foi obtida a partir de um exemplo disponibilizado pela *Microsoft*. Para se puder utilizar os serviços do *ASR Bing Speech API*, foi necessário efetuar um registo na página *web* dos serviços cognitivos da *Microsoft*. Para testar cada um dos ficheiros áudio foram utilizadas duas variantes do inglês, o inglês americano e inglês britânico. Os ficheiros testados com este sistema seguem as seguintes características: formato *WAVE*, *bit depth* de 16 *bit*, canal áudio mono e frequência de 16000 *Hz*.

Em relação ao *Speechmatics* a realização do teste foi efetuada com o *upload* dos quatro ficheiros, um de cada vez, numa página web [116] disponibilizada por este sistema de *ASR*. É, no entanto, necessário efetuar um registo para a puder usar e durante um limitado número de créditos disponíveis. A linguagem utilizada para o reconhecimento de voz foi o inglês em duas variantes, britânico e americano. Os ficheiros testados apresentavam as seguintes características: formato *mp3* e 256 *kbps*.

Os testes efetuados ao *ASR* do *iSpeech API* foram realizados através do *upload* dos ficheiros de teste numa página *web* [117] disponibilizada pelo *iSpeech*. Os ficheiros testados apresentavam as seguintes características: formato *WAVE*, *bit depth* de 16 *bit*, canal áudio mono e frequência de 16000 *Hz*. A linguagem utilizada no reconhecimento da voz foi o inglês.

Os testes do reconhecimento de voz efetuados sobre a *ASR IBM Watson Speech to Text*, foram realizados com o *upload* dos vários ficheiros numa página *web* [118] disponibilizada pela *IBM*. Cada ficheiro teve de cumprir determinados requisitos como o formato *WAV*, *bit depth* de 16 bits e a frequência ser de 16000 *Hz*. Também existia a opção de os testes serem feitos por *streaming*, no entanto, optou-se pelo *upload* dos ficheiros. Os testes foram realizados para duas variantes do inglês, britânico e americano.

Os ficheiros para serem testados devem respeitar os seguintes requisitos em relação a cada sistema *ASR*:

	Formato	Som	Bit depth	Bitrates kbps	Frequência Hz
CMUSphinx	Wav	Mono	16		8000/16000
Wit.ai	Wav; mpeg3; Ulaw; raw	Mono	8; 16; 32		8000
Bing Speech API	Wav	Mono	16		8000/16000
Speechmatics	aac; aif; m4a; mov; mp3; mp4; wav	-	-		8000 ou superior
iSpeech API	Aiff; mp3; ogg; wma; Flac; wav; alaw; ulaw; vox; mp4	-	-	(Estas bitrates são apenas para mp3) 16; 24; 32; 48 (default); 56; 64; 80; 96; 112; 128; 144; 160; 192; 224; 256; 320	8000; 11025; 16000 (default); 22050; 24000; 32000; 44100; 48000
IBM Watson Speech to Text	Wav; Ogg; Flac	-	16		8000/16000

Tabela 5. Requisitos dos ficheiros em relação aos ASR.

Alguns valores dos resultados podem depender da avaliação que é feita pela pessoa que analisa os resultados obtidos com as frases originais.

O gráfico seguinte apresenta, de uma forma geral, os resultados dos testes efetuados. Os resultados dos testes efetuados podem ser verificados com mais detalhe no ficheiro "*Resultados_WER.xlsx"*, presente no *DVD* que acompanha este documento. Para cada sistema *ASR* testado são exibidos os valores *WER* obtidos em relação a cada ficheiro áudio de testes. Como alguns *ASR* foram testados com mais de uma variante do inglês, nos gráficos seguintes aparecem os valores dos melhores resultados por *ASR*.
O sistema *Wit.ai* não conseguiu devolver resultados para o teste efetuado com o ficheiro "AudioEnglishExample_3".

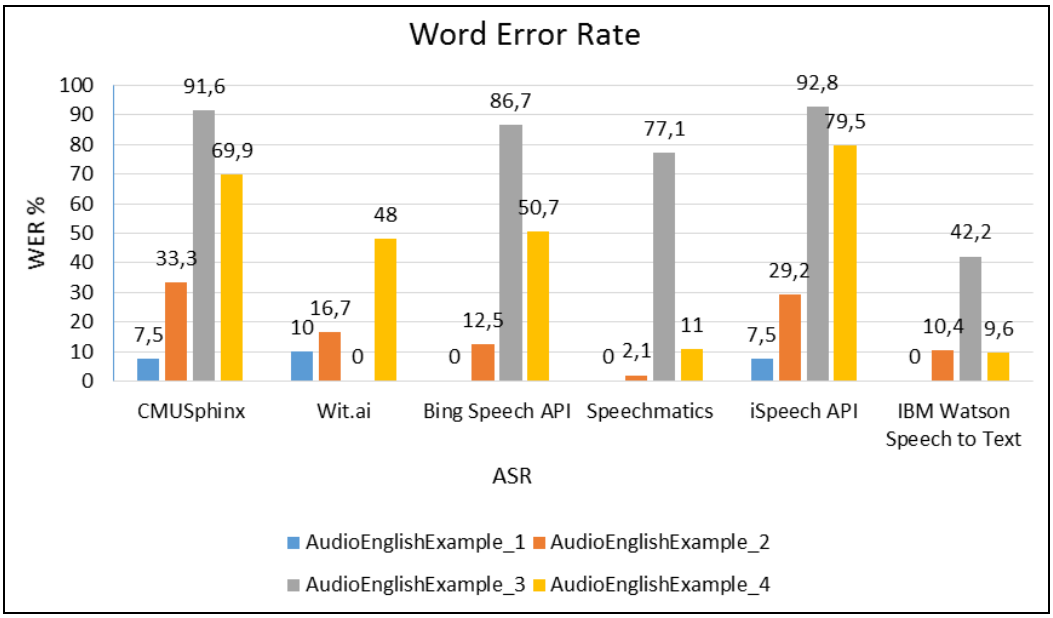

Figura 3. Gráfico com os resultados da avaliação WER.

Pode-se ter também a perceção do número de erros que cada sistema *ASR* obteve na avaliação *WER* em relação a cada ficheiro. Mais concretamente, o número de erros de substituições (S), inserções (I), eliminações (D) de cada sistema *ASR* sobre cada ficheiro áudio de teste. Os gráficos seguintes apresentam esses resultados.

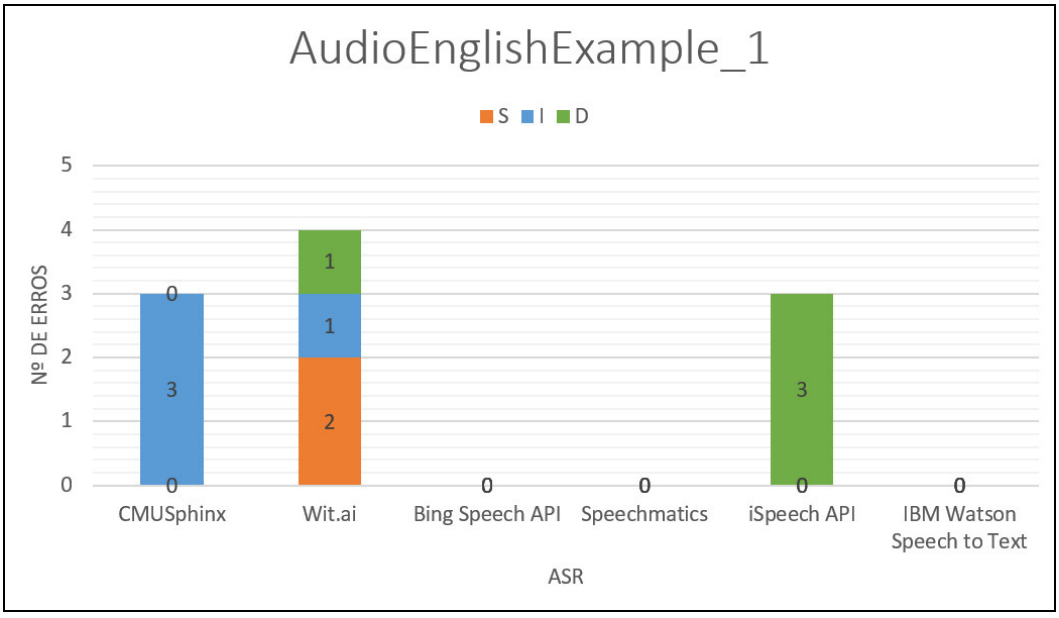

Figura 4. Gráfico que apresenta o número de erros relacionado com cada ASR para o ficheiro AudioEnglishExample_1.

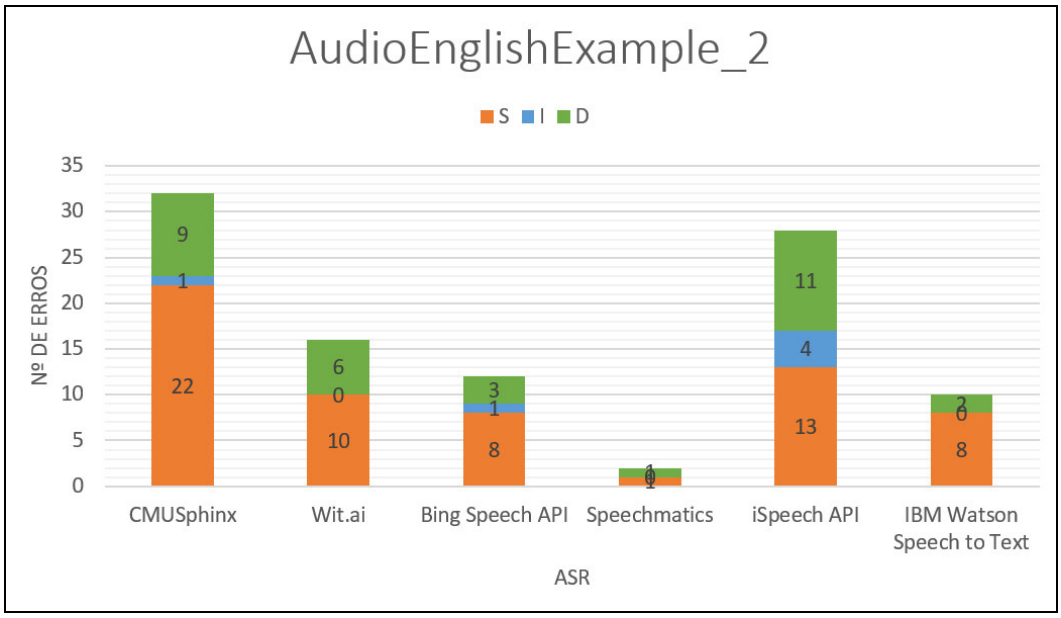

Figura 5.Gráfico que apresenta o número de erros relacionado com cada ASR para o ficheiro AudioEnglishExample_2.

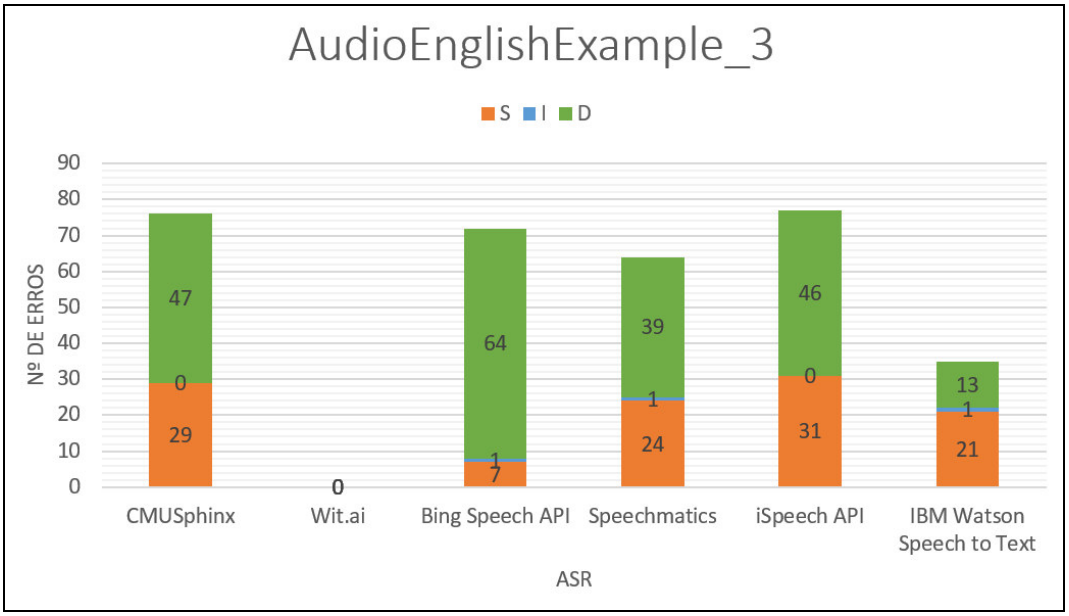

Figura 6. Gráfico que apresenta o número de erros relacionado com cada ASR para o ficheiro AudioEnglishExample_3.

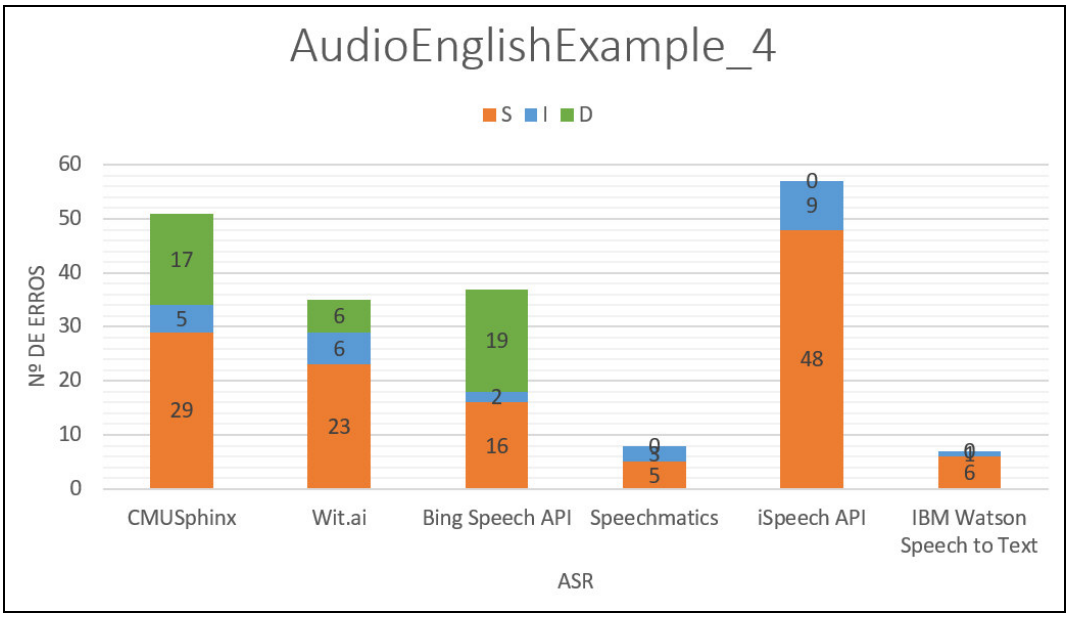

Figura 7. Gráfico que apresenta o número de erros relacionado com cada ASR para o ficheiro AudioEnglishExample_4.

Os testes foram todos efetuados com base em ficheiros áudio, onde a voz usada pelos oradores é o inglês. Como se sabe, existem várias variantes do inglês, entre os quais o britânico, americano, australiano, canadiano, entre outros. Sendo todas as variantes consideradas inglês, existem algumas diferenças que as distinguem entre si, como por exemplo, a pronúncia, a entonação, o uso de palavras diferentes para dizer a mesma coisa ou o uso das mesmas palavras para designar coisas diferentes.

Com o objetivo de validar se existem diferenças no uso de variantes diferentes do inglês no reconhecimento de voz, foram efetuados testes em que alguns *ASR* fizeram o reconhecimento com as variantes inglês americano e inglês britânico. Os *ASRs* testados com diferentes variações do inglês foram o *Bing Speech API, Speechmatics* e o *IBM Watson Speech to Text*.

Os gráficos seguintes permitem perceber a diferença do reconhecimento da voz com essas duas variantes.

- Bing Speech API

O primeiro grupo gráfico apresenta as diferenças dos resultados *WER* para as variações do inglês americano e britânico em relação ao *ASR Bing Speech API*. Os valores com menos percentagem *WER* são considerados os melhores.

O segundo grupo de gráficos apresentam uma comparação do número de erros relativos às substituições (S), inserções (I), eliminações (D) dos testes *WER*, para cada ficheiro áudio em relação às variantes do inglês, inglês americano e inglês britânico.

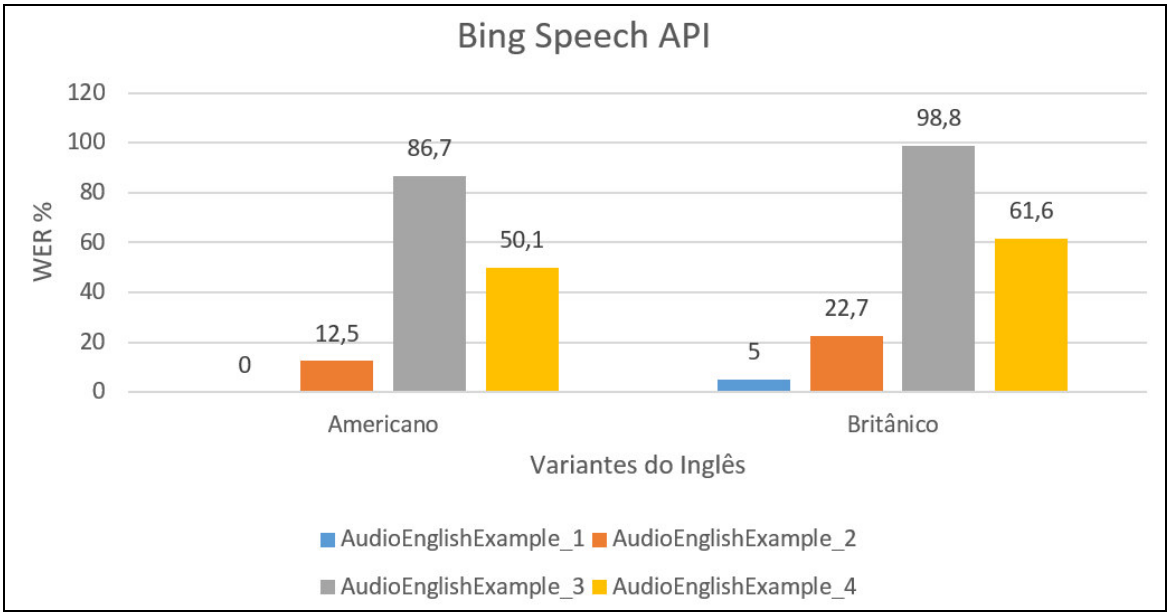

Figura 8. Comparação dos resultados WER entre as variantes, americano e britânico para o ASR Bing Speech API.

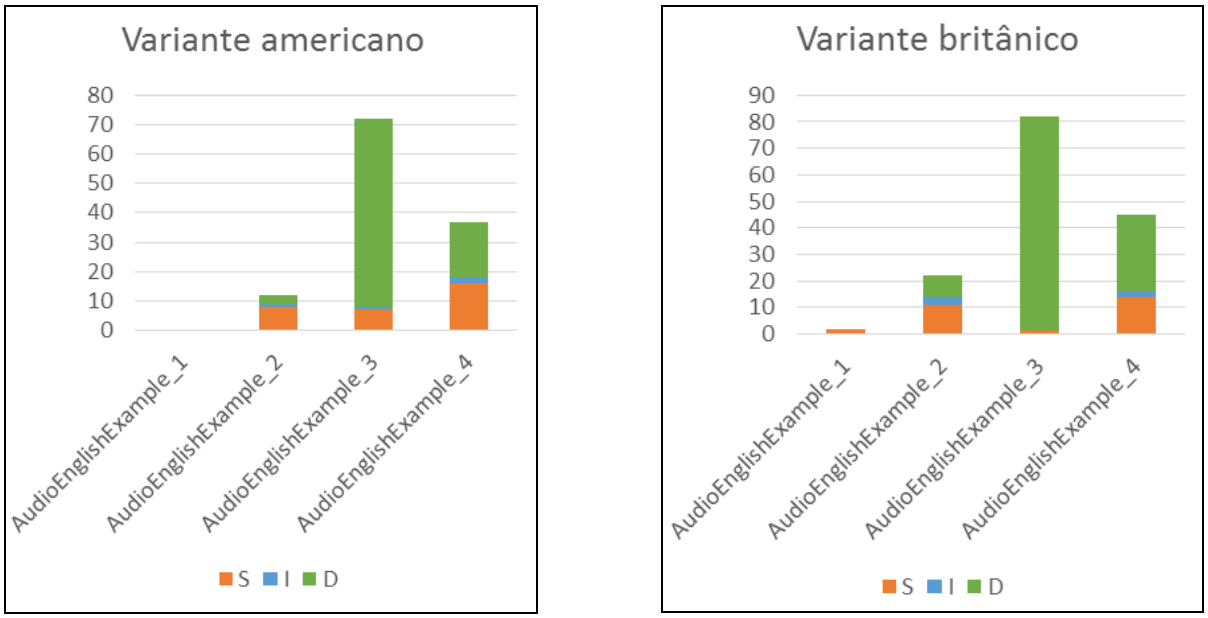

Figura 9. Comparação dos resultados em relação ao número de erros WER para cada variante do inglês, na ASR Bing Speech API.

- Speechmatics

O primeiro gráfico apresenta as diferenças dos resultados *WER* para as variações do inglês americano e britânico em relação ao *ASR Speechmatics*. Os valores com menos percentagem *WER* são considerados os melhores.

O segundo grupo de gráficos apresentam uma comparação do número de erros relativos às substituições (S), inserções (I), eliminações (D) dos testes *WER*, para cada ficheiro áudio em relação às variantes do inglês, inglês americano e inglês britânico.

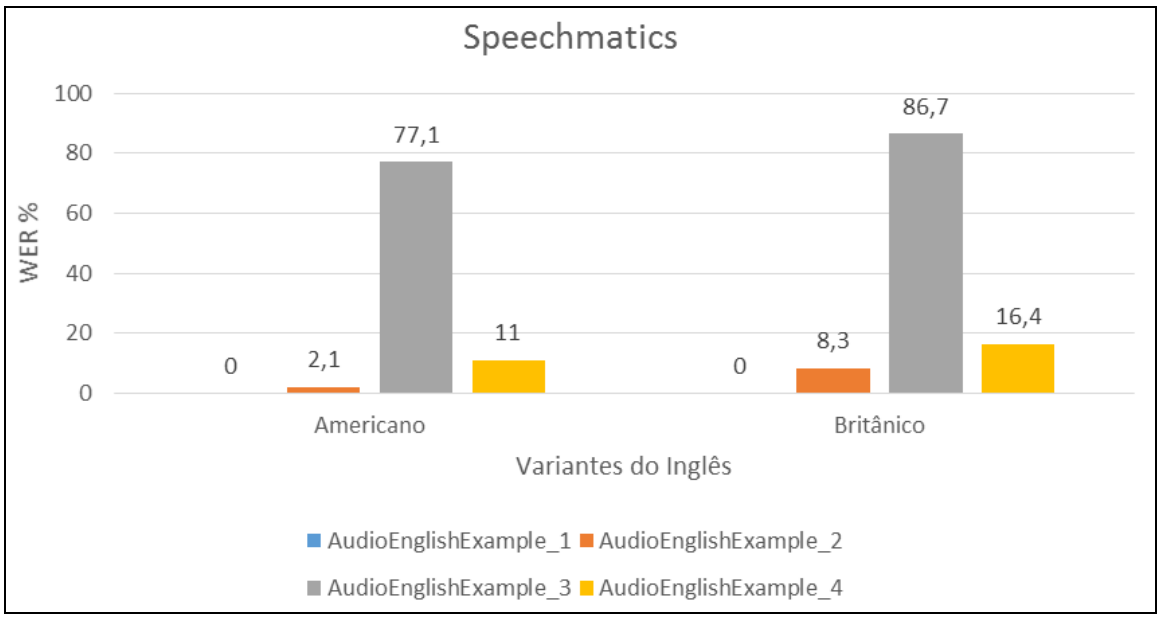

Figura 10. Comparação dos resultados WER entre as variantes, americano e britânico para o ASR Speechmatics.

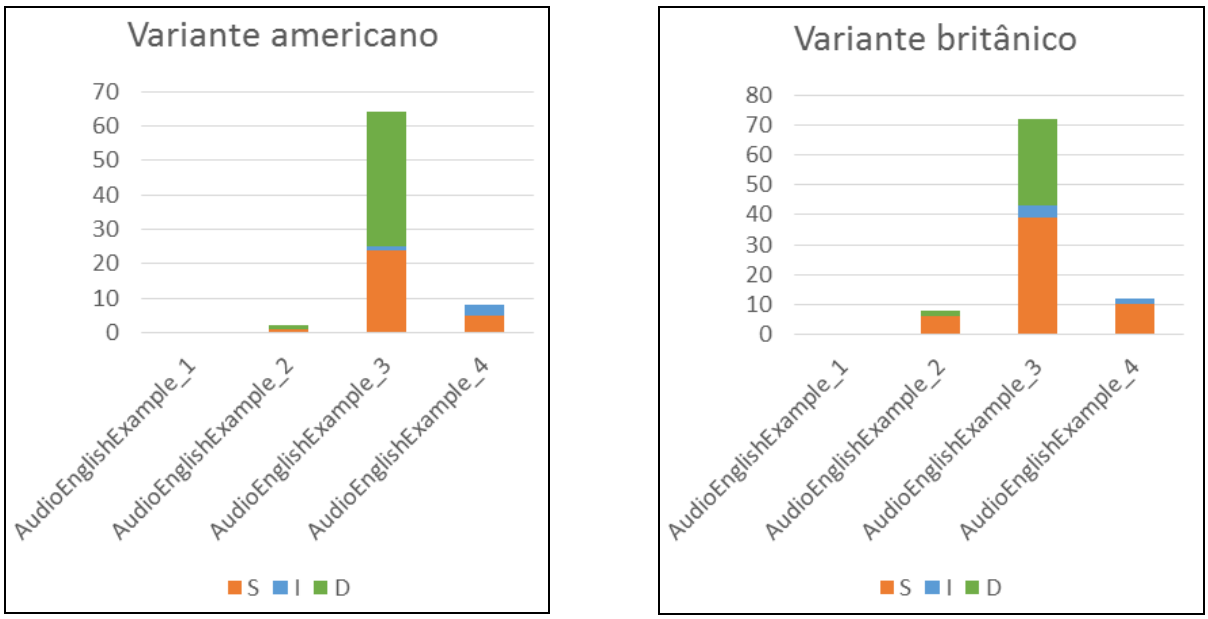

Figura 11. Comparação dos resultados em relação ao número de erros WER para cada variante do inglês, na ASR Speechmatics.

- IBM Watson Speech to Text

O primeiro gráfico apresenta as diferenças dos resultados *WER* para as variações do inglês americano e britânico em relação ao *ASR IBM Watson Speech to Text*. Os valores com menos percentagem *WER* são considerados os melhores.

O segundo grupo de gráficos apresentam uma comparação do número de erros relativos às substituições (S), inserções (I), eliminações (D) dos testes *WER*, para cada ficheiro áudio em relação às variantes do inglês, inglês americano e inglês britânico.

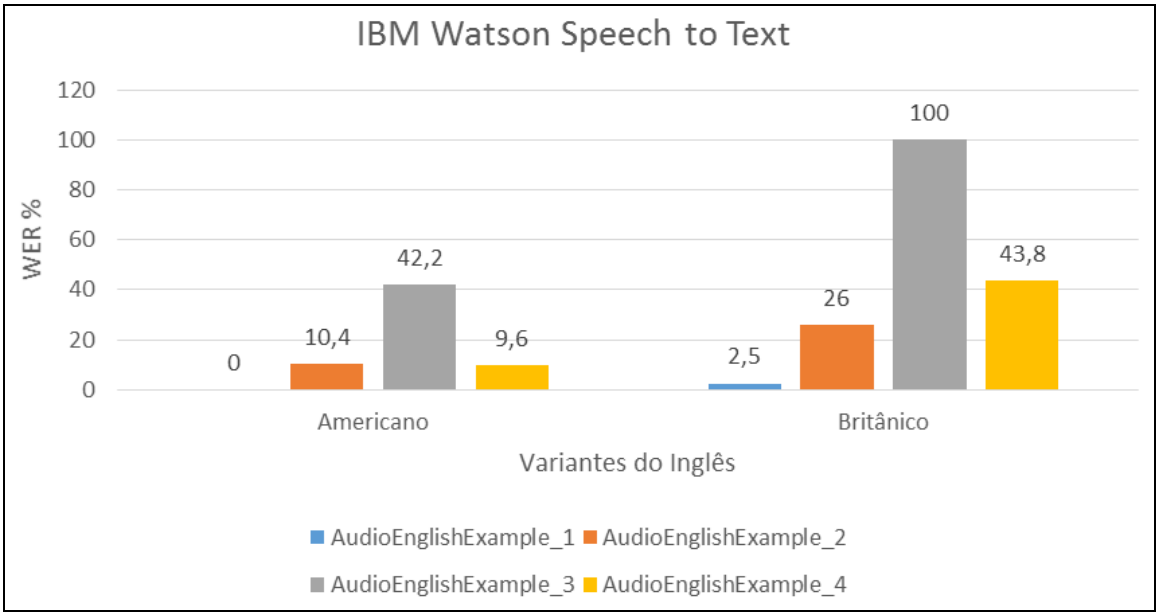

Figura 12. Comparação dos resultados WER entre as variantes, americano e britânico para o ASR IBM Watson Speech to Text.

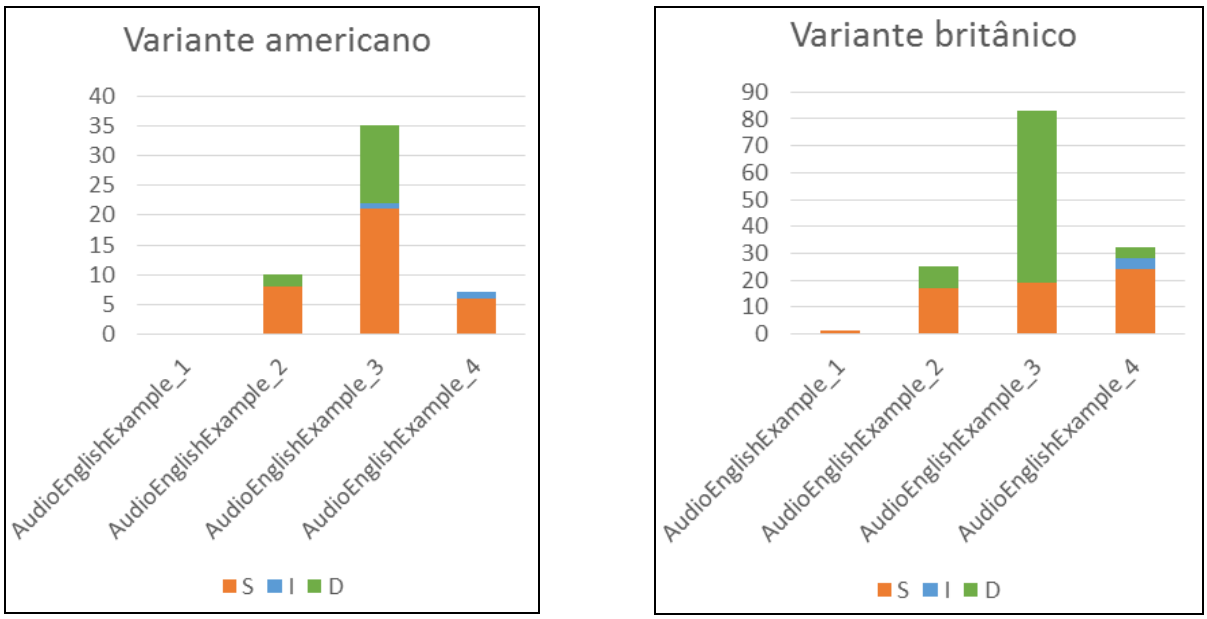

Figura 13. Comparação dos resultados em relação ao número de erros WER para cada variante do inglês, na ASR IBM Watson Speech to Text.

Implementação e teste de um sistema SDS

Com o objetivo de colocar em prática vários dos conceitos apresentados neste trabalho, foi desenvolvido uma pequena aplicação para conversação Homem-máquina. A aplicação deve ter a capacidade de reconhecer a voz Humana interpretá-la e responder ao utilizador. A resposta pode ser feita através de voz sintetizada ou pela realização de tarefas.

Inicialmente foi pensado a utilização de *frameworks*, serviços ou outro tipo de componentes *open source* e totalmente livre de custos. Contudo devido a algumas dificuldades que serão explicadas de seguida, optou-se por utilizar alguns serviços que não são *open source* nem totalmente gratuitos, mas que, devido a outras características, foram considerados os mais indicados para a realização desta aplicação.

Para a implementação da aplicação foram feitas várias tentativas. Na primeira tentativa de implementação, a linguagem de programação escolhida foi *Java* com o intuito de funcionar no maior número de dispositivos possível. A escolha do *ASR* recaiu na *library Sphinx4* [25] da *CMUSphinx*. Esta library perfilava ser a mais indicada para a obtenção dos objetivos. É *open source*, livre de usar, específica para projetos em *Java*, com alguns exemplos disponíveis, alguma documentação, dicionários, com alguns modelos acústicos e de linguagem. A língua a usar no reconhecimento de voz nesta primeira tentativa seria o inglês devido ao *CMUSphinx* não ter na altura desta implementação, modelos e dicionários para a língua portuguesa.
Com base nos exemplos e na documentação disponível, foi relativamente fácil implementar o sistema de *ASR*, contudo os resultados obtidos com o reconhecimento da voz estavam muito longe do que era pretendido. Menos de cinco por cento das palavras transmitidas por voz para a aplicação eram corretamente compreendidas.
Os maus resultados no reconhecimento da voz podiam estar relacionados com vários fatores como por exemplo, a pronúncia do utilizador devido ao inglês não ser a sua língua nativa, má receção do som por parte do microfone, modelos e dicionários inapropriados, entre outros.

Para perceber se o problema poderia estar relacionado com os modelos acústicos e linguísticos e até mesmo com o dicionário, foram feitos vários testes com outros modelos e dicionários disponíveis para a língua inglesa. Contudo os resultados foram sempre muito fracos.

Para validar se o problema estava ou não no microfone, foram utilizados três microfones diferentes. Em todos os testes efetuados com cada microfone os resultados foram sempre muito maus.

A *library Sphinx4* permite também o reconhecimento de voz a partir de ficheiros áudio. Dessa forma, foram efetuados dois testes para perceber se o problema estava na pronúncia do utilizador. No primeiro teste o reconhecimento da voz a partir de um ficheiro áudio foi feito com base num ficheiro fornecido pelos exemplos do *CMUSphinx*. Os resultados foram aceitáveis, transcrevendo corretamente cerca de 80% do áudio. O segundo teste foi feito com base num ficheiro áudio gravado com algumas palavras na língua inglesa ditas pelo utilizador. O resultado deste último teste foi muito mau, nenhuma palavra foi corretamente interpretada. Contudo, como a gravação do ficheiro áudio tem de seguir algumas especificações como, tipo de ficheiro, frequência e número de bits, ficou-se sem perceber se o problema estava na qualidade do ficheiro áudio ou na pronúncia do utilizador. Uma solução para resolver este problema seria treinar um modelo acústico específico para o microfone usado.

Devido às várias incertezas da origem dos maus resultados no reconhecimento da voz do utilizador por parte do *CMUSphinx* e da necessidade de conhecimentos mais avançados necessários, por exemplo, para o treino de modelos, e ao tempo limitado para o desenvolvimento desta aplicação, esta primeira tentativa foi abandonada.

De forma a não ter a preocupação com os modelos linguísticos e fonéticos, dicionários, *tuning* e outras configurações, o *Wit.ai* [30] surgiu como uma opção viável para o reconhecimento de voz. Com base num projeto em *java* o objetivo era utilizar algumas das referências *HTTP API* disponibilizadas por esta plataforma. Para isso foi necessário fazer um registo na plataforma *Wit.ai*. Depois criou-se através de um formulário uma aplicação à qual é atribuído um *App ID*, um *Server Access Token* e um *Client Access Token*. O *Server Access Token* é necessário para obter um *token* autorizado, que depois é usado no *header* de cada chamada às *APIs* pretendidas.

O primeiro teste efetuado consistia em obter o texto de um ficheiro áudio. Este teste foi realizado com os mesmos ficheiros áudio usados na abordagem anterior e foram

concretizados com sucesso ao contrário do que aconteceu anteriormente.
Contudo, devido à falta de exemplos do uso destas *APIs* em *java* e a várias dificuldades que surgiram, como por exemplo, captação da voz em *streaming* ou controlar quando o utilizador termina o discurso, levou que também esta abordagem fosse abandonada.

Como nas duas abordagens anteriores existiam alguns problemas difíceis de solucionar, optou-se por escolher uma nova abordagem que permitiu a implementação de um sistema de conversação Homem-máquina de forma mais acessível. Então, para o desenvolvimento da aplicação numa forma mais rápida e com menos obstáculos, optou-se por explorar os recursos que a *Speech API da Bing* [35] disponibiliza. Esta *API,* para além do reconhecimento de voz, disponibiliza também a funcionalidade de *TTS*.
A implementação foi feita com base em exemplos disponibilizados pela *Microsoft*, em *C#*, num projeto do tipo *Windows Presentation Foundation* (*WPF*). O acesso às ferramentas de *ASR* é feito através de uma *cliente library*, enquanto o acesso ao serviço de *TTS* é feito através de uma *REST API*. Neste projeto foram juntos dois exemplos disponibilizados pela *Microsoft*, um para reconhecimento de voz e outro para a sintetização de voz. Depois foi criado um *Dialogue Management* básico de forma a gerir a comunicação. O resultado final foi um sistema funcional de conversação Homem-máquina.
A utilização dos serviços de *ASR* e *TTS* do *Speech API* da *Bing* são gratuitos até atingir determinados valores, 5.000 transações por mês para ASR e outros 5000 para TTS.

A imagem seguinte representa o aspeto gráfico da aplicação e encontra-se marcado com uma numeração de forma a identificar as partes mais importantes do seu ambiente gráfico. A marcação com o número um (1) identifica o botão que depois de ser selecionado dá início à comunicação por voz entre o utilizador e a máquina. A marcação com o número dois (2) identifica o botão que depois de selecionado, termina com a comunicação. Por último, a marcação com o número três (3) identifica a área onde toda a comunicação é representada por escrito, ou seja, o texto que foi reconhecido e a resposta da aplicação.

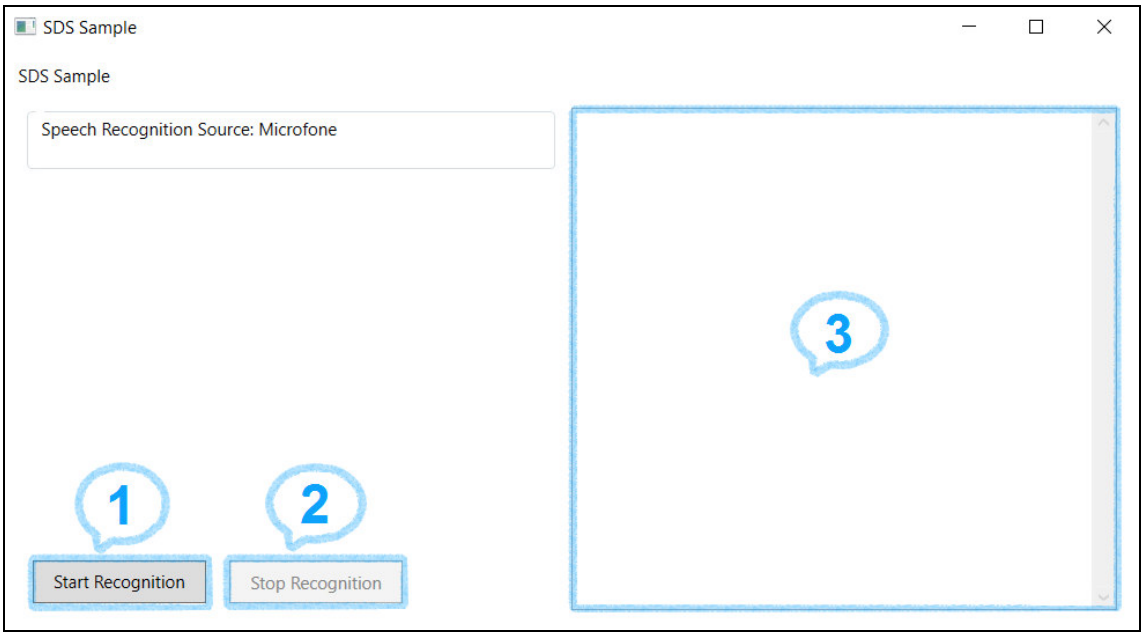

Figura 14. Layout da aplicação de conversação Homem-máquina.

A aplicação permite que um utilizador possa comunicar com a máquina, ou computador onde a aplicação está a ser executada. A voz do utilizador é compreendida com bastante precisão e, conforme algumas sequências de palavras compreendidas, a aplicação responde ou executa tarefas.

A imagem seguinte apresenta um exemplo de interação entre o utilizador e a aplicação. Neste exemplo o utilizador transmite o comando por voz "Obter horas", que é corretamente interpretado pela aplicação (retângulo azul). A taxa de sucesso registada, depois da realização de 20 tentativas, com o comando "Obter horas" foi de 90%.

A aplicação, depois de reconhecer o comando, responde por voz e também apresenta a mensagem da resposta por texto (retângulo verde).

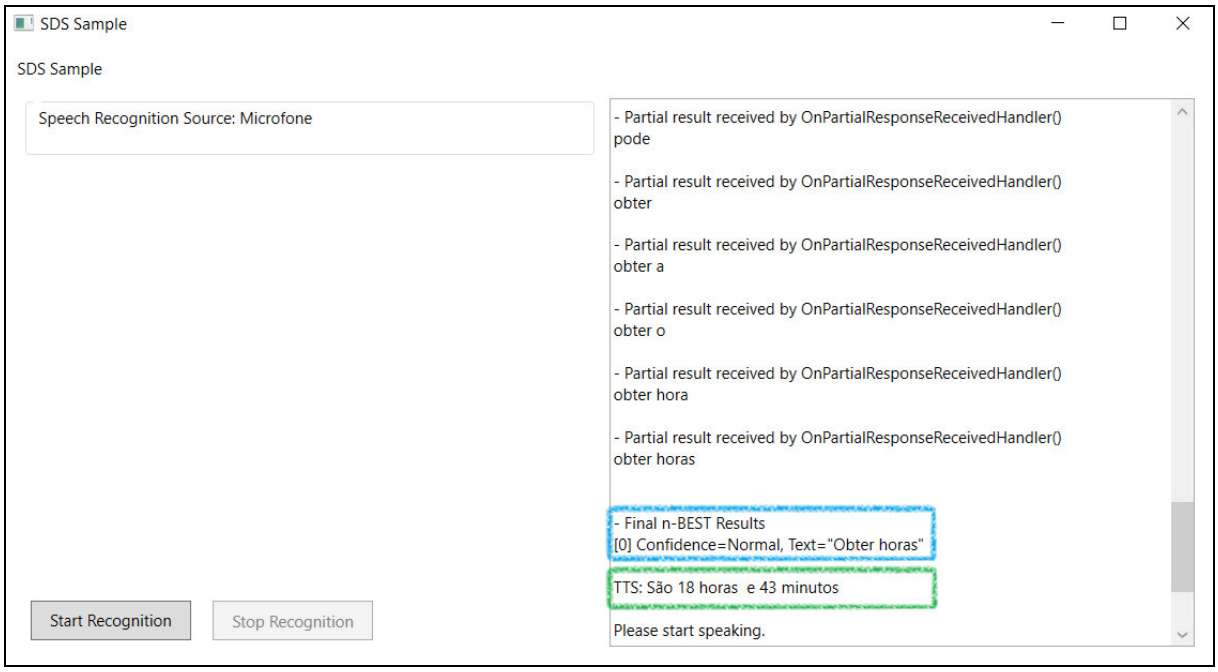

Figura 15. Aplicação de conversação Homem-máquina em funcionamento.

Discussão de Resultados

Pelos testes *Word Error Rate* é possível verificar a qualidade dos resultados obtidos para cada sistema *ASR* testado. Este método de avaliação permitiu avaliar a performance de cada sistema *ASR* relativamente a quatro conteúdos de áudio que representavam quatro níveis de dificuldades diferentes.

Pelos resultados obtidos verificou-se que os fatores: ruídos de fundo, uso de entoações, transmissão da voz de forma rápida ou mais lenta, tipo de pronúncia e qualidade da voz, são condicionantes que têm bastante relevo na procura dos melhores resultados pelos sistemas *ASR*.

Pelos valores recolhidos pode-se verificar que existe um padrão em relação ao nível de dificuldade de cada ficheiro. A ordem de dificuldade cresce pela seguinte ordem dos ficheiros, "*AudioEnglishExample_1*", "AudioEnglishExample_2", "*AudioEnglishExample_4*" e "*AudioEnglishExample_3*".

Estes testes provam que os sistemas *ASR* tem ainda um grande caminho a percorrer no objetivo de reconhecimento de voz. Podemos chegar a essa conclusão, principalmente pelos resultados obtidos a partir do ficheiro "AudioEnglishExample_3". Todos os sistemas *ASR* tiveram grandes dificuldades para o transcrever e os seus resultados foram péssimos. Contudo, um humano com conhecimentos de inglês, facilmente consegue entender a grande maioria do conteúdo do ficheiro áudio.

Se analisarmos os resultados de cada sistema *ASR* em relação aos valores *WER* obtidos para cada ficheiro, podemos concluir que, para o primeiro ficheiro os valores obtidos são razoáveis para todos os sistemas, com especial relevo para os *ASR Bing Speech API, Speechmatics* e *IBM Watson Speech to Text*, que não obterão qualquer erro. De referir que a média de resultados foi de 4.2%.

Em relação ao segundo ficheiro de áudio os resultados pioraram, sendo o *ASR Speechmatics* o que obteve melhores resultados.

O terceiro ficheiro era o mais difícil no processo de reconhecimento de voz e os resultados

foram de uma forma geral muito maus, com valores na média dos 78.1%. Devido ao elevado nível de dificuldade deste ficheiro, não foi possível efetuar o reconhecimento de voz com o *ASR Wit.ai* e apresentar os resultados do *WER*. O *ASR* com melhores resultados neste ficheiro foi o *IBM Watson Speech to Text*, com valores mais abaixo dos restantes *ASR*.
Relativamente ao quarto ficheiro, os testes também foram maus, mas não tão maus como os resultados do terceiro ficheiro com uma média de 44.8% de erro, devido ao nível de dificuldade ser também menor. O *ASR* com melhores resultados neste desafio acabou por ser o *IBM Watson Speech to Text* com valores aceitáveis.
De uma forma geral, se analisarmos o desempenho de todos os sistemas *ASR* testados em relação ao *WER*, pode-se concluir que o sistema *IBM Watson Speech to Text* foi o que apresentou melhor performance no reconhecimento de voz. Os seus resultados, com exceção dos obtidos sobre o segundo ficheiro, foram sempre os melhores.

Um facto também importante está relacionado com a diferença de resultados que podem surgir apenas com a alteração da variante de uma linguagem. Como foi apresentado nos resultados dos testes efetuados, os valores da avaliação WER são diferentes para as variantes da língua inglesa testadas, sendo mais satisfatórios para a variante americana devido às vozes presentes nos ficheiros terem características mais idênticas com a língua inglesa dos Estados Unidos da América. Com estes resultados podemos concluir que os sistemas ASR não são muito flexíveis.

Na elaboração da aplicação foram analisadas três abordagens para o reconhecimento de voz. Essa análise permitiu compreender algumas vantagens e desvantagens de cada uma. Se por um lado o *Sphinx4* tem a vantagem de ser totalmente livre e poder ser usado em modo *offline* sem qualquer ligação à internet, por outro, requer um conhecimento mais avançado para casos em que seja necessário a criação de modelos linguísticos, acústicos e dicionários, e também para treinar esses modelos.
Em relação aos *ASR* da *Wit.ai* e do *Bing* pode-se considerar como desvantagem a dependência de uma ligação à internet, uma vez que ambos usam serviços *cloud-based* e todos os sistemas de reconhecimento e de sintetização estão num servidor remoto. A grande vantagem desse aspeto é não haver a necessidade de criar e treinar modelos de linguagem e acústicos e de criar dicionários, deixando os programadores mais concentrados na verdadeira lógica da aplicação, devido a todo este trabalho ser feito pelos fornecedores dos serviços. Para além disso a maioria destes serviços tem suporte para várias linguagens.

Conclusão

A fala é o principal meio, e o mais conveniente, de comunicação entre os Humanos e, por isso, é necessário que os sistemas de comunicação entre Homem-máquina permitam que a comunicação seja o mais natural possível.

Para ultrapassar os desafios relacionados com as interfaces baseadas na comunicação por voz e permitir aos utilizadores usufruir das suas funcionalidades na resolução de tarefas do dia-a-dia, é necessário continuar a melhorar estes sistemas, procurar novas estratégias e realizar novos desenvolvimentos.

Seja devido às curiosidades tecnológicas para construir máquinas que imitam humanos ou por desejos de automatizar o trabalho com máquinas, as pesquisas realizadas em torno da fala e reconhecimento de voz, têm atraído muitos entusiastas que, ao longo dos anos, têm desenvolvido a comunicação Homem-máquina.

Ainda se está longe de chegar a alguns cenários que vemos nos filmes de ficção científica, no entanto, as máquinas já nos conseguem entender razoavelmente bem e desempenhar bastantes tarefas.

O uso de *SDS* está a crescer rapidamente. Várias aplicações comerciais e *start-ups* estão a multiplicar-se e as perspetivas de crescimento do mercado são elevadas. Os utilizadores irão aprender a usá-las de uma forma adaptativa, à medida que vão surgindo novos ecossistemas, novas tecnologias, novos poderes de computação, e novos casos de uso.

6.1 Trabalho futuro

Depois de apresentados vários conceitos presentes em sistema de conversação Homem-máquina, seria interessante implementar uma *framework* multiplataforma capaz de dotar aplicações com o controlo por voz.

CMUSphinx é uma ferramenta *open source* muito usada para projetos de investigação e projetos escolares. Outro desafio interessante seria o desenvolvimento de modelos linguísticos, acústicos e dicionários para o *CMUSphinx.*

Para dar continuidade ao trabalho desenvolvido nesta dissertação, mais testes podem ser realizados sobre os sistemas de ASR, como por exemplo, incluir mais sistemas à bateria de testes e usar a voz diretamente invés de ficheiros áudio.

Bibliografia

[1] Lai WEI, Huosheng HU, ”Towards Multimodal Human-Machine Interface for Hands-free Control: A survey”, Technical Report: CES–510, January 2011;

[2] Raveesh Meena, "Data-driven Methods for Spoken Dialogue Systems", Stockholm: KTH Royal Institute of Technology, 2016;

[3] Victor W. Zue, James R. Glass, “Conversational Interfaces: Advances and Challenges”, Published in: Proceedings of the IEEE (Volume: 88, Issue: 8, Aug. 2000);

[4] Dominique BEROULE, “Vocal Interface for a Man-Machine Dialog”, Published in: Proceeding EACL '83 Proceedings of the first conference on European chapter of the Association for Computational Linguistics;

[5] Michael F. McTear, "Spoken Dialogue Technology: Enabling the Conversational User Interface", Published in: Journal ACM Computing Surveys (CSUR) Volume 34 Issue 1, March 2002;

[6] Masahiro Araki, Daisuke Takegoshi, “A Rapid Development Framework for Multilingual Spoken Dialogue Systems”, Published in Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual;

[7] Steve Young, Milica Gašić, Blaise Thomson, Jason D. Williams, “POMDP-based Statistical Spoken Dialogue Systems: a Review”, Published in: Proceedings of the IEEE (Volume: 101, Issue: 5, May 2013);

[8] Blaise Thomson, “Statistical methods for spoken dialogue management”, Publisher Springer-Verlag London, DOI 10.1007/978-1-4471-4923-1, 2013;

[9] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, Steve Young, "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies", The Knowledge Engineering Review, Vol. 00:0, 1–24. 2006, Cambridge University Press;

[10] Mats Blomberg, Rolf Carlson, Kjell Elenius, Björn Granström, Joakim Gustafson, Sheri Hunnicutt, Roger Lindell, Lennart Neovius and Lennart Nord, “An Experimental Dialogue System: Waxholm”, Conference: Third European Conference on Speech Communication and Technology, EUROSPEECH 1993, Berlin, Germany, September 22-25, 1993;

[11] Jinyu Li, Li Deng, Yifan Gong, Reinhold Haeb-Umbach, "An Overview of Noise-Robust Automatic Speech Recognition", Published in: IEEE/ACM Transactions on Audio, Speech, and Language Processing (Volume: 22, Issue: 4, April 2014);

[12] M.A.Anusuya, S.K.Katti, "Speech Recognition by Machine: A Review", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 6, No. 3, 2009;

[13] Li Deng, Xiao Li, "Machine Learning Paradigms for Speech Recognition: An Overview", Published in: IEEE Transactions on Audio, Speech, and Language Processing (Volume: 21, Issue: 5, May 2013);

[14] Shinya Takahashi, Tsuyoshi Morimoto, “N-gram Language Model Based on Multi-Word Expressions in Web Documents for Speech Recognition and Closed-Captioning”, Published in: Asian Language Processing (IALP), 2012 International Conference;

[15] Suman Ravuri, Andreas Stolcke, "A comparative study of recurrent neural network models for lexical domain classification", Published in: Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference;

[16] Lucie Daubigney, Matthieu Geist, Olivier Pietquin, "Off-policy learning in large-scale POMDP-based dialogue systems", Published in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference;

[17] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, Alicia Abella, “PARADISE: A Framework for Evaluating Spoken Dialogue Agents”, Proceeding EACL '97 Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, Pages 271-280;

[18] Tatjana Scheffler, Roland Roller, Florian Kretzschmar, Sebastian Moller, Norbert Reithinger, “Natural vs. Synthesized Speech in Spoken Dialog Systems Research - Comparing the Performance of Recognition Results”, Published in: Speech Communication; 10. ITG Symposium, 25 September 2012;

[19] Hatim Khouzaimi, Romain Laroche, Fabrice Lefèvre, “Optimising Turn-Taking StrategiesWith Reinforcement Learning”, SIGDIAL Conference 2015;

[20] Maxim Sidorov, Alexander Schmitt, Sergey Zablotskiy, Wolfgang Minker, "Survey of Automated Speaker Identification Methods", Published in: Intelligent Environments (IE), 2013 9th International Conference;

[21] John H. L. Hansen, Sahar E. Bou-Ghazale, "Robust Speech Recognition Training via Duration and Spectral-Based Stress Token Generation", Published in: IEEE Transactions on Speech and Audio Processing (Volume: 3, Issue: 5, Sep 1995);

[22] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, Brian Kingsbury, “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”, Published in: IEEE Signal Processing Magazine (Volume: 29, Issue: 6, Nov. 2012);

[23] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geojj Zweig, Xiaodong He, Jason Williams, Yifan Gong, Alex Acero, "Recent advances in deep learning for speech research at Microsoft", Published in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference;

[24] C. Gaida, P. Lange, R. Petrick, P. Proba, A. Malatawy and a. D. Suendermann-Oeft, "Comparing Open-Source Speech Recognition Toolkits", Organisation of Alberta Students in Speech, Alberta, 2014;

[25] CMUSphinx, http://cmusphinx.sourceforge.net/

[26] Nuance, http://www.nuance.com/index.htm

[27] Kaldi, http://kaldi-asr.org/

[28] OpenEars, http://www.politepix.com/openears/

[29] Hidden Markov Model Toolkit, http://htk.eng.cam.ac.uk/

[30] Wit.ai, https://wit.ai/

[31] Speechmatics, https://speechmatics.com/

[32] VoxSigma, http://www.vocapia.com/speech-to-text-api.html

[33] AUDIMUS, https://www.l2f.inesc-id.pt/wiki/index.php/AUDIMUS

[34] AUDIMUS.SERVER, http://www.voiceinteraction.pt/?page_id=374

[35] Bing Speech API, https://www.microsoft.com/cognitive-services/en-us/speech-api

[36] Annyang, https://www.talater.com/annyang/

[37] MDN Web Speech API, https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[38] Python SpeechRecognition, https://pypi.python.org/pypi/SpeechRecognition/

[39] IBM Watson Speech to Text, http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html

[40] iSpeech API, http://www.ispeech.org/api

[41] API.AI, https://api.ai/

[42] LumenVox ASR, https://www.lumenvox.com/store/asr-sdk/

[43] EduSpeak Speech Recognition Toolkit, https://www.sri.com/engage/products-solutions/eduspeak-speech-recognition-toolkit

[44] Julius, http://julius.osdn.jp/en_index.php

[45] Pocketsphinx.js, http://syl22-00.github.io/pocketsphinx.js/

[46] Ceedvocal, http://www.creaceed.com/ceedvocal

[47] BitVoicer Server, http://www.bitsophia.com/en-US/BitVoicerServer/Overview.aspx

[48] Google Cloud Speech API, https://cloud.google.com/speech/

[49] Paul Taylor, “TexttoSpeech Synthesis”, Date Published: March 2009;

[50] Youcef Tabet, Mohamed Boughazi, "Speech Synthesis Techniques. A Survey", Published in: Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop;

[51] Zhenli Yu, Dongjian Yue, Yiqing Zu, Guilin Chen, “Word Intelligibility Testing and TTS System Improvement”, Published in Signal Processing (ICSP), 2010 IEEE 10th International Conference on 24-28 Oct. 2010;

[52] Florian Hinterleitner, Christoph Norrenbrock, Sebastian Möller, Ulrich Heute, "What Makes This Voice Sound So Bad? A Multidimensional Analysis Of State-Of-The-Art Text-To-Speech Systems", Published in: Spoken Language Technology Workshop (SLT), 2012 IEEE;

[53] SSML, https://www.w3.org/TR/2010/REC-speech-synthesis11-20100907/#S1.1

[54] PLS, https://www.w3.org/TR/pronunciation-lexicon/#S1

[55] https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

[56] DIXI, https://www.l2f.inesc-id.pt/wiki/index.php/DIXI

[57] DIXI.SERVER, http://www.voiceinteraction.pt/?page_id=367

[58] IBM Watson Text to Speech, http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/text-to-speech.html

[59] Festival Speech Synthesis System, http://www.cstr.ed.ac.uk/projects/festival/

[60] Flite (festival-lite), http://www.speech.cs.cmu.edu/flite/

[61] LumenVox TTS, https://www.lumenvox.com/store/tts-sdk/

[62] MaryTTS, http://mary.dfki.de/index.html#

[63] Vocalware, https://www.vocalware.com/#&panel1-1

[64] SpeakRight, http://speakrightframework.blogspot.pt/

[65] FreeTTS, http://freetts.sourceforge.net/docs/index.php

[66] IVONA, https://www.ivona.com/us/

[67] eSpeak, http://espeak.sourceforge.net/

[68] ReadSpeaker, http://www.readspeaker.com/

[69] Joe Buzzanga, "Beyond Keywords: The Revolution in Search", SLA Contributed Papers, Session 1885 June 15, 2015;

[70] Robert Dale, "The limits of intelligent personal assistants", Natural Language Engineering, Volume 21, Issue 2: 325–329. Cambridge University Press 2015;

[71] Nil Goksel-Canbek, Mehmet Emin Mutlu, "On the track of Artificial Intelligence: Learning with Intelligent Personal Assistants", International Journal of Human Sciences, Volume: 13 Issue: 1 Year: 2016;

[72] Feng Lin, Fuliang Weng, "Computing confidence score of any input phrases for a spoken dialog system", Published in: Spoken Language Technology Workshop (SLT), 2010 IEEE;

[73] Siri, https://www.apple.com/ios/siri/

[74] https://www.sri.com/work/timeline-innovation/timeline.php?timeline=computing-digital#!&innovation=siri

[75] Cortana, https://www.microsoft.com/en-us/mobile/experiences/cortana/

[76] Google Now, http://www.google.co.uk/landing/now/

[77] S Voice, http://www.samsung.com/ae/discover/go-hands-free-with-s-voice-for-your-mobile-device

[78] Nina, http://www.nuance.com/for-business/customer-service-solutions/nina/index.htm

[79] Vlingo, http://www.vlingo.com/

[80] Viv, http://viv.ai/

[81] http://www.wired.com/2014/08/viv/

[82] Alexa Amazon, https://developer.amazon.com/public/solutions/alexa

[83] Amazon Echo, http://www.amazon.com/dp/B00X4WHP5E

[84] BlackBerry Assistant, https://help.blackberry.com/en/blackberry-classic/10.3.1/help/amc1403813518716.html

[85] Braina, https://www.brainasoft.com/braina/

[86] HTC Hidi, http://ready2beat.com/technology/mobile/htc-hidi-siri-voice-assistant-htc/

[87] Maluuba, http://www.maluuba.com/

[88] https://www.technologyreview.com/s/601066/software-that-reads-harry-potter-might-perform-some-wizardry/

[89] http://vator.tv/news/2016-01-20-maluuba-raises-62m-series-a-for-machine-learning

[90] Cognitive Code's 'SILVIA', http://silvia4u.info/

[91] IBM's Watson, http://www.ibm.com/smarterplanet/us/en/ibmwatson/what-is-watson.html

[92] Facebook M, https://www.theguardian.com/technology/2015/aug/27/facebook-m-virtual-assistant-siri-google-now

[93] Mindmeld, https://mindmeld.com/

[94] Duolingo, https://www.duolingo.com/
[95] Dragon Assistant, http://www.nuance.com/dragon/dragon-assistant/index.htm
[96] Indigo, http://www.hello-indigo.com/
[97] Voice Actions, http://voice-actions.com/
[98] Voice Search da Soundhound, http://www.soundhound.com/hound
[99] AVX, EVA, EVAN Voice assistent, http://eva4android.com/
[100] Evi, https://www.evi.com/
[101] Andy X, http://www.etx.ca/products/android-applications-published/
[102] Memrise, https://www.memrise.com/
[103] Ariadne Spoken Dialogue System, http://www.opendialog.org/about.html
[104] Olympus, http://wiki.speech.cs.cmu.edu/olympus/index.php/Olympus
[105] SEMAINE, http://www.semaine-project.eu/
[106] TRINDI, http://www.ling.gu.se/projekt/trindi/
[107] PED, http://planeffdia.sourceforge.net/
[108] HALEF, http://sail.usc.edu/~vramanar/sds.html
[109] HOWe, https://sourceforge.net/projects/howe/
[110] Regulus, https://sourceforge.net/projects/regulus/
[111] Perlbox.org, http://perlbox.sourceforge.net/
[112] Alex, https://github.com/UFAL-DSG/alex
[113] Jindingo, http://www.speech.kth.se/jindigo/
[114] Ye-Yi Wang, A. Acero, C. Chelba, "Is word error rate a good indicator for spoken language understanding accuracy", Published in: Automatic Speech Recognition and Understanding, 2003. ASRU '03. 2003 IEEE Workshop;
[115] Gabriel Skantze, "Error Handling in Spoken Dialog Systems", Doctoral Thesis Stockholm, Sweden 2007;
[116] Speechmatics, https://www.speechmatics.com/dashboard
[117] iSpeech, http://www.ispeech.org/speech.recognition.demo
[118] IBM Watson Speech to Text, https://speech-to-text-demo.mybluemix.net/

Apêndices

Resultados_WER.xslx;