INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Modulo que permita assistir os médicos no bloco de operatório a partir de comandos de voz

MA LUOEN
novembro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Reconhecimento de fala e processamento de linguagem natural

Módulo que permita assistir os médicos no bloco de operatório a partir de comandos de voz

Ma Luoen

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Filipe De Faria Pacheco Paulo

Supervisor: Francisco Correia

Agradecimentos

Este trabalho traduz um percurso de aprendizagem com a colaboração de algumas pessoas a quem gostaria de manifestar o meu apreço.

Assim, agradeço a todos os que colaboraram e tornaram possível este trabalho:

- Aos meus pais pela oportunidade dada.
- A minha namorada pelas horas que me ajudou e encorajou;
- Aos colegas do trabalho pelas ajudas.
- Ao professor Filipe de Faria Pacheco pela disponibilidade na orientação deste projeto.
- Ao Francisco Correia por toda a ajuda disponibilizada na implementação do projeto e na construção deste documento.

Resumo

A introdução de tecnologia no mundo hospitalar veio aumentar as infeções relacionadas com os cuidados de saúde, uma vez que todos os colaboradores sejam eles médicos, enfermeiros ou técnicos, de alguma forma são quase obrigados a ter algum tipo de contacto com os equipamentos tecnológicos devido as necessidades das suas tarefas diárias.

Nem todas as infeções associadas aos cuidados de saúde são evitáveis, todavia, uma proporção significativa pode ser prevenida se os profissionais de saúde possuírem boas práticas no contexto da prevenção, procedendo a desinfeção dos equipamentos ou recorrendo a um assistente para interagir com os equipamentos em determinados contextos (exemplo: bloco operatório).

Atualmente as metodologias adotados pelos hospitais para desinfeção dos equipamentos ainda demonstram algumas falhas e são procedimentos que consomem bastante tempo ao serem executados. As interfaces no-touch oferecem uma oportunidade para resolver este problema. No entanto, os recursos para desenvolver estas interfaces são escassos, sendo que filmes de ficção científica ainda são dos melhores pontos de referência disponíveis para desenhar a interação.

Foi feita uma colaboração com a Glintt HS num contexto de bloco operatório, para que com base no caso de uso fornecido desenvolver um protótipo para adaptar facilmente as operações de pesquisa num interface web para uma interface no-touch que assenta sobre tecnologias web. Este módulo irá permite aos colaboradores efetuar pesquisas pelo historial clínico e consultar resultados.

Este documento irá detalhar o estado de arte das áreas envolventes, bem como alguns aspetos fundamentais para o desenvolvimento do protótipo tais como o design, arquitetura e testes a realizar e métodos de avaliação a aplicar.

Abstract

With the introduction of technology in hospital the number of health care associated infections has increased. Since all of the employees somehow they are almost required to have some kind of contact with technological equipment due to the needs of their daily tasks.

Not all health care associated infections are avoidable, however a significant proportion can be prevented if health professionals possess good practice in the prevention, carry out disinfection of the equipment or use a wizard to interact with the equipment in certain contexts (eg operating room).

Currently the methodologies adopted by hospitals for disinfection of equipment still shows some imperfection and time consuming when executed. The no-touch interfaces provide an opportunity to solve this problem. However, the resources to develop these interfaces are scarce, and science fiction movies like Minority Report are still the best reference points available to draw the interaction.

A collaboration was made with Glintt HS to develop a prototype easily adapt the search operations in a web interface to an no-touch interface using web technologies based on use case provided. This module will allow the users to browse through the patient's clinical history and view results.

This document will detail the state of the art of the surrounding areas as well as some fundamental aspects for the prototype development such as prototype design, testing to conduct and evaluation methods to apply.

Conteúdo

Lista de figuras

Lista de tabelas

Lista de siglas e abreviaturas

PLN – Processamento de linguagem natural

ASR – Reconhecimento automático de fala (ou voz).

TTS – Text to speech (conversão de texto em voz).

Parsing – Analise sintática

IACS – Infeções associados a cuidados de saúde

1. Introdução

1.1 Interpretar o problema a resolver

O aumento de uso de equipamentos tecnológicos num ambiente hospitalar veio aumentar as ***IACS*** (infeções associados aos cuidados de saúde), [12][15] sobretudo em locais como o bloco operatório onde muitas das vezes é imprescindível aos profissionais de saúde a utilização de dispositivos como o rato, teclado ou touch-screen para aceder às funcionalidades das aplicações com o objetivo de consultar historial clinico dos pacientes. Neste sentido e seguindo a nova tendência de serviços como Siri da Apple ou Cortana da Microsoft, a Glintt pretende criar um módulo integrável para as suas aplicações web que permita assistir os médicos a partir de comandos de voz. É fundamental que a solução seja desenhada de forma a ser facilmente integrada nas aplicações web de Glintt.

Outro dos requisitos é não limitação de processamento de comandos de voz, a solução deve ser capaz de criar um contexto, manter esse contexto e processar os comandos em linguagem natural nesse mesmo contexto, permitindo ao utilizador um "discurso" fluído e lógico.

Os maiores desafios do projeto proposto são o tratamento da linguagem natural e a definição do módulo de contextualização. O tratamento de linguagem natural é extremamente complexo, no entanto nos últimos anos houve um avanço significativo nesta área e surgiram novas tecnologias que podem ajudar a atingir um bom nível de usabilidade.

Pretende-se que seja elaborado um estudo sobre o estado da arte nesta área e se defina uma solução tecnológica exequível.

Relativamente à contextualização, é necessário que a solução desenvolvida consiga manter um contexto ao longo do tempo de utilização. Por exemplo o caso de um médico aceder aos resultados de exames de um doente:

- Input Médico: "Mostrar resultados do doente 665443"
- Sistema: Cria contexto - doente 665443. Mostra uma lista com todos os resultados de exames do doente 665443.
- Input Médico: "Mostrar apenas os resultados do último mês"
- Sistema: Utiliza contexto - doente 665443. Mostra uma lista de resultados de exames do último mês do doente 665443. Atualiza contexto - doente 665443 e datas últimos 30 dias
- Input Médico: "Apenas os de Imagiologia"
- Sistema: Utiliza contexto - doente 665443 e datas últimos 30 dias. Mostra uma lista de resultados de imagiologia do último mês do doente 665443. Atualiza contexto - doente 665443, datas últimos 30 dias especialidade Imagiologia

1.2 Objetivos

Os objetivos deste projeto podem ser traduzidos nos pontos seguintes:

- Efetuar uma investigação sobre o estado da arte sobre as tecnologias de reconhecimento de voz e definir uma solução tecnológica exequível.
- Tempo de resposta do algoritmo de PLN deve ser inferior a 1 segundo.
- O protótipo desenvolvido deve ser capaz de conhecer as seguintes formato de datas:
    - Dia [1-30/31] de [Janeiro-Dezembro] de [ano]
    - Entre o dia (data1) e o dia (data2)
    - Desde o dia (data)
    - Desde ontem
    - Entre ontem e hoje
    - Últimos X dias
    - De ontem
- Desenvolvimento de um protótipo que seja capaz de contextualizar o discurso e reconhecer no mínimo as seguintes frases:
    - Pesquisar o doente XPTO;
    - Pesquisar o prestador XPTO;
    - Pesquisar o doente XPTO1 e o doente XPTO2;
    - Pesquisar o exame do doente XPTO;
    - Mostrar o resultado do doente XPTO do dia (formatos de data identificado anteriormente);

1.3 Benefícios

Com a introdução desta nova funcionalidade nas suas aplicações a Glintt pertente beneficiar os seus clientes nos seguintes portos:

- Tornar a interação aplicação-humana mais fácil, intuitivo e natural. A interação com a aplicação será facilitado num bloco de operatório quando o prestador do serviço de saúde pretender pesquisar dados relativo ao doente. Porque em vez do prestador parar o que está a fazer e deslocar-se até ao computador e executar a tarefa de pesquisa na aplicação, pode simplesmente pronunciar o que é pretendido e a aplicação irá fazê-lo.

- Pretende reduzir os custos imputados aos seus clientes na aquisição dos equipamentos com características especiais. Por exemplo um tablet usado num bloco de operatório tem uma particularidade de ser impermeável (lavável) para poder ser devidamente esterilizada, porém estes equipamentos traduzem-se num custo aproximadamente de 1000€/unidade. (Com a introdução desta nova funcionalidade a Glintt pretende reduzir a necessidade na aquisição destes equipamentos).

- Redução de custos derivados das infeções associadas aos cuidados de saúde.

1.4 Estrutura do documento

Além da introdução este documento contem mais três capítulos. No capítulo 2 será detalhado os aspeto teórico das áreas envolventes. No capítulo 3 será apresentado os testes realizados as tecnologias identificadas bem como os aspeto técnico do desenvolvimento do protótipo e os teste realizados. Por último, no capítulo 4 são apresentadas as conclusões e os trabalhos futuros.

2. Fundamentos e conhecimentos teóricos

O protótipo a desenvolver deve ser facilmente integrável dotando as aplicações existentes da capacidade de recolha e da conversão dos dados áudios em formato de texto, para que o texto possa ser posteriormente analisado para obtenção de ação a executar.
Numa fase inicial foi elaborado um estudo sobre o estado de arte das áreas de conhecimento e as tecnologias existentes, com o objetivo de identificar as tecnologias a utilizar na implementação deste protótipo. Neste estudo foram abordados as seguintes temas:

- Reconhecimento automático de fala
- Processamento de linguagem natural.

2.1 Reconhecimento automático de fala

O estudo elaborado sobre a área de reconhecimento de fala tem como o objetivo principal entender o processo de reconhecimento de fala, como é que um software de reconhecimento funciona, quais são as tarefas a executar pelo software desde a recolha do áudio até à conversão de áudio para o texto. Durante o estudo foi efetuado uma pesquisa sobre as tecnologias existentes de reconhecimento de fala, esta pesquisa incidiu essencialmente sobre as tecnologias STT (speech to texto) ou seja as tecnologias de conversão de dados áudios em formato texto.

A expressão "reconhecimento de voz" é utilizada em vários sentidos, que na verdade referem-se a tecnologia de processamento de voz distintas que podem ser aplicadas em quatro áreas distintas [21]:

- **Comando por voz** – através de um pequeno trecho de fala o processo é capaz de identificar o comando que o sistema deve executar. Este tipo de processamento é bastante simples, uma vez que o sistema já sabe quais são os comandos disponíveis para serem utilizados [21].
- **Reconhecimento de fala natural** - envolve reconhecimento de falas compostas por várias palavras, onde após o processo de reconhecimento a fala é convertida em texto. [21]
- **A síntese de voz** - este processo é exatamente o contrário do reconhecimento de fala. Este processo produz ondas sonoras (fala humana) através de um sintetizador com base no texto recebido. [21]
- **A autenticação** – baseia-se no princípio de que a voz humana é única para cada pessoa e pode ser utilizada como a forma de identificar alguém. [21]

A área de reconhecimento de voz sofreu uma grande evolução nos últimos anos. O processo de reconhecimento passou do modo discreto para o discurso continuado, ou seja de necessidade de efetuar uma pausa entra cada palavra ditada, passou a ser possível efetuar um discurso continuado. [21]

Um software de reconhecimento automático de fala deve ser capaz de analisar um sinal de entrada e identificar os símbolos linguísticos correspondentes ao sinal de entrada analisado. O processo de reconhecimento pode ser divididos em duas fases: [27]

- Análise do sinal de entrada
- Classificação.

Na fase de análise do sinal é efetuado a conversão do sinal de entrada numa representação mais adequada ao processo de classificação. Esta conversão é executada recorrendo a um conversor A/D (Conversor analógico-digital) através de uma Transformação PCM.[27]

O PCM (Pulse Code Modulation, em Português Modulação por código de pulsos) é uma técnica utilizada para transformar o sinal analógico em digital. Esta forma digital é uma stream de amplitude que representa o sinal analógico. No entanto, ainda não é possível para o software de reconhecimento trabalhar sobre este resultado, devido a elevada dificuldade nas identificações dos padrões que possam ser relacionado de acordo com o que está a ser ditado. Para que o reconhecimento seja possível, é necessário aplicar uma técnica de extração de parâmetros (i.e analise fourier) que consiste em amostrar o sinal áudio em cada centésimo de segundo num domínio da frequência. Assim, o sinal de entrada agora é representada por um conjunto de bandas de frequência, onde é facilmente identificadas as componentes de frequência de um som. [21]

Após a transformação o passo seguinte envolve o reconhecimento das bandas de frequência amostradas. Para que este reconhecimento seja possível é utilizado uma base de dados com milhares de frequências ou "fonemas". O fonema é a designação dado ao menor elemento sonoro de um idioma ou dialeto que estabelece regras para diferenciar as palavras, tal que se substituir um fonema numa palavra pode alterar por completo o seu significado. [21]

Por exemplo: a diferença entre as palavras Jato e Fato, quando faladas, está apenas no primeiro fonema: J na primeira palavra e F na segunda palavra. [21] O fonema não deve ser confundido com a letra, enquanto o fonema é o som de fala, a letra é a representação gráfica desse som. Existe situações em que o mesmo fonema pode ser representado por diferentes letras ou vice-versa, Por exemplo o fonema Z no português pode ser representado pelas letras S (CASA), Z (ZERO) ou X (EXAME). [28] [29]

Nem sempre existe uma correspondência exata entre o número de letras e o número de fonemas, por exemplos:

- Hoje - letras H O J E = 4, fonemas /ho/ /j/ /e/ = 3
- Galho - letras G A L H O = 5, fonemas /g/ /a/ /lh/ /o/ = 4

Aos fonemas reconhecidos é atribuído ao sinal de entrada um número identificador designado de "feature number".[21]

Durante o processo de treino, o software registou as seguintes estatísticas, para os fonemas "b" e "m", respectivamente:

- No fonema "b", a probabilidade de aparecer o som associado ao *feature number* #52 é de 55%, 30% para o #189 e 15% para o #53.
- No fonema "m", a probabilidade de aparecer o som associado ao *feature number* #52 é de 10%, 10% para o #189 e 80% para o #53.

Vamos usar a análise dos dados obtidos ao longo do treino durante o processo de reconhecimento. Assumindo que foram "ouvidos" 6 *feature numbers* (#52, #52, #189, #53, #52, #52) durante o reconhecimento, vamos calcular a probabilidade de o conjunto ser o fonema "b" ou "m", respectivamente:

- 55% * 55% * 30% * 15% * 55% * 55% = 0.41%
- 10% * 10% * 10% * 80% * 10% * 10% = 0.0008%

Verifica-se assim que o fonema pronunciado foi o "b".

Figura 1 - Exemplo de como é reconhecido um fonema isolado [21]

É importante para um reconhecedor de voz identificar quando é que um fonema acaba e quando é que o outro começa. Para que esta identificação seja possível é usado uma técnica denominada de "Modelo Oculto de Markov" (HMM - Hidden Markov Models). [21]

Suponhamos que o software registou os *feature numbers* correspondentes a dois fonemas consecutivos de uma palavra, é necessário distinguir onde começa e onde acaba os fonemas. Esta distinção é um processo complicado, sobre tudo se os dois fonemas tiverem algum *feature number* em comum (ver a figura 2). Para que seja possível a identificação do início e o fim do fonema é usado o modelo Markov que consiste na explosão combinatória das possibilidades de um fonema ser seguido de um outro fonema qualquer, ligados por meio de transições com pesos associados, até que seja possível uma distinção com clareza quando é que começa e acaba um fonema. [21]

Supondo que o fonema "a", com probabilidades da ocorrência de *feature numbers* de 75% para #82, 15% para #98 e 10% para #52, surge depois do fonema "b" numa determinada palavra, gerando assim a seguinte sequência de *feature numbers:* #52, #52, #189, #53, #52, #52, #82, #52, #82, etc.

Tomando em atenção que o *feature number* #52 aparece nos fonemas "b" e "a", torna-se difícil distinguir um do outro. Apenas podemos afirmar com certeza que o fonema "b" antecede o fonema "a", dada a localização dos *feature numbers*.

Figura 2 - Exemplo do como distinguir fonemas consecutivos [21]

Tendo os fonemas identificados o passo seguinte consiste em reconhecer palavras, este reconhecimento é executado através das comparações das combinações dos fonemas identificados com as palavras contidas no dicionário utilizado pelo software.

Fatores influenciais

Teoricamente todo processo parece ser bastante simples. A cada banda de frequência é associado ao seu fonema correspondente, de seguida o software junta os fonemas em palavras, e o computador passa a ter a capacidade de reconhecer a voz humana. No entanto, o processo é algo muito mais complexo, existem fatores e variações que podem influenciar todo o processo de reconhecimento. Por exemplo a forma como as palavras são pronunciadas pode dificultar ou mesmo impossibilitar a identificação do fonema correspondente. Outro exemplo são os ruídos presentes no ambiente envolvente, o software de reconhecimento pode apresentar resultados diferentes num ambiente silencioso em comparação com um ambiente com ruído.

Arquiteturas para reconhecimento de fala

"Atualmente, os algoritmos mais populares na área de reconhecimento de fala baseiam-se em métodos estatísticos. Dentre estes, dois métodos têm-se destacado: as redes neurais artificiais e os modelos ocultos de Markov (HMM). Recentemente, as implementações híbridas que tentam utilizar as características mais favoráveis de cada um destes métodos também têm obtido bons resultados." [22].

Analise Fourier

A Transformada rápida de Fourier (em inglês Fast Fourier Transform, ou FFT) é um algoritmo que permite otimizar o cálculo da Transformada Discreta de Fourier (DFT). [14]

Os parâmetros da análise de Fourier são obtidos através de coeficientes da transformada discreta de Fourier: [18]

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} x(k).e^{-j2\Pi nk/N} \qquad \textbf{n= 0, 1, 2 ... N-1}$$

Figura 3 - Equação DFT

"A resolução em frequência é dada por $\Delta f = 1/NT$ onde o T é o período de amostragem. Assim esta resolução aumenta proporcionalmente ao número de amostras por quadro. Na parametrização temporal, contudo, a resolução aumenta a medida que o tamanho dos quadros diminuem, e o número de amostras por quadro se tornam mais concentradas." [18]

A Transformada Discreta de Fourier apresentada pela equação anterior leva ao cálculo N^2 operações de multiplicação para se obter o espectro de frequência de um sinal. Com o algoritmo da FFT (Transformada Rápida de Fourier) desenvolvido a seguir, o número de multiplicações necessárias cai para N.log2.N. [14]

Se a serie é dividida em elementos pares e ímpares temos:

$$y_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} W_N^{(2n+1)k}$$

$$y_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} W_N^{2nk}$$

devido a $W_N^{2nk} = W_{N/2}^{nk}$, isto permite escrevermos

$$y_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} W_{N/2}^{nk}$$

$$y_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} W_{N/2}^{nk}$$

Figura 4 - Algoritmo FFT P.1 [14]

Se efetuarmos o seguinte:

$x_1(m)$= AMOSTRAS PARES = x(2n) e

$x_2(m)$= AMOSTRAS IMPARES = x(2n+1)

$$y_k = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_{N/2}^{nk}$$

Figura 5 - Algoritmo FFT P.2 [14]

"Assim, se existir um número par de amostras, o problema computacional de *yk* pode ser reduzido em metade." [14]

$$y_{k+\frac{N}{2}} = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{N/2}^{n\left(k+N/2\right)} + W_N^{k+N/2} \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_{N/2}^{n\left(k+N/2\right)}$$

$$y_{k+\frac{N}{2}} = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{N/2}^{nk} - W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_{N/2}^{nk}$$

devido a $W_{N/2}^{n\left(k+N/2\right)} = W_{N/2}^{nk}$ e $W_N^{N/2} = -1$

$W_N^{N/2} = -1$	$W_{N/2}^{n\left(k+N/2\right)} = W_{N/2}^{nk}$
$W_N^{N/2} = \left(e^{\frac{-i2\pi}{N}}\right)^{N/2}$	$W_{N/2}^{n\left(k+N/2\right)} = W_{N/2}^{nk} W_{N/2}^{nN/2}$
$W_N^{N/2} = \left(e^{-i\pi}\right)$	$W_{N/2}^{n\left(k+N/2\right)} = W_{N/2}^{nk} W_{N/2}^{2nN/4}$
$W_N^{N/2} = -1$	$W_{N/2}^{n\left(k+N/2\right)} = W_{N/2}^{nk} -1^{2n}$ porque $W_N^{N/2} = -1$
	$W_{N/2}^{n\left(k+N/2\right)} = W_{N/2}^{nk}$

Com estes resultados

$$y_k = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_{N/2}^{nk}$$

Figura 6 - Algoritmo FFT P.3 [14]

Uma ***Operação Butterfly*** é constituída conforme a figura 7 :

$$X(k) = X_1(k) + W_N^k X_2(k) \quad \text{e} \quad X\left(k + N/2\right) = X_1(k) - W_N^k X_2(k) \qquad [5.26,27]$$

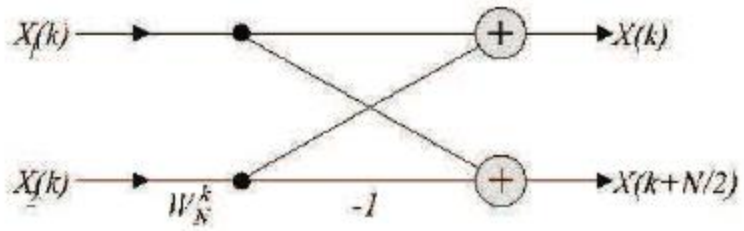

Figura 7 - Algoritmo FFT P.4 [14]

"A operação ***Butterfly*** permite que a computação da DFT seja dividida, ou seja, se existem 2n de amostras, a computação da DFT pode ser repetida exigindo somente metade do cálculo. Para uma DFT de 2 pontos (caso básico) temos " [14]:

$$y_0 = x_0 + W_2^0 x_1 = x_0 + x_1 \qquad \text{e} \qquad y_1 = x_0 - W_2^0 x_1 = x_0 - x_1$$

Figura 8 - Algoritmo FFT P.5 [14]

Sendo assim, caso existe 2^n amostras, o processamento pode ser executado através de uso da operação "Butterfly" dos quais se repetem conforme o número de bits utilizado. A figura seguinte mostra este processo [14]:

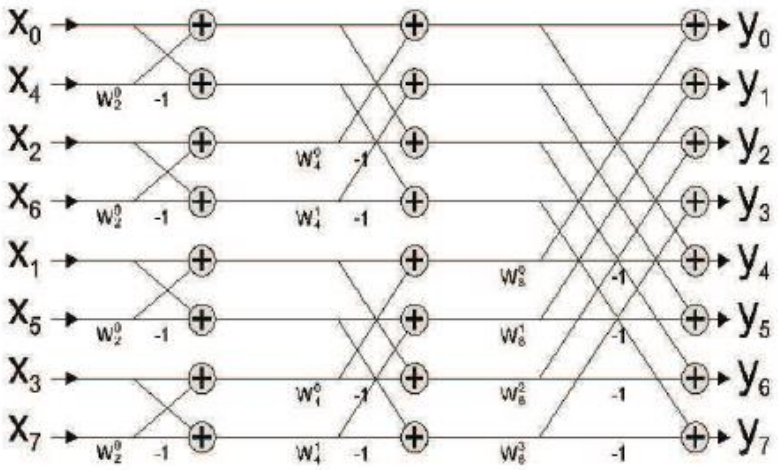

Figura 9 - Algoritmo FFT P.6 [14]

"Estas séries de operações podem ser mais facilmente implementadas se os valores de entradas forem sequenciados corretamente. A análise da representação binária dos valores de entrada revelam um padrão subjacente, ou seja, os índices correspondem à representação binária invertida do número da sequência, como mostrado na imagem seguinte". [14]

Seqüência da Operação "Butterfly"	Endereço Binário	Bit Reverso	Elementos correspondente a seqüência Original
x_0	000	000	x_0
x_4	100	001	x_1
x_2	010	010	x_2
x_6	110	011	x_3
x_1	001	100	x_4
x_5	101	101	x_5
x_3	011	110	x_6
x_7	111	111	x_7

Figura 10 - Algoritmo FFT P.7 [14]

"Devido ao fato das representações binárias se repetirem, o número de operações de multiplicações necessárias para se obter o espectro de frequência de um sinal diminui consideravelmente". [14]

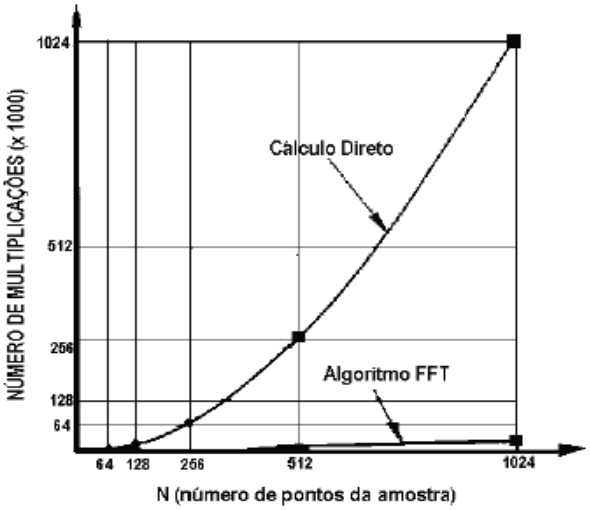

Figura 11 - Calculo necessário

"A figura anterior mostra a eficiência computacional da FFT em comparação com o método normal (transformada de Fourier simples). É possível concluir que quanto maior for o número de pontos de uma amostra, o número de multiplicações na forma direta cresce na forma exponencial, enquanto a FFT possui poucas multiplicações". [14]

HMM - Modelos Ocultos de Markov

O modelo oculto de Markov é um modelo estatístico onde os parâmetros desconhecidos (ocultos) é obtidos através de parâmetros observáveis. Num processo de reconhecimento de palavras, a sequencia de estado dos quadros representa o parâmetro desconhecido que por sua vez determinam as palavras. "Os algoritmos mais comuns utilizados para determinação da sequência de estados são: Backward, Forward e Viterb." [18]

Backward - Forward são algoritmos que permitem verificar qual a probabilidade de um conjunto das observações {O}, terem sido gerados pelo modelo de probabilidades $\lambda(A,B,\pi)$, onde $A = a_{ij} = P(s_{t+1} = j \mid s_t = i)$; $B = b_j(O_t) = P(O_t \mid s_t = j)$ e; $\pi = \pi_i = P(s_1 = i)$.[18]

"A aplicação dos algoritmos resulta para o algoritmo forward e backward, respectivamente:

$$P(O \mid \lambda) = \sum_{i \in S_F} \alpha_T(i); \qquad P(O \mid \lambda) = \sum_{i \in S_F} \pi_i b_i O_1 \beta_1(i)$$

Onde: $\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T, s_t = i \mid \lambda)$; $\alpha_t(i) = P(O_1, O_2, ..., O_t, s_t = i \mid \lambda)$."[18]

"Viterbi - O algoritmo é utilizado para encontrar a sequência de estados que tem maior probabilidade de ter gerado a sequência de estados observada. O algoritmo é semelhante ao DTW, ou seja, o algoritmo determina o caminho de maior probabilidade de transição entre

estados, encontrando a sequência de estados como maior probabilidade. Assim, definindo a variável:

$$\delta_t(t) = \max_{q_1 q_2 \Lambda\, q_{t-1}} P[q_1 q_2 \Lambda\, q_{t-1} q_t = S_i, o_1 o_2 \Lambda\, o_t | \lambda]$$

Têm-se as variáveis inicializadas em:

$$\delta_1(1) = \pi_i b_i(O_1); \qquad \psi_1(i) = 0$$

Por meio de iterações determinam-se:

$$\delta_t(j) = Max[\delta_{t-1}(i) a_{ij}] b_j(O_t); \quad \psi_t(j) = valor\,de\ i\ para Max[\delta_{t-1}(i) a_{ij}]$$

Os resultados são dados por:

$$P^* = Max[\delta_T(s)]; \qquad s_T^* = \arg\max[\delta_T(s)]$$

A sequência de estado, para t = T+1 a t = 1, é dada por:

$$s_t^* = \psi_{t+1}(s_{t+1}^*)\text{''}$$ [18]

História do Reconhecimento Automático de fala

A primeira máquina com algum interesse a reconhecer a fala foi um brinquedo chamado "Radio Rex", que foi fabricado na década de 1920. "Radio Rex" é um cão celuloide que se movia através de uma mola que é sensível a energia acústica de 500Hz. Uma energia à volta dos 500 Hz é praticamente o primeiro formante da vogal na "Rex", pronunciar "Rex" era suficiente para libertar o cão. [13]

Em 1952, nos laboratórios da Bel construíram um sistema para reconhecimento de um dígito isolado falado por uma pessoa. Este sistema media uma simples função do espectro energético no tempo, em duas vastas bandas, correspondentes, de uma forma grosseira, as duas primeiras ressonâncias do trato vocal. [13]

Em 1958, Dudly fez um classificador que avaliava o espectro de um modo contínuo. [13]

Em 1959, O Fry e Denes construíram um sistema de reconhecimento de fonemas que é capaz de reconhecer quatro vogais e nove consoantes baseado num princípio de reconhecimento de padrões. O sistema de Fry e Denes foi o primeiro a usar probabilidades de transição de fonemas para restringir o sistema reconhecedor. [13]

Em 1964 Martin desenvolveu as redes neuronais artificias para o reconhecimento de fonemas. [13]

Na década de 1960 foram desenvolvidos varias técnicas de extração de parâmetros de grande importância no que diz respeito ao reconhecimento de voz, nomeadamente a Transformada de fourier rápida (Fourier Fast Transform - FFT), a análise cepstral e a codificação de predição linear (Linear Predictive Coding - LPC). Adicionalmente foram implementados novos métodos

de comparação de padrões de sequencias, nomeadamente uma abordagem determinística chamada alinhamento Temporal Dinâmico (Dynamic Time Warp - DTW) e uma abordagem estatística chamada Modelos Escondidos de Markov (Hidden Markov Models - HMM). [13]

Na década de 1970, a Agência de projetos de Integração Avançados (Advanced Research Projets Agency - ARPA) financiou um projeto que tinha como o objetivo efetuar o reconhecimento de fala com um vocabulário de 1000 palavras usando um número limitado de locutor e uma gramatica limitada com menos de 10% de erro. [13]

No iniciou de anos 80, a ARPA voltou a financiar vários programas de pesquisa sobre o reconhecimento automático de fala, tendo sido a primeira tarefa a elaboração da "Resource Management" (RM). Esta tarefa a semelhança do primeiro projeto financiado pela ARPA consiste no reconhecimento da leitura das frases derivadas de um vocabulário de 1000 palavras, mas que agora inclui um novo componente que envolve o reconhecimento independente do locutor. As tarefas seguintes são reconhecimento de um vocabulário composto por 5000 palavras e 60000 palavras. [13]

Nos finais dos anos 1960 e início dos anos 1970 várias abordagens baseadas na codificação do conhecimento humano foram amplamente usadas. Por exemplos a utilização do conhecimento acústico-fonético nos sistemas de reconhecimento para desenvolver regras de classificação para sinais de fala. [13]

Nas últimas décadas houve uma grande esforço para melhorar a robustez no reconhecimento de fala e um aumento de enfasa de assuntos como a pronúncia, o modelo de diálogo e a modelação acústica, também tem havido uma rápida expansão de instigação nas áreas relacionadas com o reconhecimento de fala. [13]

Produtos existentes:

- **Google speech search** - é um produto do Google que permite aos utilizadores utilizarem o Google Search (efetuar pesquisa no motor google), pronunciando a pesquisa através de um dispositivos eletrónico (i.e Smartphone), ou seja, fazer com que seja efetuado uma pesquisa sobre os dados pretendidos através da voz (linguagem natural).

- **Siri** - Siri é uma aplicação de assistente pessoal para sistema iOS que é capaz de responder as perguntas colocadas, fazer recomendações, e executar as ações solicitadas.

- **Cortana** - Cortana é um/a assistente virtual do sistema operativo Windows 10. O seu nome vem da Cortana, a personagem de inteligência artificial da série Halo.

- **BMW** – Atualmente os carros de marca BMW disponibilizam uma funcionalidade designado de *Voice control system*, onde os condutores podem controlar os recursos do carro tais como o telefone, climatização, sistema de navegação através de comandos falado.

Limitações

Tendo em conta as evoluções tecnologicas nos últimos anos e os avanços consegidos na area de reconhecimento de voz, ainda não é possível garantir uma precisão de 100% no reconhecimento de fala devida algumas limitações na sua utilização:

- Mesmo os melhores sistemas de reconhecimento de voz cometem erros, por exemplo se houver ruído ou algum outro som no ambiente (por exemplo a televisão), estes ruidos tende aumentar o o número de erros no reconhecimento. [16]
- Reconhecimento de voz tem uma melhor performance caso o microfone estiver mais perto de locutor. A distância entre o microfone e o locutor tenderá a influenciar o resultado do reconhecimento. [16]
- O número de locutor existente no local influencia o resultado do processo de reconhecimento.
- O determinado software de reconhecimento de voz foi previamente treinado para efetuar o reconhecimento de um conjunto grande de palavras ou frases. Como tal se o utilizador pronunciar uma frase ou um termo técnico muito especifico o sistema pode não conseguir reconhece-lo. [17]
- O locutor tem de falar de forma consistente e clara em todos os momentos para minimizar os erros no processo de reconhecimento. Por exemplo se o locutor falar muito rápido o software poderá não ser capaz de reconhecer a frase ou a palavra pronunciada. O sistema de reconhecimento de voz também podem ter problemas em reconhecer a fala devido as alterações de voz, por exemplo quando o locutor está com tosse, sinusite ou problemas na garganta. [17]

Ao longo de pesquisa foram identificados as seguintes tecnologias de reconhecimento de fala com suporte a língua português:

- Web speech API (Google Speech API cloud service)
- Microsoft speech recognition engine (Microsoft Speech Platform SDK 11)
- Nuance natural speech (sugerida pela Glintt como uma possibilidade de teste)

2.1.1 Web speech API

A web speech API é uma biblioteca em javascript que permite implementar a funcionalidade de reconhecimento e a conversão de fala em textos de um modo continua ou não. [19] A vantagem no uso desta ferramenta para desenvolvimento desta solução é a sua simplicidade de integração e a facilidade na sua utilização.

Segurança e privacidade

Só deve existir o processo de recolhe de dados áudios depois de obter a autorização do utilizador para este tipo de operação. A autorização de utilizador pode incluir, por exemplo:

- O utilizador selecionar num elemento visível que tem uma representação gráfica óbvio a informar que vai iniciar o processo de reconhecimento de voz (fala). [19]
- Aceitar o prompt de permissão como o resultado de uma chamada a interface de reconhecimento de voz (SpeechRecognition.start). [19]
- Autorização anteriormente concedida para permitir sempre o reconhecimento de voz para uma determinada página web. [19]

É sempre fornecido ao utilizador uma indicação óbvia quando o áudio está a ser gravado.

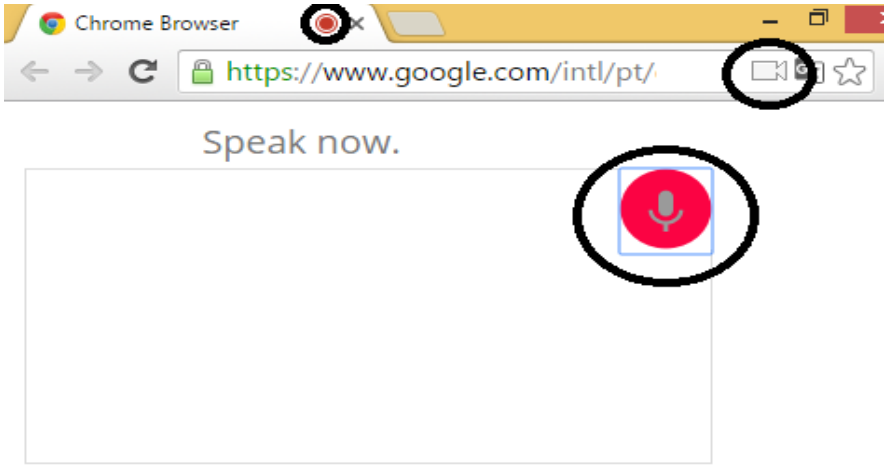

Figura 12 - Indicação fornecido no browser quando existe gravação de áudio

A desvantagem no uso desta API é em termos de suporte desta API dados nos diferentes tipos de browser, a tabela seguinte mostra o suporte dado pelos principais browsers.

IE	Firefox	Chrome	Opera	Chrome android
Não suportado	Não suportado por defeito, mas pode ser ativado atráves da abount:config =>media.webspeech.recognition.enable	Suportado (prefixo webkit) versão chrome 25 +	Suportado (prefixo webkit) versão Opera 27 +	Suportado (prefixo webkit) versão chrome android 47 +

Tabela 1 - Suporte dos browsers a API (Caniuse, 2016)

Ou seja para usar esta API é obrigatório ter um dos browsers acima identificado que suporta a API.

O Web Speech API dota as aplicações web da capacidade de lidar com os dados áudios. Há dois componentes principais nesta API:

- Speech recognition
- Speech synthesis

O serviço de reconhecimento pode ser acedido através da interface ***SpeechRecognition***, que fornece a aplicação a capacidade de reconhecer a voz a partir de uma entrada de áudio e responder apropriadamente. Para isso é necessário usar o construtor da interface para criar um novo objeto da interface SpeechRecognition, que tem uma série de manipuladores de eventos disponíveis para detetar quando a fala é recebida através do microfone do dispositivo**:** [19]

- **audiostart** - Dispara quando é dado o inicio de gravação de áudio. [19]
- **soundstart** - Dispara quando algum som, possivelmente a fala, for detetado. O evento audiostart deve ser sempre ativado antes do evento soundstart. [19]
- **speechstart** - Dispara quando o discurso a reconhecer for iniciado. O evento audiostart deve ser sempre ativado antes do evento speechstart. [19]
- **speechend** - Dispara quando o discurso a reconhecer tenha terminado. O evento speechstart deve ser sempre inativado antes do evento speechend. [19]
- **soundend** - Dispara quando não é detetado qualquer tipo de sons. O evento soundstart deve ser sempre inativado antes do evento soundend. [19]
- **audioend** - Dispara quando o agente de usuário terminar a captura de áudio. O evento audiostart deve sempre ter sido despedido antes audioend. [19]
- **result** - Dispara quando o reconhecedor de voz retorna um resultado. O evento deve usar a interface SpeechRecognitionEvent. O evento audiostart deve ser sempre ativo antes do evento de resultado. [19]
- **NoMatch** - Dispara quando o reconhecedor de voz retorna um resultado final sem qualquer hipótese de reconhecimento. O evento deve usar a interface SpeechRecognitionEvent. Os resultados atribuem em caso podem conter resultados de reconhecimento de fala que estão abaixo do limiar de confiança ou podem ser nulos. O evento audiostart deve ser sempre ativos antes do evento NoMatch. [19]
- **error** - Dispara quando ocorre um erro de reconhecimento de fala. O evento deve usar a interface SpeechRecognitionError. [19]
- **start** - Dispara quando o serviço de reconhecimento começa a captar o áudio com a intenção de reconhecer. [19]
- **end** - Dispara quando o serviço de reconhecimento é inativado. O evento deve ser sempre gerado quando a sessão termina. [19]

A interface ***SpeechGrammar*** representa um conjunto particular de gramática que a aplicação deve reconhecer. A gramática é definida usando JSpeech Grammar Format (JSGF.) [19]

SpeechRecognition Interfaces

- **SpeechRecognition** - A interface para controlar o serviço de reconhecimento. [19]
- **SpeechRecognitionAlternative** - Representa a única palavra que foi reconhecido pelo serviço de reconhecimento de fala. [19]
- **SpeechRecognitionError** - Representa mensagens de erro do serviço de reconhecimento. [19]
- **SpeechRecognitionEvent** – É um event object que contém todos os dados associados a um resultado provisório ou definitivo do serviço de reconhecimento de fala. [19]
- **SpeechGrammar** - As palavras ou padrões das palavras que o serviço de reconhecimento reconhece. [19]
- **SpeechGrammarList** - Representa uma lista dos objetos SpeechGrammar. [19]
- **SpeechRecognitionResult** - Representa o resultado do processo de reconhecimento, que pode conter vários objetos de SpeechRecognitionAlternative. [19]
- **SpeechRecognitionResultList** - Representa uma lista de objetos SpeechRecognitionResult. [19]

No anexo 1 encontram-se as informações mais detalhadas da invocação da interface do web speech API.

A utilização desta API não exige um grande esforço na integração do serviço de reconhecimento de voz, apenas é necessário a criação de uma biblioteca de javascript onde seja criado um objeto de interface speechrecognition e invocar o serviço de reconhecimento de fala.

2.1.2 Microsoft Speech Platform SDK 11

A Microsoft Speech Platform SDK 11 é uma tecnologia Microsoft que fornece um conjunto abrangente de tecnologias e ferramentas de desenvolvimento para criação de aplicações com possibilidades de ativação por voz. É possível através destas tecnologias e ferramentas existente na Plataforma, capacitar das aplicações a habilidade de reconhecer falas (reconhecimento de voz, ou STT) de um modo continuo e gerar voz sintetizada (text-to-speech, ou TTS). [4]

A plataforma Microsoft Speech SDK 11 para além das tecnologias e ferramentas, fornece ainda suporte para 26 idiomas incluindo portugês, tanto para a síntese de reconhecimento de voz e de fala (TTS ou text-to-speech). [4]

A vantagem no uso deste tecnologia está na relação de Custo-benefício, a Plataforma Speech SDK 11 pode reduzir significativamente a quantidade de trabalho necessário para implementar a funcionalidade de reconhecimento de voz nas aplicações para Windows Server ou outras plataformas, encurtando assim os ciclos de desenvolvimento e reduzindo os seus custos. [4]

Existem alguns requisitos de hardware e software na utilização deste SDK:

Sistema operativo	**Edição**	**Tipo**
Windows Vista	All editions except Starter Edition	32-bit, 64-bit
Windows 7	All editions except Starter Edition	32-bit, 64-bit
Windows Server 2008	Service Pack 2	32-bit, 64-bit
Windows Server 2008	R2	64-bit

Tabela 2 - Requisito de software [4]

Requisitos minimos	**Requisitos recomendados**
• 1 GHz CPU • 512 MB RAM • 10 GB hard drive • Ethernet network adapter • DX9-compatible video card, with 2.0 pixel shader and vertex shader support (used by some of the graphics and user interface development tools included with the XDK) • USB 2.0	• Dual 2GHz CPU • 1 GB RAM • 40+ GB SATA hard drive • Ethernet network adapter • DX9-compatible video card, with 3.0 pixel shader and vertex shader support (when available) • Two USB 2.0 host controllers, which are required when you use DVD emulation and performance analysis tools simultaneously

Tabela 3 - Requisito hardware [4]

Além dos requisitos acima referidos é necessário também instalar o SDK e o runtime da linguagem pretendida e efetuar alterações à aplicação por forma a incluir o serviço de reconhecimento de fala.

A Microsoft Speech Platform apresenta um conjunto de namespaces que são fundamentais na implementação do serviço de reconhecimento de fala nas aplicações. Aqui será abordado de uma forma geral as principais classes e membros do *namespace* Microsoft.Speech. [4]

- **Microsoft.Speech.AudioFormat –** O namespace Microsoft.Speech.AudioFormat contém informações sobre o formato do áudio de entrada suportada pelo motor de reconhecimento de voz. [4]
- **Microsoft.Speech.Recognition -** Contém os recursos necessários na implementação do serviço de reconhecimento de voz nas aplicações existentes. [4]
- **Microsoft.Speech.Recognition.SrgsGrammar** - Permite criar programaticamente gramáticas que estejam em conformidade com o W3C Speech Recognition Grammar Specification Version 1.0 (SRGS). [4]

No anexo 4 encontra-se uma listagem mais detalhada sobre as classes existente em cada um dos *namespace* acima identificado.

2.1.3 Nuance

O Nuance é uma empresa lider no mercado de reconhecimento de fala, fornece varios tipos de serviços de reconhecimento de falar em múltiplas plataformas: iOS, Android, HTTP. A interface HyperText Transfer Protocol (HTTP) permite aos utilizadores adicionar o serviço de voz nas suas aplicações de uma forma fácil e rápido. [5] OS serviços de voz disponíveis são:

- Reconhecimento Automático de Fala (ASR)
- Text To Speech (TTS)

Estas interfaces permitem o acesso aos componentes de processamento de voz hospedados num servidor através de uma solicitação HTTP, minimizando assim o consumo dos recursos. [5]

No anexo 2 enconta-se uma tabela onde estão visiveis os preços do serviço de reconhecimento de acordo com o tipo do serviço a contratar, este informação pode ser importante caso futuramente exista a possibilidade de contratação do serviço de reconhecimento de falar prestado pela a empresa Nuance.

2.2 Processamento de linguagem natural

O estudo sobre processamento de linguagem natural serviu essencialmente para entender como é feito o processamento de linguagem natural. O Processamento da Linguagem Natural (PLN), subárea da inteligência artificial e da linguística que estuda os problemas relacionados à automação da interpretação e da geração da língua humana em diversas aplicações. É o encontro de duas áreas com enfoques distintos e problemas semelhantes. O objetivo principal destes sistemas é converter as representações de linguagem humana em representações que são facilmente manipuláveis por computador. [3]

O processamento da linguagem natural pode ser dividido em quatros etapas de análises distintas:

- análise morfológica
- análise sintática
- análise semântica
- análise pragmática

2.2.1 Análise morfológica

A análise morfológica é a fase onde é analisada as palavras de uma forma isolada e classificadas de acordo com a sua estrutura gramatical. No português atualmente existem dez classes de palavras distintas: substantivo, verbo, artigo, adjetivo, numeral, pronome, advérbios, preposição, conjunção e interjeição. Palavra cuja estrutura esteja incorreta pode ser ignorada nesta fase. [25]

2.2.2 Análise sintática

A análise sintática analisa uma sequência de palavras a fim de construir uma árvore sintática onde é representado o relacionamento e posicionamento de uma palavra na frase. Comparando este processo com o estudo de gramática seria equivalente a classificar cada palavra segundo seu posicionamento na frase, como sujeito, predicado ou um outro. [25]

Numa análise sintática uma sequência de palavras pode ser ignorada caso esteja mal organizada, por exemplo: "carros o Tiago comprou todos os" este sequencia está morfologicamente correta, mas no entanto não é valida para as regras gramaticais sintáticas logo será rejeitada. [25]

Na figura 13 está uma representação gráfica de como seria a árvore sintática da frase “O Tiago comprou todos os carros”.

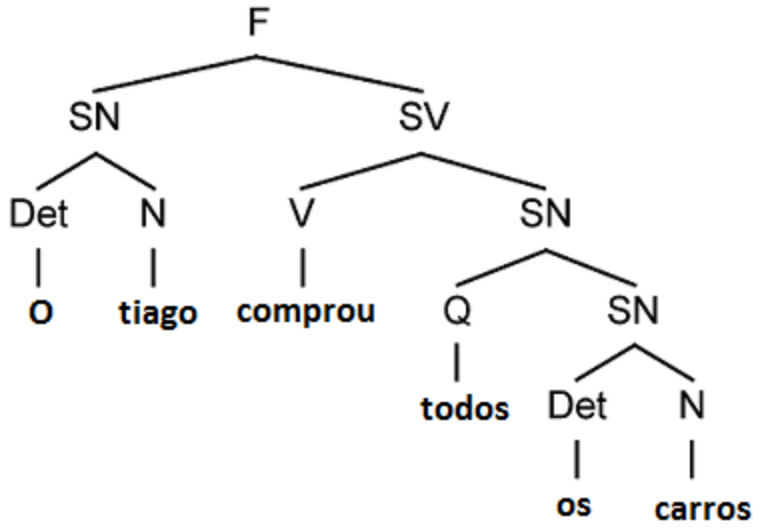

Figura 13 - Análise sintática do texto "O Tiago comprou todos os carros".

Geralmente é possivel identificar dois componentes principais num processamento sintático:

- **Gramática**, que serve de base para reconhecer e validar todas as regras sintáticas da linguagem em questão; [26]
- **Interpretador de gramática**, é responsavel na comparação da gramática com a frase de entrada a fim de produzir uma estrutura correcta a ser analiada. [26]

2.2.3 Analise semântica

O objetivo de análise semântica é analisar o significado de uma frase, as palavras de uma frase podem estar corretas e obedecerem a uma construção gramatical valida, no entanto pode não fazer sentido nenhum por ser semanticamente anómala. [25]

Exemplos de erro:

- Este carro conduziu o João.

No exemplo anterior a frase tem um erro semântico porque a frase não faz qualquer sentido mas no entanto está sintaticamente correta.

2.2.4 Analise pragmática

A análise pragmática tem como objetivo a reinterpretação de uma frase considerando que a frase podem ter significados diferentes consoante o contexto, a fim de determinar se a frase é uma solicitação, pergunta, ou afirmação. [25]

Por exemplo a frase "Qual é o teu nome?" ao ser processado pela análise pragmática a frase deve ser interpretada como sendo uma solicitação para que o programa responda qual é o seu nome. [25]

A tabela seguinte apresenta a evolução dos estudos do PLN em termos do grau de sofisticação linguística alcançado.

Década de 50: a tradução automática

- Sistematização computacional das classes de palavras da gramatica tradicional.
- Identificação computacional de poucos tipos de constituintes oracionais

Década de 60: Novas aplicações e criação de formalismos

- Primeiros tratamentos computacionais das gramáticas livres de contexto
- Criação dos primeiros analisadores sintáticos
- Primeiras formalizações do significado em termos de redes semânticas

Década de 70: Consolidação dos estudos do PLN

- Implementação das primeiras gramáticas e analisadores sintáticos

• Pesquisa de formalização de fatores pragmáticos e discursivos
Década de 80; sofisticação dos sistemas
• Desenvolvimento de teorias linguísticas
Década de 90: sistemas baseados em "representações do conhecimento"
• Desenvolvimento dos projetos de sistema de PLN complexo que integra vários tipos de conhecimentos linguísticos.

Tabela 4 - Evolução dos sistemas de PLN [23]

Ao longo de estudo sobre sistema de PLN foram identificados as seguintes ferramentas de PLN com algum interesse, mas nenhuma das ferramentas suporta a língua portuguesa:

- Stanford
- Natty

Stanford

O Stanford University tem um grupo dedicado ao estudo de PLN desde o ano 2002 e tem disponível um conjunto de tecnologias que permitem efetuar o processamento de linguagem natural, por exemplo Stanford CoreNLP e Stanford Parser. [7]

Natty

Natty é uma API de reconhecimento de datas em linguagem natural desenvolvido em linguagem Java. Dada a uma frase contendo uma expressão de data esta API é capaz de aplicar padrões de reconhecimento da linguagem natural e as técnicas de tradução para produzir uma lista de datas correspondentes. [6]

A Natty é capaz de reconhecer diversos formatos de datas, estes formatos pode ser agrupados em datas formais (i.e 1979/02/28), datas relativas (i.e o dia antes da próxima quinta-feira), e datas alternativas (i.e próxima quarta ou quinta). [6]

Para efetuar o reconhecimento este API usa o ANTLR (Another Tool for Language Recognition) para etiquetar (Tagging) e analisar os dados de entrada de forma a construir uma árvore de sintaxe abstrata (AST) que permite determinar uma data correspondente. Devido à grande variedade de formatos de data reconhecido pela Natty, muitas das vezes surgem situações de ambiguidade inerente na gramática que descrevem os formatos, por isso determinadas decisões são tomadas de uma forma arbitrária. Por exemplo, dada a expressão de entrada: '10/10 1500 ', é possível interpretar isso como "10 de outubro às 15 horas" ou como "10 de outubro do ano 1500". Dadas as opções disponíveis, Natty irá escolher a primeira opção com base no raciocínio de que as pessoas não costumam fazer referência às datas que são muito distantes do presente (séculos). [6]

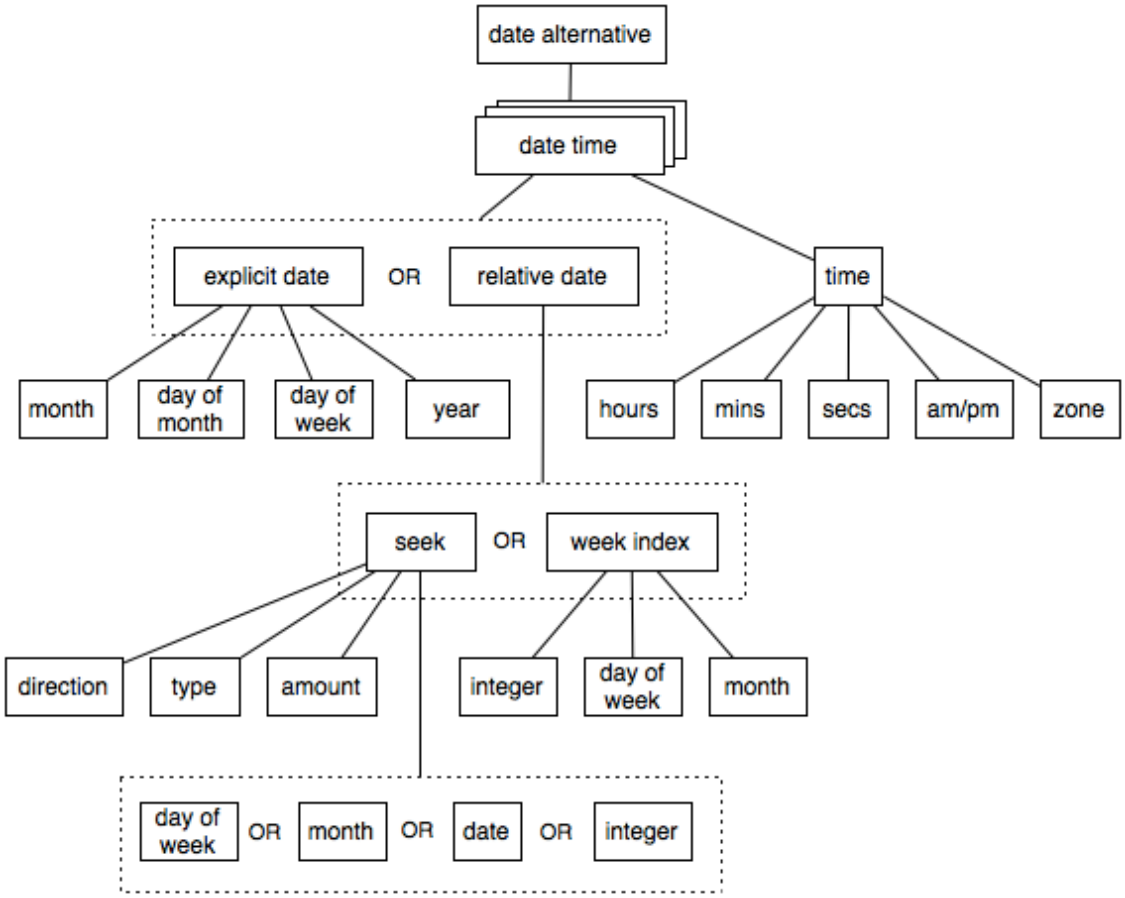

Figura 14 - Árvore sintática abstrata (Natty,2016)

A figura 14 mostra a árvore sintática abstrata que a API Natty utiliza no processo de reconhecimento de datas.

3. Implementação

Após a análise do problema apresentado, conclui-se que é necessário criar uma solução integrável que visa dotar as aplicações existentes da capacidade de entender o discurso natural (fala) do utilizador. Ou seja ao integrar a solução implementada, passa a existir uma interface no-touch que irá permitir aos utilizadores interagirem com as aplicações através de discurso natural (fala) sem recurso ao teclado nem rato. Por exemplo, o utilizador ao pronunciar "Computador pesquisa o doente XPTO", a aplicação deve ter a capacidade de reconhecer e executar uma operação de pesquisa para encontrar o doente XPTO e apresentar o resultado de pesquisa no ecrã.

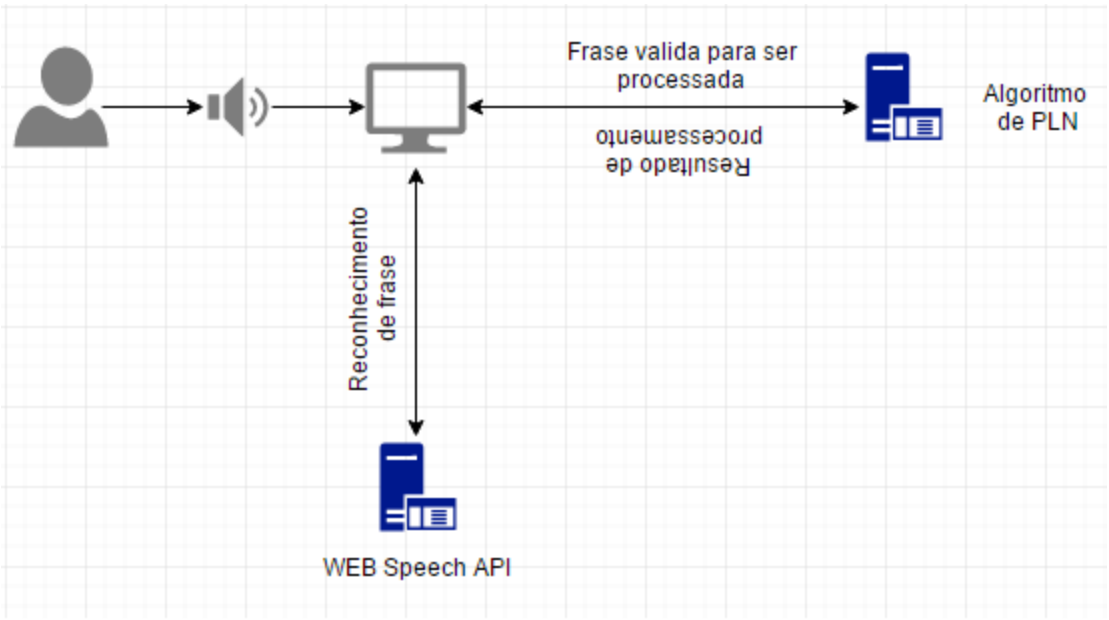

Figura 15 - Visão geral de funcionamento do protótipo

A figura anterior mostra uma visão global do funcionamento do protótipo a desenvolver. O protótipo a implementar pode ser dividido em dois componentes o front-end e o back-end, o front-end é constituído pela API de reconhecimento automático de fala que irá recolher os dados áudios (falas) e converter os dados recolhidos em formato texto. O back-end é responsável pelo processamento de linguagem natural com o objetivo de identificar a ação que deve ser executada pela aplicação.

Desta forma sempre que o utilizador pronunciar uma fala, a aplicação irá recolher e enviar a fala pronunciada para o web speech API para ser analisada e convertida em texto. Após obtida a fala pronunciada em formato texto, esta será enviado para o servidor onde será processada pelo algoritmo de processamento de linguagem natural. O resultado do algoritmo será enviado de volta para a aplicação para que seja executada a ação pretendida.

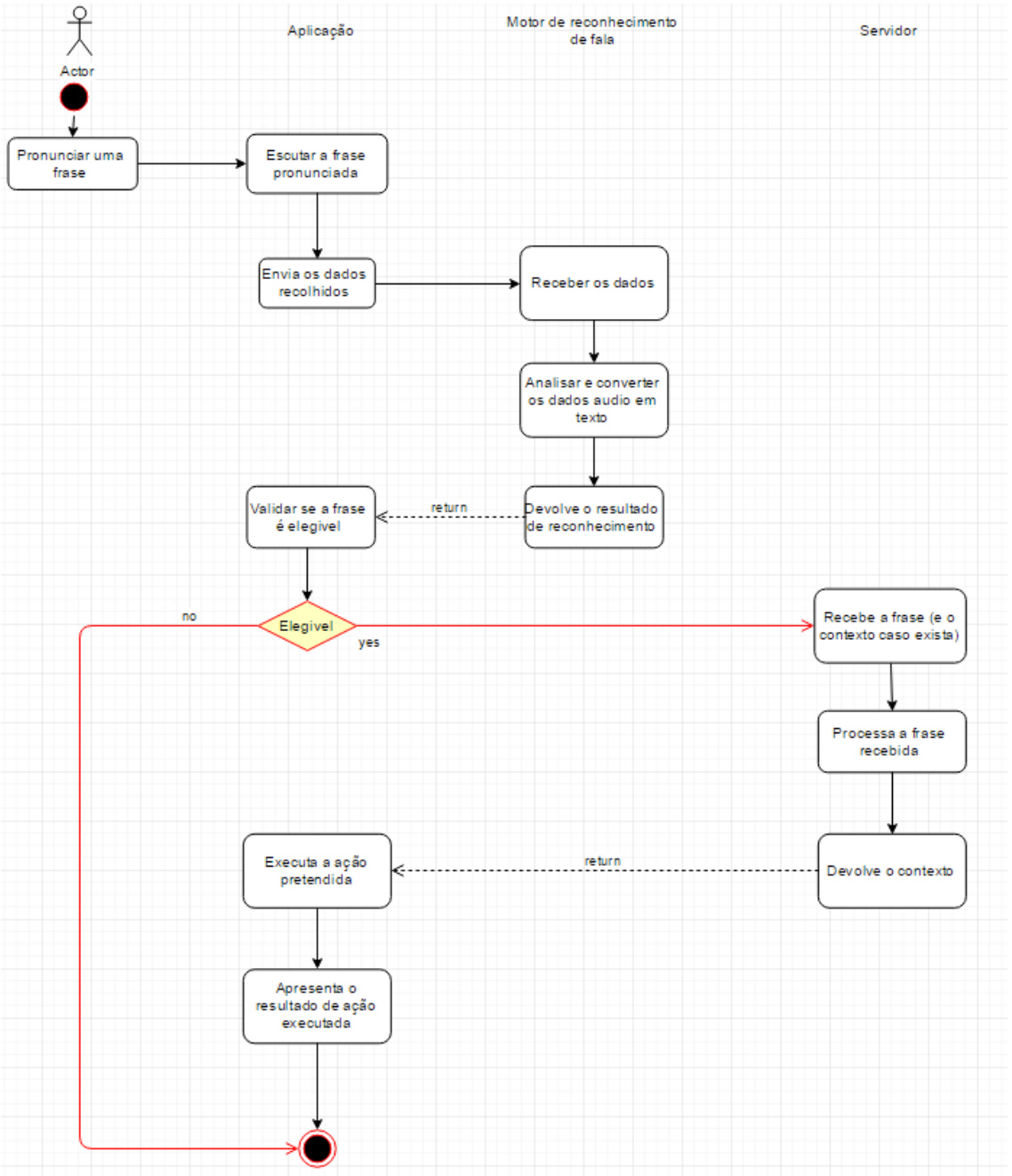

Figura 16 - Diagrama de atividade

A figura 16 é um diagrama de atividade onde está especificado o fluxo de trabalho envolvido quando aplicação escuta uma frase pronunciada pelo utilizador.

3. 1 Avaliação e Experimentação

Numa fase inicial da implementação foram executados alguns testes para avaliar o desempenho e a performance das APIs de reconhecimento automático de fala com o objetivo de identificar a API mais adequada a utilizar na implementação do protótipo. Os critérios para a escolha da API são:

- Taxa de erro apresentado.

- Esforço necessário no processo de integração do motor de reconhecimento nas aplicações.
- Performance.

Foram implementados dois tipos de testes diferentes:

- Teste A - onde não existe ruido no ambiente de teste.
- Teste B – onde existe algum ruido no ambiente de teste.

Teste realizado com um único locutor do sexo masculino de nacionalidade chinesa residente em Portugal há mais de 15 anos.

Nos testes realizados foram utilizados os seguintes equipamentos para captar os dados áudios:

1. O microfone integrado do equipamento que irá ser utilizado para o desenvolvimento do protótipo.
2. O microfone normal com as mesmas características que se encontra num local fixo:
   - Sensitivity:-46±2dB
   - Signal to Noise Ratio:≥56dB
   - Frequency Range:50-16,000Hz
3. O microfone normal com a mesma característica que se encontra junto do utilizador:
   - Sensitivity:-46±2dB
   - Signal to Noise Ratio:≥56dB
   - Frequency Range:50-16,000Hz

3.1.1 Web Speech API

Equipamento número 1

Frase dito	Tentativas	Tempo (segundos)
Pesquisar o doente	1	2
Pesquisar o doente 123	2	3.5
Mostra-me o resultado do doente 123 no dia 15 de janeiro de 2015	2	7.5
Mostra doente 123	2	4.1
Encontrar o doente 123	1	4.07

Tabela 5 - Teste sem ruído de fundo

Frase dito	Tentativas	Tempo (segundos)
Pesquisar o doente	2	2.78
Pesquisar o doente 123	1	3.78
Mostra-me o resultado do doente 123 no dia 15 de janeiro de 2015	2	8.97
Mostra o doente 123	1	4.38
Encontrar o doente 123	2	4.11

Tabela 6 - Teste com ruído de fundo

Nos testes efetuados com o equipamento número 1 na maioria das frases foi preciso pelo menos 2 tentativas para obter o resultado correto.

Equipamento número 2

Frase dito	Tentativas	Tempo (segundos)
Pesquisar o doente	1	2
Pesquisar o doente 123	2	3.3
Mostra-me o resultado do doente 123 no dia 15 de janeiro de 2015	2	7
Mostra o doente 123	1	3
Encontrar o doente 123	1	3

Tabela 7 - Teste sem ruído de fundo

Frase dito	Tentativas	Tempo (segundos)
Pesquisar o doente	2	3
Pesquisar o doente 123	2	3.75
Mostra-me o resultado do doente 123 no dia 15 de janeiro de 2015	3	8
Mostra o doente 123	1	3.1
Encontrar o doente 123	2	3.5

Tabela 8 - Teste com ruído de fundo

Nos testes efetuados com o equipamento número 2 a distancia entre o utilizador e o microfone é de aproximadamente 30 cm, e os resultados obtidos demonstram uma taxa

elevado de erro na conversão num ambiente com ruido como é possível constatar nas tabelas anteriores.

Equipamento número 3

Frase dito	Tentativas	Tempo (segundos)
Pesquisar o doente	1	2
Pesquisar o doente 123	1	3.4
Mostra-me o resultado do doente 123 no dia 15 de janeiro de 2015	2	7.1
Mostra o doente 123	1	2.1
Encontrar o doente 123	1	2.2

Tabela 9 - Teste sem ruído de fundo

Frase dito	Tentativas	Tempo (segundos)
Pesquisar o doente	1	2
Pesquisar o doente 123	1	2.8
Mostra-me o resultado do doente 123 no dia 15 de janeiro de 2015	2	7.3
Mostra o doente 123	1	2.3
Encontrar o doente 123	1	2.4

Tabela 10 - Teste com ruído de fundo

Os testes realizados com o equipamento número 3 demonstram resultados bastante positivos seja em ambiente com ou sem ruido, isso deve-se ao facto de o microfone encontrar-se muito próximo do utilizador.

Com os dados recolhidos durante os testes efetuados com os 3 equipamentos é possivel verificar um ligeiro aumento do tempo no reconhecimento de fala num ambiente com ruido em comparação com um ambiente sem ruido. Durante os testes efetuados foi possivel constatar que a API é mais precisa no reconhecimento de fala quando é projetado num local mais próximo possível do microfone. Por exemplo nos testes efetuado com o equipamento 3 como o microfone encontra-se muito próximo do utilizador (a uma distância quase nula), a taxa de erro apresentado é bastante reduzida. Um fato interessante verificado nos testes é que no caso de a fala ser projetada a uma velocidade constante, ou seja todas as palavras são pronunciadas à mesma velocidade incluído as pausas entre as palavras, neste caso a taxa de erro no reconhecimento é muito inferior em comparação com a fala projetada com uma velocidade variada.

3.1.2 Microsoft Speech Platform SDK

Foram efetuados várias tentativas na utilização desta tecnologia para reconhecimento de fala em português, mas todas as tentativas fracassaram devido ao problema técnico encontrado. No reconhecimento de inglês esta tecnologia funcionou sem problemas, no entanto depois de instalado o runtime de portuguesa e outros componentes necessários o motor de reconhecimento simplesmente não conseguiu inicializar o serviço de reconhecimento. Foram efetuados várias pesquisas focadas na resolução deste problema mas todos sem sucesso.

3.1.3 Nuance

Não foram executados testes devido à falta de licença para o produto.

3.1.4 Decisão tecnológica

Dado o problema técnico encontrado com a Microsoft Speech Platform SDK, a impossibilidade na utilização de Nuance por falta da licença do produto e após a análise e comparação do esforço necessário na integração e requisitos na utilização destas tecnologias, foi proposta à Glintt a utilização do Web Speech API como o motor de reconhecimento de fala para desenvolvimento deste protótipo pela sua simplicidade na integração.

	Integração do serviço	**Requisitos software/hardware**
Web Speech API	• Criar uma biblioteca em javascript para invocar o serviço.	Browser com suporte.
Microsoft Speech Platform SDK	• Instalar o runtime languagem pretendido. • Instalar o Microsoft .NET Framework version 4.0. • Instalar o Speech Platform SDK 11.	Sistema operativo e componentes hardware identificado no capítulo anterior
Nuance	• Não foi possível analisar.	Browser com suporte.

Tabela 11 - Comparação de esforço na integração do serviço de reconhecimento de fala

Para além das tecnologias de reconhecimento de voz também foram identificados algumas ferramentas de processamento de linguagem natural, mas nenhum deles fornece o suporte a língua Português. Apesar deste ponto impeditivo foram implementados testes com o objetivo de analisar o funcionamento e recolher dados que servirão de base para o desenvolvimento de algoritmo de PLN.

3.1.5 Stanford Parser

Nos testes efetuados foram utilizadas as seguintes frases em Inglês:

- search for the patient 123.
- search for the patient 123 and the patient 456
- search for the medical examination of the patient 123

Input	Tagging	Parse	Statics
search for the patient 123	search/VB for/IN the/DT patient/JJ 123/CD	(ROOT (S (VP (VB search) (PP (IN for) (NP (DT the) (JJ patient) (CD 123))))))	Tokens: 5 Time: 0.010 s
search for the patient 123 and the patient 456	search/VB for/IN the/DT patient/JJ 123/NN and/CC the/DT patient/JJ 456/CD	(ROOT (S (VP (VB search) (PP (IN for) (NP (NP (DT the) (JJ patient) (NN 123)) (CC and) (NP (DT the) (JJ patient) (CD 456)))))))	Tokens: 9 Time: 0.033 s
search for the medical examination of the patient 123	search/VB for/IN the/DT medical/JJ examination/NN of/IN the/DT patient/JJ 123/CD	(ROOT (S (VP (VB search) (PP (IN for) (NP (NP (DT the) (JJ medical) (NN examination)) (PP (IN of) (NP (DT the) (JJ patient) (CD 123))))))))	Tokens: 9 Time: 0.033 s

Tabela 12 – Resultado do teste efetuado com o Stanford Parser

Como podemos constatar na tabela anterior, o tempo que o algoritmo leva a processar uma frase com 9 palavras é bastante reduzido (inferior a um segundo).

3.1.6 Natty

Para efeito de teste foram utilizadas as seguintes frase em inglês que contem uma data implícita:

- search the yesterday examination of the patient 123.
- search the two days ago examination of the patient 123

Frase	Resultado
search the yesterday examination of the patient 123	Fri Sep 09 16:44:39 UTC 2016
search the two days ago examination of the patient 123	Thu Sep 08 16:45:56 UTC 2016

Tabela 13 - Resultado do teste efetuado com o Natty

Dado a ausência de ferramentas de processamento de linguagem natural com suporte a língua portuguesa será necessário implementar um algoritmo capaz de efetuar os tratamentos necessários com objetivo de extrair informações essenciais para a execução da ação pretendida.

3. 2 Biblioteca Javascript

Para garantir a comunicação entre os dois componentes do protótipo a implementar foi necessário criar uma biblioteca em linguagem javascript que será responsável pela invocação do serviço de reconhecimento de fala e a comunicação dos dados entre o motor de reconhecimento e o algoritmo de PLN. Esta biblioteca servirá também como a ponte de integração deste protótipo nas aplicações web existentes de uma forma simplificada.

A biblioteca é composta por dois componentes simples, a invocação da interface da API do reconhecimento automático de fala e a invocação do serviço de PLN para extração do contexto da frase obtida.

```
if (('webkitSpeechRecognition' in window)) {
    var recognition = new webkitSpeechRecognition();
    recognition.continuous = true;
                        :
    recognition.onresult = function (event) {

        while (final_transcript.charAt(0) === ' ')
            final_transcript = final_transcript.slice(1);
        if (final_transcript.startsWith(prefix)) {
            serviceCall2(final_transcript.replace(prefix,""));
        }
                        :
    };
}

function serviceCall2() {
        :
}

function call_action() {
        :

}
```

Figura 17 - Biblioteca Javascript

A figura 17 ilustra a invocação da interface do reconhecimento de fala efetuada pela biblioteca de javascript, em anexo 6 encontram-se informações mais detalhadas sobre esta biblioteca implementada.

3. 3 Reconhecimento automático de fala

Antes de invocar a interface de reconhecimento de fala e iniciar o processo de gravação de áudio é necessário verificar se o browser utilizado suporte o Web Speech API através de validação de existência do objeto *webkitSpeechRecognition*.

Caso afirmativo é necessário criar uma instância do objeto *webkitSpeechRecognition* que irá fornecer a interface de voz e definir alguns dos seus atributos bem como os manipuladores dos eventos.

```
if (('webkitSpeechRecognition' in window)) {
    var recognition = new webkitSpeechRecognition();
    recognition.continuous = true;
    recognition.onstart = function () {...};
    recognition.onend = function () {...};
    recognition.onerror = function(event) { ... }
    recognition.onresult = function (event) {...};
}
```

Figura 18 – webkitSpeechRecognition interface

Nesta API existem duas possibilidades de reconhecimento de fala, o reconhecimento continuado e o reconhecimento simples. No reconhecimento simples o processo de recolha de dados áudio termina quando o utilizador deixa de pronunciar falas. Ou seja neste modo, sempre que o utilizador pretender que a aplicação efetue o reconhecimento de fala pronunciada tem de iniciar o processo de gravação de áudio através de um elemento gráfico existente na aplicação. No processo continuado, o browser está constantemente a recolher os dados áudio para serem convertidos em textos pela API, e apenas termina quando o utilizador selecionar o elemento gráfico destinado a interromper para este fim.

A escolha do tipo de reconhecimento a efetuar é definida na variável ***continuos*** da interface *speechrecognition*.

Para o desenvolvimento deste protótipo o interesse é possuir um processo de reconhecimento de fala continuado, onde a aplicação está constantemente à escuta da fala dos utilizadores. No entanto existe um problema na implementação deste tipo de reconhecimento de fala, porque nem todas as frases pronunciadas pelo locutor devem ser processados pela aplicação, ou seja, quando é que uma frase deve ser enviada para o servidor onde será analisado pelo algoritmo de PLN. Por exemplo, um médico pronunciou uma frase de pesquisa “Pesquisar o doente XPTO” e de seguida pede o enfermeiro para ir buscar algo. O processo de reconhecimento de fala irá capturar ambas as falas, mas apenas a frase de pesquisa deve ser enviada para o servidor para ser processada pelo algoritmo de PLN.

Para solucionar este problema, foi proposta a definição de uma palavra-chave com o objetivo de validar se uma frase é elegível para ser enviado para o servidor para ser processada. Ou seja uma frase só é elegível para ser enviada quando iniciar por uma palavra específica definida

previamente, neste caso será a palavra “computador”. Por exemplo “computador pesquisar o doente XPTO”, onde a palavra “computador” será a palavra de validação para que o processo possa decidir se a frase deve ou não ser enviada para o servidor onde será processada pelo algoritmo de PLN.

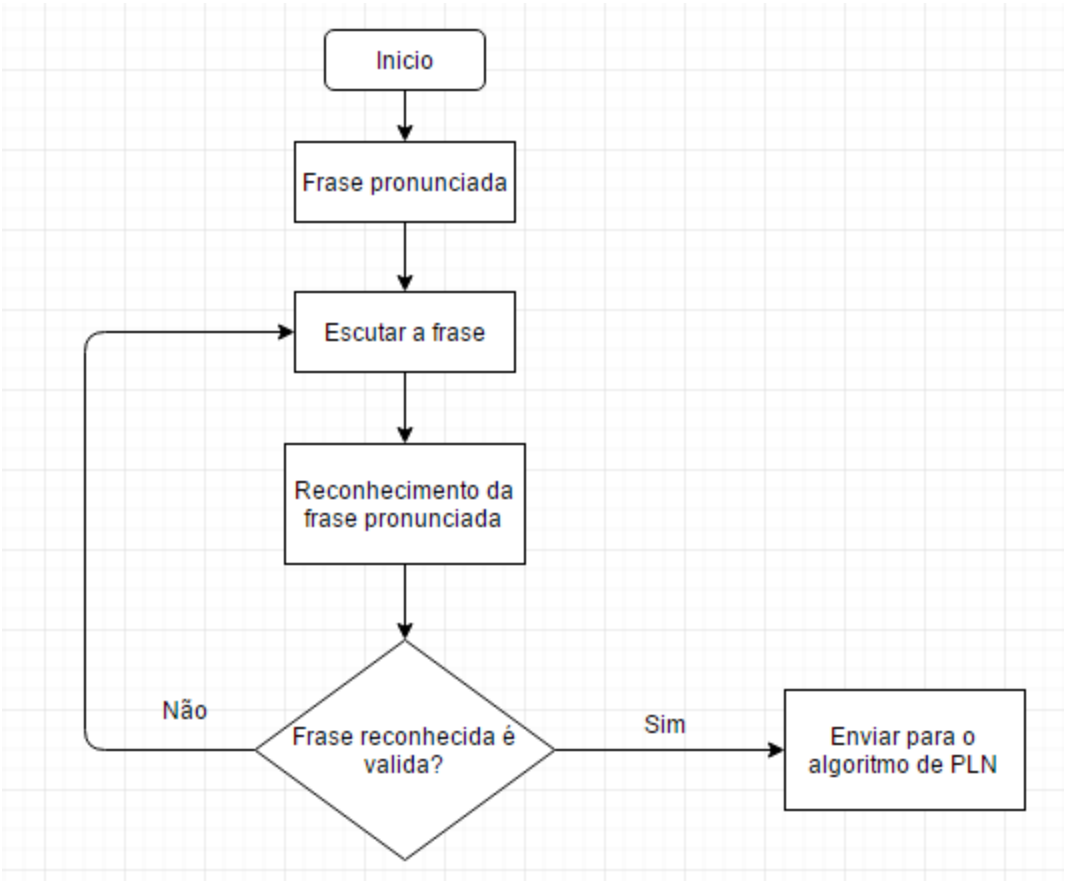

Figura 19 - Fluxograma 1

A figura 19 demonstra o comportamento do protótipo na validação de uma frase a fim de determinar se a frase é elegível ou não para ser enviada para o algoritmo de PLN para ser processado. De uma forma geral a biblioteca invoca o serviço de reconhecimento continuo, o serviço recolhe e procede à conversão da falar, a fala convertida é devolvida pelo motor de reconhecimento e posteriormente analisada para validar a existência de palavra-chave, caso se confirme a existência da palavra-chave a frase é enviada para o algoritmo para ser processada, caso contrario a frase será ignorada e o protótipo fica à escuta de uma próxima fala.

3. 4 Algoritmo de PLN

O algoritmo de PLN implementado tem como objetivo efetuar vários tratamentos necessários à frase recebida, a fim de extrair um contexto válido contendo as informações essenciais para a execução da ação pretendida com a frase pronunciada.

Antes de proceder à extração do contexto é necessário validar se a frase recebida está corretamente formulada ou não. Para isso é efetuado um conjunto de validações que incluem a análise morfológica, sintática e semântica.

Para proceder às validações necessárias, primeiramente as palavras que compõem a frase são classificadas de acordo com o seu significado.

Figura 20 - Conversão de frase numa estrutura manipulável

Após a classificação das palavras é feita a validação sintática para análisar o relacionamento e emprego das palavras na frase e por fim a validação semântica para validar o significado da frase.

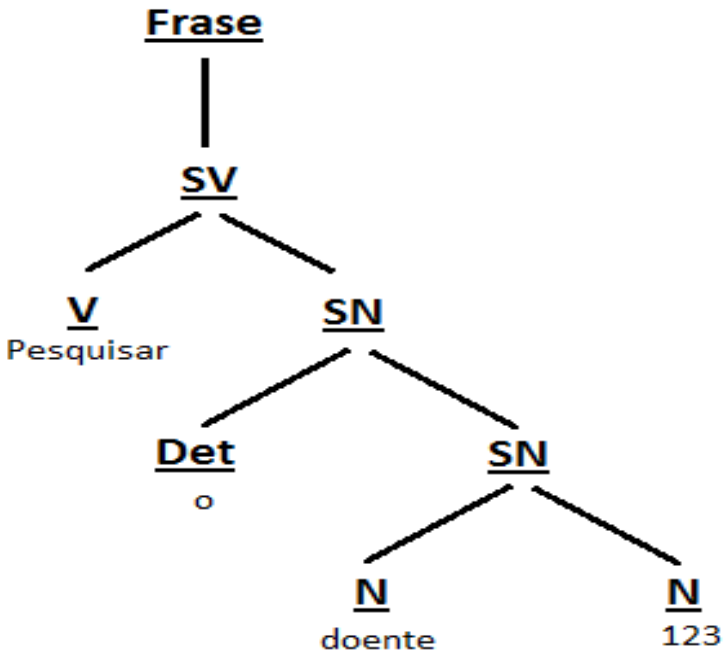

Figura 21 - Arvore de parse da frase "Pesquisar o doente 123"

Durante o desenvolvimento foi proposta pela Glintt a possibilidade de tratamento das frases que estão sintaticamente incorretas mas que é possivel extrair da frase um contexto valido. Quando as pessoas falam, são geralmente bastante descuidadas no que toca à formulação de uma frase, e isso leva à produção de erros fonéticos que podem influenciar o resultado da API de reconhecimento de voz. Dada a frase "computador pesquisar o doente XPTO", ao ser pronunciado muitas das vezes o artigo pode ser esquecido, ou devido aos fatores ambientais (i.e ruídos) a API converteu erradamente a frase, ficando apenas "computador pesquisar doente XPTO", para um ser humano é perfeitamente possível reconhecer a ação pretendida com a frase, no entanto para um algoritmo de PLN ao efetuar a validação sintática, esta não seria uma frase valida porque a sua construção sintática está incorreta.

Face a esta situação e após a análise em conjunto com a Glintt, a solução passaria por retirar do algoritmo de PLN as validações sintáticas e semânticas. Com esta solução proposta o que se pretende é que o protótipo passe a ter um comportamento semelhante ao de um ser humano, ou seja mesmo que uma frase esteja mal formulada ou que não faça sentido, desde que seja possível extrair da frase uma ação e um sujeito então esta frase passa a ser válida. Por exemplo:

- Frase 1 - Pesquisar doente XPTO.
- Frase 2 - Pesquisar ontem doente XPTO

Para estas duas frases apresentadas, apesar de estarem mal formuladas, as frases são consideradas como sendo válidas porque é possível para o algoritmo extrair das frases uma ação e um sujeito.

Com esta alteração é possível que o algoritmo venha extrair ações falsas ou seja ações que não são o que é pretendido pelo locutor ao pronunciar uma determinada frase, essas ações falsas podem resultar de um erro de conversão no processo de reconhecimento de fala, descuido de fala por parte de utilizador ou um qualquer outro motivo mas a Glintt pretende assumir esta margem de erro e avançar com esta solução.

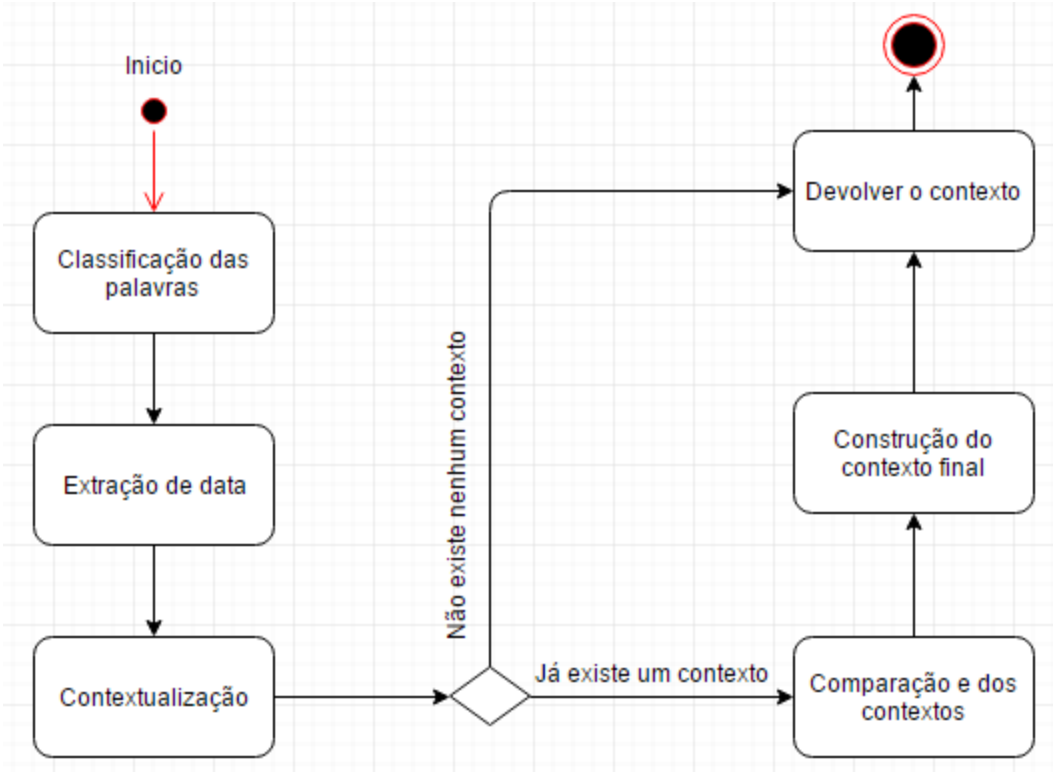

Figura 22 – Funcionamento de algoritmo

A figura 22 demonstra o comportamento do algoritmo de PLN no processamento de uma frase. A frase recebida pelo algoritmo é classificado de acordo com a base de gramática, de seguida o algoritmo procede à extração de datas existentes na frase de acordo com os formatos de data definidos no algoritmo. Após a extração de datas é construído o contexto com base na frase recebida, caso já exista um contexto (extraído no processamento de uma

outra frase pronunciada num momento anterior) o algoritmo procede à comparação dos contextos a fim de determinar o contexto final a devolver.

3. 5 Extração de data em formato de texto

Para identificar os possíveis formatos de datas que se encontram numa frase foi implementado um processo de identificação que recorre às expressões regulares para os identificar. A semelhança da API Natty os formatos de datas são divididos em dois grupos:

- Explicitas – Quando existe um formato claramente percetível. Por exemplo: 15 de janeiro de 2016
- Implícitas – Quando não existe um formato percetível. Por exemplo: últimos 2 dias.

O algoritmo aplica um conjunto de expressões regulares para identificar as datas na frase recebida, caso exista a correspondência o processo efetua a extração de data de acordo com o formato de data detetado.

Caso não encontre nenhuma correspondência o algoritmo assume que não existe nenhuma data a extrair e prossegue com o processo de PLN.

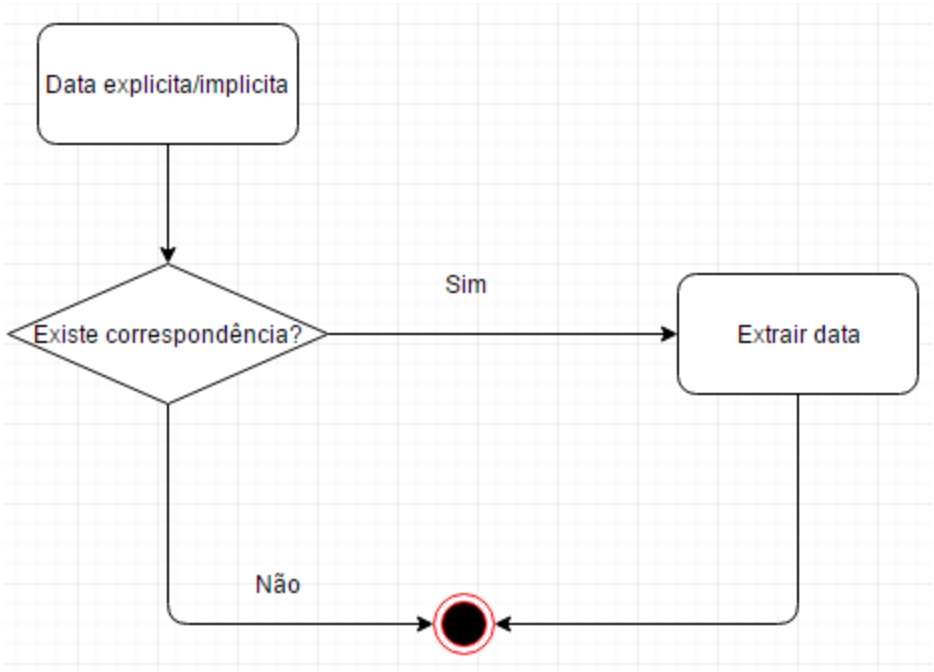

Figura 23 - Extração de data

A figura anterior demonstra o comportamento do algoritmo na extração de datas existentes na frase a processar.

3. 6 Contexto

O contexto é o resultado final do algoritmo de PLN em formato **JSON** que contém os dados essenciais para execução de ação pretendida com a frase pronunciada.

A criação do contexto é essencial para que sejam possíveis as operações de pesquisa continuadas sob um determinado contexto, por exemplo o locutor pronuncia a frase "computador pesquisar o exame dos doentes XPTO1 e XPTO2" após o processamento da frase será criado um contexto de acordo com os dados extraídos, de seguida é pronunciada uma segunda frase "computador pesquisa apenas de ontem", através do contexto anteriormente criado o processo é capaz de identificar a ação correta que deve ser executada ao processar a segunda frase, ou seja pesquisar o exame efetuados no dia anterior ao atual do doente XPTO1 e XPTO2. Caso não existisse o contexto, para o processo de PLN a segunda frase seria inválida porque não é possível identificar o sujeito.

O processo de contextualização funciona de uma forma simples, a cada frase recebida é extraído um contexto, a esse contexto extraído é efetuada uma comparação com um contexto extraído num momento anterior (caso exista). Esta comparação serve essencialmente para validar se o contexto atual extraído contém apenas dados a acrescentar ao contexto anterior ou se deve descartar o contexto anterior mantendo o contexto atual. Um dos principais problemas encontrados na implementação deste processo consiste em dotar o algoritmo da capacidade de decisão sobre quando deve manter o novo contexto descartando o antigo ou alterar o contexto antigo acrescentando dados novos do contexto atual.

Foi proposta e validada pela Glintt a criação de uma regra de validação que permite ao processo contextualização decidir quando deve alterar um contexto e quando deve descartar um antigo mantendo o novo contexto. Com esta solução proposta, sempre que o contexto extraído atualmente possuir um sujeito que não existe no contexto anteriormente criado, então o algoritmo deve descartar o contexto antigo e manter o novo contexto extraído na frase atual, caso contrário deve acrescentar dados do contexto novo ao contexto já existente.

Com a definição desta regra de validação sempre que surge uma frase com um novo sujeito é substituído o contexto antigo pelo novo contexto extraído.

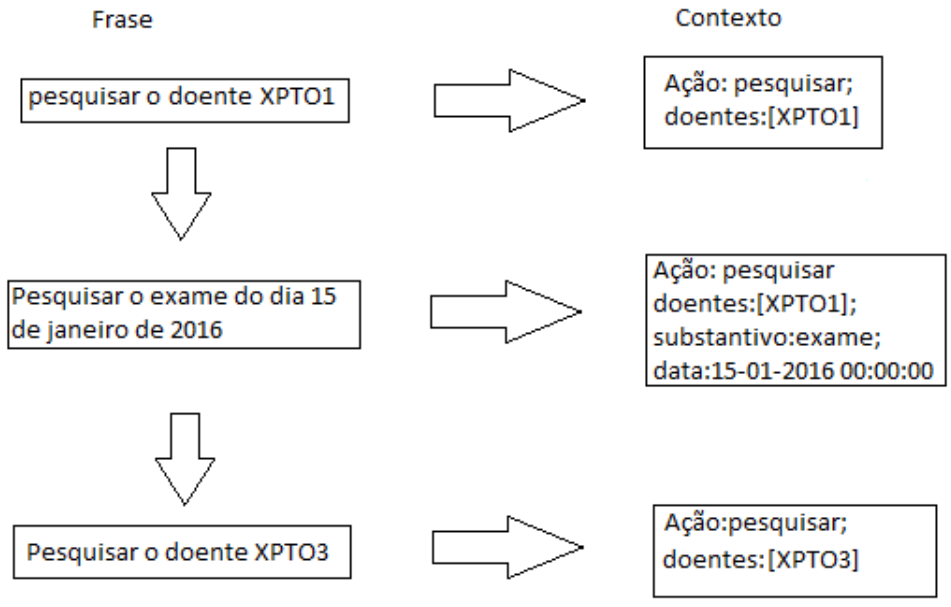

Figura 24 – Contextualização

A figura 24 é possivel visualizar os contextos extraidos pelo algoritmo no processamento das três frases pronunciadas em momentos distintos:

- Momento 1 – Pesquisar o doente XPTO1.
- Momento 2 – Pesquisar o exame do dia 15 de janeiro de 2016.
- Momento 3 – Pesquisar o doente XPTO3.

No momento 1 o utilizador pronunciou a frase "pesquisar o doente XPTO1", a frase é enviada para o algoritmo PLN para que seja processado e extraído o contexto. Como ainda não existe nenhum contexto, o contexto devolvido será "ação:pesquisar; doente: XPTO1".

A frase pronunciada no momento 2 é enviada para o algoritmo juntamente com o contexto extraído no momento 1. Após o tratamento e extração do contexto da frase pronunciada no momento 2, o algoritmo verifica a existência de um contexto criado num momento anterior e procede a comparação dos 2 contextos, neste caso o contexto extraído no momento 2 não possui nenhum sujeito, o algoritmo entende que este contexto possui apenas dados que devem ser acrescentado ao contexto extraído no momento 1. Após a alteração do contexto, o contexto devolvido pelo algoritmo será "ação:pesquisar; substantivo:exame; doente: XPTO1; data: 15-01-2016; data_relation:single".

Ao pronunciar a 3ª frase o contexto extraído possui um novo sujeito, após a comparação dos contextos usando a regra de validação definida, o algoritmo decide que deve descartar o contexto devolvido pelo algoritmo no momento 2 e manter o contexto extraído no momento 3 devolvendo assim o contexto "ação:pesquisar; doente:XPTO3".

3. 7 Integração

Para integrar o protótipo nas aplicações existentes existem alguns requisitos a implementar:

- Criar uma nova página onde existe a possibilidade de invocar o serviço de reconhecimento de fala.
- Implementar um serviço que irá analisar o resultado devolvido pelo serviço de PLN.

A implementação do serviço de análise de resultado devolvido pelo serviço de PLN tem de ser feito nas aplicações existentes porque cada aplicação representa um conjunto de conceitos de negócios diferentes, e esses conceitos não devem ser replicados no protótipo visto ser desnecessária, e também tornaria a integração mais complexa caso os conceitos fossem implementados no protótipo.

Nas aplicações existentes é necessário implementar uma nova página web contendo um elemento gráfico para ativar o serviço de reconhecimento de fala, e uma secção para apresentar o resultado de ação pretendida com a frase pronunciada. Esta nova página é necessária porque o serviço de reconhecimento é interrompido quando existe uma mudança de página, para que o reconhecimento seja contínuo é necessário que todas as operações sejam feito numa mesma página web.

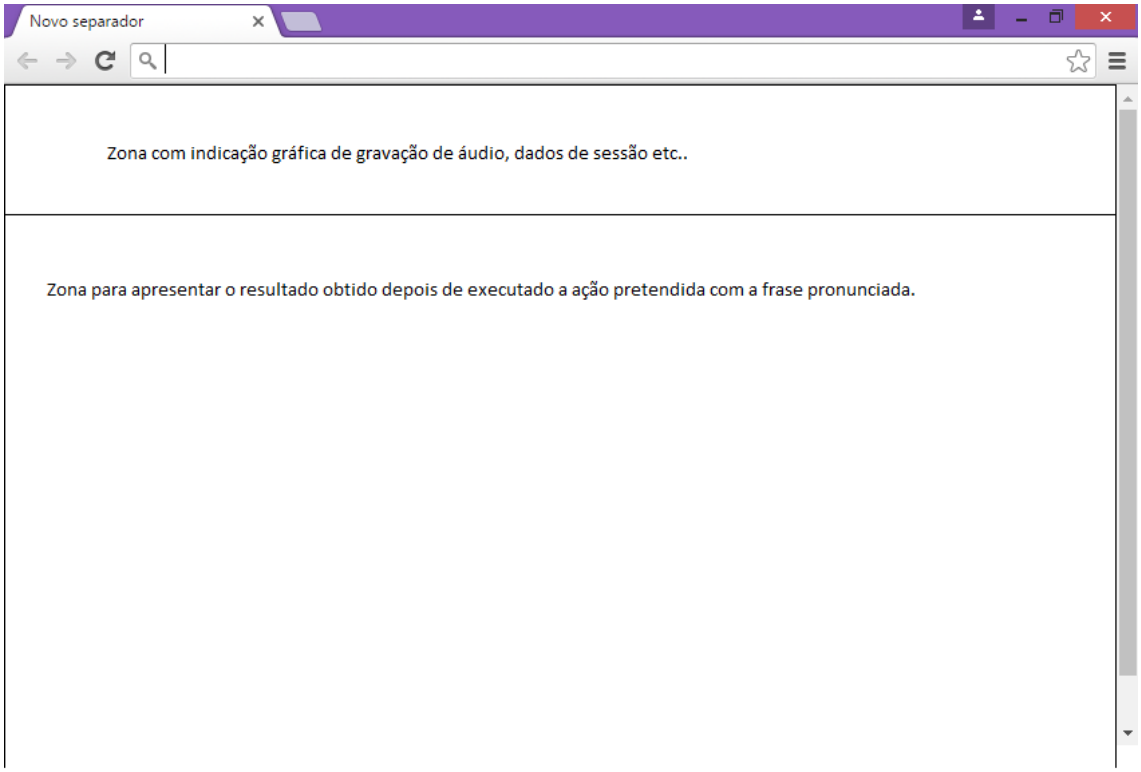

Figura 25 - Estrutura de nova pagina a criar

Está disponível no anexo 5 o diagrama de classe representativo do algoritmo de PLN implementado.

3. 8 Teste

A fim de validar o protótipo implementado a partir de um ponto de vista de usabilidade, foram efetuados testes de usabilidade com diferentes frases de pesquisas com o objetivo de reunir os dados objetivos e subjetivos de dois utilizados com idade compreendido entre os 25 e os 35 anos, de nacionalidade portuguesa com nenhuma experiência neste tipo de ação.

3.8.1 Avaliação e testes

Os testes foram executados executado seguindo uma metodologia de sondagem, onde após o teste o utilizador responde a um questionário acerca da usabilidade do protótipo e as suas opiniões acerca do protótipo. Além disso são registados dados relativos aos testes com o objetivo de identificar a taxa de erro apresentado pelo protótipo no reconhecimento de fala, e o tempo necessário para completar as tarefas.

A sessão seguinte descreve como os testes foram realizados e os resultados destes testes.

Cada sessão de teste terá uma duração aproximadamente de 8 minutos, com as seguintes repartições:

- É apresentado ao utilizador os objetivos do protótipo implementado e explicado o teste a ser realizado (1 minuto).
- Testes por parte de utilizador tem uma duração de 5 minutos, os testes seguiram as condições:
    - Máximo de 3 tentativas por frase.

- Microfone do portátil ASUS K53S com o utilizador posicionado a uma distância não superior a 30 centímetros. Cada frase pronunciada em condições com e sem ruídos.
- Um microfone que se encontra junto do utilizador. Cada frase projetada em condições com e sem ruídos.

- Resposta ao questionário 2 minutos.

3.8.2 Resultados

Durante os testes os utilizadores pronunciaram um conjunto de frases em ambiente com e sem ruídos, e os resultados obtidos não foram muito positivos. Apesar dos testes iniciais realizado com o WEB Speech API apresentarem um resultado positivo no reconhecimento de voz, durante os testes do protótipo a taxa de erros apresentado foi bastante elevada seja em ambiente com ruído ou sem ruído.

Frase pronunciada	Tentativas	Tempo (segundos)
Computador pesquisar o doente 123 e o 124.	3	7.91
Computador pesquisar apenas o doente 124	3	Sem resultado
Computador pesquisar o resultado do doente 123	2	7
Computador pesquisar apenas o resultado do dia 15 de janeiro de 2016	3	11.3

Tabela 14 - Teste efetuado pelo utilizador 1

Frase pronunciada	Tentativas	Tempo (segundos)
Computador pesquisar o doente 123 e o 124.	3	7.91
Computador pesquisar apenas o doente 124	3	Sem resultado
Computador pesquisar o resultado do doente 123	3	Sem resultado
Computador pesquisar apenas o resultado do dia 15 de janeiro de 2016	3	Sem resultado

Tabela 15 - Teste efetuado pelo utilizador 2

Nas tabelas 14 e 15 é possível visualizar os resultados obtidos nos testes efetuados pelos 2 utilizadores no ambiente sem ruído com uma distância de cerca de 30 centímetros entre os utilizadores e o microfone. Como é possível constatar durante os testes a taxa de erro no reconhecimento de falar é bastante elevado, existe situação em que é atingido o limite

máximo de tentativas sem obter um resultado esperado. Em anexo 3 encontra-se uma lista completa dos dados recolhidos durante os testes.

Frase pronunciada	Tempo (segundos)
Computador pesquisar o doente 123 e o 124.	0.059
Computador pesquisar apenas o doente 124	0.052
Computador pesquisar o resultado do doente 123	0.054
Computador pesquisar apenas o resultado do dia 15 de janeiro de 2016	0.066

Tabela 16 - Tempo que algoritmo necessita para processamento de uma frase

Na tabela 16 é possível visualizar o resultado do teste realizado com o objetivo de determinar o tempo que o algoritmo necessita para efetuar o processamento de uma frase, com os dados presente na tabela anterior é possível constatar que nenhuma das quatros frases utilizadas para o teste levou mais de que um segundo a ser processada.

4. Conclusão

Analisando os dados recolhidos durante o teste final podemos concluir que o motor de reconhecimento de fala está muito dependente da qualidade de áudio recolhido, da velocidade e a forma como a frase é pronunciada e dos equipamentos utilizados para recolha dos dados áudio. Tudo isso pode influenciar o resultado produzido pelo motor de reconhecimento de voz e posteriormente influenciar o resultado do serviço de PLN.

É de salientar que a constante repetição de uma frase sem resultado pretendido pode levar a um cansaço emocional e gerar um clima de frustração na utilização desta solução que pode colocar em causa o sucesso desta solução, durante testes foi possível observar um certo grau de frustração por parte dos utilizadores.

Foi também possível verificar um desvio de tempo de resposta do protótipo implementado com o objetivo inicial de ser inferior a 1 segundo. Esse desvio deve se principalmente ao tempo que a API necessita para efetuar o reconhecimento, quando mais complexa for a frase (i.e número de palavras) pronunciada mais tempo a API necessita para efetuar o reconhecimento, o ambiente envolvente, a forma (com clareza ou não) e a velocidade com que a frase é pronunciada também podem influenciar o tempo do processo de reconhecimento. Conclui-se de que este objetivo inicial de tempo de resposta do protótipo inferior a 1 segundo definido é algo que não corresponde à realidade tecnológica existente, pelo menos com a tecnologia atual não é possível atingi-lo. Devido a esta limitação foi alterado este objetivo de tempo de resposta do protótipo para o tempo de resposta do algoritmo de PLN.

Podemos concluir que o protótipo cumpriu os objetivos definidos, no entanto a nível de usabilidade este protótipo não atingiu as expectativas devido à elevada taxa de erro apresentado no reconhecimento de fala (web speech API como o motor de reconhecimento de fala).

A tabela seguinte demonstra as restrições encontradas na implementação.

Restrição	**Descrição**
Ruídos e capacidade de microfone.	Além ruído presente no ambiente as limitações dos equipamentos de recolhe de áudios podem influenciar o resultado.
Unidades acústicas	Existem fonemas que produzem sons surdos que pode ser interpretado com sendo separações das palavras.
Locutor	Diferentes locutores podem falar de uma forma diferente e esta forma de falar pode influenciar o resultado de reconhecimento, por exemplo: velocidade com que fala, dialetos, etc.
Distanciação entre o locutor e o microfone	A distância entre o locutor e o microfone pode influenciar a qualidade dos dados áudios recolhidos.

Tabela 17 - Restrições na utilização do motor de reconhecimento de fala

4. 1 Trabalhos Futuros

Entende-se por trabalhos futuros, encontrar meios para solucionar os problemas encontrados na implementação do protótipo.

É necessário conduzir uma nova análise no sentido de encontrar um motor de reconhecimento com menor taxa de erro em comparação com a web speech API.

É necessário implementar uma estrutura gramatical eficiente, a base gramatical utilizada no protótipo é uma base pequena com dezenas de entradas. É necessário rever e otimizar o algoritmo implementado para extração de datas, pois este está limitado a uma gramática de formatos de datas previamente identificadas, caso seja necessário implementar um novo formato é preciso efetuar uma alteração ao algoritmo por forma a acrescentar o novo formato.

A construção do contexto deve melhorado, atualmente o processo de construção de contexto segue uma regra simples de criação do contexto, ou seja quando detetado um novo sujeito dá lugar a construção de uma novo contexto, descartando o contexto atual existente.

Analisar e estudar a possibilidade de incluir a funcionalidade de manipulação de ficheiro, vídeo e imagens, atualmente o protótipo apenas permite efetuar as operações de pesquisar.

5. Referencias

[1] API, W. S. (2013). Obtido de https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API.

[2] Caniuse. (2016). Obtido de http://caniuse.com/#feat=speech-recognition.

[3] ISEP. (2013). Documentação utilizada na disciplina Inteligencia artificial ISEP.

[4] Microsoft. (2016). Obtido de https://msdn.microsoft.com/en-us/library/hh362873(v=office.14).aspx.

[5] Nuance. (2016). Obtido de https://developer.nuance.com/public/index.php?task=prodDev

[6] Natty, N. d. (2016). Obtido de http://natty.joestelmach.com/

[7] Stanford. (2016). *Stanford University*. Obtido de http://nlp.stanford.edu/software/lex-parser.shtml

[8] Docsoft.inc. (06 de 2009). What is Automatic Speech Recognition?

[9] Qin, L. (2013). Learning Out-of-VocabularyWords in Automatic Speech Recognition.

[10] BEARDON, C., LUMSDEN, D., & HOLMES, G. (1991). G. Natural Language and Computational Linguistics.

[11] Mahesh, K., & Nirenburg, S. (s.d.). Knowledge-Based Systems for Natural Language Processing.

[12] Elaine Pina, Etelvina Ferreira, Alexandre Marques e Bruno Mato. Martin. Infeccoes associadas aos cuidados de saude e seguranca do doente.

[13] Daniel Jurafsky and James H. Martin. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.

[14] Adriano de Andrade Bresolin. Estudo do Reconhecimento de Voz para o Acionamento de Equipamentos Elétricos via Comandos em Português.

[15] Rui Alexandre dos Reis Marques Cardoso. As infeções associadas aos cuidados de saúde.

[17] Match-project.http://www.match-project.org.uk/resources/tutorial/Speech_Language/Speech_Recognition/Rec_4.html.

[17] Techwalla. https://www.techwalla.com/articles/the-disadvantages-of-voice-recognition-software.

[18] ALLAN SHIGUETO AKISHINO. Reconhecimento automático de fala.

[19] Web Speech API. https://dvcs.w3.org/hg/speech-api/raw-file/tip/webspeechapi.html#use_cases.

[20] Direção-Geral da Saúde. Programa Nacional De Prevenção E Controlo Da Infeção Associada Aos Cuidados De Saúde.

[21] Paulo Guilhoto, Susana Rosa. Reconhecimento de Fala (Universidade de Coimbra).

[22] Carlos Alberto Ynoguti. Reconhecimento de Fala Contínua Usando Modelos Ocultos de Markov.

[23] Bento Silva,Osvaldo Jr.. Introdução ao Processamento das Línguas Naturais e Algumas Aplicações.

[24] Ricardo Annes. Aplicações De Sistemas Multiagentes e Aprendizagem Automática No Processamento Da Linguagem Natural.

[25] José Maria Cesário Júnior. Sobre O Conceito De Processamento De Linguagem Natural.

[26] Danilo Martins, Karina Kataoka e Leonardo Trindade. Processamento De Linguagem Natural.

[27] Luís Filipe Moreira. Reconheciemnto automático de fala contínua.

[28] Só Português. http://www.soportugues.com.br/secoes/fono/fono1.php.

[29] Flip. http://www.flip.pt/Duvidas-Linguisticas/Duvida-Linguistica/DID/3357.

Anexo 1

INTERFACE WEB Speech API

```
[Constructor]
  interface SpeechRecognition : EventTarget {
    // recognition parameters
    attribute SpeechGrammarList grammars;
    attribute DOMString lang;
    attribute boolean continuous;
    attribute boolean interimResults;
    attribute unsigned long maxAlternatives;
    attribute DOMString serviceURI;
    // methods to drive the speech interaction
    void start();
    void stop();
    void abort();
    // event methods
    attribute EventHandler onaudiostart;
    attribute EventHandler onsoundstart;
    attribute EventHandler onspeechstart;
    attribute EventHandler onspeechend;
    attribute EventHandler onsoundend;
    attribute EventHandler onaudioend;
    attribute EventHandler onresult;
    attribute EventHandler onnomatch;
    attribute EventHandler onerror;
    attribute EventHandler onstart;
    attribute EventHandler onend;
  };

  interface SpeechRecognitionError : Event {
    enum ErrorCode {
      "no-speech",
      "aborted",
      "audio-capture",
      "network",
      "not-allowed",
      "service-not-allowed",
      "bad-grammar",
      "language-not-supported"
    };
    readonly attribute ErrorCode error;
    readonly attribute DOMString message;
  };
  // Item in N-best list
  interface SpeechRecognitionAlternative {
    readonly attribute DOMString transcript;
    readonly attribute float confidence;
  };
```

```
// A complete one-shot simple response
interface SpeechRecognitionResult {
   readonly attribute unsigned long length;
   getter SpeechRecognitionAlternative item(in unsigned long index);
   readonly attribute boolean isFinal;
};
// A collection of responses (used in continuous mode)
interface SpeechRecognitionResultList {
   readonly attribute unsigned long length;
   getter SpeechRecognitionResult item(in unsigned long index);
};
// A full response, which could be interim or final, part of a continuous response or not
interface SpeechRecognitionEvent : Event {
   readonly attribute unsigned long resultIndex;
   readonly attribute SpeechRecognitionResultList results;
   readonly attribute any interpretation;
   readonly attribute Document emma;
};
// The object representing a speech grammar
[Constructor]
interface SpeechGrammar {
   attribute DOMString src;
   attribute float weight;
};
// The object representing a speech grammar collection
[Constructor]
interface SpeechGrammarList {
   readonly attribute unsigned long length;
   getter SpeechGrammar item(in unsigned long index);
   void addFromURI(in DOMString src,
                   optional float weight);
   void addFromString(in DOMString string,
                   optional float weight);
};
```

Anexo 2

Serviço de speech-to-text fornecido pela empresa NUANCE.

Silver	Gold	Esmerald
Cloud speech recognition (ASR) with models for dictation, websearch, and TV	Cloud speech recognition (ASR) with models for dictation, websearch, and TV	Cloud speech recognition (ASR) with models for dictation, websearch, and TV
Vocabularies for ASR Customization Up to 200 phrases Up to 40 vocabularies	Vocabularies for ASR Customization Up to 1,000 phrases Up to 80 vocabularies	Vocabularies for ASR Customization Up to 3,000 phrases Up to 160 vocabularies Multiple vocabularies / languages / topics
Cloud Text-to-Speech (TTS) for 40+ languages and 60+ voices	Cloud Text-to-Speech (TTS) for 40+ languages and 60+ voices	Cloud Text-to-Speech (TTS) for 40+ languages and 60+ voices
Android, iOS, and Windows Phone 7 & 8 SDKs	Android, iOS, and Windows Phone 7 & 8 SDKs	Android, iOS, and Windows Phone 7 & 8 SDKs
HTTP REST Interface Evaluation only	HTTP REST Interface	HTTP REST Interface
SSL	SSL	SSL
Support	Support	Support Dedicated
Production Pricing Free up to 20,000 transactions / Mo. / App. Above 20,000 : $.008 / transaction	SLA	SLA Advanced
Payment Methods Paypal prepay	Production Pricing $.008 / transaction	Consulting SDK support Speech science consulting User experience & Design Advanced solutions
	Payment Methods Paypal prepay	Payment Methods $10k prepay Postpaid billing

Tabela 18 - Preçário de Nuance

Anexo 3

Utilizador -1

Teste efetuado sem ruido

Frase pronunciada	Tentativas	Tempo (segundos)
Computador pesquisar o doente 123 e o 124.	3	7.91
Pesquisar apenas o doente 124	3	Sem resultado
Computador pesquisar o resultado do doente 123	2	7
Computador pesquisar apenas o resultado do dia 15 de janeiro de 2016	3	11.3

Teste efetuado com ruido

Frase pronunciada	Tentativas	Tempo (segundos)
Pesquisar o doente 123 e o doente 123	2	7
Pesquisar apenas o doente 124	3	Sem resultado
Pesquisar o resultado do doente 123	3	Sem resultado
Pesquisar apenas o resultado do dia 15 de janeiro de 2016	3	Sem resultado

Utilizador 2

Teste efetuado sem ruido

Frase pronunciada	Tentativas	Tempo (segundos)
Computador pesquisar o doente 123 e o 124.	3	7.91
Pesquisar apenas o doente 124	3	Sem resultado
Computador pesquisar o	3	Sem resultado

resultado do doente 123		
Computador pesquisar apenas o resultado do dia 15 de janeiro de 2016	3	Sem resultado

Teste efetuado com ruido

Frase pronunciada	**Tentativas**	**Tempo (segundos)**
Pesquisar o doente 123 e o doente 123	3	Sem resultado
Pesquisar apenas o doente 124	3	Sem resultado
Pesquisar o resultado do doente 123	3	Sem resultado
Pesquisar apenas o resultado do dia 15 de janeiro de 2016	3	Sem resultado

Anexo 4

Microsoft.Speech.Recognition Namespace

Class	Description
AudioLevelUpdatedEventArgs	Provides data for the AudioLevelUpdated event.
AudioSignalProblemOccurredEventArgs	Provides data for the AudioSignalProblemOccurred event.
AudioStateChangedEventArgs	Provides data for the AudioStateChanged event of theSpeechRecognitionEngine class.
Choices	Represents a list of alternative items to make up an element in a grammar.
DtmfHypothesizedEventArgs	Provides data for the DtmfHypothesized event.
DtmfRecognitionEngine	Provides access to DTMF recognition services.
DtmfRecognitionRejectedEventArgs	Provides data for the DtmfRecognitionRejected event.
DtmfRecognizeCompletedEventArgs	Returns information from the RecognizeCompleted event.
DtmfRecognizedEventArgs	Provides data for the DtmfRecognized event.
EmulateRecognizeCompletedEventArgs	Provides data for the EmulateRecognizeCompleted event.
Grammar	A run-time object that references a speech recognition grammar, which an application can use to define the constraints for speech recognition.
GrammarBuilder	Provides a mechanism for programmatically building the constraints for a speech recognition grammar.
GrammarInfo	Represents an object that contains static information about a speech recognition grammar. In contrast, a Grammar object contains information about a speech recognition grammar that is loaded into a speech recognizer at run time.
GrammarInfoPartsCollection	Represents a collection of speech recognition engine parts.
InvalidCultureException	Thrown when a recognizer attempts to reference a grammar that specifies an unsupported or uninstalled language.
LoadGrammarCompletedEventArgs	Provides data for the **LoadGrammarCompleted** event.
RecognitionEventArgs	Provides information about speech recognition events.
RecognitionResult	Contains detailed information about input that was recognized by instances ofSpeechRecognitionEngine or DtmfRecognitionEngine.
RecognizeCompletedEventArgs	Provides data for the RecognizeCompleted event.
RecognizedAudio	Represents audio input that is associated with

	a RecognitionResult.
RecognizedPhrase	Contains detailed information, generated by the speech recognition engine, about the recognized input.
RecognizedWordUnit	Provides the atomic unit of recognized speech.
RecognizerInfo	Represents information about a SpeechRecognitionEngine object.
RecognizerUpdateReachedEventArgs	Provides data for the RecognizerUpdateReached event.
ReplacementText	Contains information about a speech normalization procedure that has been performed on recognition results.
SemanticResultKey	Associates a key string with SemanticResultValue values to define theSemanticValue objects.
SemanticResultValue	Represents a semantic value and optionally associates the value with a component of a speech recognition grammar.
SemanticValue	Represents the semantic organization of a recognized phrase.
SpeechDetectedEventArgs	Provides data for the SpeechDetected event.
SpeechHypothesizedEventArgs	Infrastructure. Provides data for the SpeechHypothesized event.
SpeechRecognitionEngine	Provides the means to access and manage a speech recognition engine.
SpeechRecognitionRejectedEventArgs	Provides information for the SpeechRecognitionRejected event.
SpeechRecognizedEventArgs	Provides information for the Grammar.SpeechRecognized andSpeechRecognitionEngine.SpeechRecognized events.

Microsoft.Speech.Recognition.SrgsGrammar Namespace

Class	Description
MssGrammarCompiler	Represents a compiler for speech recognition grammars.
SrgsDocument	Defines a design-time object that is used to build strongly typed runtime grammars that conform to the Speech Recognition Grammar Specification (SRGS) Version 1.0.
SrgsElement	Defines the base class for classes in the Microsoft.Speech.Recognition.SrgsGrammarnamespace that correspond to the elements in an SRGS grammar.
SrgsGrammarCompiler	Compiles SrgsDocument and XML-format grammar files into binary grammar files with the .cfg extension and sends the output to a stream.
SrgsItem	Represents a grammar element that contains phrases or other entities that a user can speak to produce a successful recognition.

SrgsOneOf	Represents a list of alternative words or phrases, any one of which may be used to match speech input.
SrgsRule	Represents a grammar rule.
SrgsRuleRef	Represents the grammar element that specifies a reference to a rule.
SrgsRulesCollection	Represents a collection of SrgsRule objects.
SrgsSemanticInterpretationTag	Represents a tag that contains ECMAScript that is run when the rule is matched.
SrgsText	Represents the textual content of grammar elements defined by the World Wide Web Consortium (W3C) Speech Recognition Grammar Specification (SRGS) Version 1.0.
SrgsToken	Represents a word or short phrase that can be recognized.

Microsoft.Speech.AudioFormat Namespace

Class	Description
SpeechAudioFormatInfo	Represents information about an audio format.

Microsoft.Speech Namespace

Interface	Description
ISpeechResourceLoader	Gives applications control over loading resources.

Anexo 5

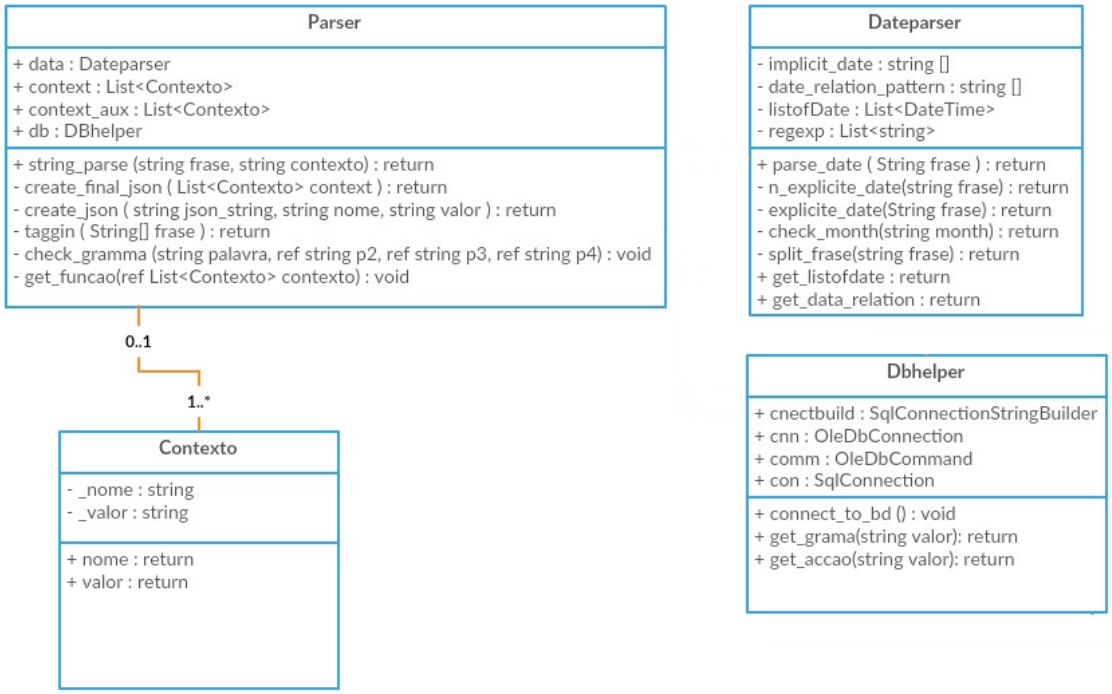

Figura 26 -Diagrama de Classe do algoritmo de PLN

Anexo 6

Biblioteca implementada para garatir a comunicação entre os dois componentes do prototipo.

```
var final_transcript = '';
var recognizing = false;
var ignore_onend;
var start_timestamp;
var prefix = 'computador';

if (('webkitSpeechRecognition' in window)) {

	//start_button.style.display = 'inline-block';
	var recognition = new webkitSpeechRecognition();
	recognition.continuous = true;
	//recognition.interimResults = true;
	recognition.onstart = function () {
		recognizing = true;
		showInfo('info_speak_now');
	};

	recognition.onend = function () {
		recognizing = false;
	};

	recognition.onresult = function (event) {
		var interim_transcript = '';
		for (var i = event.resultIndex; i < event.results.length; ++i) {
			if (event.results[i].isFinal) {
				final_transcript += event.results[i][0].transcript;
			} else {
				interim_transcript += event.results[i][0].transcript;
			}
		}
		//final_transcript = upperfirst(final_transcript);
		alert(final_transcript);
		while (final_transcript.charAt(0) === ' ')
			final_transcript = final_transcript.slice(1);

		if (final_transcript.startsWith(prefix)) {
			serviceCall2(final_transcript.replace(prefix,""));
		} /*else {
			alert("prefix em falta");
		}*/
		final.innerHTML = linebreak(final_transcript);
		nao_total.innerHTML = linebreak(interim_transcript);
		final_transcript = '';
```

```
        };
}

var two_line = /\n\n/g;
var one_line = /\n/g;
function linebreak(s) {
        return s.replace(two_line, '<p></p>').replace(one_line, '<br>');
}
var first_char = /\S/;
function upperfirst(s) {
        return s.replace(first_char, function (m) { return m.toUpperCase(); });
}

function startButton(event) {
        if (recognizing) {
                recognition.stop();
                return;
        }
        final_transcript = '';
        recognition.lang = 'pt-PT';
        recognition.start();
        ignore_onend = false;
        final.innerHTML = '';
        nao_total.innerHTML = '';
        showInfo('info_allow');
        start_timestamp = event.timeStamp;
}
function showInfo(s) {
        if (s) {
                for (var child = info.firstChild; child; child = child.nextSibling) {
                        if (child.style) {
                                child.style.display = child.id == s ? 'inline' : 'none';
                        }
                }
                info.style.visibility = 'visible';
        } else {
                info.style.visibility = 'hidden';
        }
}

function serviceCall2() {
        if (arguments[0].length > 0) {
                $.ajax({
                        type: "POST",
                        url: 'WebService1.asmx/parse_string',
```

```
                        data: '{ frase: \" + arguments[0] + '\', contexto: \" +
window.global_contexto + '\'}',
                        contentType: "application/json; charset=utf-8",
                        dataType: "json",
                        success: function (msg) {

                                if (msg.d != "false") {
                                        window.global_contexto = msg.d;
                                        call_action(window.global_contexto);

                                } else {
                                        alert("Frase não reconhecida.");
                                }
                                $("#divResult").html(msg.d);
                        },
                        error: function (e) {
                                $("#divResult").html("WebSerivce unreachable");
                        }
                });

        }
}

function call_action() {
        var global_contexto;
        obj = JSON.parse(arguments[0]);
        if (arguments[0].length > 0) {
                $.ajax({
                        type: "POST",
                        url: obj.accao,
                        data: '{ contexto: \" + arguments[0] + '\'}',
                        contentType: "application/json; charset=utf-8",
                        dataType: "json",
                        success: function (msg) {
                                $("#divResult1").html(msg.d);
                        },
                        error: function (e) {
                                $("#divResult").html("WebSerivce unreachable");
                        }
                });
        }
}
```