CURSO DE CIÊNCIA DA COMPUTAÇÃO

João Luís Farinon

# ANÁLISE E CLASSIFICAÇÃO DE CONTEÚDO TEXTUAL

Santa Cruz do Sul
2015

**ANÁLISE E CLASSIFICAÇÃO DE CONTEÚDO TEXTUAL**

Trabalho de Conclusão apresentado ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul, para obtenção parcial do título de Bacharel em Ciência da Computação.

Orientadora: Prof.ª Daniela Bagatini.

Santa Cruz do Sul
2015

## AGRADECIMENTOS

Agradeço de forma especial todas as pessoas que me ajudaram a desenvolver este trabalho de conclusão, dentre elas meu pai, minha mãe, minhas duas irmãs, minha vó, minha namorada, minha orientadora, meus amigos, colegas de aula e de trabalho.

Por fim, agradeço também a todos que não foram citados.

## RESUMO

Desde meados dos anos 50 vêm sendo desenvolvidas técnicas para tornar o mais natural possível a comunicação entre seres humanos e computadores. Uma das áreas responsáveis por estudar este tipo de comunicação é denominada de Processamento da Linguagem Natural (PLN). Para este trabalho de conclusão de curso, foi feito um estudo da área de PLN baseado em autores como Stuart J. Russell, Peter Norvig, Nils John Nilsson, James F. Allen, entre outros. O presente trabalho utilizou recursos de PLN para analisar textos, dando ênfase na análise léxica em conjunto com abordagens estatísticas. Com base nesta premissa, foi desenvolvido um processador textual tendo como principal função classificar, através de cálculos e contagens, as palavras mais relevantes de textos da língua portuguesa. Tal solução integra os comportamentos do agente inteligente da aplicação MaRE (Mapeamento de Rota de Estudo), que é responsável por identificar conteúdos que sejam relevantes ao objetivo de uma pesquisa web. Portanto, com este processador foi possível identificar que a solução estatística para PLN é capaz de processar satisfatoriamente as palavras que aparecem com maior frequência em textos. Também, tal solução, incrementou o comportamento do agente inteligente do MaRE na classificação de conteúdo textual.



**Palavras-chave:** Processamento de Linguagem Natural, conteúdo relevante na web, Agentes Inteligentes.

## ABSTRACT

Since the mid-50s has been developed techniques to make as natural as possible the communication between humans and computers. One of the areas responsible for studying this type of communication is called Natural Language Processing (NLP). For this final paper, a study was made in NLP area based on authors like Stuart J. Russell, Peter Norvig, Nils Nilsson John, James F. Allen, among others. This final paper used NLP resources to analyze texts, emphasizing the lexical analysis together with statistical approaches. Based on this premise, it developed a processor of texts with the main function sort, through calculations and counts, the most important words of Portuguese language texts. This solution integrates the behavior of intelligent agent of the MaRE (Study Route Mapping) application, which is responsible for identifying content that is relevant to the purpose of a web search. Therefore, with this processor was possible to identify the statistical solution to PLN is able to successfully process the words that appear more frequently in texts. Also, the solution increased the behavior of MaRE intelligent agent on textual content rating.



**Keywords:** Natural Language Processing, relevant web content, Intelligent Agents.

## LISTA DE FIGURAS

## LISTA DE TABELAS

**SUMÁRIO**

# 1. INTRODUÇÃO

Aplicações computacionais auxiliam na resolução de diversos problemas com o propósito de buscar soluções que possam melhorar a vida das pessoas. Dentre os problemas de pesquisa, pode-se citar o Processamento de Linguagem Natural (PLN).

Segundo Russell e Norvig (2013), PLN trata a possibilidade de humanos se comunicarem com máquinas da forma mais natural possível, sem ser preciso aprender línguas artificiais muito específicas, que são os comandos nos quais manipula-se um computador, por exemplo, visto que muita gente ainda tem dificuldades.

O objetivo deste trabalho de conclusão foi desenvolver um processador de textos da língua portuguesa, utilizando uma solução estatística de PLN, com o propósito de identificar as palavras mais relevantes, que aparecem com mais frequência em um texto. Alcançado tal objetivo, o processador textual foi integrado com a aplicação MaRE (Mapeamento de Rota de Estudo, conforme capítulo 4), com o propósito de incrementar o comportamento do agente inteligente da aplicação.

Alguns dos objetivos específicos para o desenvolvimento do presente trabalho foram:

- Aprofundar estudos sobre PLN e sobre cálculos de relevância de palavras em documentos textuais.
- Conhecer trabalhos relacionados.
- Definir tecnologias para o desenvolvimento da aplicação de PLN.
- Testar resultados e integrar o processador textual desenvolvido com a aplicação MaRE.

A principal motivação deste trabalho foi investigar sobre a abordagem estatística de PLN como forma de indicar entendimento do significado de um texto. Atualmente, as soluções existentes, baseadas apenas no estudo linguístico, para descoberta de significado, são bastante complexas e os resultados ainda estão longe da perfeição, resultando assim em um ótimo cenário para pesquisas. Portanto, uma possível solução para esta complexidade foi partir da extração das palavras de maior relevância pela sua frequência

no texto, uma vez que elas geralmente representam a ideia principal do texto analisado (WIVES, 2002).

Este trabalho está dividido nos seguintes capítulos:

O capítulo 2 apresenta o conceito de linguagem natural, linguística de corpus e PLN, que foi a principal área de estudo para a realização deste trabalho.

No capítulo 3 são apresentados trabalhos que se relacionam com o presente trabalho, no que diz respeito a PLN.

No capítulo 4 é apresentado o aplicativo MaRE, que foi desenvolvido no trabalho de conclusão de curso do aluno Kelvin S. Teixeira, sob título de Mapeamento de Rota de Estudo na Web Baseada na Estratégia de Gamificação, trabalho este que serviu de cenário para teste do presente trabalho.

No capítulo 5 são discutidas as soluções utilizadas no desenvolvimento deste trabalho de conclusão, que é a análise e classificação do conteúdo textual baseado em PLN estatístico.

O capítulo 6 apresenta os testes do sistema desenvolvido, bem como detalhes sobre a integração com o aplicativo MaRE.

O capítulo 7 apresenta a conclusão e os trabalhos futuros.

Para finalizar, o capítulo 8 lista as referências que foram úteis para o desenvolvimento deste trabalho.

# 2. TRATAMENTO DA LINGUAGEM NATURAL

Este capítulo é chamado de tratamento da linguagem natural pois trata da manipulação direta da linguagem natural. É abordado o estudo sobre a linguagem natural, linguística de *corpus* e processamento de linguagem natural, porém é preciso compreender também o que é linguística computacional, para Vieira e Lima (2001 apud Muller, 2011, p. 20), é "a área de conhecimento que explora as relações entre linguística e informática, tornando possível a construção de sistemas com capacidade de reconhecer e produzir informação apresentada em linguagem natural" já, segundo Othero e Menuzzi (2005, p. 22), "a Linguística Computacional pode ser dividida em duas subáreas distintas: a Linguística de *Corpus* e o Processamento de Linguagem Natural".

## 2.1. Linguagem Natural

Para Rich e Knight (1994), a linguagem natural destina-se à comunicação dos seres humanos sobre o mundo, sendo que a maior parte da comunicação linguística ocorre através da fala. A linguagem escrita ainda é muito recente se comparado à fala, e para um computador, mais fácil de ser interpretada.

Segundo Rabuske (1995, p. 117),

> O entendimento da linguagem natural é difícil, pois requer conhecimentos de linguística e do domínio do discurso. Estes conhecimentos, em boa dose, o ser humano adquire naturalmente, desde tenra idade. À medida que a pessoa se desenvolve intelectualmente, acumula mais e mais conhecimentos e experiências que vão enriquecendo a possibilidade de comunicação.

A tabela 1 apresenta os diferentes tipos de abordagens que cada área de pesquisa recebe em relação à linguagem natural.

**Tabela 1 - Diferentes abordagens do estudo da linguagem natural**

| Área de Pesquisa | Problemas típicos | Ferramentas |
|---|---|---|
| Linguistas | Como palavras formam frases e sentenças? O que limita os possíveis | Intuições sobre a consistência e sentido; modelos matemáticos de |

| | | |
|---|---|---|
| | significados de uma sentença? | estrutura (por exemplo, a teoria da linguagem formal, a semântica do modelo teórico). |
| Psicolinguistas | Como as pessoas podem identificar a estrutura das frases? Como os significados das palavras são identificados? Quando é que o entendimento ocorrerá? | Técnicas experimentais com base na medição do desempenho humano; análise estatística das observações. |
| Filósofos | Qual é o significado, e como palavras e frases adquirem-no? Como as palavras identificam objetos no mundo? | Argumentação utilizando linguagem natural usando a intuição sobre contra exemplos; modelos matemáticos (por exemplo, modelo de lógica matemática). |
| Linguista Computacional | Como identificar a estrutura das frases? Como modelar o conhecimento e o raciocínio? Como utilizar a linguagem para realizar tarefas específicas? | Algoritmos, estruturas de dados, modelos formais de representação e de raciocínio; técnicas de IA (métodos de pesquisa e representação). |

**Fonte: ALLEN (1995 apud MULLER, 2011)**

A linguagem natural tem o propósito de possibilitar a comunicação entre as pessoas. No entanto, percebe-se na tabela 1, que a manipulação da linguagem natural, vantajosamente, pode ser utilizada no auxílio a outras áreas, como na computação, por exemplo: na recuperação de informações, sistemas de banco de dados, sistemas especialistas, documentação automatizada, entre outros.

Os estudos na área de linguística computacional permitem ao homem manter sua forma de expressão mais antiga, sua própria língua, sem necessitar de conhecimentos adicionais, ao contrário, grandes esforços são concentrados para tornar esta comunicação (homem-computador) cada vez mais recíproca.

Porém processar a linguagem natural de forma irrestrita, sem condicionamento a um vocabulário, a uma gramática simplificada ou a um contexto, qualquer que seja esta

linguagem, pode ser um problema. Assim, também é importante falar das gramáticas, são elas que fundamentam e regulam o uso correto das linguagens faladas e escritas, que especificam as sentenças que são permitidas em uma linguagem. Segundo Rich e Knight (1994, p. 443), "os formalismos das gramáticas fundamentam muitas teorias linguísticas, que por sua vez dão base para muitos sistemas de compreensão da linguagem natural".

Portanto o formalismo das gramáticas (estruturas gramaticais que darão base a sistemas de compreensão em linguagem natural), dicionários compatíveis com as aplicações a explorar, o conhecimento implícito associado aos próprios elementos que irão compor analisadores de textos, além do conhecimento da linguística de *corpus* são componentes de compreensão essenciais quando se trata de linguagem natural.

## 2.2. Linguística de Corpus

A linguística de *corpus* é uma área razoavelmente nova. Tagnin (2008) apresenta um breve histórico, afirmando que a linguística de *corpus* vem sendo desenvolvida desde os anos 80, começando as primeiras pesquisas na Europa, e mais tarde em outras partes do mundo, inclusive no Brasil.

Para Russell e Norvig (2013, p. 807),

> Um *corpus* é uma grande coleção de texto, como os bilhões de páginas que constituem a World Wide Web. O texto é escrito por seres humanos e para seres humanos, e a tarefa do software é facilitar a localização das informações corretas pelos seres humanos. Essa abordagem implica o uso de estatística e aprendizagem para tirar proveito do corpus, e em geral impõe modelos probabilísticos de linguagem que podem ser aprendidos a partir de dados.

Segundo Othero e Menuzzi (2005), a linguística de *corpus* preocupa-se mais com a corpora (plural de corpus) digital, que nada mais é do que grandes locais de armazenamento (banco de dados) que contenham amostras de linguagem natural. Estas amostras não necessariamente precisam ser somente escritas, mas podem ser de forma falada também.

Os trabalhos que envolvem linguística de *corpus* nem sempre são voltados para a construção de softwares (em contraste com PLN). Em sua grande maioria, estes trabalhos são voltados para o estudo de fenômenos linguísticos envolvendo amostras de

determinadas linguagens naturais (OTHERO, MENUZZI, 2005). Para Berber Sardinha (2000 apud Othero e Menuzzi, 2005, p. 23),

> […] a linguística de corpus se ocupa da coleta e explicação de corpora, ou conjuntos de dados linguísticos textuais que forma coletados criteriosamente com o propósito de servirem para a pesquisa de uma língua ou variedade linguística. Como tal, dedica-se à explicação da linguagem através de evidências empíricas, extraídas por meio de computador.

Resumidamente, o corpus pode então ser definido como sendo uma grande coleção de textos, como por exemplo as inúmeras páginas dos sites que constituem a internet, e serão estas páginas uma das fontes de informações de testes para realizar o PLN no presente trabalho.

## 2.3. Processamento de Linguagem Natural (PLN)

A seguir é apresentado a um breve histórico e definição de Processamento de Linguagem Natural, suas propriedades, áreas que fazem o uso de PLN e tipos de abordagens de uso.

### 2.3.1. Breve Histórico

Desde meados dos anos 50, PLN vem sendo estudada e aplicada em casos que é necessário utilizar linguagem natural, como meio de comunicação com computadores. O principal precursor foi Alan Turing, propondo o teste de Turing, que é um teste de inteligência artificial onde uma máquina simula o comportamento humano (SETZER, 2015).

Os avanços nesta área aconteceram também por muitas necessidades, como traduções automáticas de textos, por exemplo. Diante disso, os pesquisadores perceberam que era possível solucionar computacionalmente parte destas necessidades. Para Othero e Menuzzi (2005, p. 25), “A linguística computacional teve grande impulso graças a esforços para o desenvolvimento de programas de tradução automática nas décadas de 1950 e 1960, e seu surgimento está ligado a desenvolvimentos na área da inteligência artificial”.

Anos mais tarde, é criado pelo pesquisador Joseph Weinzenbaum, em 1966, um dos primeiros chatterbots (programas que interagem com seres humanos através de diálogos em linguagem natural) (OTHERO, MENUZZI, 2005). O chatterbot em questão foi batizado de ELIZA e sua função era simular um diálogo entre um paciente e um psicólogo. ELIZA utilizava técnicas de PLN e o curioso é que na época muitas pessoas achavam que estavam conversando com uma pessoa de verdade.

A figura 1 mostra um trecho de uma conversa entre ELIZA e um usuário humano.

**Usuária:** Os homens são todos iguais.

**ELIZA:** EM QUE SENTIDO

**Usuária:** Eles estão sempre nos importunando por algum motivo ou outro.

**ELIZA:** PODE PENSAR EM UM EXEMPLO ESPECÍFICO

**Usuária:** Ele diz que eu fico muito tempo deprimida.

**ELIZA:** FICO TRISTE EM SABER QUE VOCÊ FICA DEPRIMIDA.

**Usuária:** É verdade. Eu sou infeliz.

**ELIZA:** VOCÊ ACHA QUE VIR AQUI VAI AJUDAR VOCÊ A NÃO SER INFELIZ

**Usuária:** Eu preciso de ajuda, acho que isso é certo.

**ELIZA:** O QUE SIGNIFICARIA PARA VOCÊ SE VOCÊ CONSEGUISSE AJUDA

**Usuária:** Talvez eu pudesse aprender a conviver com a minha mãe.

**ELIZA:** FALE-ME MAIS SOBRE A SUA FAMÍLIA

**Usuária:** Minha mãe cuida de mim.

**ELIZA:** QUEM MAIS EM SUA FAMÍLIA CUIDA DE VOCÊ

**Usuária:** Meu pai.

**ELIZA:** SEU PAI

**Usuária:** De certa forma, você é como meu pai.

**ELIZA:** QUE SEMELHANÇA VOCÊ VÊ

**Usuária:** Você não é muito agressiva, mas acho que você não quer que eu perceba isso.

**Figura 1- Conversa entre ELIZA e um ser humano**

**Fonte: Adaptado de OTHERO, MENUZZI, 2005**

Mesmo sendo um programa desenvolvido na década de 1960, o realismo da conversa é muito próximo a um diálogo entre humanos. O programa foi considerado um avanço na computação, e a partir daquela época as pesquisas na área aumentaram de forma significativa (MITTMANN, 2015).

### 2.3.2. Definição

Com base em Rich e Knight (1994), PLN pode ser entendido como aplicações computacionais que compreendem linguagens naturais (compreensão e geração de textos), tanto na fonética quando na escrita, sendo que a última é mais fácil de um computador conseguir interpretar de maneira correta, pois não tem que lidar com problemas do sinal do áudio, como o ruído por exemplo.

Para Vieira e Lopes (2010, p. 184), "Processamento de Linguagem Natural (PLN) é uma área de Ciência da Computação que estuda o desenvolvimento de programas de computador que analisam, reconhecem e/ou geram textos em linguagens humanas, ou linguagens naturais".

Já segundo Kao (2004 apud Muller, 2011, p. 22), "PLN é a tentativa de extrair uma representação com significado mais amplo, a partir de um texto livre. Por exemplo, descobrindo quem fez o quê a quem, quando, onde, como e por que".

Portanto, PLN diz respeito à construção de softwares que são capazes de processar (compreender) informações em linguagem natural, conseguindo obter algum significado do texto que por ele é interpretado. Por exemplo, se perguntamos para o computador: "Que horas são?" ou "Que horas são agora?" ou ainda "Você sabe que horas são?", o seu desafio seria entender estas perguntas sintaticamente diferentes, mas com o mesmo significado (semântica), passar pelos processos de análise, saber interpretar que é uma solicitação para que as horas sejam informadas, e retornar a hora para o usuário.

Outro exemplo é a aplicação de PLN em consultas de bancos de dados, onde uma pessoa leiga, que desconhece de comandos SQL, faria a seguinte pergunta para o computador: "Quais são os nomes dos alunos ativos nesta universidade". O computador

então realiza, para este caso, um processo de análise do que foi informado em linguagem natural, executando o comando "SELECT nome FROM aluno WHERE ativo = 'S'" e retornado os registros dos alunos.

A figura 2 mostra uma ideia bem geral de como são as etapas e um exemplo de uso de PLN, onde tem-se entradas por textos ou falas que são processadas utilizando PLN, gerando um resultado (estruturas) que pode, ou não, ser armazenado em um banco de dados para posteriormente ser utilizado em aplicações.

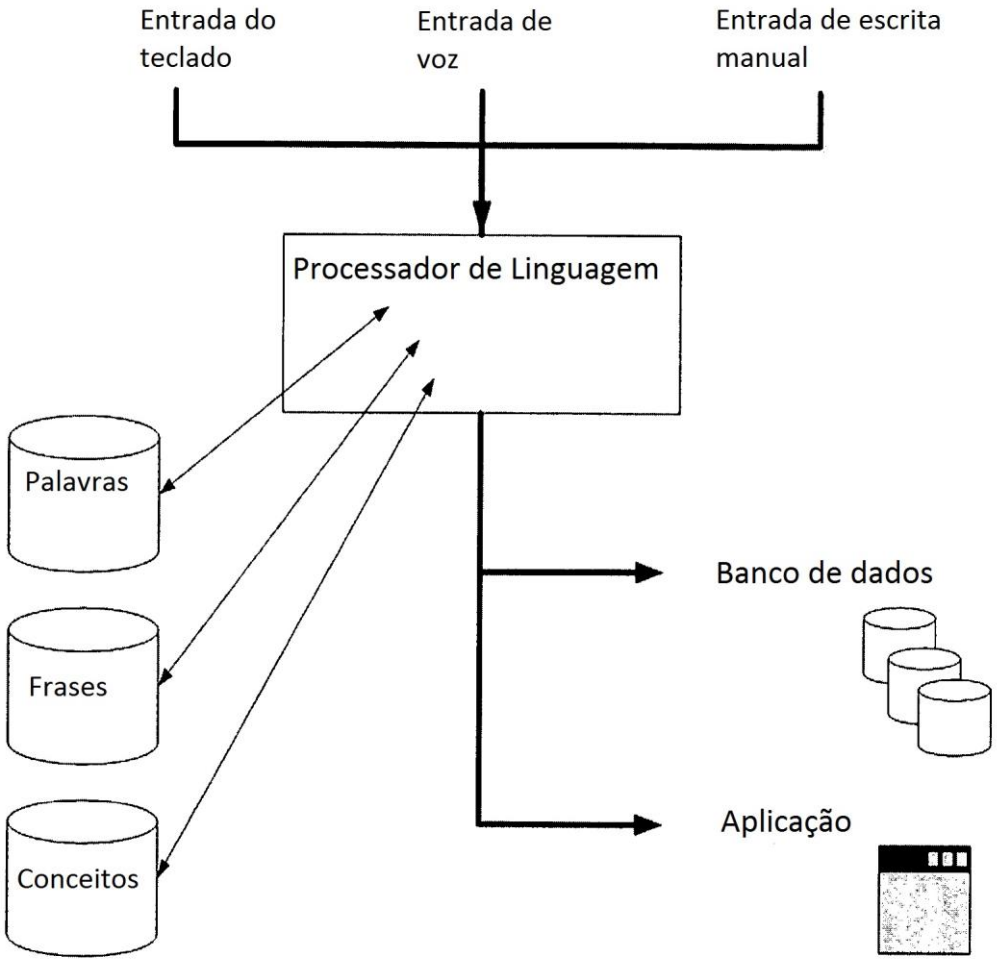


**Figura 2- Etapas de PLN, de forma geral**

**Fonte: https://lbsitbytes2010.wordpress.com/2013/03/27/scientist1-makoto-nagao-roll-no6/**

PLN geralmente faz uso de conceitos linguísticos, como partes do discurso (substantivo, verbo, adjetivo, entre outros), e estruturas gramaticais. Também, em alguns casos, é necessário lidar com anáfora (repetição da mesma palavra em frases ou versos

consecutivos) e ambiguidades. Para fazer isso, são utilizadas diversas representações de conhecimento, tais como um léxico de palavras, seus significados, propriedades e regras gramaticais, bem como outros recursos como ontologias (estudo do ser, da realidade, da existência) de entidades e ações, ou um dicionário de sinônimos e abreviaturas (KAO, 2004 apud MULLER, 2011).

### 2.3.3. Etapas de PLN

Alguns módulos componentes de um sistema para o processo de compreensão de textos escritos são: analisadores léxicos-morfológicos (lê e realiza o tratamento a nível de palavras), sintáticos (produz um conjunto de estruturas sintáticas relacionando os termos de uma oração) e semânticos (cria uma representação final do significado de uma oração).

A seguir, apresenta-se as etapas de PLN. Embora uma ferramenta de PLN não necessariamente precisa seguir (implementar) todas estas etapas, principalmente quando é decidido por implementar um sistema bem específico de uma determinada língua (RICH, KNIGHT, 1994). As etapas podem ser divididas da seguinte maneira:

- **Análise Léxica**

Segundo Gonzalez e Lima (2003, p. 30),

> Na etapa da análise léxica, ocorre a conversão de uma cadeia de caracteres (o texto da consulta) em uma cadeia de palavras. Assim, o principal objetivo desta etapa é a identificação das palavras que constituem a consulta. Na fase seguinte, artigos, preposições e conjunções são candidatos naturais à lista de *stopwords*, ou seja, a serem eliminados. Então, pode ser executada a normalização lexical através de *stemming*, pois frequentemente o usuário especifica uma palavra na consulta, mas somente variações desta palavra estão presentes em um documento relevante.

Resumindo, uma das funções da análise léxica é *tokenização*, que é uma técnica para decompor um texto em palavras, e estas palavras são denominadas de tokens. Uma vez que os tokens foram definidos, o sistema de PLN, se desejado, pode avançar para a próxima etapa, a análise morfológica.

- **Análise Morfológica**

Nesta etapa, a análise da forma das palavras mostra a existência de vários elementos, que lhe constituem e estruturam. São exemplos de elementos mórficos: raiz, radical e terminações. As palavras isoladas são analisadas pelos seus componentes. Por exemplo: Se informar o texto “Abrir o arquivo teste.txt”, a análise morfológica tem que saber que teste.txt é um arquivo, e que .txt é a extensão deste arquivo (RICH, KNIGHT, 1994).

Assim, a análise morfológica é responsável pela análise de palavras isoladas, onde cada palavra é classificada em uma categoria de acordo com as regras que regem a língua portuguesa (substantivo, adjetivo, numeral, verbo, advérbio, pronome, artigo, preposição, conjunção e interjeição). Por exemplo: caracteres de entrada: cliente, identifica a entrada sendo um substantivo.

- **Análise Sintática**

De modo geral, diz respeito à estrutura das palavras em uma frase. Os resultados obtidos na etapa da análise morfológica servem de entrada para a análise sintática. Estes resultados servem para criar uma descrição estruturada (RICH, KNIGHT, 1994). Por exemplo, se informar a frase “Menino o vai loja à”, o analisador sintático rejeitaria ela. O primeiro erro nesta frase, está em “Menino o”. O substantivo “Menino” e o artigo definido “o” estão fora de ordem, pois, segundo a gramática da língua portuguesa, um artigo deve antepor um substantivo. O segundo erro é da mesma natureza do primeiro, e está nas palavras “loja à”.

Por exemplo, ao analisar sintaticamente a frase “O João ama a Maria”, tem-se como resultado uma descrição bem estruturada desta frase, destacando as suas unidades, como artigo, sujeito, verbo, entre outros. Estas unidades serão importantes quando for realizada a análise semântica (RICH, KNIGHT, 1994). A figura 3 apresenta o resultado de uma análise sintática.

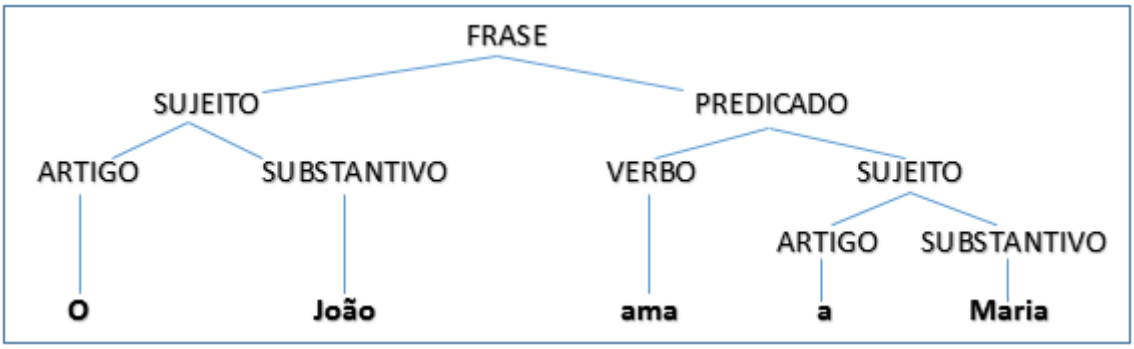


**Figura 3- Resultado de uma análise sintática**

**Fonte: Adaptado de OTHERO, MENUZZI, 2005**

- **Análise Semântica**

De maneira bem geral, análise semântica é dar significado às estruturas criadas na análise sintática. Analisa o significado das palavras em uma frase. Por exemplo, a frase "O pensamento azul ficou molhado" não tem um significado muito válido, e seria rejeitada pelo analisador semântico.

Para Rezende (2003, p. 342),

> A análise semântica dos textos é feita para tentar identificar a importância das palavras dentro da estrutura da oração. Quando se utiliza um único texto, algumas funções podem ser identificadas e pela função identifica um grau de importância. Por exemplo, um nome tem grande chance de representar informação relevante na estrutura da sentença. Para tarefas como categorização, o interessante seria analisar um documento comparando-o a Bases de Conhecimento de diferentes assuntos, para descobrir a qual categoria ele pertence. Um conjunto bem-selecionado de textos sobre um assunto compõe uma base de Conhecimento.

Para Rich e Knight (1994), a linguagem final da representação do significado é chamado de *linguagem-objeto*. Esta linguagem-objeto é considerada a principal finalidade do processamento semântico, pois inclui tanto a estrutura representacional quando o vocabulário de significado específico. Mas também há outro ponto que é muito importante falar, que é o Processamento Léxico.

A primeira etapa de qualquer sistema de processamento semântico é procurar palavras em um dicionário (ou léxico) e extrair seu significado. Como existem palavras com vários significados, pode não ser possível extrair o significado correto olhando apenas para a palavra isolada. Por exemplo, a palavra "dente" pode ter muitos significados, pode ser dente-de-alho, dente de uma engrenagem ou dente da boca de uma pessoa. Neste caso é

preciso analisar a sentença em que a palavra está contida. Se a frase for “O dente doía muito”, um processamento correto deveria saber que dente-de-alho e de engrenagem não dói, mas de pessoas sim. Como pode-se observar, a ambiguidade acontece em muitas palavras das linguagens naturais. Para amenizar este problema, e facilitar a vida de um analisador léxico, é preciso colocar marcadores semânticos cada vez mais detalhados, como na frase anterior, sobre o dente dolorido. Faz-se necessário marcar um sentido de dente como parte de animal (para seres humanos), e outro como parte de não animal (para dente-de-alhos e de engrenagens) (RICH, KNIGHT, 1994). Somente dessa forma um processamento léxico faria a distinção correta. Este processo de marcação de sentido nas palavras é conhecido formalmente como *Desambiguação Léxica.*

Por fim, existem as *Gramáticas Semânticas*, que são construídas ao final da análise semântica. Pode-se observar na figura 4 o resultado de uma análise feita a partir da frase “Eu quero imprimir o arquivo .init do Mário”. Tem-se uma gramática com marcações e significados bem definidos, para cada parte que a compõe (RICH, KNIGHT, 1994).

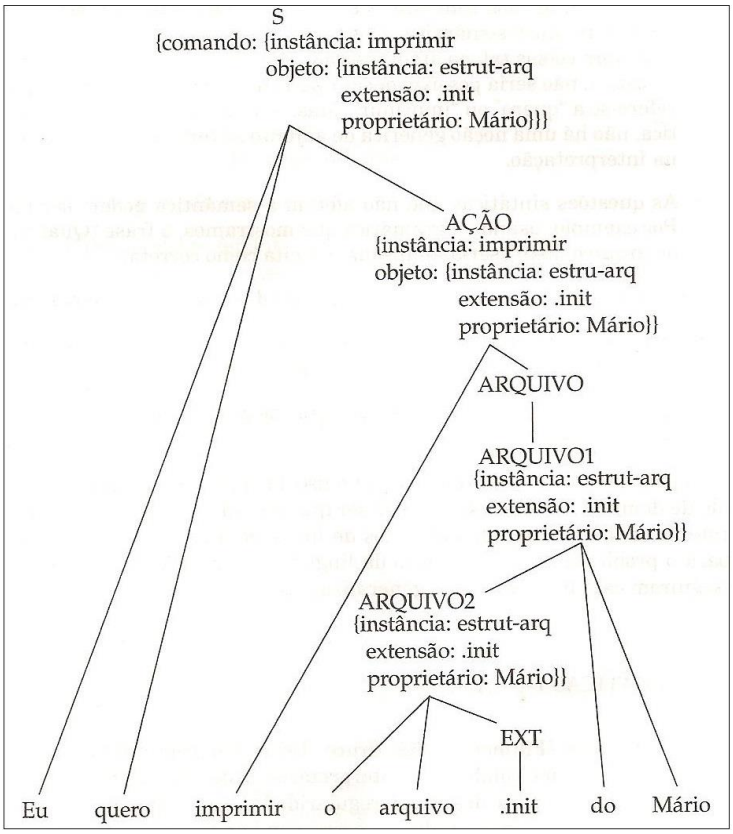


**Figura 4- O resultado da análise com uma gramática semântica**

**Fonte: RICH, KNIGHT, 1994**

É importante frisar que não é necessário criar estas gramáticas, mas quando usadas, há vantagens para PLN. Para Rich e Knight (1994), as principais vantagens são que com uma gramática já construída, é possível pegar a sua estrutura e usar diretamente na aplicação de PLN. Outra vantagem bastante evidente é com relação às ambiguidades léxicas que são evitadas, pois a gramática já deixa bem definido o significado de cada palavra.

- **Integração do Discurso e Análise Pragmática**

Na integração do discurso, o significado de uma frase isolada muitas vezes depende do significado da frase anterior, e pode dar um significado para frase posterior. Por exemplo, a frase isolada "Eu estou muito cansado" dá a ideia de uma pessoa fisicamente cansada. Mas se em seguida acrescentar a frase "Nós estamos sempre discutindo!", nos dá a ideia de uma pessoa emocionalmente cansada (RICH, KNIGHT, 1994).

A análise pragmática reavalia a estrutura do que foi dito para determinar o que realmente se quis dizer. Por exemplo, no caso de uma interação homem-máquina, onde o ser humano informa para máquina que ela deve abrir um arquivo com máxima velocidade. A máquina então tem que saber que não basta só abrir o arquivo, mas que é preciso também diminuir o processamento de outras tarefas para atender o usuário de forma mais satisfatória (RICH, KNIGHT, 1994).

Após realizar o processo de PLN, é possível avaliar a eficácia do seu uso através de medições. Esta etapa é conhecida como Pós-Processamento dos dados. Para Rezende (2003, p. 362), "O pós-processamento dos dados consiste da fase da validação das descobertas efetuadas pela etapa de processamento dos dados e da visualização dos resultados encontrados". Tem-se duas formas bem básicas de métricas de avaliação dos resultados obtidos, *Precisão* e *Recall*. Estas técnicas foram adotadas da área de Recuperação da Informação (RI) e se baseiam em noções de relevância, por exemplo, se um documento atender a necessidade de informação do usuário, então ele é considerado relevante à solicitação do usuário. A seguir apresenta-se as métricas de avaliação. Rezende (2003) caracteriza estas duas técnicas como:

- **Precisão**

Esta técnica de precisão avalia o quanto o modelo acerta. Sua fórmula é a seguinte: (*número de itens relevantes recuperados / número total de itens recuperados*).

- **Recall**

Por outro lado, a técnica de Recall avalia o quanto o modelo contabiliza. Sua fórmula é definida da seguinte maneira: (*número de itens relevantes recuperados / número de itens relevantes na coleção*).

### 2.3.4. Tipos de Abordagens

Há diferentes tipos de abordagens em PLN, sendo as mais comuns a abordagem semântica, baseada nas funcionalidades de cada palavra em um texto, e a estatística, baseada em frequência (MULLER, 2011). Para Gonzalez e Lima (2003, p. 41), "Os dois grandes recursos de que dispõe o PLN são o conhecimento linguístico e a estatística". Ainda segundo Gonzalez e Lima (2003, p. 11), "o PLN pode se apoiar no conhecimento linguístico e em métodos estatísticos, não necessariamente de forma excludente".

- **Abordagem Semântica**

Este tipo de abordagem emprega técnicas que avaliam a sequência dos termos no contexto da frase, para a correta identificação da função de cada termo. A análise semântica utiliza fundamentos e técnicas baseadas no processamento de linguagem natural (MULLER, 2011).

Segundo os autores Dong e Ebecken, Lopes, Costa (2008, 2003 apud Muller, 2011), o entendimento semântico da linguagem natural geralmente necessita duas competências:

- Análise dos elementos que constituem a linguagem;
- Conhecimento da finalidade dos componentes constituintes, o que requer:

- Conhecimento morfológico: conhecimento da estrutura, da forma e das inflexões das palavras.

- Conhecimento sintático: conhecimento estrutural das listas de palavras e como as palavras podem ser combinadas para produzir sentenças.

- Conhecimento semântico: o que as palavras significam independentes do contexto, e como significados mais complexos são formados pela combinação de palavras.

- Conhecimento pragmático: o conhecimento do uso da língua em diferentes contextos, e como o significado e a interpretação são afetados pelo contexto.

- Conhecimento do discurso: como as sentenças imediatamente precedentes afetam a interpretação da próxima sentença.

- Conhecimento do mundo: conhecimento geral do domínio ou o mundo que a comunicação da linguagem natural se relaciona.

Segundo Voorhees e Allen (1999, 2000 apud Gonzalez e Lima, 2003), abordagens puramente linguísticas também encontram dificuldades pois o conhecimento linguístico é muitas vezes aplicável em domínios muito específicos, não sendo portável, e as técnicas linguísticas (como etiquetagem de categorias gramaticais, resolução de ambiguidade, análise sintática, outras) são bastante complexas e necessitam ter alto grau de precisão para trazer benefícios.

- **Abordagem Estatística**

Segundo Dong (2008 apud Muller 2011, p. 26), “Em contraste à análise semântica, o processamento estatístico (PLN estatístico), faz uso bastante limitado de gramáticas e a priori, do conhecimento semântico. Na verdade, PLN estatístico tenta aprender modelos de linguagens a partir de exemplos em corpus de linguagem natural usando o formalismo da teoria da probabilidade”.

Bod (1995 apud Muller 2011) afirma que a utilização das abordagens estatísticas em geral, para PLN, apontam caminhos promissores para o entendimento do significado, uma vez que geram modelos matematicamente precisos para frequências de ocorrência e por permitirem suposições valiosas em casos de incertezas.

Para Gasperin e Lima (2001, p. 19), “aquelas palavras e expressões que ocorrem relativamente mais vezes no texto de assunto específico são provavelmente parte do vocabulário específico do domínio deste texto”.

Segundo Wives (2002), “Existe uma série de fórmulas desenvolvidas ou aplicadas com o intuito de calcular a importância de uma palavra baseando-se em sua frequência. Essa importância costuma ser chamada de peso e indica o grau de relação entre a palavra e os documentos em que ela aparece”. Ainda segundo ele, há três fórmulas que são simples e aplicáveis a praticamente todo o tipo de aplicação, a frequência absoluta, a relativa, e a inversa:

- A frequência absoluta é a mais simples e a menos precisa, pois apenas conta a quantidade de vezes que as palavras são apresentadas em um texto.

- A frequência inversa é obtida através do cálculo TF-IDF, onde TF (Term Frequency) representa a frequência absoluta que a palavra aparece em determinado documento, e IDF (Inverse Document Frequency) é a escala de importância do termo em um conjunto de documentos (MULLER, 2011). Este cálculo avalia a importância de uma palavra para um documento com base em um conjunto de documentos pré-cadastrados.

- A frequência relativa se obtém com a contagem da frequência absoluta de cada palavra, dividido pelo total de palavras do texto analisado, podendo ainda ser multiplicado por 100 para se ter uma representação em percentual. A sua fórmula é:

$$FR = (QTD\_PALAVRA / QTD\_TOTAL) * 100$$

  Onde: QTD_PALAVRA representa a frequência absoluta em que a palavra aparece no texto, e QTD_TOTAL é o total de palavras de todo o texto.

### 2.3.5. Utilidades de PLN

Algumas áreas em que o uso de PLN se torna útil são:

- **Recuperação de Informação (RI)**

Segundo Frantz (1997 apud Gonzalez e Lima, 2003, p. 17), "Um sistema de RI tem como meta encontrar a informação exigida para satisfazer a necessidade de informação (NI) do usuário". RI também deve armazenar a informação. Para Strzalkowski (1999 apud Gonzalez e Lima, 2003), além do procedimento de buscas, RI pode também incluir catalogação, categorização e classificação de informação na forma textual. O uso de PLN pode melhorar os resultados através da interpretação do texto obtido.

- **Web Mining**

Baseado em Rezende (2003), é uma metodologia de recuperação da informação que utiliza ferramentas de mineração de dados para extrair informações. Web Mining pode ser dividida em três abordagens diferentes, Web Mining de conteúdo (analisa textos, imagens e outros conteúdos presentes nos documentos HTML), de estrutura (estuda o relacionamento das páginas através dos hiperlinks) e de uso (análise dos registros de navegação dos usuários).

- **Tradução automática**

Consiste em programas que fazem uso de PLN para traduzir um texto em linguagem natural de um idioma para outro. Ainda nos dias de hoje, a tradução automática é considerada um grande desafio, pois há muitas diferenças entre as características das linguagens entre os mais diversos idiomas, como as gramaticas, sintaxe, semântica, entre outros. Toda essa diferença acaba por dificultar o trabalho dos tradutores (OTHERO, MENUZZI, 2005). Mesmo assim, há bons tradutores, como o *Google Tradutor,* que com a ajuda da interpretação humana, é possível chegar a bons resultados.

- **Resumo de textos**

É possível resumir um texto de forma automática utilizando PLN. Para isso, são identificadas as partes mais importantes em um texto, e usadas no resultado final. Porém,

tem um bom resultado somente em textos que são bem estruturados, como artigos, documentos científicos, entre outros (BATISTA, 2010).

- **Correção de textos**

PLN pode ser utilizado para fazer correção gramatical e ortográfica de textos. São utilizados conceitos de linguística, estatística e a utilização de corpus (BATISTA, 2010). Um exemplo de software que faz correção de textos é o *Microsoft Word*.

## 2.4. Considerações

Este capítulo discutiu sobre linguagem natural, linguística de corpus e processamento de linguagem natural, que juntas podem formar a base para a construção de sistemas que utilizam PLN.

Uma ferramenta de PLN precisa ter amplo conhecimento da linguagem natural que irá ser manipulada, e para realizar o processamento da linguagem é preciso ter fontes de informações. Para o presente trabalho foram utilizados conteúdos de textos e de páginas web.

O estudo das etapas de PLN se fez importante para o presente trabalho, no qual foi abordada a etapa de análise léxica fazendo uso da técnica de *tokenização.*

Das abordagens de PLN vistas neste capítulo, o presente trabalho fez uso de uma abordagem estatística da linguagem natural, utilizando a fórmula da frequência relativa. Esta técnica é mais eficiente do que o cálculo da frequência absoluta, pois não contabiliza somente as palavras em separado, e ao mesmo tempo mantém a simplicidade e fácil adaptação em relação ao cálculo da frequência inversa, já que não necessita ter uma coleção de textos pré-cadastrados em uma base de dados, como é o caso do cenário de aplicação deste trabalho de conclusão.

# 3. TRABALHOS RELACIONADOS

Neste capítulo serão apresentados alguns trabalhos relacionados. O estudo destes trabalhos serviu de fonte de inspiração e contribuiu para a realização do presente trabalho, já que todos eles fazem uso de PLN.

## 3.1. Evi (Evi, 2012)

Evi é um software de busca do tipo pergunta e resposta, disponível somente para língua inglesa. Ele analisa semanticamente a pergunta feita, e apresenta a resposta em linguagem natural. Para processar tal resposta, o sistema analisa as principais informações da pergunta e busca em separado o significado de cada parte, conforme sua base de dados de conhecimento, para no final responder de forma coerente à pergunta do usuário (EVI, 2015).

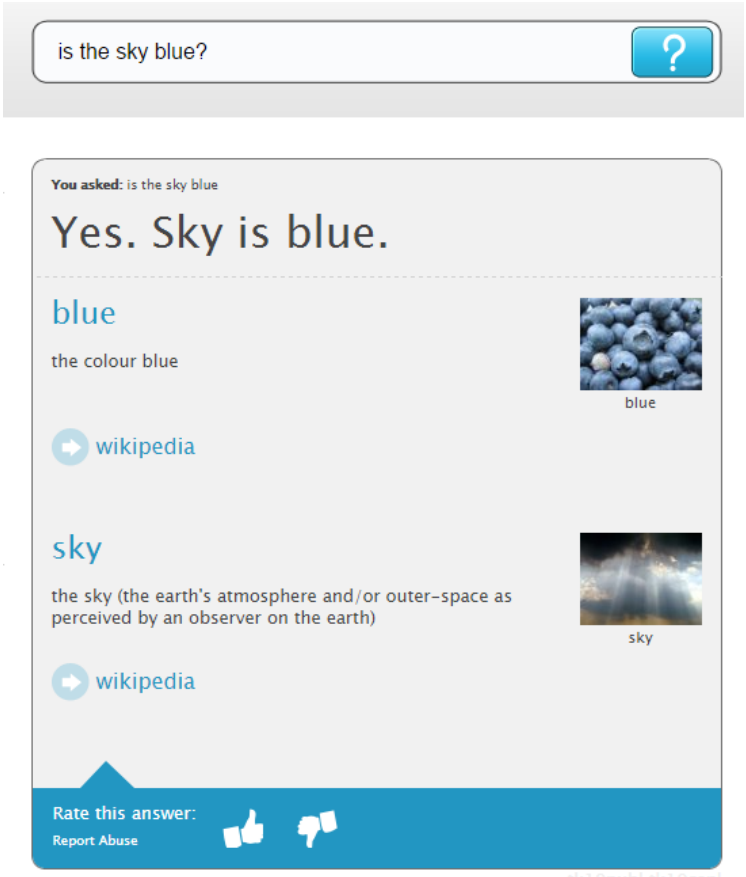


**Figura 5- Exemplo de pergunta e resposta do aplicativo Evi**

**Fonte: https://www.evi.com/q/is_the_sky_blue**

A figura 5 mostra um exemplo simples de pergunta e resposta em linguagem natural. Foi feita a pergunta “O céu é azul?” e o aplicativo respondeu “Sim. O céu é azul”, retornando ainda breves explicações dos termos de maior relevância, como azul e céu. Também é possível realizar perguntas do tipo “Como está o tempo agora em Nova York”, e serão retornados para o usuário informações em tempo real da temperatura e do clima da cidade de Nova York.

O interessante é que é possível fazer qualquer tipo de pergunta, e caso o aplicativo não souber a resposta, o próprio usuário pode dar a sua colaboração, adicionando conhecimento à base de dados do Evi através de formulários em que é selecionado, entre alguns campos, o assunto da pergunta, por exemplo. Também é possível avaliar as respostas obtidas, aprimorando assim a qualidade de busca do software (EVI, 2015).

Inicialmente o aplicativo foi batizado de *True Knowledge*. Foi desenvolvido em 2005 e lançado em 2007 na internet, pela empresa inglesa de mesmo nome. Em 2011 foi iniciado o desenvolvimento do aplicativo EVI, baseado no motor de busca do True Knowledge. Em 2012 a empresa mudou o seu nome também para EVI, mostrando que seu foco era somente no desenvolvimento deste aplicativo. Ainda em 2012, foi vendida para a empresa *Amazon*.

Atualmente o software está disponível gratuitamente em versão web browser, disponível no endereço https://www.evi.com, ou na forma de aplicativo para smartphones. Nos smartphones a experiência é mais rica, pois possui um sintetizador de voz, permitindo ao usuário fazer perguntas através da fala, recebendo respostas também faladas.

### 3.2. CoGroo (Universidade de São Paulo, 2006)

Desenvolvido em 2006 por alunos da Universidade de São Paulo (USP), é um corretor gramatical de textos gratuito, disponível na forma de extensão (API Java), acoplável ao *libreOffice* ou em outros projetos. Atualmente está na versão 4.0.0. Realiza análise morfológica, sintática e regras gramaticais para detectar erros em textos (BATISTA, 2010). Por ser um software de código livre, é possível colaborar com o projeto através do endereço http://comunidade.cogroo.org/.

O CoGroo detecta erros nas relações entre as palavras, por exemplo, na sentença “Os menino estudam demais”, o corretor gramatical irá detectar erro de concordância entre o artigo “os” e o substantivo “menino”. Também sugere uma medida corretiva, como a substituição da palavra “menino” pela mesma palavra flexionada no plural, que é “meninos”. A seguir são listados os principais erros gramaticais que o CoGroo corrige (COGROO, 2015):

- Colocação pronominal.
- Concordância pronominal.
- Concordância entre sujeito e verbo.
- Concordância verbal.
- Uso de crase.
- Regência nominal.
- Regência verbal.
- Erros comuns da língua portuguesa escrita.

Para realizar a análise gramatical de uma frase, o sistema basicamente passa pelas seguintes etapas (COGROO, 2015):

- **Separador de sentenças e tokenização**: Separa as sentenças de uma frase, e cada sentença é transformada em tokens (palavras isoladas que compõem uma frase).
- **Etiquetador morfológico**: Cada token recebe uma etiqueta morfológica (se é um verbo, adjetivo, substantivo, entre outros), baseado em um dicionário de classificação morfológica.).
- **Agrupador**: Os tokens são agrupados conforme as suas classificações morfológicas, com o intuito de encontrar sequências que podem ser tratadas como se fossem um único elemento, como por exemplo “uma casa”.

- **Analisador sintático**: Analisa o agrupamento feito na fase anterior, realizando uma nova classificação (verbo, sujeito, adjetivo, entre outros) de cada parte resultante deste agrupamento, onde ao final é gerada uma árvore sintática.

A figura 6 mostra o resultado de uma análise gramatical para a frase "Ele foi procurar uma casa", onde é possível ver de maneira clara o que foi descrito nos passos anteriores.

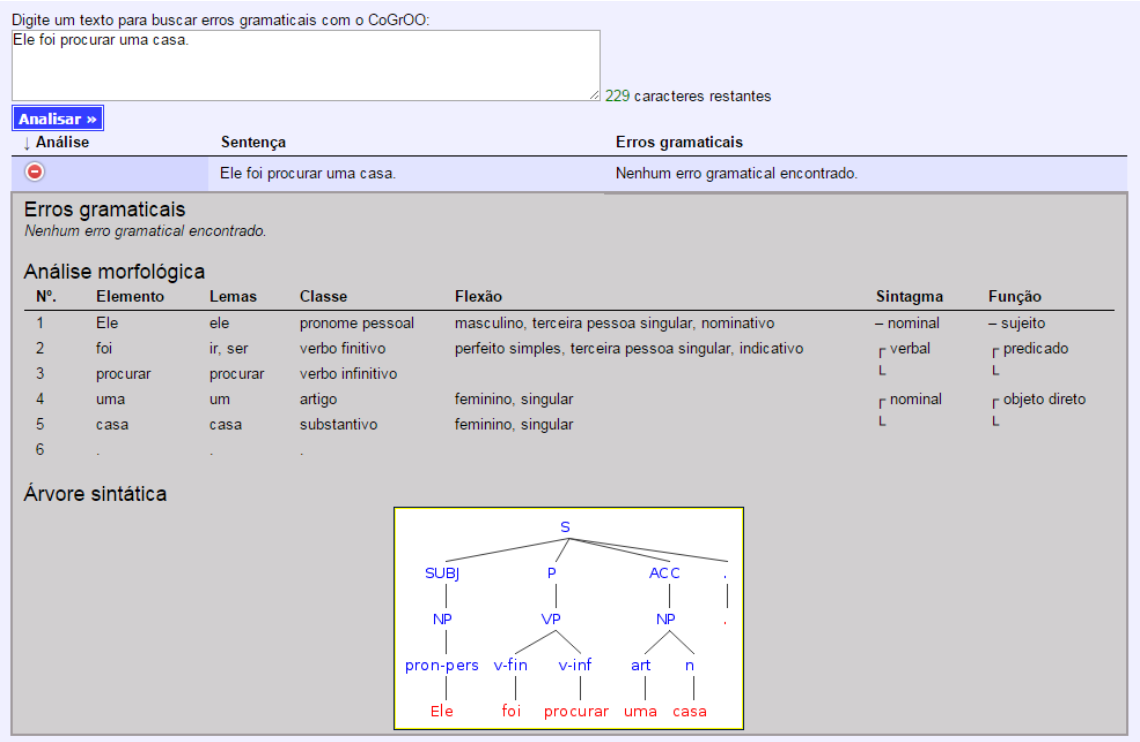


**Figura 6- Resultado de uma análise gramatical**

**Fonte: http://comunidade.cogroo.org/grammar**

No site do CoGroo Comunidade (http://comunidade.cogroo.org/grammar), é possível realizar análises gramaticais, como a feita na figura 6, com frases de até 255 caracteres. Além de mostrar a análise morfológica e sintática para cada sentença, também são apresentados erros gramaticais, contendo a classificação de cada erro e sugestões para correção.

### 3.3. ALICE (Richard S. Wallace, 1995)

ALICE (Artificial Linguistic Internet Computer Entity) é um chatterbot criado em 1995 nos Estados Unidos para língua inglesa. Utiliza PLN para realizar a conversa em

linguagem natural com o usuário. A sua criação foi fortemente baseada no chatterbot ELIZA de 1966. O programa foi desenvolvido na linguagem de programação JAVA e usa uma espécie de arquivo XML chamado AIML (Artificial Intelligence Markup Language) para especificar as regras da conversa e criar a base de conhecimento, que é o lugar onde ficam armazenadas as regras gramaticais e palavras da língua inglesa. O AIML foi desenvolvido pelo mesmo criador de ALICE, Richard S. Wallace, e desde então este padrão tem sido usado por muitos outros chatterbots (ALICE, 2015).

Na figura 7, é possível ver um exemplo de conversa com ALICE.

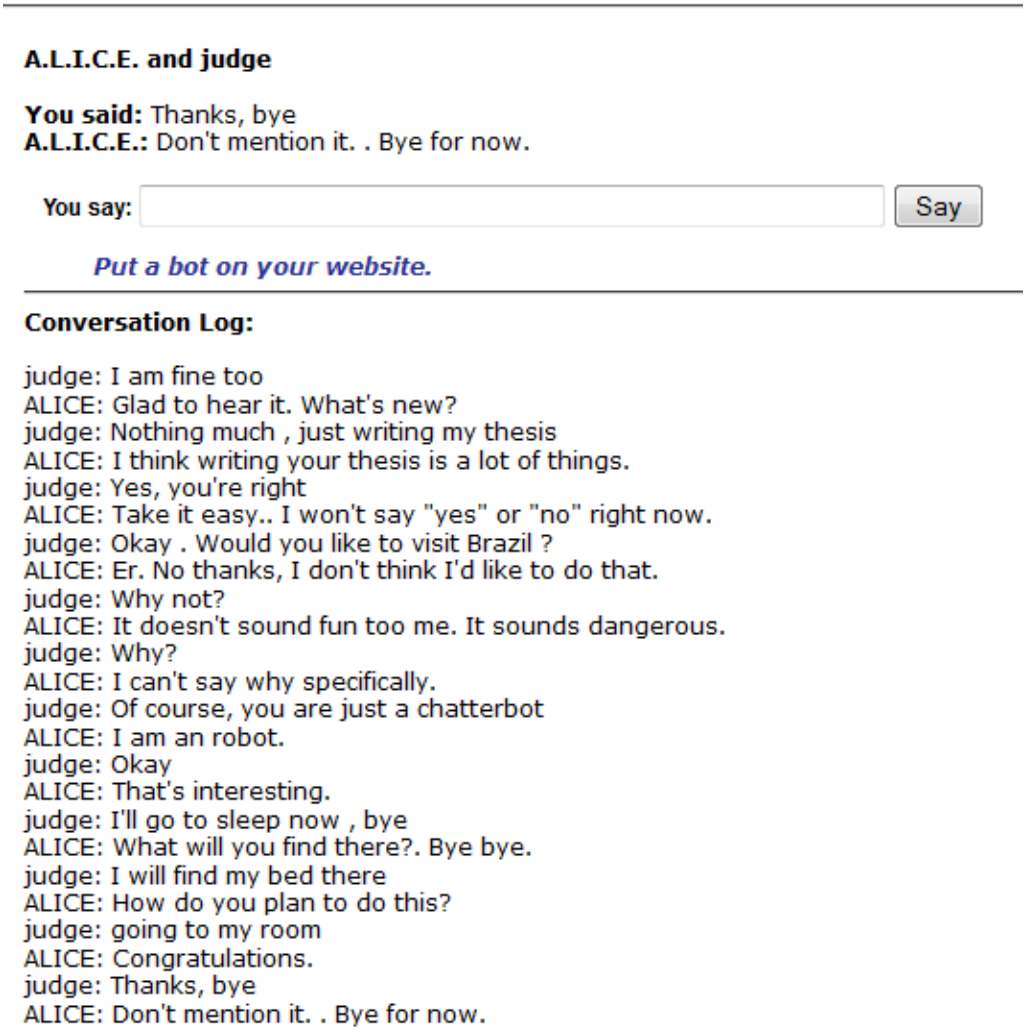


**Figura 7- Exemplo de conversa com ALICE**

**Fonte: http://alice.pandorabots.com/**

Para conversar com ALICE, basta preencher o campo de pergunta e clicar no botão Say, e logo em seguida é retornada uma resposta, às vezes com outra pergunta. Todo o histórico da conversa é gravado na forma de logs. Para exemplificar, na conversa da figura 7, em um momento perguntou-se para ALICE se ela gostaria de visitar o Brasil, e esta respondeu que não, que isso soa perigoso. É no mínimo interessante programas deste tipo, passando muitas vezes a impressão de que estamos realmente falando com um ser humano de verdade.

É possível também desenvolver um chatterbot próprio baseado em ALICE, através da edição de um arquivo AIML, mas no entanto é preciso utilizar um interpretador. No site oficial do ALICE está disponível para download gratuito o interpretador de arquivos AIML, escrito para diferentes linguagens de programação.

### 3.4. Ogma (Luiz Cláudio Gomes Maia, 2008)

Ogma é um software para análise de textos criado em 2008 na Universidade Federal de Minas Gerais, pelo professor doutor Luiz Cláudio Gomes Maia. A ferramenta foi desenvolvida para auxiliar na sua tese de doutorado. Utiliza PLN na aplicação de tarefas como etiquetagem morfossintática, contagem de termos, entre outros.

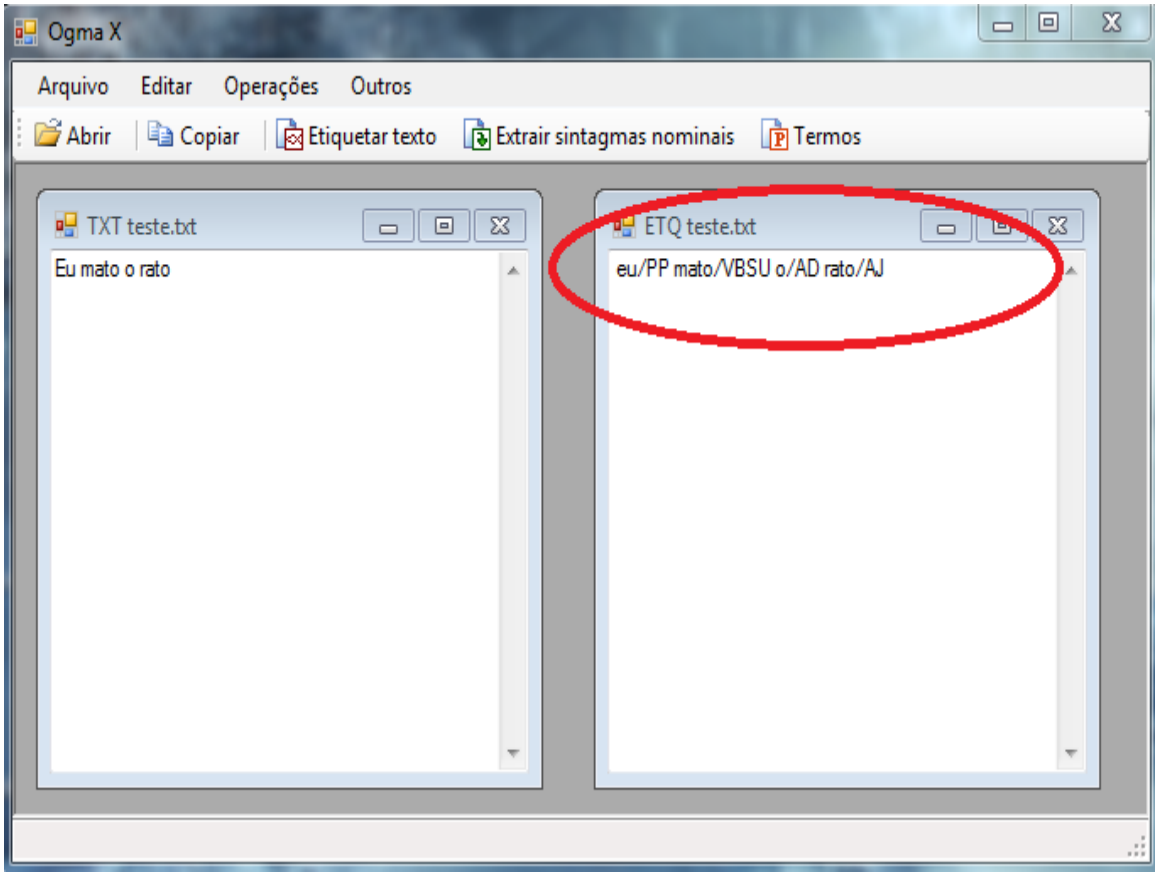


**Figura 8- Exemplo de etiquetagem**

**Fonte: MAIA, 2009**

A etiquetagem morfossintática consiste em identificar as categorias gramaticais das palavras em uma sentença, e é importante em aplicações que precisem extrair a categoria gramatical das palavras (TAGNIN, 2008). A figura 8 mostra um exemplo de etiquetagem morfossintática do software Ogma. A frase "Eu mato o rato" foi analisada e etiquetada, tendo como resultado "eu/PP mato/VBSU o/AD rato/AJ". É possível reparar que os termos da frase foram divididos e etiquetados por categorias gramaticais, onde PP refere-se ao pronome pessoal "eu", VBSU refere-se à palavra "mato" do verbo matar, AD refere-se ao artigo definido "o" e AJ refere-se ao adjetivo "rato".

O software pode ser baixado gratuitamente no endereço http://www.luizmaia.com.br/ogma/, e é compatível com o sistema operacional Microsoft Windows.

### 3.5. Sistema de Recomendação para Apoio à Análise Fundamentalista do Mercado de Capitais Utilizando Processamento de Linguagem Natural (Tomás Augusto Müller, 2011)

Desenvolvido em 2011 por Tomás Augusto Müller como trabalho de conclusão de curso pela Universidade de Santa Cruz do Sul (UNISC), é um sistema de recomendação para apoio à análise fundamentalista do mercado de ações. Utiliza PLN para descobrir o conhecimento existente no conteúdo de notícias informadas pelo usuário, e com base nestas notícias, recomenda ações para serem tomadas conforme as operações do mercado de capitais (MULLER, 2011).

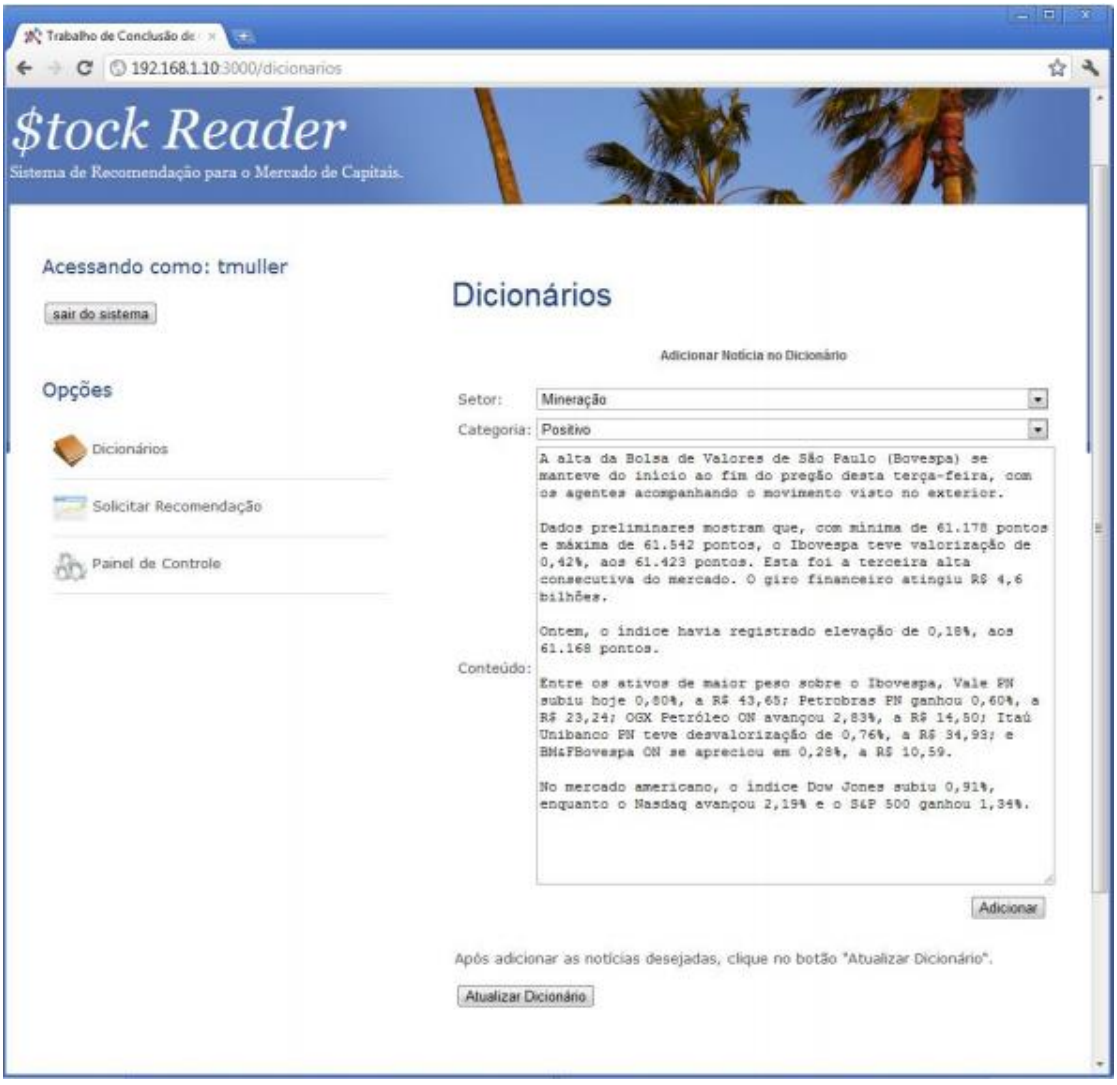


**Figura 9- Adicionando uma notícia no sistema**

**Fonte: MULLER, 2011**

A figura 9 mostra o layout do sistema em um dos momentos de seu fluxo, o cadastro de uma nova notícia. É informado o setor de interesses que a notícia pertence, a categoria da notícia, e o seu conteúdo. As categorias das notícias são três:

- **Positivo**: Representa uma tendência de alta nos preços de determinada empresa. O sistema recomenda realizar uma operação de venda.

- **Negativo**: Representa uma tendência de baixa nos preços de determinada empresa. O sistema recomenda realizar uma operação de compra.

- **Neutro**: O mercado encontra-se neutro. O sistema recomenda manter as ações de determinada empresa, não realizando operação de compra nem de venda.

Ao cadastrar uma notícia, o texto é então submetido para o motor do sistema, onde ocorre o PLN. Nesta etapa, a análise e processamento do texto pode ser dividida em outras duas etapas: Tratamento da Linguagem Natural (TLN) e cálculo de pesos.

Na etapa de TLN, a notícia que o usuário cadastrou é modificada. São removidas palavras que não iniciem com letras, e *stopwords* (palavras não representativas para um documento). No final, o resto do texto é transformado em tokens.

Após executar o tratamento do texto, os tokens recebem valores, denominados de pesos, e após são armazenados em dicionários. Estes pesos determinam a relevância que cada token tem no dicionário em que faz parte. Cada setor tem os seus dicionários, e estes são classificados pelos tipos positivos, negativos e neutros, onde desta forma é possível agrupar palavras-chave que representam determinado sentimento para os setores, permitindo assim realizar uma análise estatística da linguagem, oferecendo um interessante modelo para representar e manipular o conhecimento não estruturado (MULLER, 2011).

A equação de atribuição de pesos aos tokens é *TF-IDF = TF x IDF*. Segundo Muller (2011, p. 71), "TF (Term Frequency) representa a frequência que a palavra aparece em determinado documento, e IDF (Inverse Document Frequency) é a escala de importância da palavra em um conjunto de documentos". Esta equação consegue representar o grau de

importância de uma palavra dentro de uma coleção de documentos. Na figura 10, é possível observar o cálculo TF-IDF para a palavra “aumento”.

| TF | IDF |
|---|---|
| 1 documento com 100 palavras<br>A palavra “aumento” aparece 3 vezes.<br>TF = (3 / 100) = 0.03 | Conjunto de 10.000.000 documentos.<br>A palavra “aumento” aparece em 1000 documentos.<br>IDF = Log(10.000.000 / 1000) = 4 |
| O valor do índice TF-IDF é dado pela multiplicação:<br>TF-IDF = 0.03 x 4 = 0.12 | |

**Figura 10- Exemplo de cálculo de peso**

**Fonte: MULLER, 2011**

As descrições anteriores dizem respeito ao cadastro de uma notícia. Porém, o objetivo da aplicação é recomendar notícias. Para isso o sistema dispõe uma tela onde o usuário solicita uma recomendação, informando o setor ou empresa, junto com a notícia que deseja analisar. Essa notícia passa pelos mesmos processos de TLN e cálculo de pesos, e logo após é realizado o cálculo de recomendação com base nos dicionários do setor que foi especificado. Muller (2011, p. 72) explica que no cálculo de recomendação, “é identificado qual o dicionário (positivo, negativo ou neutro) para o setor em questão, que possui o maior valor acumulado após proceder com o somatório dos pesos existentes em cada dicionário para as palavras da notícia avaliada”. Após o cálculo, o sistema apresenta em uma nova tela o resultado da recomendação. A figura 11 mostra uma parte deste resultado.


**Figura 11- Resultado de uma recomendação**

**Fonte: MULLER (2011)**

### 3.6. DOSVOX (Universidade Federal do Rio de Janeiro, 1993)

Criado exclusivamente para deficientes visuais em 1993 na Universidade Federal do Rio de Janeiro (UFRJ), é um sistema que se comunica com o usuário através da voz, processando textos escritos e sintetizando-os através da fala. O sistema operacional DOSVOX permite que pessoas cegas utilizem um microcomputador comum (PC) para desempenhar uma série de tarefas, adquirindo assim um nível alto de independência no estudo e no trabalho (DOSVOX, 2015).

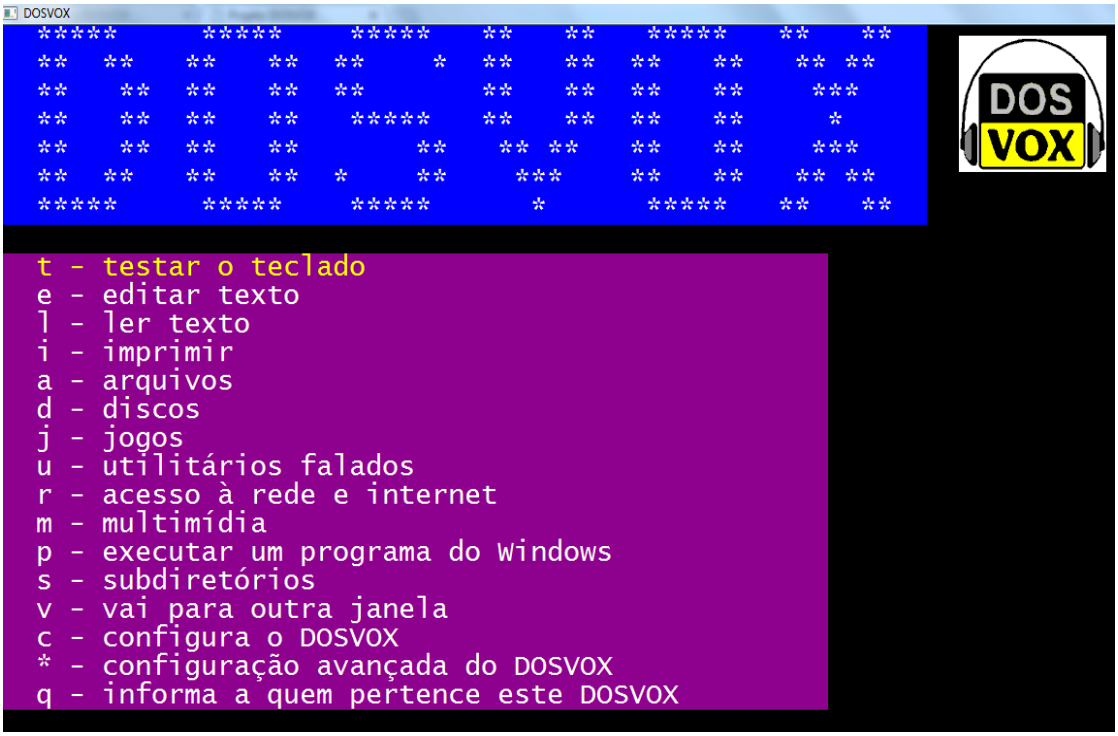


**Figura 12- Tela inicial do DOSVOX**

**Fonte: DOSVOX, 2015**

A figura 12 mostra a tela inicial do sistema. É possível navegar pelo menu através das setas direcionais do teclado ou pelas teclas correspondentes de cada opção, e a cada escolha feita pelo usuário é sintetizado o texto em voz natural. Praticamente toda entrada de teclado é falada, ajudando bastante o deficiente visual, que dificilmente irá ficar perdido durante o uso no sistema.

Segundo informações do site oficial (http://intervox.nce.ufrj.br/dosvox/), o DOSVOX possui mais de 80 programas internos que abrangem diferentes áreas, como: Jogos

didáticos, editor, leitor e impressor de textos, impressor para braile, ampliador de telas para pessoas com visão reduzida, acesso à internet, possibilitando o deficiente visual estar em contato com o mundo, entre outros. Estes programas são escolhidos através do menu inicial do DOSVOX (figura 12). Um destes programas internos muito interessante é o *GoogleVox*, com ele é possível realizar pesquisas na internet utilizando o motor de buscas do Google. A figura 13 mostra os resultados obtidos ao consultar a palavra "UNISC":

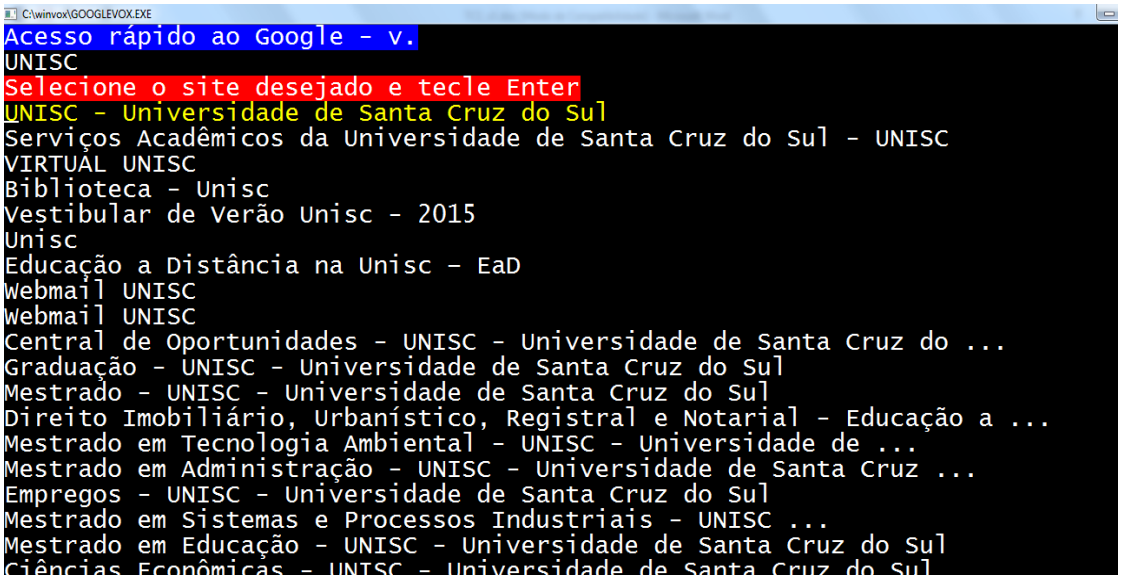


**Figura 13- Utilização do programa GoogleVox**

**Fonte: DOSVOX, 2015**

Para navegar pelos resultados obtidos, basta o usuário escolher o site e teclar ENTER. Feito isso, o DOSVOX realiza a leitura em linguagem natural do título inicial do site escolhido, e apresenta duas opções extras. A primeira é para navegar pelo site através da interface do sistema, mapeando o site e realizando uma leitura de todos os links e possibilidades de navegação daquele site. A segunda opção é abrir o site em um web browser comum.

A versão atual do programa é 4.5, e pode ser adquirido de forma gratuita por download através do endereço http://intervox.nce.ufrj.br/dosvox/download.htm. É compatível com o sistema operacional Microsoft Windows. Estima-se que o número de usuários do sistema seja de mais de 60000 pessoas (DOSVOX, 2015).

## 3.7. Comparação entre os trabalhos

A tabela 2 apresenta um comparativo entre os trabalhos apresentados neste capítulo, mostrando as características de cada um.

**Tabela 2 - Comparativo entre os trabalhos apresentados**

| Trabalho Relacionado | Entrada de dados | Saída de dados | Tipo / Técnica de PLN | Resultado do PLN | Ambiente |
|---|---|---|---|---|---|
| **Evi** | Texto / Voz | Texto / Voz | Análise semântica do texto, conforme base de dados de conhecimento. | Resposta de pergunta em linguagem natural. | Disponível em web site e na forma de aplicativo para smartphones. |
| **CoGroo** | Texto | Texto | Utiliza análise morfológica, sintática e regras gramaticais para detectar erros. | Marca o local do erro gramatical no texto, sugerindo correções. | Disponível em web site, e na forma de API Java acoplável ao libreOffice ou em outros projetos. |
| **ALICE** | Texto | Texto | Utiliza AIML (Artificial Intelligence Markup Language) para especificação e criação de regras e base de dados de conhecimento. | Resposta de pergunta em linguagem natural. | Disponível em web site. |
| **Ogma** | Texto | Texto | Etiquetagem morfossintática, contagem de termos, entre outros. | Na etiquetagem morfossintática, marca cada termo do texto com a sua | Disponível em ambiente desktop. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | respectiva categoria gramatical. | |
| **Sistema de Recomendação para Investimentos no Mercado de Capitais** | Texto | Texto | Tokenização e atribuição de pesos para realizar análise estatística da linguagem natural. | Recomendação do texto analisado em linguagem natural. | Disponível em web site. |
| **DOSVOX** | Texto | Texto / Voz | Sintetização de voz através de comandos do teclado ou arquivos de textos. | Sintetização de voz. | Disponível em ambiente desktop. |

**Fonte: (AUTOR, 2015)**

Todos os trabalhos estudados contribuíram para conhecer e compreender a aplicação das técnicas de PLN. No entanto, destaca-se a pesquisa "Sistema de Recomendação para Investimentos no Mercado de Capitais", que apresenta uma solução de PLN similar à proposta neste trabalho. Porém, cabe destacar que diferente desta pesquisa, que utilizou uma solução estatística de PLN fazendo uso do cálculo da frequência inversa, optou-se por uma solução baseada no cálculo da frequência relativa, conforme justificada na seção 2.4.

## 3.8. Considerações

Todos os trabalhos apresentados neste capítulo possuem uma relação em comum com o presente trabalho, todos utilizam PLN para análise da linguagem natural, seja escrita ou falada. Com o estudo dos mesmos, fica clara a importância do uso desta técnica. Por exemplo, quando o programa DOSVOX realiza a leitura de um arquivo de texto, ele não fala simplesmente palavra por palavra em separado como se não tivessem ligação, mas sim respeitando pontuações, vírgulas, início de frase, fim de frase, respeita o tempo entre um parágrafo e outro, entre outros, sendo que tais análises só são possíveis graças ao PLN.

Ou quando é feita a pergunta “O céu é azul?” para ELIZA, e esta por sua vez responde “Sim, exceto à noite”. Esta resposta foi dada utilizando técnicas de PLN, onde foi preciso analisar que a frase escrita é do tipo pergunta, analisar a relação entre céu e azul, e buscar na sua base de conhecimento a resposta certa, com o acréscimo da informação de que à noite o céu não é azul.

Considera-se então, adequado o uso de PLN para o presente trabalho, que consiste principalmente em analisar textos escritos em linguagem natural e o conteúdo das páginas de sites.

# 4. MAPEAMENTO DE ROTA DE ESTUDO NA WEB BASEADA NA ESTRATÉGIA DE GAMIFICAÇÃO

Neste capítulo, será apresentado o trabalho de conclusão de curso "Mapeamento de Rota de Estudo na Web Baseada na Estratégia de Gamificação" (MaRE – Mapeamento da Rota de Estudo), do aluno Kelvin S. Teixeira, que foi desenvolvido em 2014/2015 na UNISC, para obtenção do título de Bacharel em Ciência da Computação. Compreender a etapa de definição do agente inteligente utilizado é importante, pois é nesta parte que o presente trabalho procurou dar a sua contribuição com o uso de PLN.

## 4.1. Propósito e objetivo

Segundo Teixeira (2014), devido ao crescimento da web nos últimos anos, os usuários encontram dificuldades em filtrar informações em suas pesquisas, por vezes se sentindo frustrados por não terem encontrado o que desejam. Diante deste cenário, surgem artefatos tecnológicos com o objetivo de auxiliar o usuário durante a sua navegação e busca por informação na *web*. Muitos deles centralizam informações em determinados assuntos, para manter um conteúdo de qualidade. Exemplos destes sistemas são:

- **LinguaLeo**: Site especializado no ensino da língua inglesa, de forma gratuita.
- **Khan Academy**: Site que oferece ensino em diferentes áreas, como matemática, português, programação, entre outros, também de forma gratuita. O aprendizado acontece através de vídeo aulas.
- **Codecademy**: É um site voltado para o aprendizado em programação. Nele são disponibilizadas vídeo aulas sobre diversos temas envolvendo programação, e o usuário pode ir colocando em prática o que for aprendendo através de um editor de código que o próprio site oferece.

Sabendo do problema da busca por informações de qualidade e da influência da *web* como espaço de estudo, o propósito do trabalho MaRE é mapear a rota de estudo na web. Para tal, foi utilizado agente inteligente e elementos de gamificação (explicado no

decorrer do capítulo), com o objetivo de tornar a experiência do usuário mais interessante durante a sua navegação na *web* (TEIXEIRA, 2014).

## 4.2. Funcionamento da Aplicação

A aplicação foi desenvolvida na linguagem de programação C#, e possui um WebView (*container* no qual são exibidos conteúdos web dentro do próprio aplicativo) onde o usuário é capaz de navegar na internet. No início da navegação, o usuário irá informar o assunto que ele deseja pesquisar, para que o agente inteligente possa saber qual página será importante para ele. Durante essa navegação, o sistema analisa e armazena todos os sites que foram visitados e, ao término da pesquisa, é apresentada a rota de estudo do usuário, permitindo assim que este possa refletir sobre tudo o que foi pesquisado e o que de fato tenha importância para ele.

O tipo de agente utilizado possui duas características, reativas e baseado em modelo, podendo ser considerado um agente híbrido. Para a primeira característica, Teixeira (2014, p. 43) cita que "As características do agente reativo serão úteis no que tange à detecção do conteúdo das páginas e posterior aviso ao usuário de que aquele assunto não é pertinente ao objetivo definido". Basicamente, durante a navegação do usuário, o agente irá verificar quando há mudança de uma página, armazena o conteúdo dela (que serve como dados de entrada) e, se necessário, interage com o usuário, por exemplo, questionando sobre o objetivo de pesquisa e a relevância da página para isto. A segunda característica vem dos agentes baseados em modelo, ele irá guardar um histórico das páginas do usuário, para quando este acabar com a sua pesquisa, o agente poder resgatá-las, apresentando-as em forma de grafos (o objetivo dos grafos é abordado adiante).

Com o intuito de tornar mais agradável e engajador o uso do sistema, utilizou-se os elementos de gamificação, como Conquistas, Progressão, Missões, Personalização e Pontuação. Neste sistema, as conquistas são representadas através de medalhas, e ganhas quando o usuário executa determinadas ações. A progressão é visualizada através de uma barra de experiência. Esta barra será preenchida conforme o usuário desempenha tarefas, como por exemplo, acessar um site que seja de relevância para ele. As missões são impostas pelo sistema quando o usuário deseja pesquisar algo, e este pode especificar

quais objetivos deseja alcançar com a pesquisa. A personalização é utilizada junto com a barra de experiência, para quando o usuário atingir diferentes níveis, o sistema libere ferramentas para que este possa personalizar o seu perfil. A pontuação serve para o usuário ter uma ideia estatística de como está o seu uso no sistema (TEIXEIRA, 2014).

Como resultado final, a aplicação apresenta a rota de estudo na forma de grafos direcionados. O grafo mostra todas as páginas visitadas pelo usuário, incluindo as que foram e não foram relevantes para a sua pesquisa. A figura 14 apresenta o grafo da rota de estudo. Os números das arestas representam a ordem em que as páginas foram acessadas. Os nós de cor laranja representam páginas que não possuem relevância para o usuário, os de cor cinza e azul, que possuem, contudo os nós azuis são marcados como relevantes pelo próprio usuário.

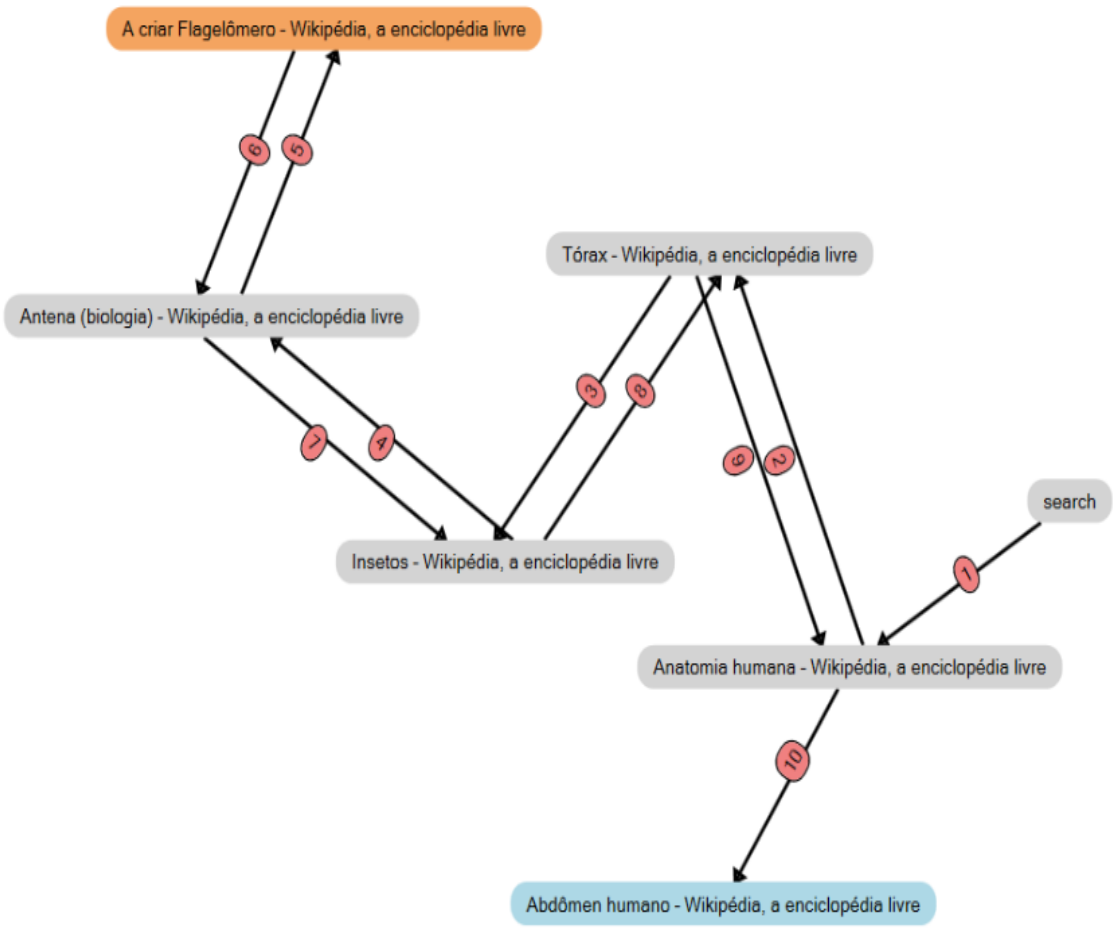


**Figura 14- Grafo contendo a rota de estudo do usuário**

**Fonte: TEIXEIRA, 2014**

Toda vez que o usuário acessar uma página, o agente inteligente vai analisá-la. Caso o conteúdo da página não seja relevante, o agente irá marcar de laranja (página sem importância). Se o assunto tiver relação com o que o usuário deseja procurar, o agente irá marcar a página de cinza (página normal, de relevância). Caso a página for muito importante e de total relevância, neste caso o usuário pode vir a fazer uma marcação, e esta será representada de azul no grafo.

## 4.3. Considerações

Este capítulo apresentou a aplicação desenvolvida pelo aluno Kelvin S. Teixeira, mostrando a explicação do problema e objetivos, o funcionamento geral do sistema, bem como o uso e definição de agentes inteligentes e elementos de gamificação. A aplicação apresentada serviu de cenário de testes para o desenvolvimento do presente trabalho, no qual o agente inteligente (da aplicação MaRE) foi incrementado com o uso de PLN estatístico.

Inicialmente, para descobrir se uma página é ou não relevante, o agente somente fazia uma simples comparação das palavras de pesquisa do usuário com o conteúdo das páginas web, sem usar outra técnica. Mas essa simples comparação de palavras, sem analisar o significado e contexto onde estão inseridas, pode vir a trazer resultados equivocados. Se, por exemplo, o usuário pesquisar sobre “presente de aniversário”, e estiver navegando em uma página que fale sobre o “tempo presente” (no sentido de “nos dias de hoje”, “atualmente”), o agente vai marcar esta página como sendo relevante, quando na verdade não é. Por este motivo, se fez necessário o uso de PLN, pois utilizando esta técnica é possível extrair representações mais válidas, no que diz respeito à relevância de palavras em textos, realizando assim a análise do conteúdo das páginas web e classificando-as como relevante de forma mais precisa.

# 5. ANÁLISE E CLASSIFICAÇÃO DE CONTEÚDO TEXTUAL

Este capítulo mostra como ocorreu o desenvolvimento da análise e classificação de conteúdo textual para este projeto. Incialmente apresenta-se uma visão geral e solução algorítmica, finalizando com possibilidades de integração e considerações.

## 5.1. Visão geral do processamento

Para este trabalho de conclusão, foi desenvolvido um processador de conteúdo textual que identifica, através de cálculos e contagens, as palavras de maior relevância contidas em um texto, com o objetivo de descobrir o seu assunto de foco. Com este processador é possível obter informações estatísticas de um determinado texto, separá-lo em *tokens*, e classificá-lo de acordo com a sua frequência relativa.

A figura 15 ilustra, com base no texto exemplo apresentado, a ação do processor, ou seja, como ocorre a análise e a classificação das palavras do texto.

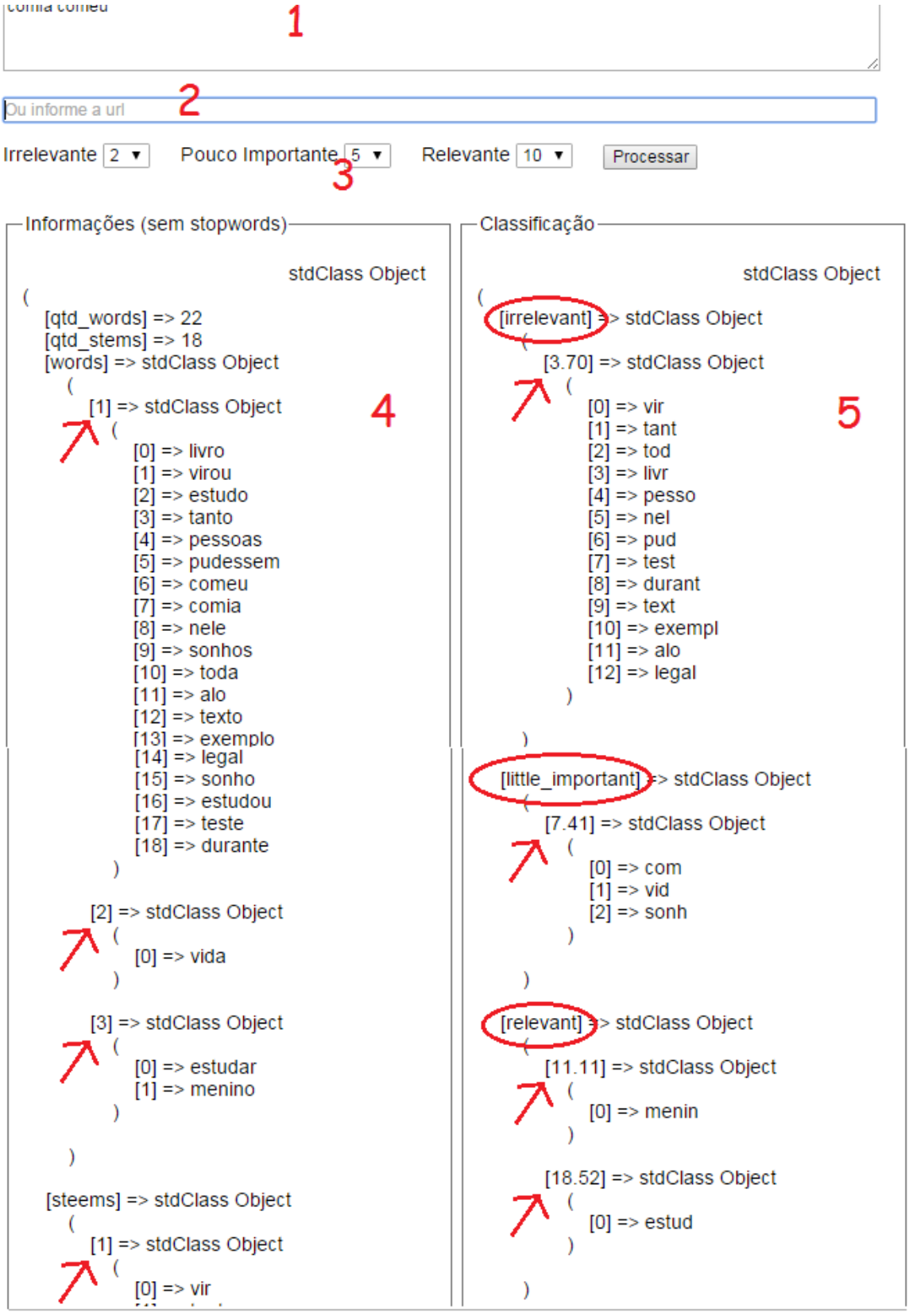


**Figura 15 – Página demo do processador de textos**

**Fonte: (AUTOR, 2015)**

Explicando detalhadamente cada numeração da figura 15, tem-se que:

1. Local onde é informado o texto a ser processado.

2. Entrada para informar o endereço do site e buscar o seu conteúdo para ser processado. Caso seja informado o endereço e o texto (item 1) juntos, o processamento para o texto irá prevalecer.

3. Percentual de classificação dos grupos de relevância. Estes percentuais definem em quais grupos de classificação de relevância as palavras irão fazer parte. Para o exemplo, foi marcado que 2 por cento do total dos grupos de frequência relativa (será explicado a seguir, no item 5) fazem parte do grupo irrelevante, entre 2 e 5 por cento do grupo pouco importante, e entre 5 e 10 por cento, do grupo relevante.

4. Informações resultantes do processamento do texto. No exemplo, o texto processado apresentou uma contagem de 22 palavras (sem stopwords, que será explicado na seção 5.2 deste capítulo), 18 stems (explicado na seção 5.2). Também é possível observar cada palavra e cada stem do texto, em separado, e a sua frequência absoluta no texto através das flechas contidas na imagem.

5. Grupos de classificação de relevância. Estes grupos são divididos em três categorias: irrelevante, pouco importante e relevante (circulados). As flechas representam o valor da frequência relativa atribuído para cada *token*, que foi obtido através do cálculo de frequência e são ordenados de forma crescente. O objetivo de separar os *tokens* em grupos é o de possibilitar o destaque para os *tokens* mais relevantes, podendo, por exemplo, em uma comparação de palavras, excluir o grupo dos irrelevantes por saber que este grupo não tem muito significado no texto, já os relevantes e pouco importantes sim.

## 5.2. Etapas do processamento textual

O processamento do conteúdo textual envolve três etapas: a manipulação da linguagem natural, a análise e a classificação. O enfoque do desenvolvimento deste trabalho de conclusão foi para a análise de textos obtidos através de web sites, que

possuem marcações HTML espalhadas em seu conteúdo, porém vale ressaltar que textos normais (sem marcações HTML) também são tratados normalmente.

Desta forma, a etapa de manipulação do conteúdo textual corresponde aos seguintes passos:

- Eliminação de tags HTML, códigos javascript e estilos CSS (cascading style sheets ou folhas de estilo em cascata) do conteúdo de uma página web.

- Remoção de palavras que não iniciem com letras (exemplo: números, ou outros caracteres).

- Utilização da técnica de *tokenização*, vista na etapa de análise léxica de PLN, onde o texto é transformado em *tokens*. Como é processado textos, foi utilizado como delimitador o espaço em branco (\n), para então obter os tokens (palavras).

- Remoção de *stopwords*. Segundo Gonzalez e Lima (2003, p. 13), "A eliminação de *stopwords* pode ser, também, uma estratégia adotada no PLN. *Stopwords* são palavras funcionais, como artigos, conetivos e preposições". Alguns exemplos de *stopwords* são: as, e, os, de, para. Estas palavras acabam sendo removidas porque são irrelevantes para o processamento do texto. O arquivo de *stopwords*, denominado de stoplist.txt, foi retirado do site http://snowball.tartarus.org/algorithms/portuguese/stemmer.html. Foi optado por colocar as *stopwords* em um arquivo externo para ser fácil de adicionar ou retirar outras *stopwords* conforme a necessidade.

- Normalização dos *tokens* para letras minúsculas.

- Redução dos *tokens* para o seu respectivo *stem* (palavras relacionadas são reduzidas a partes iguais). Desta forma, palavras relacionadas, no plural ou escritas de forma incorreta, serão tratadas com o mesmo significado. Esta técnica é conhecida como *stemming*. Por exemplo, ESTUD é o *stem* das palavras estudante, estudou, estudo, estudar. O algoritmo utilizado para realizar o *stemming* foi criado pelo cientista da computação Martin Porter e é chamado de *snowball*. O módulo em PHP deste algoritmo foi retirado do site

http://snowball.tartarus.org/algorithms/portuguese/stemmer.html. Caso o servidor PHP que o processador textual for instalado não suporte este módulo, é usado um arquivo denominado stemm.txt, que contém mais de 32 mil palavras e seus respectivos *stems*. Conforme surgem novas palavras é possível atualizar este arquivo facilmente. A figura 16 apresenta como é a construção deste arquivo:

```
abaixa=abaix
abaixe=abaix
abaixei=abaix
abaixo=abaix
abaixou=abaix
abalada=abal
abalado=abal
abalaram=abal
abalos=abal
abalou=abal
abalroado=abalro
abandona=abandon
abandoná=abandon
abandonada=abandon
abandonadas=abandon
abandonado=abandon
abandonados=abandon
abandonam=abandon
```

**Figura 16 – Construção do arquivo de stems**

**Fonte: (AUTOR, 2015)**

Uma vez feita a manipulação inicial do texto, para a etapa de análise foi utilizado o critério de similaridade e relevância, aplicando um valor de frequência relativa para cada palavra do texto analisado, conforme a fórmula matemática de frequência relativa, vista no capítulo 2.

Para a etapa de classificação de relevância das palavras em relação ao texto analisado, foi utilizado como critério o valor de frequência relativa atribuído para cada palavra, que irá definir em qual grupo de classificação a mesma fará parte. Para definir o intervalo de classificação de cada grupo, é estipulado um ponto de corte. A tabela 3 apresenta um exemplo válido de classificação, o qual foi utilizado na implementação, conforme descrito na seção 5.1 (percentual de classificação dos grupos de relevância). Os valores de porcentagem de classificação podem ser ajustados conforme a necessidade.

**Tabela 3 - Grupos de classificação de relevância**

| Porcentagem de classificação | Grupo de classificação |
|---|---|
| 0 – 20 | Irrelevante |
| 20 – 50 | Pouco importante |
| 50 – 100 | Relevante |

**Fonte: (AUTOR, 2015)**

### 5.3. Tecnologias utilizadas e exemplos de uso

O sistema foi desenvolvido na linguagem de programação PHP 5.5 (PHP: Hypertext Preprocessor), e funciona nos sistemas operacionais Windows e Linux. Foi criado uma classe chamada textProcessor que contém 3 métodos de acesso público:

- getTextInfo. Aceita como parâmetro de entrada o texto a ser processado (tipo string). Tem como retorno um array formatado em string JSON, contendo as seguintes informações: Total de palavras, total de stems, as palavras e os stems. A figura 17 mostra um exemplo de chamada, já a figura 18 mostra o retorno esperado.

```php
<?php
require_once __DIR__ . '/TextProcessor/textProcessorClass.php';

try {
    $texto = "Exemplo de texto para ser processado. Estudo estudar estudou comer comeu";

    $obj = new textProcessor();
    $ret = $obj->getTextInfo($texto);
} catch(Exception $e) {
    echo $e->getMessage();
}
```

**Figura 17 – Exemplo de chamada do método getTextInfo**

**Fonte: (AUTOR, 2015)**

```
{
    "qtd_words": 8,
    "qtd_stems": 5,
    "words": {
        "1": {
            "0": "estudou",
            "1": "comer",
            "2": "comeu",
            "3": "estudar",
            "4": "estudo",
            "5": "texto",
            "6": "processado",
            "7": "exemplo"
        }
    },
    "steems": {
        "1": {
            "0": "exempl",
            "1": "process",
            "2": "text"
        },
        "2": {
            "0": "com"
        },
        "3": {
            "0": "estud"
        }
    }
}
```


**Figura 18 – Retorno do método getTextInfo**

**Fonte: (AUTOR, 2015)**

É importante notar na figura 18 que foi retornado um total de 8 palavras processadas, no entanto somente 5 *stems*, isso porque as palavras comer e comeu foram identificadas pelo processador como COM, por terem um significado muito parecido. O mesmo valeu para as palavras estudar, estudo e estudou, resultando apenas em ESTUD.

- getTextStems. Aceita como parâmetro de entrada o texto a ser processado (tipo *string*). Tem como retorno um *array* formatado em *string* JSON, contendo os *stems* extraídos do texto.

- getTextClassification. Aceita 3 parâmetros de entrada, o texto a ser processado (tipo *string*), o percentual do grupo irrelevante (tipo inteiro de 0 a 8), e o percentual do grupo pouco importante (tipo inteiro de 1 a 9). Tem como retorno um *array* formatado em *string* JSON, contendo os *stems* extraídos do texto. A

figura 19 mostra um exemplo de chamada, já a figura 20 mostra o retorno esperado.

```php
<?php
require_once __DIR__ . '/TextProcessor/textProcessorClass.php';

try {
    $texto = "Exemplo de texto para ser processado. Estudo estudar estudou comer comeu";

    $obj = new textProcessor();
    $ret = $obj->getTextClassification($texto, 2, 5);
} catch(Exception $e) {
    echo $e->getMessage();
}
```

**Figura 19 – Exemplo de chamada do método getTextClassification**

**Fonte: (AUTOR, 2015)**


```
{
    "irrelevant": {
        "12.50": {
            "0": "exempl",
            "1": "process",
            "2": "text"
        }
    },
    "little_important": {
        "25.00": {
            "0": "com"
        }
    },
    "relevant": {
        "37.50": {
            "0": "estud"
        }
    }
}
```


**Figura 20 – Retorno do método getTextClassification**

**Fonte: (AUTOR, 2015)**

A motivação por ter utilizado PHP para o desenvolvimento ocorreu por ser uma linguagem robusta, de fácil aprendizado, amplamente utilizada pela comunidade de programadores, mas que carece de algoritmos que tenham relação com uma manipulação mais profunda da linguagem natural, sendo então uma ótima oportunidade de contribuição para a comunidade da computação.

## 5.4. Possibilidade de integração

O processador textual também ficará disponível através de um web service. Desta maneira poderá ser consumido por qualquer linguagem de programação, e facilmente integrado com outros projetos. É possível visualizar a documentação do web service implementado a partir do endereço http://<IP>/TextProcessor/ws/ws.php. A figura 21 apresenta esta documentação.

# textProcessor SOAP WebService interface description

Endpoint URI: http://localhost/TextProcessor/ws/ws.php

WSDL URI: http://localhost/TextProcessor/ws/ws.php?WSDL

PHP SOAP client download URI: http://localhost/TextProcessor/ws/ws.php?PHPSOAPCLIENT

## Index

Public methods:

- getTextClassification
- getTextInfo
- getTextStems

## Public methods

### getTextClassification

```
string getTextClassification (
    string text,
    int irrelevant,
    int little
)
```

Processa o texto e retorna um json contendo os tokens
ordenados e classificados pelo cálculo da frequência relativa

- string **text**
  Texto
- int **irrelevant**
  Grupo irrelevant, de 0 a 8
- int **little**
  Grupo little_important, de 1 a 9

### getTextInfo

```
string getTextInfo ( string text )
```

Obtém informações quantitativas do texto

### getTextStems

```
string getTextStems ( string text )
```

Obtem stems de cada token do texto

---

Powered by PhpWsdl - PDF download: Download this page as PDF

**Figura 21 – Documentação do web service**

**Fonte: (AUTOR, 2015)**

Na figura 22, é possível ver um exemplo de como acessar os métodos da classe através do consumo do web service, utilizando o PHP como linguagem para o exemplo.

```php
<?php
try {
    $wsdl = "http://localhost/TextProcessor/ws/ws.php?WSDL";
    $client = new SoapClient($wsdl);

    $texto = "Exemplo de texto para ser processado";

    $info = $client->getTextInfo($texto);
    $stems = $client->getTextStems($texto);
    $classific = $client->getTextClassification($texto, 2, 5);

} catch (Exception $e) {
    echo $e->getMessage();
}
```

**Figura 22 – Acesso do web service via PHP**

**Fonte: (AUTOR, 2015)**

## 5.5. Considerações

Este capítulo apresentou a implementação e desenvolvimento do presente trabalho. Foi abordado o objetivo do trabalho, a ideia principal de desenvolvimento, definição das tecnologias, exemplos de uso e possibilidade de integração com outros aplicativos. A tabela 4 faz uma síntese das características do processador textual, características também utilizadas na comparação dos trabalhos relacionados apresentados no capítulo 3. Destaca-se que quanto a técnica de PLN, diferente dos demais trabalhos, utilizou-se uma solução estatística baseado no cálculo de relevância de palavras, ou seja, cálculo da frequência relativa.

**Tabela 4 - Características do processador textual**

| | **Entrada de dados** | **Saída de dados** | **Tipo / Técnica de PLN** | **Resultado do PLN** | **Ambiente** |
|---|---|---|---|---|---|
| **Processador textual** | Texto | Texto | Solução estatística de PLN (cálculo de relevância | Texto formatado em JSON, separado por categorias (irrelevante, | Ambiente servidor, disponível através de webservic |

| | | | | | |
|---|---|---|---|---|---|
| | | | de palavras) | pouco importante e relevante). | es para possível integração. |

**Fonte: (AUTOR, 2015)**

# 6. TESTES E RESULTADOS

Este capítulo apresenta os testes do processador textual, a integração com o MaRE e os resultados obtidos. Os valores de porcentagem de classificação usados para os testes foram definidos em 0 até 20 por cento para o grupo irrelevante, de 21 até 50 por cento para o grupo pouco importante, e de 51 até 100 por cento para o grupo relevante, uma vez que com está configuração foi possível chegar a um resultado satisfatório.

## 6.1. Testes do processador textual

Para verificar a eficiência do processador, primeiramente o sistema foi testado com pequenos textos onde o assunto de foco era bem explícito. Nestes casos, o retorno do processamento foi positivo, contendo nos grupos pouco importante e relevante as palavras que condiziam com a ideia dos textos. Um exemplo deste teste é apresentado a seguir, com o texto retirado do site http://webtextos.blogspot.com.br/ que aborda sobre amor:

> Penso que tudo que se refere ao amor, pressupõe o bem, o bom, o belo. Não podemos aceitar coisas ruins e dolorosas em nome do amor. Sofrer por amor não é vantagem nenhuma, embora a mídia tente nos convencer dia após dia que sofrimentos sentimentais são algo bonito e digno. Não caia nessa. Amor bom é amor equilibrado.
>
> Quando o amor diz respeito à nossa pessoa, ou seja, quando o assunto é se amar, é preponderante que levemos a assertiva acima quase como uma lei. Se amar é sairmos fora de toda e qualquer possibilidade destrutiva com a qual possamos estar envolvidos. E ao contrário, é fazermos todo o bem possível à nós mesmos. Pode parecer uma atitude hedonista e narcísica, mas não é. Somente teremos plenas condições de amar aos outros depois que estivermos fortalecidos, amando o nosso próprio ser e aceitando nossa existência do jeito que nos foi concedida.

O resultado foi que a palavra AMAR e AMOR ocuparam o grupo relevante, ou seja, condiz com o tema principal do texto, como ilustrado na figura 23.

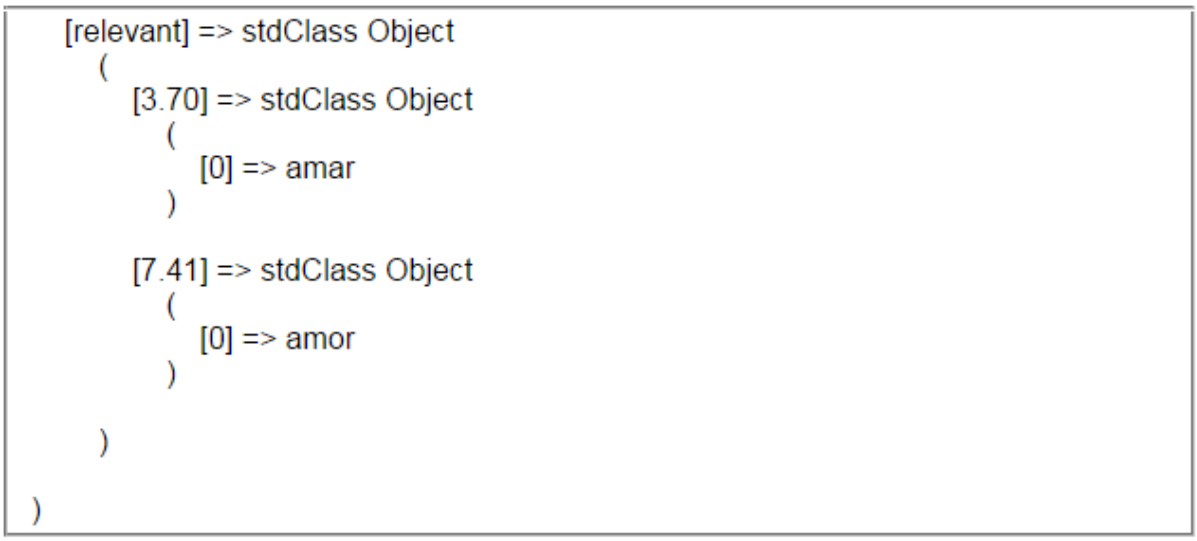


**Figura 23 – Resultado do processamento do texto sobre amor**

**Fonte: (AUTOR, 2015)**

Já quando se analisa textos complexos, como um poema por exemplo, o resultado não é tão satisfatório, como pode ser observado no texto a seguir, retirado do site http://pensador.uol.com.br/:

> Com boca, dentes e saliva, você desmente minhas neuras, desarma meus complexos infantis de feiura, um por um, cheio de malícia, paciência e atenção, intercalando lambidas, paradas estratégicas e olhares experimentais. É madrugada, é tarde, eu preciso ir, melhor nem começar, vamos finalizar isso separados, cada um na sua casa, no chuveiro, vai ser melhor.

O resultado retornado no grupo relevante foi a palavra MELHOR, ou seja, neste caso não foi possível chegar a uma conclusão, que é descobrir o assunto de foco do texto. Acredita-se que a análise estatística, por ter características baseadas na frequência da palavra, não consiga atingir resultados satisfatórios para textos que contenham termos distintos e que não se repetem.

A segunda etapa de testes foi feita utilizando a página demo do processador, informando o endereço de web sites. Quando é analisado o conteúdo de uma página web, identificou-se que a construção destas páginas, por não existir padronização obrigatória, torna difícil identificar o que é um texto válido e o que não é. Outro problema é a tarefa de fazer uma limpeza correta, para posterior análise, de todos os textos invisíveis, ou seja, tags HTML, códigos em javascript, estilos CSS, entre outros componentes que fazem

parte do conteúdo das páginas web, e que são misturados com os textos que realmente interessam.

A figura 24 apresenta um exemplo com um resultado positivo, podendo encontrar facilmente o texto. O conteúdo da página foi retirado do site da Wikipedia, tendo como pesquisa da página o vídeo game Super Nintendo. Neste caso, o processador apresentou como stems relevantes SUP, NINT, CONSOL, JOG e GAM, ou seja, fazem referência com palavras que realmente tem a ver com o texto analisado (o resultado do processamento pode ser conferido na figura 26). Já a figura 25 apresenta o conteúdo retirado da página inicial do site Hotmail. Neste exemplo, o processador não retornou nada, possivelmente porque o conteúdo é muito confuso, tendo pouco texto e muito código *javascript* envolvido.

```
<li class="toclevel-1 tocsection-7"><a href="#Refer.C3.A9ncios"><span class="tocnumber">7</span> <s
<li class="toclevel-1 tocsection-8"><a href="#Jogos"><span class="tocnumber">8</span> <span class="
<li class="toclevel-1"><a href="#Refer.C3.AAncias"><span class="tocnumber">9</span> <span class="to
<li class="toclevel-1 tocsection-9"><a href="#Ver_tamb.C3.A9m"><span class="tocnumber">10</span> <s
<li class="toclevel-1 tocsection-10"><a href="#Liga.C3.A7.C3.B5es_externas"><span class="tocnumber"
</ul>
</div>
<script>document.documentElement.className = document.documentElement.className.replace( /(^|\s)cli
<p></p>
<h2><span class="mw-headline" id="Hist.C3.B3ria">História</span><span class="mw-editsection"><span
<p>Tudo começou quando a <a href="/wiki/NEC_Corporation" title="NEC Corporation">NEC</a> decidiu co
<p>Como os dois videogames tinham processadores de <a href="/wiki/16-bits" title="16-bits" class="m
<p>Fora lançado ao fim de 1990 no Japão, nos EUA em Novembro de 1991 e depois em 1992 na Europa.</p
<p>A versão européia do console (lançado em 1992) é visualmente idêntica ao modelo japonês. O contr
<p>No Brasil, chegou oficialmente apenas em <a href="/wiki/30_de_agosto" title="30 de agosto">30 de
<p>A Nintendo garantiu seu sucesso no Japão especialmente por manter velhos parceiros, como <a href
<p>O Super NES e Super Famicom foram lançados com apenas alguns jogos, mas esses jogos foram bem re
<p>O Super NES foi sucedido pelo <a href="/wiki/Nintendo_64" title="Nintendo 64">Nintendo 64</a> em
<h2><span class="mw-headline" id="Emula.C3.A7.C3.A3o">Emulação</span><span class="mw-editsection"><
<p>Projetos para emular o Super Nintendo começaram com o lançamento do VSMC em 1994 e Super Pasofam
<p>Nintendo of America alega que o uso de ROMS e emuladores para o SNES é pirataria.</p>
<p>Emulação do SNES está agora disponível em aparelhos portateis, como celulares Android, Iphone e
<h2><span class="mw-headline" id="Era_32-bit">Era 32-bit</span><span class="mw-editsection"><span c
<p>Enquanto outras empresas estavam se movendo para sistemas de 32 bits, <a href="/wiki/Rare_Ltd."
<h2><span class="mw-headline" id="Vers.C3.B5es">Versões</span><span class="mw-editsection"><span cl
<table class="wikitable" align="right">
<tr>
<td>
<div class="floatnone"><a href="/wiki/Ficheiro:Super-Famicom-Console-Set.jpg" class="image" title="
</td>
<td>
```

**Figura 24 – Parte do conteúdo da página da Wikipedia**

**Fonte: https://pt.wikipedia.org/wiki/Super_Nintendo_Entertainment_System**

```
<!DOCTYPE html><!-- ServerInfo: BL2IDSLGN3B045 2015.11.18.19.58.06 Live1 Unknown LocVer
<!-- PreprocessInfo: BTSA007:RR1BLDF059,  - Version: 16,0,25956,0 -->
<!-- RequestLCID: 1033, Market:EN-US, PrefCountry: US, LangLCID: 1033, LangISO: EN -->
<html dir="ltr" lang="EN-US"><head><meta http-equiv="Content-Type" content="text/html; 
        <link rel="stylesheet" title="R3CSS" type="text/css" href="https://auth.gfx.ms/
a:hover{color:#338ED1;}
a:hover:active{color:#66AADC;}
:root input[type=button].default, :root input[type=submit].default{background-color:#00
:root input[type=button]:hover.default, :root input[type=submit]:hover.default, :root bu
</style><style type="text/css">body{display:none;}</style><script type="text/javascript"
  var g_dtFirstByte=new Date();var g_objPageMode = null;</script><link rel="image_src" h
<script type="text/javascript">var ServerData = {Be:'https://auth.gfx.ms/16.000.25956.00
aY:'87f7d5b732414410a290b00f7ac31a3e',aZ:'',o:'',AW:'https://sc.imp.live.com/content/dar
z:{'Logo':'','LogoAltText':'','LogoText':'','ShowWLHeader':true},a0:'http://mail.live.co
A1:false,Ag:2,BF:'https://auth.gfx.ms/16.000.25956.00/Microsoft_Logotype_Gray.png',ak:tr
'Passport',P:'https://login.live.com/cookiesDisabled.srf?mkt=EN-US&lc=1033',A7:'',Q:true
aE:'',aF:2,Ax:'https://login.live.com/login.srf?wa=wsignin1.0&rpsnv=12&ct=1448055502&rve
wreply=https:%2F%2Fmail.live.com%2Fdefault.aspx%3Frru%3Dinbox&id=64855&cbcxt=mai&mkt=EN-
  <script type="text/javascript" src="https://auth.gfx.ms/16.000.25956.00/Login_Core.js"
  <script type="text/javascript">SRSRetry("__Login_Strings", "https://auth.gfx.ms/16.000
<body onload="evt_Login_onload(event);" uiTheme="Web">
</body></html>
```

**Figura 25 – Parte do conteúdo da página inicial do Hotmail**

**Fonte: http://www.hotmail.com/**

```
        [1.16] => stdClass Object
            (
                [0] => jog
                [1] => gam
            )

        [1.21] => stdClass Object
            (
                [0] => consol
            )

        [1.35] => stdClass Object
            (
                [0] => -
            )

        [3.03] => stdClass Object
            (
                [0] => sup
            )

        [3.07] => stdClass Object
            (
                [0] => nint
            )

    )

)
```

**Figura 26 – Resultado do processamento da página da wikipedia sobre Super Nintendo**

**Fonte: (AUTOR, 2015)**

O processador também apresenta um resultado negativo quando analisa sites que utilizam a tecnologia flash (apresentam o seu conteúdo através imagens e vídeos). O site http://www.martinanderle.de/ é um exemplo e neste caso também não foi retornado nada. A figura 27 mostra o código da página inicial do site, onde é possível perceber a ausência de textos:

```
    {
        background-color:#000000;
        height: 100%;
        margin: 0;
        padding: 0;
    }
</style>
<script type="text/javascript">
        var flashvars = {};
        var params = {menu:'true', allowFullScreen:'true',wmode:"direct"};
        var attributes = {id: 'alternative',bgcolor:'#000000'};
        swfobject.embedSWF("Main.swf?201202290415", "alternative", "100%", "100%", "10.1.0", "js/ex

    <!--
    function resizeBrowser() {}

    function track(str)
    {
        //alert(str);
        //pageTracker._trackPageview(str);
    }
    //-->
</script>
</head>
<body onload="resizeBrowser()">
    <script type="text/javascript">
    var pageTracker = _gat._getTracker("UA-2065410-1");
    </script>
    <div id="alternative">
        <a href="http://www.adobe.com/go/getflashplayer">
            <img src="http://www.adobe.com/images/shared/download_buttons/get_flash_player.gif" alt
        </a>
    </div>
```

**Figura 27 – Exemplo de site desenvolvido em flash**

**Fonte: http://www.martinanderle.de/**

A seguir são listadas algumas observações feitas sobre os resultados dos testes do processador textual:

- Quando são analisados textos escritos de forma clara e objetiva, o processador apresenta um resultado dentro do esperado.

- Quando o assunto do texto não é muito claro, o processador não apresenta um bom resultado, muitas vezes retornando palavras sem um significado para que se consiga saber do que o texto se trata.

- Para a análise do conteúdo de web sites, o processador retorna um resultado satisfatório para páginas contendo bastante texto, como é o caso de blogs, a wikipedia, reportagens, entre outros.

- Sites que possuem muitas fotos, pouco texto objetivo, vídeos, como é o caso do youtube, ou o site do google maps, por exemplo, o processador não apresenta um bom resultado, retornando palavras aleatórias ou em alguns casos sem retorno algum, provavelmente porque o conteúdo analisado não contém textos suficientes ou as palavras não se repetem com frequência.

## 6.2. Testes da integração com a aplicação MaRE

Primeiramente, a figura 28 apresenta o esquema de integração entre o aplicativo MaRE com o processador textual. As etapas circuladas destacam a colaboração do presente trabalho com o agente inteligente do MaRE.

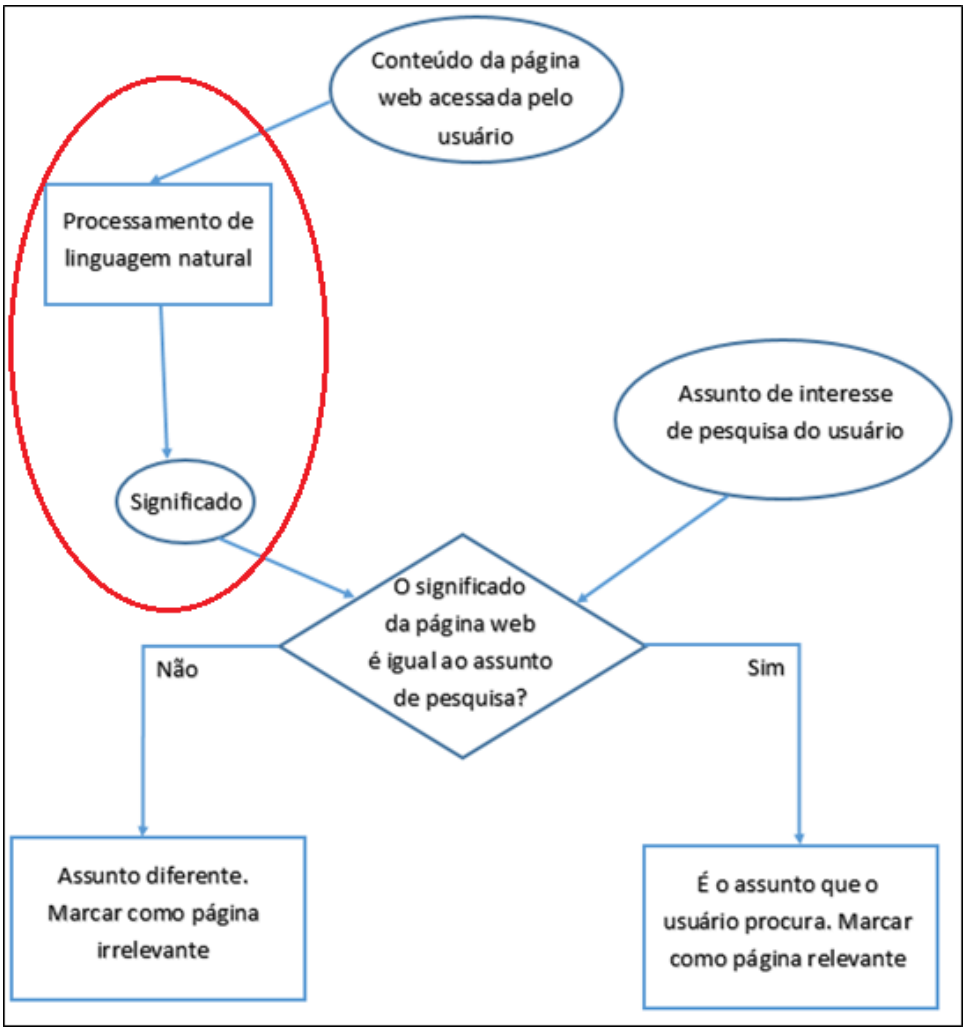


**Figura 28 – Esquema de integração**

**Fonte: (AUTOR, 2015)**

No desenvolvimento do algoritmo de integração, foi feita a alteração na classe Agente, especificamente no método que corresponde ao comportamento VerificaConteudoPagina da aplicação MaRE. As figuras 29 e 30 mostram como era este método antes e como ficou depois da alteração.

```
private bool VerificaConteudoPagina()
{
    string urlAddress = BrowserInstance.Url.AbsoluteUri;

    List<string> contents = null;

    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(urlAddress);
    request.Method = "GET";
    using (var response = request.GetResponse())
    using (var stream = response.GetResponseStream())
    using (var reader = new StreamReader(stream))
    {
        HttpStatusCode statusCode = ((HttpWebResponse)response).StatusCode;
        contents = new List<string>(reader.ReadToEnd().Split(' '));
        contents.Sort();
    }

    bool contemSimilaridade = false;

    foreach (var palavraChave in AppGlobalData.LstPalavrasChave)
    {
        contemSimilaridade = contents.Where(o => o.ToUpper().Contains(palavraChave.ToUpper())).Count() > 0;

        if (contemSimilaridade)
            break;
    }

    return contemSimilaridade;
}
```

**Figura 29 – Método de verificação de conteúdo antes da alteração**

**Fonte: TEIXEIRA, 2014**

```
private bool VerificaConteudoPagina()
{
    string strHtml = "";
    string strKeyWords = "";
    List<string> stems = new List<string>();
    List<string> keyWords = new List<string>();

    string urlAddress = BrowserInstance.Url.AbsoluteUri;
    List<string> contents = null;
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(urlAddress);
    request.Method = "GET";
    using (var response = request.GetResponse())
    using (var stream = response.GetResponseStream())
    using (var reader = new StreamReader(stream))
    {
        strHtml = reader.ReadToEnd();
    }

    contents = new List<string>(strHtml.Split(' '));
    contents.Sort();

    ServiceReference1.textProcessorSoapClient x = new ServiceReference1.textProcessorSoapClient();

    var json = x.getTextClassification(strHtml, 2, 5);
    var result = JsonConvert.DeserializeObject<dynamic>(json);

    var limportantGroup = result.little_important;
    var relevantGroup = result.relevant;
    foreach (var n3 in limportantGroup)
    {
        foreach (var n2 in n3)
        {
            foreach (var n1 in n2)
            {
                foreach (var n0 in n1)
                {
                    string stem = n0;
                    stems.Add(stem);
                }
            }
        }
    }
    foreach (var n3 in relevantGroup)
    {
        foreach (var n2 in n3)
        {
            foreach (var n1 in n2)
            {
                foreach (var n0 in n1)
                {
                    string stem = n0;
                    stems.Add(stem);
                }
            }
        }
    }
```

```
    bool contemSimilaridade = false;
    if (stems.Count() > 0) {
        // Com PLN
        foreach (var palavraChave in AppGlobalData.LstPalavrasChave)
        {
            strKeyWords = strKeyWords + " " + palavraChave.ToLower();
        }

        json = x.getTextStems(strKeyWords);
        result = JsonConvert.DeserializeObject<dynamic>(json);

        foreach (var n3 in result)
        {
            foreach (var n2 in n3)
            {
                foreach (var n1 in n2)
                {
                    foreach (var n0 in n1)
                    {
                        string stem = n0;
                        keyWords.Add(stem);
                    }
                }
            }
        }

        foreach (var key in keyWords)
        {
            contemSimilaridade = stems.Contains(key);
            if (contemSimilaridade)
                break;
        }
    else {
        // Sem PLN
        foreach (var palavraChave in AppGlobalData.LstPalavrasChave)
        {
            contemSimilaridade = contents.Where(o => o.ToUpper().Contains(palavraChave.ToUpper())).Count() > 0;

            if (contemSimilaridade)
                break;
        }
    }

    return contemSimilaridade;
}
```

**Figura 30 - Método de verificação de conteúdo depois da alteração**

**Fonte: TEIXEIRA, 2014**

Detalhes de como ocorreu a integração são listados passo a passo a seguir:

1. O conteúdo da página acessada é processado através da comunicação com o web *service*.

2. O retorno é transformado em um *array* e o grupo irrelevante é ignorado, evitando assim que o programa sinalize de forma equivocada uma página como sendo relevante, quando na verdade poderia não ser.

3. Os retornos dos grupos pouco importante e relevante são salvos em um novo *array*, contendo assim o valor dos dois grupos.

4. É verificado se o novo *array* possui resultados. Se sim, então é aplicado o processo de *stemming* (através do método getTextStems da classe textProcessor) nas palavras-

chave (para corrigir palavras escritas erradas ou agrupar palavras semelhantes) e verificado se alguma destas palavras está contida no novo *array*.

5. Caso o novo *array* não possua resultados (resultado do processamento ocorrido por causa de má padronização do conteúdo das páginas HTML), então é utilizada a antiga solução do MaRE, sem PLN. Esta alternância entre utilizar ou não a integração via web service é feita através de um comando de decisão.

6. Uma vez definido o tipo de comparação a ser feita (com PLN ou sem PLN), o método VerificaConteudoPagina retorna TRUE quando a página acessada é de relevância para o usuário, e FALSE em caso negativo.

Um detalhe importante a ser pontuado, é que no acesso da primeira página após iniciar uma pesquisa, a aplicação demora um pouco para responder, justamente pela troca de informações com o web service, mas para as pesquisas posteriores o sistema passa a responder no tempo normal.

Com os testes efetuados, foi possível perceber uma clara evolução na interação do agente com o usuário, podendo destacar especificamente duas melhorias. A primeira delas foi que o processador textual veio a incrementar a aplicação MaRE na questão de eliminar resultados equivocados quanto ao resultado da relevância, com a eliminação da comparação com o grupo irrelevante, que por sua vez retorna palavras sem muito significado para o texto. A segunda melhoria foi a limpeza dos textos invisíveis (tags HTML, código javascript e estilo CSS) do conteúdo da página acessada, o que eliminou o risco de serem feitas comparações errôneas das palavras-chave do usuário com algum destes textos.

Por exemplo, ao iniciar uma pesquisa com a palavra-chave CONSOLE (de vídeo game) e acessar a página https://www.wikipedia.org/, o MaRE sem PLN sinalizou como sendo de relevância para o usuário, quando na verdade não é. Neste caso o que aconteceu foi que no conteúdo da página existe um comando em javascript chamado de CONSOLE. Quando foi feita a mesma pesquisa, mas integrando o MaRE com o processador textual, o comportamento do agente foi correto, não marcando a página como relevante. A figura 31 apresenta a reação do agente, exemplificando este comportamento:

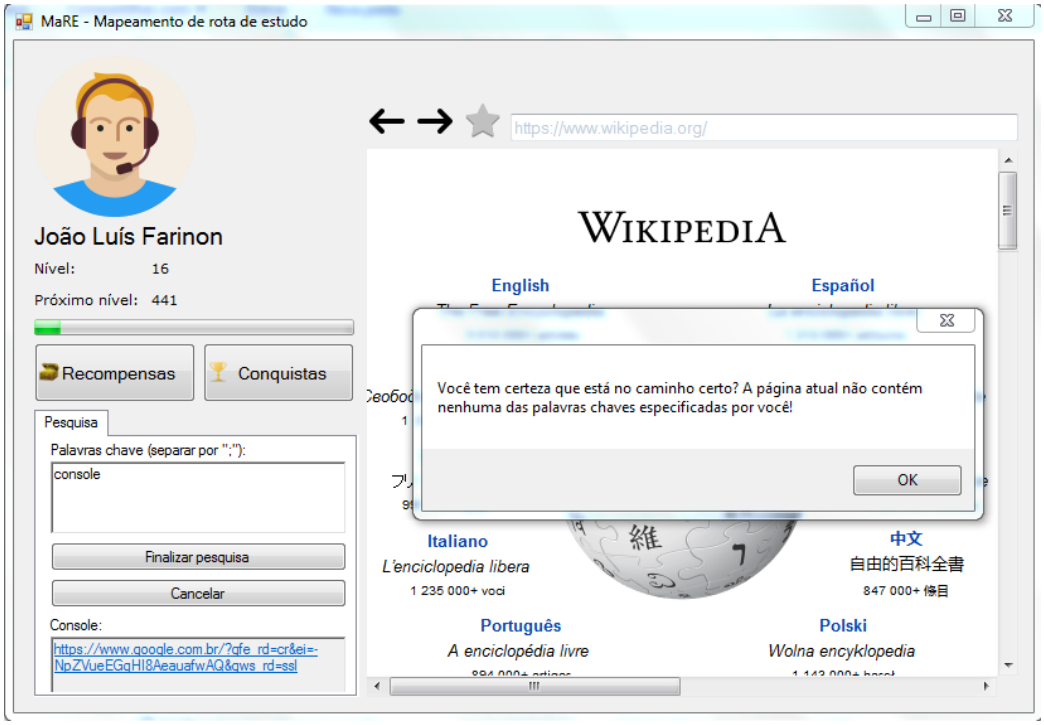


**Figura 31 – Reação do agente quando página visitada não é relevante**

**Fonte: TEIXEIRA, 2014**

## 6.3. Considerações

Este capítulo mostrou o resultado dos testes efetuados com o processador textual desenvolvido e a integração com a aplicação MaRE. A integração do sistema MaRE com a classe textProcessor ocorreu sem maiores problemas, funcionando dentro do esperado, de forma que o agente responde de forma mais adequada em situações onde é necessário limpar os textos invisíveis do conteúdo da página acessada, por exemplo.

Porém, algumas observações são destacadas a seguir:

- Quando analisados textos complexos, o processador textual retorna palavras que por si só não conseguem transmitir a ideia principal do texto.

- Em alguns casos quando o conteúdo analisado, principalmente das páginas web, é muito poluído e sem textos aparentes, o sistema retorna um conjunto vazio. Na integração com o MaRE, quando isso acontece, então não é utilizado o processador textual desenvolvido, e sim a forma antiga de comparação entre o conteúdo e as palavras-chave.

# 7. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho de conclusão abordou o tema PLN, apresentando o conceito, etapas, tipos de abordagens e as utilidades e vantagens do processamento da linguagem em conteúdo textual. Dentre os desafios desta área está conseguir extrair significados a partir de textos escritos em linguagem natural. A dificuldade de analisar de maneira correta a relação entre palavras e o contexto que estão inseridas é bastante desafiador, somado com problemas de variações morfológicas e ambiguidades que as línguas possuem, em especial a portuguesa. Embora soluções completas de PLN ainda sejam um desafio, abordagens mais simples, baseadas em estatísticas servem como possibilidades para PLN, podendo ser chamado assim de uma solução de PLN estatístico (MORAIS, AMBRÓSIO, 2007).

Também foram apresentados ao longo deste trabalho, pesquisas relacionadas que fazem o uso de PLN, tais como: Evi, CoGroo, ALICE, Ogmma, Sistema de Recomendação para Apoio à Análise Fundamentalista do Mercado de Capitais Utilizando Processamento de Linguagem Natural e DOSVOX.

Outro estudo de destaque foi o trabalho Mapeamento de Rota de Estudo na Web Baseada na Estratégia de Gamificação (MaRE), onde o presente trabalho deu a sua contribuição ao agente inteligente aprimorando o comportamento que verifica o conteúdo da página web.

Como solução PLN foi desenvolvido um processador textual abordado no capítulo 5. Tal processador realiza a contagem da frequência absoluta de cada palavra, e divide pelo total de palavras contidas no texto analisado, posteriormente classificando-as em grupos de relevância. Este processador foi integrado, através de um web service, com a aplicação MaRE.

Através dos testes efetuados, foi possível concluir que o agente inteligente da aplicação MaRE teve uma melhora significativa em sua tarefa. Com a integração do processador textual, a comparação das palavras-chave do usuário com o conteúdo das páginas web passou a ter um maior aprimoramento, principalmente com a eliminação dos termos menos significativos e com a limpeza dos textos invisíveis do conteúdo analisado. Desta forma, o agente teve um comportamento mais adequado, por exemplo, isto foi

percebido em situações em que o agente informa para o usuário quando um site visitado não é de relevância para o objetivo da sua pesquisa.

Para os trabalhos futuros, sugere-se incrementar o processo de limpeza de textos invisíveis que fazem parte do conteúdo de páginas web. Atualmente, quando este conteúdo é muito confuso e sem padronização, o processador textual não se comporta como deveria, retornando para o usuário um *array* vazio.

Além de analisar textos, uma possibilidade de complementação deste trabalho é fazer com que o processador interprete imagens e vídeos, como por exemplo o site do youtube, que tem seu conteúdo formado basicamente por vídeos ou sites que fazem uso da tecnologia flash.

Outra possibilidade de incremento é utilizar, em conjunto com PLN estatístico, uma solução semântica. Uma forma de utilização seria encontrar sinônimos para as palavras classificadas como pouco importante e relevante. Utilizando como exemplo o MaRE, caso o usuário iniciasse uma pesquisa procurando pela palavra-chave PRATICAR, e navegasse por uma página web que tivesse a palavra ENSAIAR ou TREINAR, então neste caso a aplicação iria marcar como conteúdo relevante para o usuário.

Para obter melhor desempenho na aplicação MaRE, considera-se mais adequando implementar a solução desenvolvida de forma nativa em C#. Desta forma não seria preciso depender de uma comunicação externa, que é o caso do consumo do web service, e nem do PHP, eliminando dessa forma dependências que podem ser desagradáveis para o usuário, por precisar de instalações extras para o aplicativo funcionar.

## 8. REFERÊNCIAS


1. ALICE. *Artificial Linguistic Internet Computer Entity.* Disponível em: <http://alice.pandorabots.com/>. Acessado em: 05 de abril de 2015.

2. ALLEN, J. *Natural language understanding*. 2. ed. Pearson Education, 1995.

3. ALLEN, J. *Natural Language Processing for Information Retrieval*. Seattle, Washington: tutorial apresentado em NAACL/ANLP Language Technology Joint Conference, 2000.

4. BATISTA, T. M. *Corretor Gramatical para o emacs*. 2010. Disponível em: <http://www.ime.usp.br/~cef/mac499-10/monografias/thiago/monografia/monografia.pdf>. Acessado em: 21 de março de 2015.

5. BERBER SARDINHA, T. *Linguística de corpus: histórico e problemática*. 2000. Revista D.E.L.T.A., vol. 16, n. 2.

6. BOD, R. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Tese de Doutorado. Institute for Logic, Language and Computation (ILLC), University of Amsterdam. Holanda: Academichepers, 1995.

7. CARVALHAL, C. A. M. *Desenvolvimento e aplicação de agentes Web inteligentes para pesquisa de publicações científicas nas áreas da biomedicina e bioinformática*. Instituto politécnico de Bragança, 2012.

8. CMU SPHINX. *Open source speech recognition*. Disponível em: <http://cmusphinx.sourceforge.net/>. Acessado em: 20 de março de 2015.

9. COGROO. *Corretor Gramatical acoplável ao LibreOffice*. Disponível em: <http://cogroo.sourceforge.net/>. Acessado em: 21 de março de 2015.

10. COULON, Daniel de; KAYSER, Daniel. *Informática e linguagem natural: uma visão geral dos métodos de interpretação de textos escritos*. Brasília: IBICT, 1992. 96 p.

11. DONG, A. *The Language of Design*. Theory and Computation. Springer, 2008.

12. DOSVOX. *Projeto DOSVOX*. Disponível em: <http://intervox.nce.ufrj.br/dosvox>. Acessado em: 20 de março de 2015.

13. EBECKEN, Nelson Francisco Favilla; LOPES, Maria Celia Santos; COSTA, Myrian Christina de Aragão. *Mineração de Textos*. Sistemas Inteligentes: fundamentos e aplicações, 2003.

14. EVI. *Ask me anything.* Disponível em: <https://www.evi.com/>. Acessado em: 05 de abril de 2015.

15. FAYYAD, U.; PIATESKI-SHAPIRO, G.; SMYTH, P. *The KDD Process for Extracting Useful Knowledge from Volumes of Data*. In: Communications of the ACM, p. 27-34, Nov.1996.

16. FRANTZ, V.; SHAPIRO, J.; VOISKUNSKII, V. *Automated Information Retrieval: Theory and Methods*. San Diego, CA: Academic Press, 1997. 365 p.

17. GASPERIN, V. C.; LIMA, V. L. S. *Fundamentos do Processamento Estatístico da Linguagem Natural*. PUCRS, faculdade de informática, 2001.

18. GONZALEZ, M.; LIMA, V. *Recuperação de Informação e Processamento da Linguagem Natural*. PUCRS, faculdade de informática, 2003. Disponível em: <https://www.inf.pucrs.br/~gonzalez/docs/minicurso-jaia2003.pdf>. Acessado em 21 de março de 2015.

19. I.L.A. *Voice Assistant*. Disponível em: <https://sites.google.com/site/ilavoiceassistant>. Acessado em: 20 de março de 2015.

20. KAO, Anne; POTEET, Stephen R. *Report on KDD conference 2004 panel discussion can natural language processing help text mining?* SIGKDD Explorations, 6(2):132–133, 2004.

21. KYTEA. *Kyoto Text Analysis Toolkit*. Disponível em: <http://www.phontron.com/kytea>. Acessado em: 20 de março de 2015.

22. MAIA, L. C. G. *OGMA, ferramenta para análise de textos*. 2008. Disponível em: <http://www.luizmaia.com.br/ogma/>. Acessado em: 09 de abril de 2015.

23. MEDEIROS CASELI, H. *Tradução automática*. Disponível em: <http://www.bv.fapesp.br/pt/auxilios/24244/portal-de-traducao-automatica-recursos-e-ferramentas-para-o-portugues-do-brasil/>. Acessado em: 20 de março de 2015.

24. MITTMANN, A. *Implementação do chatterbot ELIZA na linguagem multiparadigma Oz*. Universidade Federal de Santa Catarina. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_304/TCC-Adiel.pdf>. Acessado em: 25 de março de 2015.

25. MORAIS, E. A. M.; AMBRÓSIO, A. P. L. *Mineração de Textos*. Universidade Federal de Goiás. 2007. Disponível em: <http://www.portal.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_005-07.pdf>. Acessado em: 22 de junho de 2015.

26. MOSES. *Statistical machine translation system*. Disponível em: <http://www.statmt.org/moses/>. Acessado em: 20 de março de 2015.

27. MULLER, D. N. *Processamento de Linguagem Natural*. 2003. Relatório para a disciplina CMP187 - Mentes e Máquinas. Universidade Federal do Rio Grande do Sul. Disponível em <http://www.inf.ufrgs.br/~danielnm/docs/pln.pdf>. Acessado em: 10 de março de 2015.

28. MULLER, T. A. *Sistema de Recomendação para Apoio à Análise Fundamentalista do Mercado de Capitais Utilizando Processamento de Linguagem Natural*. 2011. Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade de Santa Cruz do Sul.

29. NAGAO, Makoto. *Natural Language Processing (NLP).* LBS kuttipedia. Disponível em <https://lbsitbytes2010.wordpress.com/2013/03/27/scientist1-makoto-nagao-roll-no6/>. Acessado em: 15 de abril de 2015.

30. NILSSON, Nils J. *Artificial intelligence : a new synthesis*. San Francisco: M. Kaufmann, 1998. 513 p.

31. OTHERO, Gabriel de Ávila; MENUZZI, Sérgio de Moura. *Linguística computacional: teoria & prática*. São Paulo: Parábola, c2005. 126 p.

32. PORTER, M. *Snowball Algorithm.* Disponível em: <http://snowball.tartarus.org/credits.php>. Acessado em 21 de setembro de 2015.

33. PROFESSEURS. *Processamento de Linguagem Natural*. Disponível em: <http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/IA/PLN/pln.html>. Acessado em: 10 de março de 2015.

34. PUCRS. *Processamento de Linguagem Natural*. Disponível em: <https://www.inf.pucrs.br/linatural/>. Acessado em: 10 de março de 2015.

35. RABUSKE, Renato Antônio. *Inteligência artificial*. Florianópolis: Ed. da UFSC, 1995. 240 p.

36. REZENDE, Solange Oliveira (Org.). *Sistemas inteligentes: fundamentos e aplicações.* Barueri: Manole, 2003. 525 p.

37. RICH, Elaine; KNIGHT, Kevin. *Inteligência artificial*. 2. ed. São Paulo: Makron Books do Brasil, c1994. 722 p.

38. RUSSELL, Stuart J.; NORVIG, Peter. *Inteligência artificial*. 3ed., Rio de Janeiro: Elsevier, 2013.

39. SETZER, V. W. *Alan Turing e a Ciência da Computação*. Departamento de Ciência da Computação da USP. Disponível em: <http://www.ime.usp.br/~vwsetzer/Turing-teatro.html>. Acessado em: 25 de março de 2015.

40. STRZALKOWSKI, T. *Natural Language Information Retrieval*. Kluwer Academic Publishers, 1999. 385 p.

41. TAGNIN, Stella E. O.; VALE, Oto Araújo (Org.). *Avanços da linguística de corpus no Brasil*. São Paulo: Humanitas, 2008. 437 p.

42. TEIXEIRA, Kelvin S. *Mapeamento de Rota de Estudo na Web Baseada na Estratégia de Gamificação.* 2014. Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade de Santa Cruz do Sul.

43. UFBA. *Processamento de Linguagem Natural*. Disponível em: <http://homes.dcc.ufba.br/~leotavo/index.html/artigo2.pdf>. Acessado em 20 de março de 2015.

44. VIEIRA, R. *Linguística computacional: fazendo uso do conhecimento da língua*. Entrelinhas, ano 2, n. 4, São Leopoldo: UNISINOS, 2002.

45. VIEIRA, R.; LIMA, V. L. S. *Linguística computacional: princípios e aplicações*. In: IX Escola de Informática da SBC-Sul. Luciana Nedel (Ed.) Passo Fundo, Maringá, São José. SBC-Sul, 2001.

46. VIEIRA, R.; LOPES, L. *Processamento de linguagem natural e o tratamento computacional de linguagens científicas.* Linguagens especializadas em corpora, modos de dizer e interfaces de pesquisa. PUCRS, p. 183-201, 2010. Disponível em: <http://www.pucrs.br/edipucrs/linguagensespecializadasemcorpora.pdf>. Acessado em 16 de dezembro de 2015.

47. VOORHEES, E. M. Natural Language Processing and Information Retrieval. In: Information Extraction: towards scalable, adaptable systems. Lecture notes in Artificial Intelligence, N.1714, 1999. p.32-48.

48. WIVES, L. *Tecnologias de descoberta de conhecimento em textos aplicadas à inteligência competitiva*. Universidade Federal do Rio Grande do Sul. 2002. Disponível em <http://www.leandro.wives.nom.br/pt-br/publicacoes/eq.pdf>. Acessado em: 22 de junho de 2015.