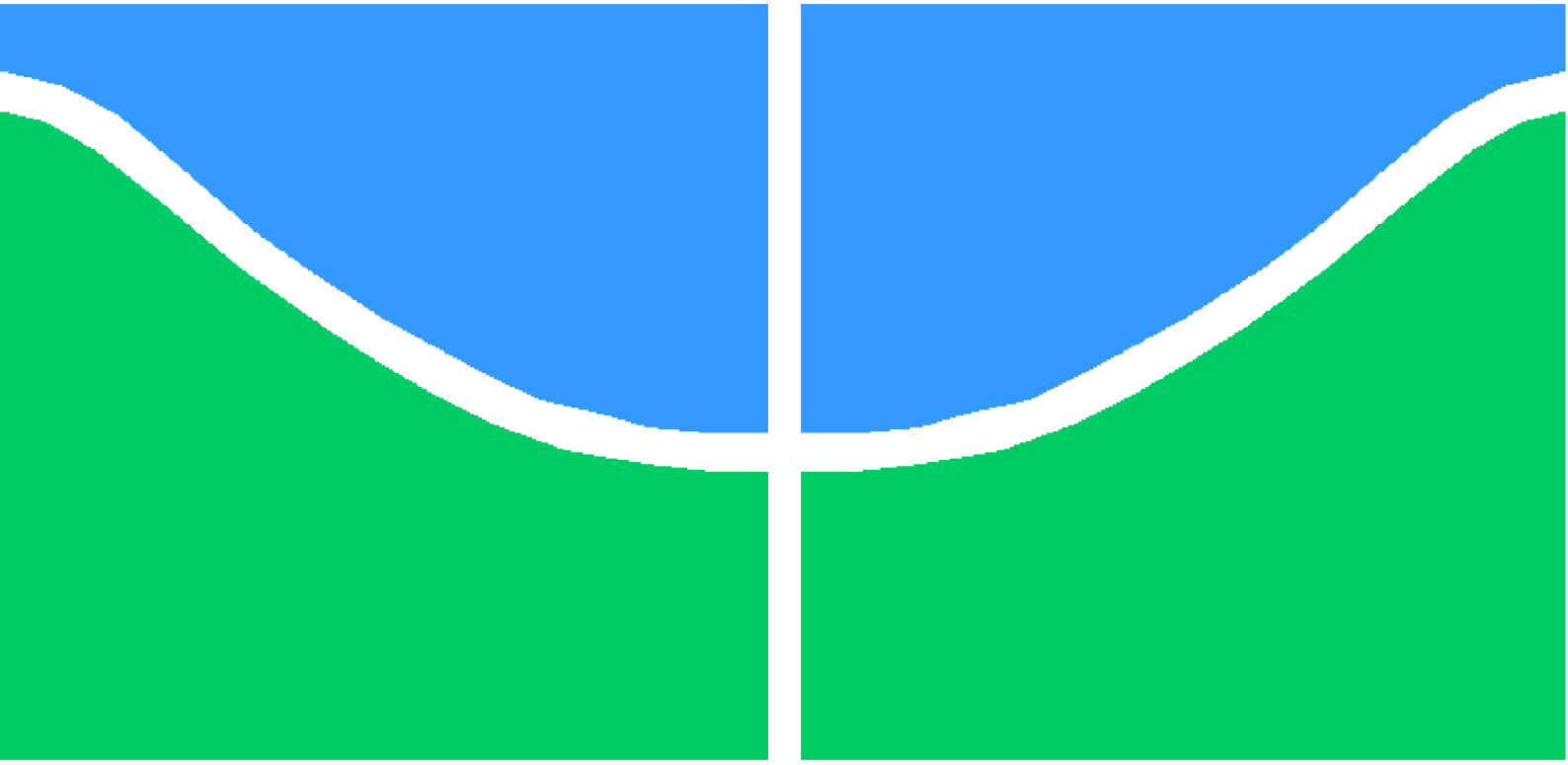


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de software

Aplicação de processamento de linguagem natural: Uma ferramenta de apoio à correção de questões dissertativas

Autor: Cristóvão de Lima Frinhani

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2016

Cristóvão de Lima Frinhani

Aplicação de processamento de linguagem natural: Uma ferramenta de apoio à correção de questões dissertativas

Monografia submetida ao curso de graduação em (Engenharia de software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Coorientador: Mestre Maurício Vidotti Fernandes

Brasília, DF

2016

Cristóvão de Lima Frinhani

Aplicação de processamento de linguagem natural: Uma ferramenta de apoio à correção de questões dissertativas/ Cristóvão de Lima Frinhani. – Brasília, DF, 2016-

85 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Processamento de linguagem natural. 2. Algoritmo genético. I. Professor Doutor Sérgio Antônio Andrade de Freitas. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicação de processamento de linguagem natural: Uma ferramenta de apoio à correção de questões dissertativas

CDU 02:141:005.6

Cristóvão de Lima Frinhani

Aplicação de processamento de linguagem natural: Uma ferramenta de apoio à correção de questões dissertativas

Monografia submetida ao curso de graduação em (Engenharia de software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de software).

Trabalho aprovado. Brasília, DF, 01 de julho de 2016:

Professor Doutor Sérgio Antônio Andrade de Freitas
Orientador

Professora Doutora Carla Silva Rocha Aguiar
Convidado 1

Professor Doutor Edgard Costa Oliveira
Convidado 2

Brasília, DF
2016

Este trabalho é dedicado a todos que,
um dia já finalizaram uma monografia.

Agradecimentos

Agradeço primeiramente a Deus, por está vivo até hoje, agradeço aos meus pais, minha mãe Sebastiana, ao meu pai Leandro, por sempre me apoiarem, e desde a simplicidade nunca deixaram me faltar nada, me ensinado sempre a ser uma pessoa digna, agradeço ao meu irmão Fábio, que agora também segue seu caminho na graduação, por me apoiar e está comigo em algumas das noites em claro de estudos, agradeço a minha namorada Anacarina, por está comigo em meus momentos de lazer e descanso necessários para fazer esse TCC, agradeço aos meus amigos presentes da FGA que estão comigo por todo esse caminho, agradeço ao meu professor orientador Sérgio por me guiar nesse tema de TCC, agradeço ao meu coorientador Maurício por todas as longas conversas de fim de tarde, agradeço a todos os professores da FGA, por sempre fazerem o possível por uma faculdade melhor.

Sapere aude

Resumo

O ensino a distância é uma modalidade de ensino que tem crescido. Porém tem encontrado barreiras em sua escalabilidade. A correção de questões dissertativas normalmente leva um maior tempo que questões de certo ou errado e múltiplas escolhas. Buscando auxiliar o tutor em seu trabalho de correção de questões dissertativas, se pensou em um sistema de apoio de pré-avaliação.

Um sistema onde um tutor pode cadastrar uma questão e um gabarito, e os alunos responderem a essa questão, automaticamente o sistema processa as respostas dos alunos, e uma nota preliminar é mostrada ao tutor, podendo aprovar ou não a nota.

Para isso, foi estudado o processamento de linguagem natural, onde a aplicação de similaridade semântica pode quantificar uma resposta com um gabarito. Para minimizar o erro da quantificação, foi estudado o aprendizado de máquina, para se adquirir conhecimento com a interação do tutor.

O sistema desenvolvido se divide em duas partes principais, a primeira sendo a comparação entre o gabarito e uma resposta, onde se usa os algoritmos de similaridade semântica, dando uma nota a resposta, e a segunda a interface web, que permite o cadastro das questões, responder questões e avaliar respostas. A avaliação das respostas está diretamente ligada ao aprendizado do sistema, com o *feedback* do tutor, o aprendizado de máquina pode analisar os elementos que diferiram e ajustar a nota.

Para testar o sistema, foi utilizado a disciplina de fundamentos de arquitetura de computadores, tendo o objetivo de avaliar a diferença da nota dada pelo sistema e autor da questão.

Os testes apresentaram uma avaliação inicial do sistema, demostrando sua capacidade de dar notas significativas para as respostas. O aprendizado de máquina demostrou, inicialmente, uma melhoria dos valores dados as notas das respostas.

Palavras-chaves: Similaridade semântica. Processamento de linguagem natural. Aprendizado de máquina. Algoritmo genético. Questões dissertativas. Apoio a correção.

Abstract

Distance education is a teaching method that has grown. But have found barriers in its scalability. Seeking assist the tutor in his work of correction of essay questions, was considered a pre-assessment support system.

A system where a tutor can register an question and a template, and students answer this question, the system automatically processes student responses, and a preliminary note is shown to tutor, the tutor can accept or not the note.

For this we was studied the natural language processing, the application of semantic similarity can quantify an answer with a template. To minimize the quantification error, was studied machine learning, to gain knowledge on the interaction with the tutor.

The developed system is divided into two main parts, the first being the comparison between the template and response, which use the semantic similarity algorithm, giving a note the answer, and the second the web interface, that allows the registration of the issues, answer questions and evaluate responses. The evaluation of responses is directly connected to the system of learning, with tutor feedback, the machine learning can analyze the elements that differed and adjust the note.

To test the system, we used the discipline of computer architecture basics, with the objective of evaluating the difference of note given by the system and asker.

The tests presented an initial assessment of the system, demonstrating its ability to make significant scores for the answers. The machine learning initially demonstrated an improvement of the values given the notes of the answers.

Key-words: Semantic Similarity. Natural language processing. Machine learning. Genetic algorithm. Essay questions. Support for correction.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PLN	Processamento de linguagem natural
LSA	*Latent Semantic Analysis*
AVA	Ambiente virtual de aprendizado
MVC	*Model View Controller*
EAD	Educação a distância
NLTK	*Natural Language Toolkit*
ISA	*Is a kind of*
URL	*Uniform Resource Locator*
IP	*Internet Protocol*
FAC	Fundamentos de arquitetura de computadores

Lista de símbolos

$\sum$	Letra grega sigma
$=$	Igualdade
δ	Letra grega minúscula delta
$\|$	Norma
$\approx$	Aproximadamente igual
$\neq$	Diferente

Sumário

1 INTRODUÇÃO

Com o avanço das tecnologias, os métodos de ensino e aprendizagem se modificaram, cada vez mais é necessário se atualizar para se manter dentro do contexto competitivo e social da sociedade, não sendo isso mais uma exclusividade dos jovens(PEREIRA; SCHMITT; DIAS, 2007).

Com isso os Ambientes Virtuais de Aprendizagem (AVAs), estão sendo cada vez mais utilizados pela academia e corporações como uma opção tecnológica para suprir a demanda educacional(PEREIRA; SCHMITT; DIAS, 2007).

O Moodle, um AVA, possui o recurso de questionário que permite o professor aplicar questionários nesses ambientes, em sua maioria a correção desses é automatizada, como os tipos verdadeiro ou falso e de múltipla escolha, questões dissertativas não são corrigidas automaticamente(FILHO, 2009).

	Ano Declarado																
	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008*
Nº de Cursos	1	1	1	3	6	9	6	10	14	9	10	22	45	78	194	320	8

Figura 1 – Relação de número de cursos à distância (virtuais) por ano no Brasil. Fonte:(ABRAEAD, 2008)

Esse crescimento pode ser visto na figura 1 do Anuário Brasileiro Estatístico de Educação Aberta e a Distância (ABRAEAD, 2008), no ano de 2007 com total de 320 cursos oferecidos à distância.

Foram escritos dois artigos desse trabalho de conclusão de curso, sendo visto a primeira página de cada uma deles nos apêndices A e B.

1.1 Objetivos

São apresentados nesta seção os objetivos gerais, objetivos específicos e a questão de pesquisa.

1.1.1 Objetivos Gerais

Desenvolver um sistema de apoio a correção de questões dissertativas, utilizando técnicas de processamento de linguagem natural, para a similaridade dos textos e *Machine Learning* para criar um fator de aprendizado que melhore sua eficiência com as interações.

1.1.2 Objetivos específicos

Os objetivos deste trabalho são:

- Criar uma sintaxe para a definição dos gabaritos, os gabaritos serão usados para o calculo da similaridade.
- Construir um sistema capaz de fazer pré-avaliações de respostas de questões dissertativas dando uma nota.
- Construir um sistema capaz de usar *Machine Learning* para minimizar o erro da valoração das respostas dissertativas.
- Avaliar a eficiência do sistema implantado.

1.1.3 Questão de pesquisa

A questão que esse trabalho tenta responder é: É possível criar um sistema de apoio que pré-avalie questões dissertativas, que guie o tutor na correção, e ao mesmo tempo aprenda com essa interação, para sua melhoria?

1.2 Motivação

Diversos fatores despertam a motivação deste trabalho, sendo alguns deles de curiosidade e estudo, e outros para auxiliar tutores e estudantes em seu caminho de aprendizado. Posso descrever que são os principais motivos:

- Entendimento do processamento de linguagens naturais.
- Entendimento do processo de aprendizado de maquina.
- Auxiliar tutores na correção de respostas dissertativas.
- Ajudar os alunos a receberem suas notas em menor tempo.

1.3 Metodologia

Esta seção apresenta a metodologia que será usada no desenvolvimento deste trabalho.

1.3.1 Classificação da pesquisa

Segundo (GIL, 2008) com base nos objetivos, uma pesquisa é classificada como exploratório, descritiva ou explicativa.

A pesquisa exploratória tem como objetivo a familiaridade com o problema, tornando ele mais explicativo ou construindo hipóteses, sendo flexível em seu planejamento, assume na maioria dos casos a forma de pesquisa bibliográfica ou de estudo de caso(GIL, 2008).

Com base nos objetivos do trabalho, ele se classifica como exploratório.

1.3.2 Referencial teórico

Para a pesquisa de referencial teórico de similaridade semântica e algoritmos genéticos, serão pesquisados em livros e artigos, tanto por meio físico e digital. Os conceitos estudados serão sumariados e descritos no capítulo 3 de referencial teórico.

1.3.3 Proposta

Uma vez estudado o referencial teórico e tendo os objetivos definidos, foi proposto um escopo do sistema onde foi definido o passo a passo do processo que o sistema irá seguir para alcançar os objetivos.

1.3.4 Engenharia de software

Para a implementação serão consideradas as praticas da *framework Scrum*, criando um *backlog* de produto e dividindo seu desenvolvimento em *sprints*, para o desenvolvimento do sistema serão utilizados o paradigma de programação orientado a objetos e um padrão arquitetural MVC.

1.4 Estrutura da Monografia

Este trabalho possui 5 partes: Introdução, Referencial teórico, Proposta, Desenvolvimento da proposta e Conclusão e trabalhos futuros, no capítulo 1 é apresentada a introdução, no capítulo 2 é feito um estudo sobre o referencial teórico, no capítulo 3 é apresentada a proposta de sistema e no capítulo 4 é feito o levantamento necessário de engenharia de software para o desenvolvimento do sistema, no capítulo 5 é feito a conclusão e trabalhos futuros.

2 Referencial Teórico

Este capítulo tem como objetivo apresentar o referencial teórico que embasa a pesquisa deste trabalho, incluindo conceitos abordados, tais como o processamento de linguagem natural, algoritmos de similaridade e *Machine Learning*.

2.1 Processamento de Linguagem Natural

Com uma quantidade de páginas, a web se constitui hoje quase toda de linguagem natural, com isso um agente que deseja adquirir conhecimento tem que entender a ambiguidade e confusão da linguagem humana(RUSSELL; NORVIG, 2013).

As linguagens naturais não podem ser definidas como um conjunto de sentenças definidas. Com isso se torna melhor definir um modelo de linguagem natural usando probabilidade sobre sentenças em vez de um conjunto definido(RUSSELL; NORVIG, 2013).

Linguagens naturais são ambíguas, podemos tomar como exemplo o “Ele viu o banco”, onde pode significar que ele viu uma peça de mobília ou uma instituição financeira. Com isso não podemos falar de um significado e sim de uma probabilidade sobre o significado(RUSSELL; NORVIG, 2013).

O Processamento de Linguagem Natural (PLN) tal como construído por (JUNIOR, 2008), tem duas abordagens principais, são elas, a baseada em texto e a baseada em diálogo.

A abordagem baseada em texto tem como principal foco a busca de documentos, resumos e compreensão de textos(JUNIOR, 2008).

A abordagem baseada em diálogo tem crescimento com a interface entre máquinas e humanos(JUNIOR, 2008).

O Processamento de Linguagem Natural tem como principal etapa o pré-processamento, onde serão definidas, reconhecidas às sentenças e classificações das palavras a sua função sintática(JUNIOR, 2008).

2.2 Algoritmos de Similaridade

Nesta seção são descritos os algoritmos de similaridade semântica que foram encontrados nos artigos e livros de referência.

2.2.1 Pré-processamento

O pré-processamento é a etapa onde se pega um texto bruto e o transforma em uma estrutura para os algoritmos usarem, com o principal objetivo de aumentar a qualidade dos dados. Considerada à etapa mais onerosa do processo, por não contar com uma única técnica, mas sim por varias técnicas(JUNIOR, 2008).

Para a realização do processamento do texto pelos algoritmos existentes é necessário aplicações de algumas técnicas, as principais são:

2.2.1.1 *Tokenization*

Tokenization tem como objetivo, a partir de um texto de entrada, a extração de unidades mínimas do texto, sendo o primeiro processo a ser feito no texto de entrada. Cada unidade é chamada de *token*, que normalmente representam uma palavra no texto, mas podem também estar relacionados a mais de uma palavra, símbolo ou caractere de pontuação(JUNIOR, 2008).

Tendo o caractere espaço sempre descartado, uma das abordagens de identificar *tokens* é a quebra do texto em seus delimitadores, como além do espaço: () <>!-?.;'-“|(JUNIOR, 2008).

Entretanto, uma tarefa de identificar *tokens*, simples para um humano, pode ser complexo para um algoritmo, temos o exemplo de que o “.” pode ser usado em abreviações ou em números(JUNIOR, 2008).

Uma abordagem descrita por (JUNIOR, 2008) tem os seguintes passos:

- Geração simples: Utilizando uma lista de delimitadores são extraídos os *tokens* preliminares.
- Identificação de abreviações: Com a ajuda de dicionários pré-estabelecidos, são identificados os diversos tipos de abreviaturas.
- Identificação de palavras Combinadas: Algumas palavras compostas, que são interligadas por caracteres como “&” e “-”, devem ser consideradas um único *token*.
- Identificação de símbolos de internet: Extraído os endereços de e-mail, endereços de sites (URLs) e endereços IP.
- Identificação de números: É identificada toda e qualquer forma de apresentação de número, incluindo também medidas e valores, como R$100,00.
- Identificação de *tokens* multivocabulares: Para manter o sentido original de algumas palavras se tem a necessidade de unir elas em um único *token*.

2.2.1.2 Remoção de *Stopwords*

As *stopwords* são palavras em um texto que não possuem um valor semântico, são normalmente as preposições, artigos, conjunções, pontuação e pronomes de uma língua. As *stopwords* são removidas de acordo com uma *stoplist*, essa pode ser definida manualmente, tendo as palavras cadastradas manualmente, ou por forma automatizada, através da frequência de aparição das palavras, um percentual maior de aparição defini a *stoplist*(JUNIOR, 2008).

2.2.1.3 *Lemmatization*

Lemmatization é o processo de recuperar a primitiva de uma palavra, tendo como benefício uma estrutura que conserva o sentido das palavras(JUNIOR, 2008).

2.2.2 *Latent Semantic Analysis*(LSA)

Sendo uma técnica matemática e estatística, o *Latent Semantic Analysis(LSA)*, não sendo um processamento de linguagem natural tradicional, ele não utiliza qualquer dicionário, redes semânticas ou semelhantes, ele se utiliza do texto puro para a analise.(LANDAUER; FOLTZ; LAHAM, 1998).

O primeiro passo é representar o texto como uma matriz, onde cada palavra se torna uma coluna, cada linha é um trecho de texto, cada célula contém a quantidade de vezes que a palavra se repete no trecho de texto. Temos a figura 2 como exemplo.(LANDAUER; FOLTZ; LAHAM, 1998).

É aplicado a decomposição em valores singulares, com isso a matriz é decomposta em produto de outras três matrizes. A primeira matriz, como vetores de valores de fator ortogonal as linhas, a segunda é uma matriz diagonal, a terceira matriz como vetores de valores de fator ortogonal as colunas. A multiplicação dessas matrizes é nossa matriz inicial(LANDAUER; FOLTZ; LAHAM, 1998), temos a figura 3 como exemplo.

Para finalizar, se ordena a matriz S por tamanho, escolhendo os dois maiores valores, pegando as linhas e colunas equivalentes nas matrizes W e P. Com isso se tem uma matriz Xhihat reduzida, é aproximadamente igual a X, com um grau menor(LANDAUER; FOLTZ; LAHAM, 1998). Temos a figura 4 como exemplo.

Para o calculo da similaridade Deerwester(DEERWESTER et al., 1990) descreve as similaridades.Comparação de dois termos, é o produto escalar entre os dois vetores linhas da matriz Xhihat. Comparação de dois documentos, é o produto escalar entre as duas colunas da matriz Xhihat. Comparação de um termo e um documento, é o valor da célula da intersecção da linha e coluna.

Example of text data: Titles of Some Technical Memos

c1: *Human* machine *interface* for ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

$\{X\} =$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Figura 2 – Exemplo da representação do LSA. Fonte:(LANDAUER; FOLTZ; LAHAM, 1998)

Landauer (LANDAUER; FOLTZ; LAHAM, 1998) descreve que o *Latent Semantic Analysis*(LSA) não leva em conta a ordem das palavras no processamento.

Cada célula contém a quantidade de repetições da palavra.(LANDAUER; FOLTZ; LAHAM, 1998).

2.2.3 *STASIS*

Com os avanços nos projetos de linguísticos computacionais, temos bases de conhecimento semântico prontas e disponíveis. Essas bases de conhecimento constroem uma hierarquia comum ao mundo, essa hierarquia está ligada aos temos "*is a*" ou "*is a kind of*" (ISA), com a ISA podemos determinar a distância semântica entre palavras.

Dadas as palavras p1 e p2, podemos encontrar a semelhança semântica através da base de conhecimento, propostos diversos métodos testados por (LI; BANDAR; MCLEAN, 2003), a de melhor desempenho para a semelhança semântica, s(palavra1,palavra2), é uma função entre o comprimento do caminho dos atributos e profundidade.

$$s(palavra1, palavra2) = e^{-al}\frac{e^{bh}-e^{-bh}}{e^{bh}+e^{-bh}}$$

Onde l é o comprimento do caminho, h a profundidade, e os parâmetros $a = 0.2$ e $b = 0.6$(LI; BANDAR; MCLEAN, 2003).

$$\{X\} = \{W\}\{S\}\{P\}'$$

$\{W\} =$

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

$\{S\} =$

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

$\{P\} =$

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

Figura 3 – Exemplo da representação da decomposição SVD. Fonte:(LANDAUER; FOLTZ; LAHAM, 1998)

$\left\{\hat{X}\right\} =$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Figura 4 – Exemplo da matriz Xhihat. Fonte:(LANDAUER; FOLTZ; LAHAM, 1998)

Para a similaridade de sentenças descrito por (LI et al., 2006) duas formulas que se completam na formula:

$$s(sentenca1, sentenca2) = \delta \frac{S1 \cdot S2}{\|S1\| \cdot \|S2\|} + (1 - \delta) \frac{\|r1 - r2\|}{\|r1 + r1\|}$$

A formula pode ser dividida em duas partes, sendo o δ o peso que defini a importância da ordem das palavras sobre a similaridade das sentenças, dentro de um valor entre (0.5,1]. Quanto maior o δ menor será a importância da ordem das palavras(LI et al., 2006).

Para semelhança de sentenças, temos uma frase1 e frase2, fazemos a união dessas, assim não temos palavras repetidas, criamos um vetor S, com o mesmo tamanho da quantidade de palavras da união da frase1 e frase2. Esse vetor é preenchido com a função de se a palavra i da união está na frase1, é atribuído 1 em S[i], caso não esteja, é calculado a semelhança da palavra i com cada palavra de frase1, pegando a maior similaridade, e se essa similaridade for maior que um valor predefinido o valor da similaridade é atribuído em S[i], se não é atribuído 0(LI et al., 2006).

Após isso é aplicado um peso para cada S[i], assim S[i]=S[i]*I(palavra i), onde I(palavra i) $= 1 - \frac{log(n+1)}{log(N+1)}$, N é o número total de palavras no corpus, n é a frequência da palavra i no corpus(LI et al., 2006).

A primeira equação $\frac{S1 \cdot S2}{\|S1\| \cdot \|S2\|}$ defini a similaridade com o coeficiente de cosseno entre os dois vetores(LI et al., 2006).

A segunda equação $\frac{\|r1 - r2\|}{\|r1 + r1\|}$ defini a ordem das palavras como sendo a diferença normalizada da ordem das palavras.(LI et al., 2006). Se temos a frase1={A caixa dentro da cesta} e frase2={A cesta dentro da caixa}, os vetores r1={1,2,3,4,5} e r2={1,5,3,4,2}.

$$1 - \frac{\|\{1,2,3,4,5\}\| - \|\{1,5,3,4,2\}\|}{\|\{1,2,3,4,5\}\| + \|\{1,5,3,4,2\}\|} \approx 0.701$$

2.3 *Machine Learning*

Como subárea da inteligência artificial, o aprendizado de máquina tem como objetivo criar técnicas computacionais e sistemas que automaticamente adquirem conhecimento(REZENDE, 2003).

Como uma ferramenta de aquisição automático de conhecimento, a técnica de *Machine Learning* tem diversos algoritmos com diferentes desempenhos para cada problema, sendo importante compreender suas limitações de desempenho(REZENDE, 2003).

(CARRIJO, 2004) cita as divisões de estratégias de aprendizado de maquina: hábito, instrução, dedução, analogia e indução, considerando a complexidade da inferência. A estratégia de aprendizado por hábito é a mais simples, enquanto que as por analogia e indução são as mais complexas, pois o aprendiz necessita de um maior esforço para o

aprendizado.

Na figura 5, podemos ver a hierarquia do aprendizado focando o caminho que leva a classificação.

> Aprendizado indutivo é efetuado a partir de raciocínio sobre exemplos fornecidos por um processo externo ao sistema de aprendizado. O aprendizado indutivo pode ser dividido em supervisionado e não-supervisionado. No aprendizado supervisionado é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido(REZENDE, 2003).

Os exemplos usados para a indução são em geral descritos por vetores de valores, onde cada valor representa uma característica ou atributos e um rótulo da classe associada. O algoritmo de indução tenta classificar novos exemplos em suas devidas classes, para os rótulos de classe discretos, esse problema é conhecido como classificação e para valores contínuos como regressão(REZENDE, 2003).

Para o aprendizado não supervisionado temos que o indutor irá analisar e agrupar os exemplos de maneira a utilizar da medida de similaridade, após o agrupamento é necessário uma interpretação dos agrupamentos para saber seu significado(CARRIJO, 2004).

Outro fator é o esclarecimento das classificações, em (MICHALSKI; BRATKO; KUBAT, 1998) são divididos em dois tipos de sistemas de aprendizado:

- Sistemas considerados caixa-preta, tem representações de conceitos que não são interpretados com facilidade por humanos, não fornecendo nem explicando o processo de reconhecimento.
- Sistemas voltados a conhecimento, tem estruturas para a compreensão do conhecimento por humanos.

Na Tabela 1 são resumidas por (REZENDE, 2003) as características gerais dos sistemas de aprendizado de máquina.

Tabela 1 – Características gerais de sistemas de aprendizado de máquina. Fonte: (REZENDE, 2003)

Modos de formas de aprendizado	Paradigmas de aprendizado	Formas de aprendizado
Supervisionado	Simbólico	Incremental
Não supervisionado	Estatístico	Não incremental
	Baseado em instâncias	
	Conexionista	
	Genético	

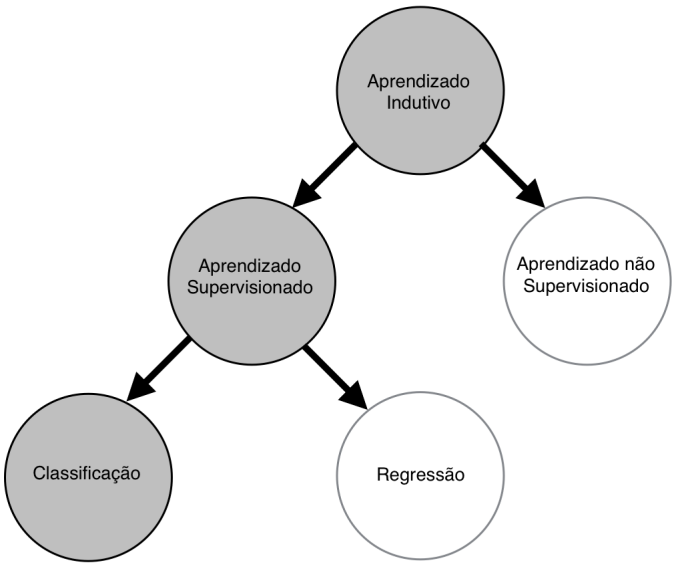


Figura 5 – Hierarquia do aprendizado adaptado de Fonte:(REZENDE, 2003)

2.3.1 Aprendizado Supervisionado Conceitos e Definições

Nessa seção são descritos algumas definições de termos e problemas de classificação no aprendizado supervisionado definidos por (REZENDE, 2003) e (CARRIJO, 2004):

Indutor: O objetivo do indutor é encontrar um classificador com base em conjunto de exemplos rotulados(REZENDE, 2003) e (CARRIJO, 2004).

Exemplo: O exemplo descreve o objeto de interesse na forma de valores em um vetor(REZENDE, 2003) e (CARRIJO, 2004).

Atributo: Descreve características de um exemplo, existem ao menos dois tipos de atributos, são eles o nominal ou discreto, quando não existe uma ordem entre os valores, e contínuo, quando existe uma ordem entre valores(REZENDE, 2003) e (CARRIJO, 2004).

Existem também símbolos especiais para os atributos, são eles, o de ausência do atributo, que é diferente de zero, normalmente representado pelo símbolo "?", e o não-se-aplica, quando um atributo não se aplica ao exemplo, normalmente representado pelo símbolo "!"(REZENDE, 2003) e (CARRIJO, 2004).

Classe: No aprendizado supervisionado o exemplo possui um atributo especial, chamado de rótulo ou classe, que descreve o fenômeno de interesse, isso é a meta que se deseja aprender para fazer previsões, tipicamente pertencentes a um conjunto discreto (nominal) de classes C1, C2, ..., Ck no caso de classificação ou de valores reais no caso de regressão(REZENDE, 2003) e (CARRIJO, 2004).

Conjunto de exemplos: Um conjunto de exemplos é composto por exemplos contendo valores de atributos bem como a classe associada. Na Tabela 2 é mostrado o

Tabela 2 – Conjunto de exemplos no formato atributo-valor. Fonte:(REZENDE, 2003)

	X1	X2	...	Xm	Y
T1	x11	x12	...	x1m	y1
T2	x21	x22		x2m	y2
...	...	...	...	...	...
Tn	xn1	xn2	...	xnm	yn

formato padrão de um conjunto de exemplos T com n exemplos e m atributos. Nessa tabela está representado um conjunto de dados com n exemplos e m atributos. Nesse formato, as colunas (X1, ..., Xm) representam os diferentes atributos e as linhas (T1, ..., Tn) os diferentes exemplos, na ultima coluna temos o Y, yi = f(x_1, ..., xn), é a função que se tenta predizer a partir dos atributos(REZENDE, 2003) e (CARRIJO, 2004).

Ruído: Um ruído pode ser causado por diversos fatores, pelo próprio processo de aquisição dos dados, transporte dos dados, entre outros, exemplificando temos dois elementos com atributos iguais e classes diferentes, isso é um ruído(REZENDE, 2003) e (CARRIJO, 2004).

Conhecimento do domínio: É o conhecimento prévio sobre o domínio trabalhado, como limites ou valores válidos para atributos, nem todos os indutores são capazes de usar o conhecimento do domínio, eles usam apenas os exemplos fornecidos(REZENDE, 2003) e (CARRIJO, 2004).

Classificador: Como falado acima, um indutor gera como saída um classificador, de forma a dado uma entrada não rotulada possa classifica-la com maior precisão possível sua classe. A tarefa de um indutor é, dado um conjunto de exemplos, induzir uma função h que aproxima f, normalmente desconhecida, onde h(xi) aproximasse de f(xi)(REZENDE, 2003) e (CARRIJO, 2004).

Bias: Qualquer escolha de hipótese além da consistência dos exemplos se chama de *bias* de aprendizado. Devido ao fato de que quase sempre existe um número grande de hipóteses consistentes, todos os indutores possuem alguma forma de *bias*(MITCHELL, 1997).

Modo de aprendizado: Dividido em dois modos, o não-incremental, necessita de todo o conjunto de treinamento para o aprendizado, conhecido como *batch*, e o incremental, onde o indutor tenta atualizar a função h, sempre que novos exemplos são adicionados ao conjunto de treinamento(REZENDE, 2003) e (CARRIJO, 2004).

Espaço de descrição: Cada atributo xij corresponde a uma coordenada em um espaço de m dimensões, cada ponto no espaço tem uma classe associada de yi, com isso o classificador divide este espaço em regiões, e cada região rotulada com uma classe. Quando um novo exemplo é classificado é determinado seu ponto no espaço e atribuído a classe daquela região(REZENDE, 2003) e (CARRIJO, 2004).

Erro de precisão: É uma medida de desempenho dada ao classificador, ou taxa de classificação incorreta, para isso se usa a formula erro(h):

$$erro(h) = \frac{1}{n} \sum_{i=1}^{n} \parallel yi \neq h(xi) \parallel$$

O operador ‖ E ‖ retorna 1 se a expressão for verdadeira e 0 se falso, complementando a taxa de erro o acc(h) é a precisão:(REZENDE, 2003) e (CARRIJO, 2004)

$$acc(h) = 1 - erro(h)$$

Para problemas de regressão, o erro(h) pode ser calculado pela distância do valor real para o valor dado de classificador. São dadas duas equações, sendo elas o erro médio quadrado (mse-err ou mean squared error) e a distância absoluta média (mad-err ou mean absolute distance), apresentadas como:(REZENDE, 2003) e (CARRIJO, 2004)

$$mse - err(h) = \frac{1}{n} \sum_{i=1}^{n} (yi - h(xi))^2$$

$$mad - err(h) = \frac{1}{n} \sum_{i=1}^{n} \mid yi - h(xi) \mid$$

Distribuição de classes: Dado um conjunto de T exemplos, a distribuição de classes da a porcentagem de cada classe sobre o modelo, dada pela formula distribuicao(Cj), onde Cj é parte do conjunto T, e n o total de exemplos:(REZENDE, 2003) e (CARRIJO, 2004)

$$distribuicao(Cj) = \frac{1}{n} \sum_{i=1}^{n} \parallel yi = Cj \parallel$$

Como exemplo temos que em um conjunto de 100 elementos, temos as classes C1 com 60 elementos, C2 com 30 elementos e C3 com 10 elementos, a distribuição ficaria *distribuicao*(C1,C2,C3)=(0,60, 0,30, 0,10) = (60,00%, 30,00%, 10,00%), a C1 se define como classe majoritária e o C2 se define como classe minoritária(REZENDE, 2003) e (CARRIJO, 2004).

Erro majoritária: É o erro máximo que um classificador pode cometer, sua formula:(REZENDE, 2003) e (CARRIJO, 2004)

$$maj - err(T) = 1 - max(distribuicao(C1, .., Cn))$$

No exemplo acima o erro majoritário é maj-err(T) = 1 - 0,60 = 40,00%.

Prevalência de classes: Quando uma classe tem uma alta classificação gera um desbalanceamento do conjunto de exemplos, como exemplo a *distribuicao*(C1,C2,C3)=(98%, 1,0%, 1,0%) de um classificador, que classifique exemplos como pertencentes à classe majoritária C1 teria uma precisão de 98,00% (maj-err(T) = 2,00%)(REZENDE, 2003) e (CARRIJO, 2004).

Under* e *Overfitting: Ao induzir uma hipótese a partir de um conjunto de exemplos podemos ter que essa seja muito especifica para esse conjunto. O exemplo é uma parte do conjunto domínio. É possível induzir hipóteses que melhorem o conjunto de treinamento, quando a hipótese se ajusta ao treinamento e seu desempenho piora com

exemplos diferentes ouve um *overfitting* ou seja ajustou-se em excesso ao conjunto de treinamento(REZENDE, 2003) e (CARRIJO, 2004).

O contrário de um *overfitting*, seria quando um conjunto muito pequeno de exemplos fazendo a hipótese se ajustar pouco ao conjunto de testes gerando um *underfitting*(REZENDE, 2003) e (CARRIJO, 2004).

Overtuning: O excesso de ajustes, para que todos os parâmetros sejam otimizados para todos os exemplos se chama *Overtuning*, assim como o *overfitting*, pode ser evitado utilizando-se apenas uma parte dos exemplos disponíveis para a execução do indutor e o restante dos exemplos para um teste final do classificador induzido. Os exemplos agrupados em conjuntos de treinamento e teste, uma vez treinado se aplica os exemplos de testes, para avaliar a confiabilidade das estimativas futuras do sistema(REZENDE, 2003) e (CARRIJO, 2004).

Poda: A poda é um método que se usa para lidar com o ruído e *overfitting*, há dois métodos para a poda. O pré-poda, são removidos alguns exemplos antes da geração da hipótese, pós-poda, uma hipótese é gerada e algumas partes eliminadas(REZENDE, 2003) e (CARRIJO, 2004).

Completude e Consistência: Uma hipótese pode ser avaliada em duas formas, a completude, se ela classifica todos os eventos, e a consistência, se classifica corretamente os exemplos. Temos quatro combinações possíveis, são elas (a) completa e consistente; (b) incompleta e consistente; (c) completa e inconsistente ou (d) incompleta e inconsistente. Temos na figura 6 um exemplo considerando dois atributos X1 e X2, três classes, bola azul, adição azul, e asterisco verde, as hipóteses induzidas para cada classe são representadas por duas regiões indicadas por linhas sólidas para a classe bola azul, duas regiões indicadas por linhas pontilhadas para a classe adição vermelha e uma região indicada por linhas tracejadas para a classe asterisco verde(REZENDE, 2003) e (CARRIJO, 2004).

Matriz de confusão: Ela é uma forma de medir a efetividade do modelo, onde mostrar o número de classificações corretas, versus as classificações preditas para cada classe sobre um conjunto de exemplos T. Como na tabela 3, temos K classes verdadeiras e K classes preditas, cada elemento representa o número de exemplos em T, que pertenciam a Ci, mas foram classificados como Cj, calculado por:(REZENDE, 2003) e (CARRIJO, 2004)

$$M(Ci, Cj) = \sum_{\forall(x,y)\in T: y=Ci} \parallel h(x) = Cj \parallel$$

O número de acertos, para cada classe, se localiza na diagonal principal M(Ci,Ci) da matriz, os demais elementos M(Ci,Cj), para i $\neq$ j, representam erros na classificação. A matriz de confusão de um classificador ideal possui todos esses elementos que não são da diagonal principal iguais à zero, podendo ser representado por uma matriz identidade multiplicado por um vetor contendo as quantidade de (C1,C2..., Ck)(REZENDE, 2003)

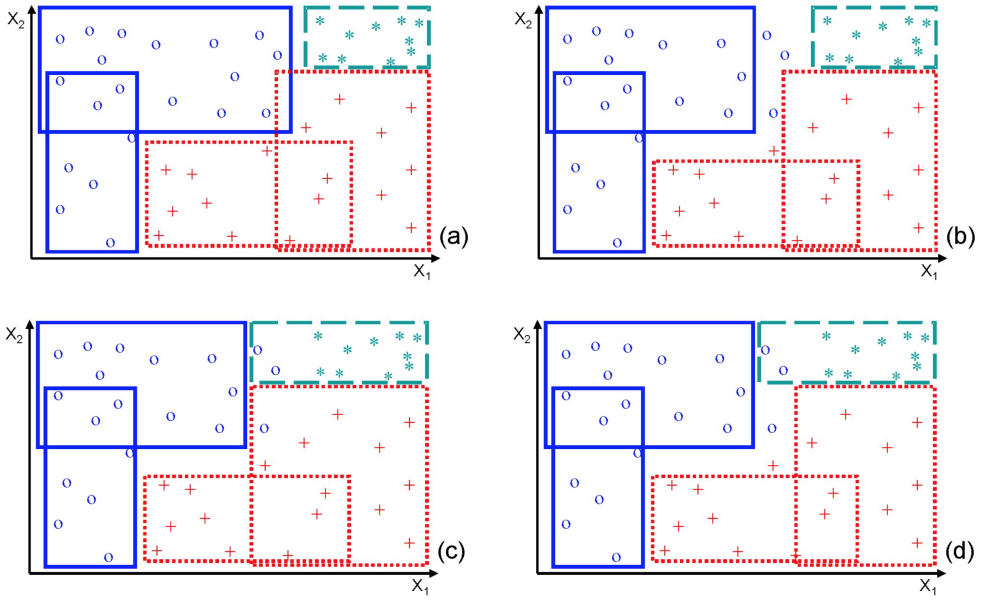


Figura 6 – Completude e consistência de uma hipótese. Fonte:(REZENDE, 2003)

Tabela 3 – Matriz de confusão de um classificador. Fonte:(REZENDE, 2003)

Classe	predita C1	...	predita Ck
Verdadeira C1	M(C1,C1)	...	M(C1,Ck)
...	...	...	...
Verdadeira Ck	M(Ck,C1)	xn2	M(Ck,Ck)

e (CARRIJO, 2004).

Custo de erro: O custo do erro, representado por cost(Ci,Cj), é a penalidade aplicada ao classificador que classifica um exemplo em Ci, quando deveria classificar como Cj, assim o cost(Ci,Ci) = 0, uma vez que não é um erro, logo cost(Ci,Cj) > 0, para i ≠j, em geral é dado cost(Ci,Cj) = 1. No calculo do custo os erros são convertidos em custos pela multiplicação do erro pelo custo correspondente, pela formula:(REZENDE, 2003) e (CARRIJO, 2004)

$$errocost(h) = \frac{1}{n}\sum_{i=1}^{n} \parallel yi \neq h(xi) \parallel Xcost(yi, h(xi)$$

Também pode ser obtido pela matriz de confusão, pela formula:(REZENDE, 2003) e (CARRIJO, 2004)

$$errocost(h) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} M(Ci, Cj)Xcost(Ci, Cj)$$

Complexo: É uma disjunção de conjunções para testar atributos, tendo a forma de X operador Valor, onde X é um atributo, operador é um operador pertencente ao conjunto $=, \neq, >, <, \geq, \leq$ e Valor é um valor constante válido para o atributo X, retornando verdadeiro ou falso(REZENDE, 2003) e (CARRIJO, 2004).

Regra: Uma regra assume a forma se L então R, podendo ser escrita como $L \rightarrow R$,

normalmente L e R são complexos e sem atributos em comum. A parte esquerda L é denominada condição, premissa, cauda ou corpo da regra, e a parte direita R é denominada conclusão ou cabeça da regra(REZENDE, 2003) e (CARRIJO, 2004).

Regra de classificação: Uma regra de classificação assume a forma de uma regra se L então classe = Ci, onde Ci pertence ao conjunto de K valores de classe C1, ..., Ck(REZENDE, 2003) e (CARRIJO, 2004).

Cobertura: A cobertura de uma regra $L \rightarrow R$, pode ser descrita por quatro estados. Exemplos que cumprem a parte L da regra constituem o seu conjunto de cobertura. Exemplos que cumprem tanto a condição L como a conclusão R são inclusos pela regra. Exemplos que cumprem a condição L mas não a conclusão R são inclusos incorretamente pela regra. Exemplos que não cumprem a condição L não são inclusos pela regra(REZENDE, 2003) e (CARRIJO, 2004).

2.4 Algoritmo genético

Carrijo (CARRIJO, 2004) divide os períodos do desenvolvimento de inteligência artificial em quatro fases, são elas:

- Período sub-simbólico, entre 1950 e 1965, tinha como principal característica a representação do conhecimento de forma numérica.
- Período simbólico, entre 1962 e 1975, tinha como principal característica o desenvolvimento dos algoritmos de representações simbólicas como a lógica de predicados e redes semânticas.
- Período intensivo, entre 1976 e 1988, tinha como principal característica a quantidade de conhecimentos que era incorporados aos sistemas de aprendizagem.
- Atualmente, tem como característica tornar o computador *expert* no desenvolvimento de determinadas tarefas.

2.4.1 Terminologia

A área de algoritmos genéticos existem termos que são voltados a biologia, abaixo existe a explicação dos principais, citados por (GOLDBERG, 1989 apud CARRIJO, 2004) (GALVAO, 1999 apud CARRIJO, 2004) (MICHALEWICZ; ATTIA, 1994 apud CARRIJO, 2004):

- Cromossomo ou Genoma: Representa a estrutura de dados compostas por genes, que codifica uma solução para um problema.

- Gene: Sendo uma característica do cromossomo, representa uma variável que de decisão do problema em um vetor que representa o cromossomo.
- Indivíduo: É representado pelo seu cromossomo e valor de aptidão.

2.4.2 O algoritmo

O processo do algoritmo genético descrito por Holland é conhecido na literatura como *Simple Genetic Algorithm ou Standard Genetic Algorithm* ou, simplesmente, SGA, descrito por (DAVIS, 1991 apud CARRIJO, 2004), são seis passos simples, A figura 7 mostra um fluxograma com os passos básicos de um algoritmo genético *standard*:

1. Inicia a população de tamanho fixo N.
2. Avalie a função de aptidão para cada solução dessa população.
3. Cria novos cromossomos(filhos) através de recombinação da população atual(pais).
4. Elimine membros da antiga população, para inserir os filhos, mantendo o número fixo N da população.
5. Aplica-se a função de aptidão nos novos filhos e os coloca na população.
6. Verifica a condição de parada, podendo ser, a solução ideal ou um número pré-estabelecido de gerações criadas, e retorna a solução com a melhor aptidão. Caso contrário, volte ao passo três.

2.4.2.1 Iniciação da população

Seguindo a metodologia do algoritmo temos o primeiro passo, a iniciação, descrita por (SILVA, 2005), ela pode ser de três maneiras:

- Inicialização aleatória: Os indivíduos da população são gerados de forma aleatória.
- Inicialização determinística : Os indivíduos da população são gerados seguindo uma função predeterminada.
- Inicialização aleatória com nicho: Os indivíduos da população são gerados com características semelhantes.

A iniciação não necessariamente precisa ser aleatória, podendo ser gerado uma população em um intervalo onde se acredita ter a resposta, ou podemos ter um escalonador para a população, tendo como exemplo uma população de 50 indivíduos distribuídos igualmente entre 0 a 10(SILVA, 2005).

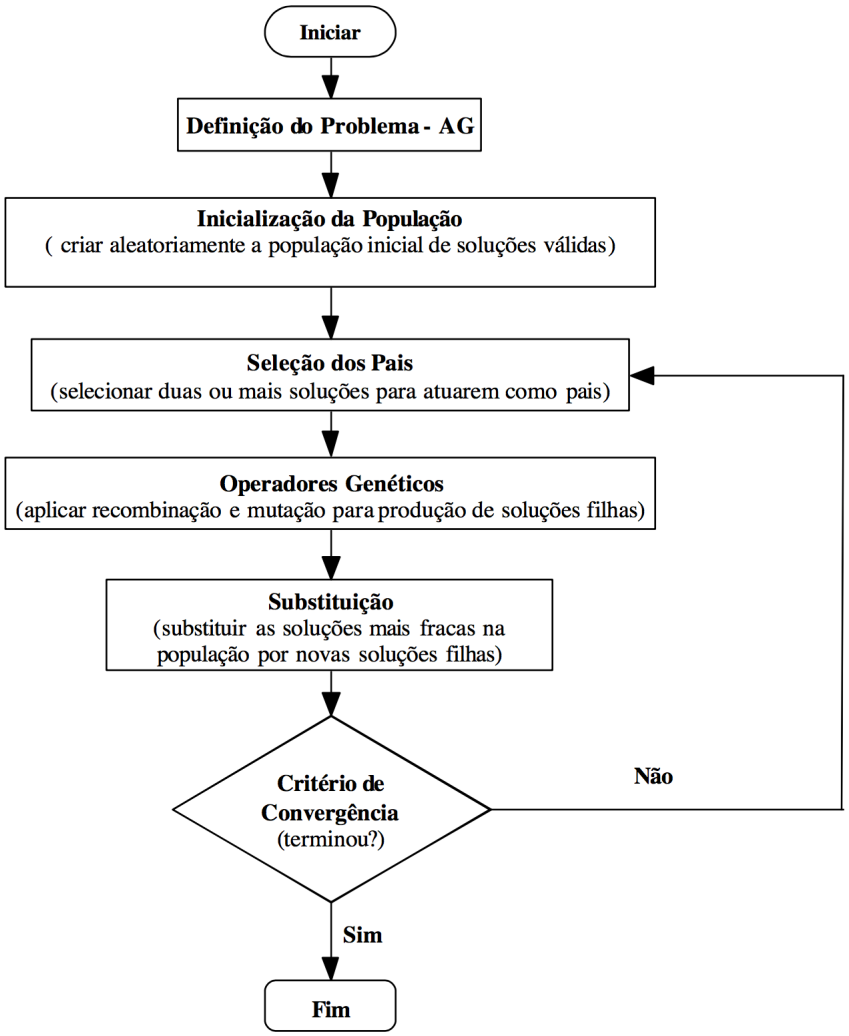


Figura 7 – Fluxograma do algoritmo genético *standard*. Fonte:(CARRIJO, 2004)

2.4.2.2 Seleção dos Pais

Para a seleção dos pais são usadas a função de aptidão e métodos de seleção.

2.4.2.3 Função de aptidão

Uma função de aptidão é o que liga o código a solução, toma como entrada um cromossomo, que é uma tentativa de resolver o problema, e retorna um número real que informa o desempenho dele, esse valor é usado no processo de seleção e reprodução(CARRIJO, 2004).

Para a seleção, ao selecionar os indivíduos mais aptos, após a passagem desses pela função de aptidão, para alguns métodos é desejável que a soma dos valores de aptidão seja 1, para isso pode ser usado a formula abaixo, que é a aptidão relativa:(SILVA, 2005)

$$f(x_i)_{rel} = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}$$

Onde X_i é um cromossomo, $f(x_i)$ é a função de aptidão e $f(x_i)_{rel}$ é o valor relativo

Tabela 4 – Exemplo de seleção pelo método da roleta. Fonte: (SILVA, 2005)

Indivíduo	Aptidão	Roleta	Aptidão Relativa	Porcentagem
I1	22	22	0.27160493	27%
I2	25	47	0.30864197	31%
I3	17	64	0.20987654	21%
I4	7	71	0.08641975	9%
I5	10	81	0.12345679	12%

dele.

2.4.2.4 Métodos de seleção

Os métodos de seleção, tem a proposta de procurar favorecer os indivíduos mais aptos, mas tentando manter a diversidade, citados por (SILVA, 2005) temos:

- Roleta.
- Torneio.
- Amostragem estocástica.

2.4.2.4.1 Roleta

Esse é o método mais simples descrito, e normalmente o mais utilizado, os indivíduos são representados como uma roleta, seu espaço é proporcional ao número dado pela função de aptidão a ele. Com isso são gerados números aleatórios entre 0 e o total do somatório da aptidão. O número de vezes gerado depende do tamanho da população, indivíduos selecionados são inseridos na população intermediária(SILVA, 2005).

Na figura 8 e na tabela 4 temos um exemplo referente ao uso do método (SILVA, 2005):

2.4.2.4.2 Torneio

Os indivíduos são separados em grupos com n indivíduos, o melhor de cada grupo é selecionado para a população intermediaria, geralmente o n é usado como três. Exemplo na figura 9 (SILVA, 2005).

2.4.2.4.3 Amostragem Estocástica

É uma variação do método da roleta, ao em vez de se rodar a roleta a quantidade de vezes necessárias, é dado o número n necessário onde é dividido igualmente seu espaço a partir do ponto escolhido, com isso só é necessário uma rodada(SILVA, 2005).

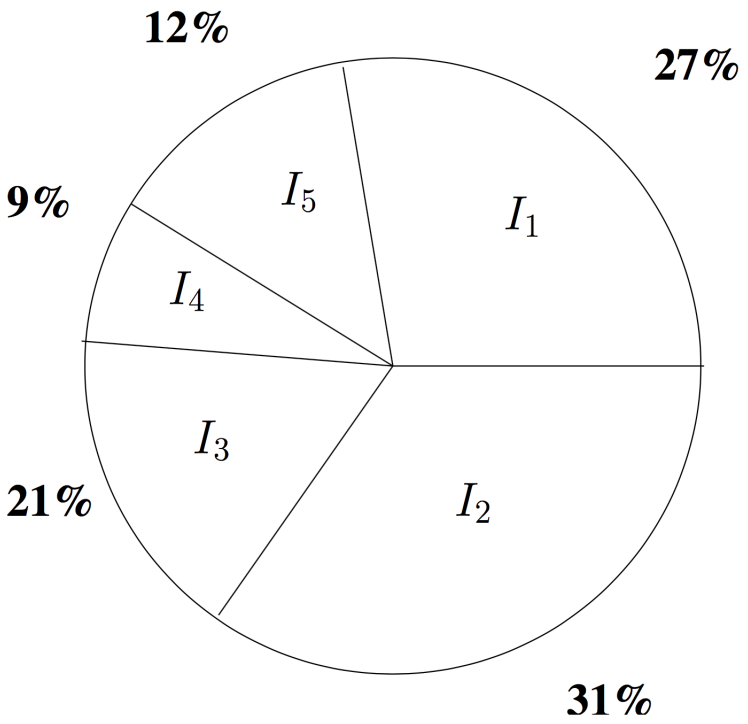


Figura 8 – Esquema da Roleta. Fonte:(SILVA, 2005)

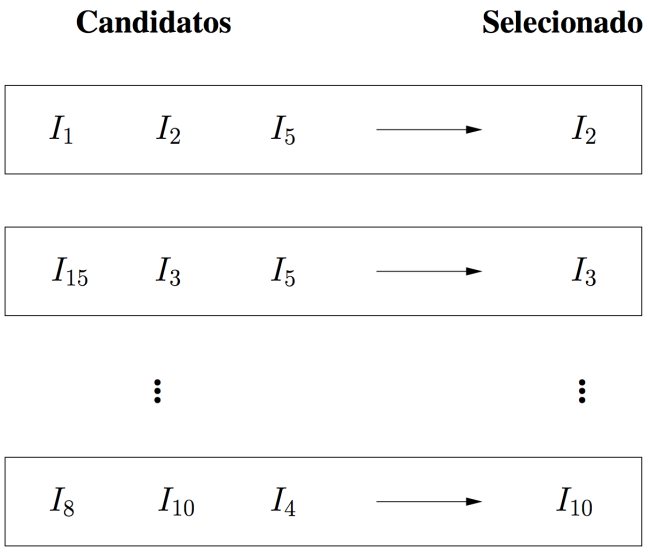


Figura 9 – Esquema do Torneio. Fonte:(SILVA, 2005)

2.4.2.5 Operadores Genéticos

As operações genéticas descritas por (SILVA, 2005) e (CARRIJO, 2004) são:

- Cruzamento, Recombinação ou *Crossover*.
- Mutação.

2.4.2.5.1 Cruzamento, Recombinação ou *Crossover*

Geralmente usando uma permutação simples entre os pais, ou da combinação entre características correspondentes entre soluções pais, o primeiro caso é mais usado em representação através de código binário, enquanto o segundo, no caso de representação real das soluções(CARRIJO, 2004).

A recombinação é dada por uma probabilidade denominada de taxa de recombinação, que varia entre 60% e 100%. Isso implica que os filhos podem ser iguais aos pais, gerando um número entre [0,1], podemos sortear quais seriam permutados ou não(CARRIJO, 2004).

Seguindo os seguintes passos descritos por (CARRIJO, 2004), podendo ser visto na figura 10:

- Gera-se um número aleatório entre 0 e 1.
- Se o número gerado for maior ou igual à probabilidade de recombinação, ocorrer recombinação, caso não seja não ocorrerá recombinação.
- Escolhe-se aleatoriamente um ponto, no qual ocorrerá a quebra.
- Quebram-se os dois cromossomos neste ponto.
- Formam-se dois novos cromossomos combinando-se a primeira parte do cromossomo A com a segunda parte do cromossomo B, e a primeira parte do cromossomo B com a segunda parte do cromossomo A.

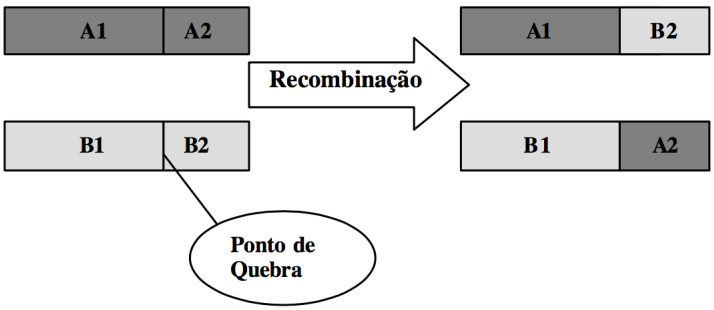
2.4.2.5.2 Mutação

O processo de mutação geralmente controlado por um fator de porcentagem, que indica a probabilidade de um gene sofrer mutação. Com isso é inserido novo material genético em cada geração(CARRIJO, 2004) (SILVA, 2005).

A operação de mutação pode ser aplicada antes ou depois da recombinação, a operação de mutação percorre todos os bits ou genes da solução e com uma probabilidade predeterminado de o gene ser trocado ou não. A probabilidade de afetar um gene deve ser definida de forma que seja bem pequena, para não causar destruição de um cromossomo(CARRIJO, 2004).

2.4.2.6 Substituição

Elimine membros da antiga população, para inserir os filhos, mantendo o número fixo N da população(DAVIS, 1991 apud CARRIJO, 2004).



Pais: **(recombinação após o oitavo bit)** **Filhos:**

1 1 1 1 1 1 1 1 | **1 1 1** ---------- **1 1 1 1 1 1 1 0 0 0**

0 0 0 0 0 0 0 0 | 0 0 0 ---------- 0 0 0 0 0 0 0 **1 1 1**

Pais: **(recombinação após o segundo bit)** **Filhos:**

1 0 | **1 1 0 1 0 0 0 1** ---------- **1 0** 0 1 1 0 0 0 1 1

1 1 | 0 1 1 0 0 0 1 1 ---------- 1 1 **1 1 0 1 0 0 0 1**

Figura 10 – Esquemas de uma recombinação simples. Fonte:(CARRIJO, 2004)

2.4.2.7 Parâmetros Genéticos

Descritos por (SILVA, 2005) e (CARRIJO, 2004) são listados alguns parâmetros usados em Algoritmos genéticos, são eles:

- Tamanho da População.
- Taxa de Cruzamento.
- Taxa de Mutação.
- Taxa de Substituição.
- Critérios de parada.
- Problemas de convergência.

2.4.2.7.1 Tamanho da População

Com uma população grande pode afetar o desempenho geral do sistema, assim como uma população pequena pode não ter uma cobertura boa do espaço da busca do problema. Uma grande população nos dar uma grande cobertura do espaço de busca da solução, além de evitar a escolha de uma máximo local, ao em vez do máximo global. No entanto há necessidade de recursos computacionais mais poderosos ou de maior tempo para processamento.

2.4.2.7.2 Taxa de cruzamento

Uma taxa de cruzamento alta, poderá retirar estruturas com boas aptidões, uma taxa de cruzamento baixa, o algoritmo poderá ficar lento ou trava.

2.4.2.7.3 Taxa de Mutação

Uma taxa baixa de mutação nos previne perda de cromossomos com uma boa aptidão, e previne também a estagnação, um fator muito grade de mutação faz nossa busca ser essencialmente aleatória.

2.4.2.7.4 Taxa de Substituição

É a taxa que será substituída a geração anterior, com uma taxa alta pode ocorrer perda de população com alta aptidão, um valor baixo pode tornar o algoritmo lento.

2.4.2.7.5 Critérios de parada

Como critérios de parada se usam normalmente um número máximo de gerações, tempo de processamento ou uma estagnação da melhoria, isso é mínimas variação dos valores da aptidão ou do melhor membro ou da população.

2.4.2.7.6 Problema de Convergência

A convergência prematura é um problema comum em algoritmos genéticos, ocorre quando são achados soluções com aptidão elevada, mas não ótimas, com isso esses cromossomos podem dominar a população. Esse problema pode ser evitado limitando o numero de filhos por cromossomo, pode-se também aumentar a taxa de mutação para aumentar a diversidade ou evitar inserir filhos duplicados.

2.4.3 Notas

Todos os conceitos dão apoia para construção do sistema proposto, usando o processamento de linguagem natural, para tratamento do texto resposta e gabarito, e sua similaridade semântica, e *Machine Learning*, para o aprendizado do sistema, que será adquirido com a interação com as correções das respostas do sistema.

2.5 Engenharia de Software

2.5.1 Metodologia Scrum

O SCRUM é um *framework* desenvolvido por Ken Schwaber e Jeff Sutherland, usado desde 1990 para o gerenciar o desenvolvimento de produtos complexos, ele tenta entregar produtos com o maior valor agregado possível(SCHWABER; SUTHERLAND, 2013).

Os times, que estão associados a papeis, eventos, artefatos e regras, seu fundamento vem o empirismo, ou seja a experiência gera o conhecimento, utilizando o interativo e incremental para adequar as previsões e controle de riscos(SCHWABER; SUTHERLAND, 2013).

Segundo (SCHWABER; SUTHERLAND, 2013) os três pilares que sustentam o SCRUM são: transparência, inspeção e adaptação.

Transparência: Sempre manter visível os resultados aos interessados, requerendo um padrão comum para que todos os observadores possam ter o mesmo ponto de vista.

Inspeção: Sendo benéficas quando realizadas de forma correta, devesse frequentemente se inspecionar os artefatos, desde que não atrapalhe a real trabalho, sendo melhor realizadas por especialistas em suas áreas.

Adaptação: Uma vez encontrado, através de inspeções, um erro ou desvio dos limites aceitáveis em qualquer artefato, devesse ajustar o mesmo o mais cedo possível para minimizar maiores desvios.

Formalmente se tem quatro eventos envolvendo o inspeção e a adaptação(SCHWABER; SUTHERLAND, 2013):

- Reunião de planejamento da *sprint*.
- Reunião diária.
- Reunião de revisão da *sprint*.
- Retrospectiva da *sprint*.

2.5.1.1 O time *scrum*

O time *scrum* se auto-organiza, tendo autonomia para escolher a melhor forma de trabalhar, e é multifuncional, onde pode completar qualquer tarefa independente de outros fora da equipe. Três parte distintas se envolvem no time *scrum* são essas *product owner*, o Time de desenvolvimento e o *scrum master*(SCHWABER; SUTHERLAND, 2013).

Product Owner: Responsável por manter os itens no *backlog* do produto, tornando ele visível e claro a todos, garantindo que o time de desenvolvimento entenda-os. O *product owner* é representado por uma pessoa.

Time de Desenvolvimento: Tem como principal função transformar o *backlog* do produto em incrementos funcionais para o cliente, tendo ele todos os requisitos para o desenvolvimento dos item *backlog* do produto, sem dependência externa. No time todos são desenvolvedores, não podendo ter outros títulos ou agrupamentos de sub-times com especializações.

Scrum Master: Sendo o responsável pelo cumprimento das praticas do *scrum*, trabalho, com o *product owner*, esclarecendo os itens no *backlog* do produto, compreender o projeto a longo-prazo e facilitando os eventos do *scrum*, com o Time de desenvolvimento, liderando o time nas praticas de agregar o maior valor de produto, autogerenciamento e *scrum*, além de ser o facilitador de impedimentos do time.

2.5.1.2 Eventos do Scrum

No *scrum* todos os eventos são *time-boxed*, isso é, tem um tempo fixo máximo para duração, cada evento foi criado para ser possível a realização da transparência e inspeção(SCHWABER; SUTHERLAND, 2013).

2.5.1.2.1 *Sprint*

No *scrum* a *sprint* é o principal evento, que engloba todos os demais, de forma que o tempo dela é fixo não podendo ser alterado posteriormente, com um limite de um mês, sendo que uma *sprint* começa assim que termina a anterior. Compostas por uma reunião de planejamento da *sprint*, reuniões diárias, trabalho de desenvolvimento, uma revisão da *sprint* e a retrospectiva da *sprint*(SCHWABER; SUTHERLAND, 2013).

2.5.1.2.2 Reunião de planejamento da *sprint*

Todo trabalho a ser realizado na *sprint* é definido nesta reunião, com um limite de oito horas para uma *sprint* de um mês, com dois tópicos que são (SCHWABER; SUTHERLAND, 2013):

- Quais incrementos podem ser entregues na próxima *sprint*?

- Como será realizado o trabalho para incremento?

Ao responder essas questões se tem a meta da *sprint*, que guia o time de desenvolvimento para a construção de algo de valor ao cliente, a partir dos itens escolhidos a serem feitos na *sprint* se cria o *backlog* da *sprint*, que será desenvolvido(SCHWABER; SUTHERLAND, 2013).

2.5.1.2.3 Revisão da *sprint*

Essa reunião inclui todos envolvidos no *scrum*, tendo um *time-boxed* de quatro horas. São os principais itens de pauta da reunião, a apresentação dos itens do *backlog* que foram definidos pelo *product owner* que estão prontos. E os que não estão prontos, o time de desenvolvimento demostra o que está pronto e responde as duvidas sobre o incremento, análise de possíveis mudanças no mercado.

2.5.1.3 Artefatos do Scrum

2.5.1.3.1 *Backlog* do Produto

Onde se encontra uma lista de tudo que é necessário para o desenvolvimento do sistema. O *backlog* do Produto evolui junto com o projeto e o desenvolvimento(SCHWABER; SUTHERLAND, 2013).

2.5.1.3.2 *Backlog* da *sprint*

Onde se encontra os itens selecionados do *backlog* do Produto para se desenvolver no período da *sprint*(SCHWABER; SUTHERLAND, 2013).

2.5.2 Arquitetura *Model View Controller*(MVC)

Segundo (KRASNER; POPE, 1988) o MVC é dividido em três camadas. Sendo a *Model*, é uma simulação do domínio especificado, podendo ser um simples digito a um objeto complexo. A *View*, solicita dados da *Model*, contendo as informações visuais e os componentes necessários para a exibição. A *Controller*, é a responsável por capturar ações de dispositivos de entrada, e com isso conectar as chamadas entre a *Model* e *View*.

O MVC tem o ciclo começo por uma ação do usuário, a *Controller* notifica a *Model*, para que seu estado seja atualizado, a *Model* efetua as modificações necessárias e alerta as suas dependências que foi alterada, assim a *View* consulta o novo estado da *Model*, e atualiza a sua visualização. Temos a figura 11 como exemplo desse ciclo(KRASNER; POPE, 1988).

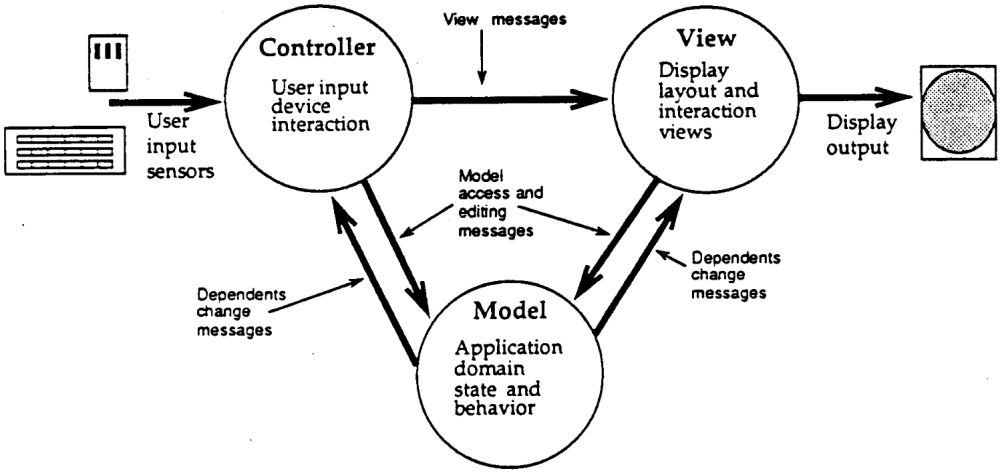


Figura 11 – Ciclo de interação do MVC. Fonte:(KRASNER; POPE, 1988)

2.5.3 Paradigma Orientado a Objetos

Permitindo o tratamento de substantivos como objetos e verbos como ações, um objeto pode ser um ator, agente ou servidor, podemos definir um objeto como uma entidade que (BOOCH, 1986):

- Tem estado.
- Sofre ações de outros objetos.
- É uma instância de classe.
- É expresso por um nome.
- Restringe visibilidade para outros objetos.

Se descreve os seguintes passos para se desenvolver um objeto (BOOCH, 1986):

- Identificar seus atributos.
- Identificar suas operações.
- Estabelecer a visibilidade do objeto.
- Estabelecer a interface.
- Implementar o objeto.

3 A PROPOSTA

Neste capítulo é descrita a proposta do trabalho, detalhando os passos a serem realizados, considerando o referencial teórico apresentado e estudos já realizados.

3.1 Introdução

Com a evolução da tecnologia, os computadores se tornaram cada vez mais parte do nosso cotidiano, com as mais diversas utilidades, desde cálculos complexos e previsão do tempo, diversão, comunicação, educação entre outros. Com isso o computador se tornou algo fundamental na sociedade.

Para a educação, a utilização cada vez maior de Ambientes Virtuais de Aprendizado(AVA), que podem ser usados tanto para alunos de cursos presenciais ou de educação a distância (EAD), melhora o acompanhamento pelo professor. Este tem a possibilidade de oferecer as matérias de estudo, recolher atividades, aplicar testes entre outras atividades oferecidas.

Nesse trabalho iremos focar a parte de avaliação, nas AVA's podem-se ter avaliações por meio de perguntas de múltipla escolha, verdadeiro ou falso, entre outros, a avaliação de questões discursivas pode ser realizada, mas sua correção tem que ser feita manualmente. Com o processamento de linguagem natural (PLN), uma subárea da inteligência artificial, será realizada a partir de uma pré-avaliação de questões discursivas de forma a guiar os tutores na correção.

A proposta é desenvolver uma forma de analisar a resposta discursiva com um gabarito definido, com algoritmos de similaridade de texto como *latent semantic analysis* (LSA) e STASIS, podemos comparar o quanto que dois textos são similares. Utilizando a similaridade entre as frases dos textos, será calculado uma nota para a resposta discursiva. Uma vez que o tutor der uma nota que não seja a proposta pelo sistema, um algoritmo genético analisa quais elementos do texto implicou na diferença na nota. Essa diferença é armazenada, para o sistema aprender qual parte do texto tem maior ou menor nota, quais partes do texto do gabarito tem maior ou menor similaridade com a resposta, isso altera a nota final.

3.2 O processo

O processo da figura 12 foi estabelecido em fases, são elas: Inserir a questão, Disponibilizar questão, Responder questão, Correção da resposta enviada, Correção da

resposta pelo Tutor, Algoritmo genético, Disponibilizar notas, Olhar notas.

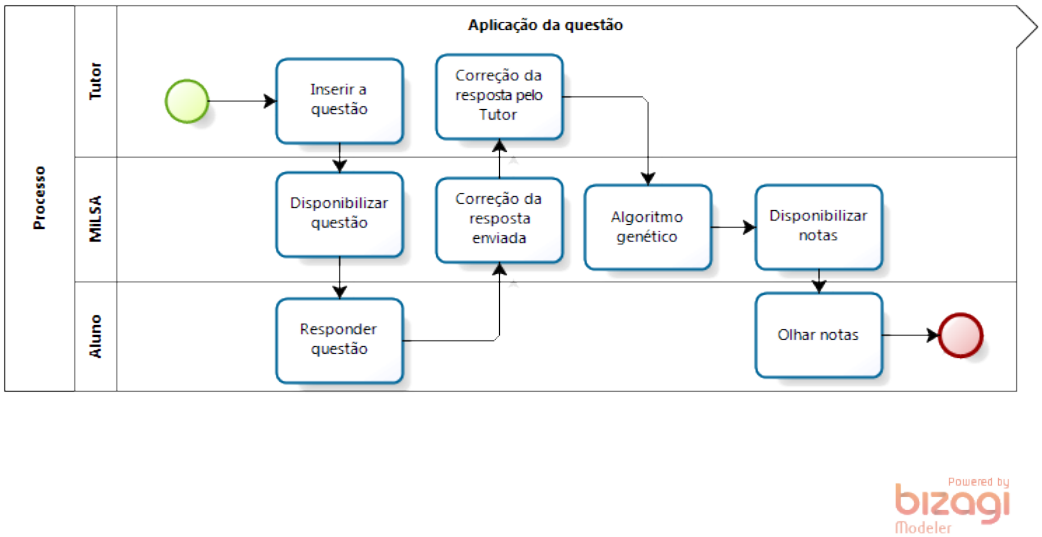


Figura 12 – Representa os passos no processo do sistema de apoio de correção de respostas dissertativas.

Inserir a questão: Para aplicar uma questão é necessário cadastra-la. Uma questão é composta por tipo, pergunta e gabarito, para os textos de gabarito foi definida uma linguagem de marcação, que permitisse marcar partes mais importantes do texto gabarito, ela foi definida da seguinte maneira:

Foram divididas as perguntas em dois tipos, as “Quais”, que serão uma enumeração sem necessariamente uma ordem dos itens, e as “Como”, onde a ordem dos itens enumerados será de importância à resposta.

A sintaxe:

Serão colocadas palavras, essas terão uma ordem, que pode ou não importar, dependendo do tipo da pergunta, essas palavras permitem agrupamentos por colchetes, [ ], que poderá ser alinhadas, e cada agrupamento permitirá operadores. São eles = (igual), $\sim$ (til), - (menos), $>$ (maior que), $<$ (menor que), que poderão ser combinados entre si, esses agrupamentos que definiram quais palavras serão importantes dentro da resposta.

Marcadores

- Colchetes [ ] defini prioridade de agrupamento nas palavras.
- - operador que defini que o agrupamento pode diminuir a pontuação do gabarito.
- = operador que defini que o agrupamento pode ser substituído por sinônimos.
- $\sim$ operador que defini que o agrupamento pode ser substituído por um tesauro.

- > operador que defini que o agrupamento pode ser substituído a parte pelo todo.
- < operador que defini que o agrupamento pode ser substituído o todo pela parte (metonímia).

Os operadores > (maior que),< (menor que), além de definir sobre os agrupamentos, também aceitam um peso, logo após sua ocorrência, para saber quantos níveis deveram subir ou descer.

Exemplo: Como se faz miojo?

Tenha um [pacote]= de [miojo], tenha um [fogão]= , tenha [agua] e uma [panela]=, quebre o miojo dentro do pacote, coloque a [[panela]= com água] no [fogão]= , [coloque o [conteúdo]= do [pacote]= de [miojo na [panela]=]], [espera] de 5 a 10 [minutos]>< , está pronto.

Outro conceito que foi definido é o de "termos", onde seria extraído do texto gabarito as palavras que estão demarcadas com colchetes, e elas virariam termos que o texto resposta posteriormente terá que ter.

Uma vez escolhido o tipo de questão, escrito a questão e definido um gabarito para a questão usando a sintaxe definida, essa questão seria salva pelo tutor.

Disponibilizar questão: O tutor disponibiliza uma questão para os alunos enviarem respostas.

Responder questão: O aluno responde a questão.

Correção da resposta enviada: Uma vez disponibilizado a questão pelo tutor o sistema irá receber as respostas enviadas pelos alunos, e irá definir uma nota as respostas recebidas.

Correção das respostas pelo tutor: Uma vez fechada a questão, será mostrado ao tutor à nota, o texto resposta e as frases associadas. As opções que ele tem no sistema são o de aceitar a nota dada, que o sistema seguirá normalmente, ou poderá recusar a nota, onde o sistema irá perguntar qual nota seria mais adequada e pedir para que marque as frases relacionadas a questão. Seguindo essa linha, o sistema altera a nota e aprende com o novo exemplo, sem alterar as notas anteriores e posteriores, mas aplicará aquele novo exemplo em aplicações futuras.

Algoritmo genético: Uma vez que o tutor tenha corrigido uma questão, o sistema analisa as alterações feitas pelo tutor, com algoritmo genético essas alterações se tonam um aprendizado para o sistema.

Disponibilizar notas: Uma vez que o tutor tenha corrigido todas as notas, ele disponibilizará no sistema as notas.

Olhar notas: O aluno pode entrar no sistema para visualizar a nota.

3.3 Análise do texto

A análise do texto, ou análise das respostas, foi definido no processo da figura 13, dividido em: Quebrar o texto em frases, Marcar frases com termos, Verificar melhor aderência e Calcular nota.

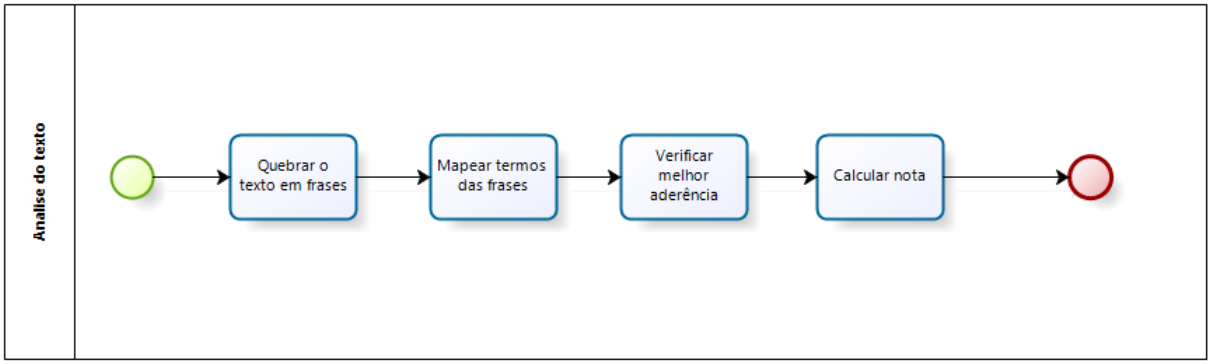


Figura 13 – Representa os passos no processo de análise de texto.

Quebrar o texto em frases: Para a análise do texto foi considerado a quebra do mesmo de acordo com as virgulas.

Mapear termos das frases: Cada frase será analisada para verificar se ela contém algum termo incluso, caso tenha, será analisada a aderência dessa frase com a frase do gabarito que contenha o mesmo termo.

Verificar melhor aderência: O peso de aderência é o valor mínimo a ser alcançado pela similaridade entre as frases, a partir da divisão feita no cadastro das questões. O *Latent Semantic Analysis* (LSA) será usado para as questões definidas como "Quais", e o STASIS para as questões definidas como “Como”. Isso se deve ao fato que o STASIS leva em consideração á ordem das sentenças. O fator de aderência será iniciado em 0,3, uma vez que temos mapeado quais frases tem quais termos, se duas frases contenham o mesmo termo será passado a que tem maior similaridade com a frase gabarito correspondente. Caso uma frase não seja mapeada a nem um termo, ela será testada com as frases do gabarito que não encontraram nem uma frase que passaram do fator de aderência. Para ser aceito a similaridade terá que ser no mínimo o mesmo valor de aderência. Por exemplo, se temos uma frase1 com o termo1, da resposta, e uma frase2 com o termo1 do gabarito, que foi considerado a melhor similaridade entre os termos, e tenha similaridade de 80% e o fator de aderência seja de 90%, esse termo não foi aceito. Podemos ver na figura 14, o pseudocódigo de criação da matriz de similaridade.

```
matrizDeSimilaridade = [[][]]

for i in len(frasesGabarito):
    for j in len(frasesResposta):
        nota=0
        if (tipo == 'Quais'):
            nota = LSA(frasesGabarito[i],frasesResposta[j])
        elif(tipo == 'Como'):
            nota = stasis(frasesGabarito[i],frasesResposta[j])
        matrizDeSimilaridade[i][j]=nota
```

Figura 14 – Pseudocódigo de criação da matriz de similaridade.

Calcular nota: A nota será calculada de acordo com as frases aceitas, e o peso que essas frases tenham no sistema, assim sendo um somatório das frases aceitas vezes os pesos da frases.

3.4 Algoritmo genético

O algoritmo genético nivela os pesos de cada frase sobre o peso da nota e o fator de aderência para cada frase. Inicialmente cada frase terá seu peso igual na soma final de cada nota. Por exemplo, se temos quatro frases, cada uma tem influência inicial em 25% da nota final, um fator de aderência mínimo foi arbitrado em 0.3, para o algoritmo genético tem as propostas de:

- Ajustar a influência das frases na nota final, acordo com a nota do tutor.
- Ajustar os pesos do fator de aderência, se a frase foi aceita ou não pelo tutor.

Caso o tutor, aceite a frase1 da resposta e na frase2 do gabarito, ele pode ou não alterar a nota, e esses dois fatores serão levados ao algoritmo genético para que seja recalculado tanto o fator de aderência quanto o peso que a frases tem sobre a nota final.

O gene montado para o algoritmo genético será composto com elementos decimais para o peso de cada frases sobre a nota final, elementos decimais para o fator de aderência e elementos decimais para a similaridade. A aderência ser aceita ou não, será o objetivo do algoritmo genético, encontrar a melhor forma de moldar os elementos anteriores com o *feedback* do tutor.

4 Desenvolvimento da Proposta

4.1 Descrição do projeto

O sistema de apoio à correção de respostas dissertativas é uma solução de software, para plataforma *web*, onde um tutor pode cadastrar questões de respostas dissertativas. Quando respondidas o sistema fará uma pré-avaliação da resposta, dando ao tutor uma nota que o guie na correção. Sendo aberto para o aprendizado, com as técnicas de aprendizado de maquina, o sistema irá melhorar a pré-avaliação com a interação do tutor.

4.2 Proposta

O sistema desenvolvido para plataforma web, onde um tutor consiga gerenciar suas questões dissertativas, assim sendo, ele cadastrará uma questão e gabarito, disponibilizará para receber respostas, corrigirá questão e disponibilizará notas. A partir do gabarito da resposta, que foi inserido usando a sintaxe de marcação, o processamento de linguagem natural e algoritmos de similaridade determinam uma nota para a resposta. Com a interação com o tutor, o algoritmo genético melhora a função da pré-avaliação, com isso se espera demandar menos tempo de correção pelo tutor.

Para o seu desenvolvimento será usado algumas práticas do *Scrum*, o uso de *sprints*, de duração de duas semanas, a organização das histórias em um *backlog* do produto e o *backlog* da *sprint*.

4.3 O sistema

Foram identificados no sistema os perfis de Tutor e Aluno, e o comum entre eles o usuário, de acordo com o perfil o sistema dará as seguintes ações:

- Tutor: Cadastrar questões, Avaliar as notas, Disponibilizar notas.
- Aluno: Responder as questões, Visualizar correção.
- Usuário: Logar no sistema.

Foram derivadas histórias de usuário para as ações, na tabela 5 podemos ver o *Backlog* do produto inicial.

Para todas as histórias de usuário foram derivadas tarefas a serem realizadas, abaixo temos as tabelas 6, 7, 8, 9, 10, 11, 12.

Tabela 5 – *Backlog* do Produto

Número	História de usuário	Pontos
1	Eu, como Usuário, quero me autenticar no sistema para que meu perfil seja identificado.	5
2	Eu, como Tutor, quero cadastrar questões e gabaritos para aplicar aos meus alunos.	18
3	Eu, como Tutor, quero que as respostas inseridas há minha questão sejam corrigidas pelo sistema.	18
4	Eu, como Aluno, quero responder a questões para que eu possa ser avaliado.	2
5	Eu, como Tutor, quero poder mudar a nota dada pelo sistema para corrigir a divergência de nota dada.	8
6	Eu, como Tutor, quero poder disponibilizar as notas no sistema para meus alunos.	2
7	Eu, como Aluno, quero ver minha correção para que eu possa entender como minha resposta foi avaliada.	2
	Total	55

Tabela 6 – Eu, como usuário, quero logar no sistema para que meu perfil seja identificado.

Tarefa	Dificuldade
Criar tabela no banco para usuários	2
Criar tabela no banco para perfis	2
Criar tela para login	2

Tabela 7 – Eu, como Tutor, quero cadastrar questões e gabaritos para aplicar aos meus alunos.

Tarefa	Dificuldade
Criar tabela no banco para questões	2
Criar algoritmo para extrair as frases do gabarito	5
Criar algoritmo para extrair as termos das frases	5
Criar tela para cadastrar questões	5

Tabela 8 – Eu, como Tutor, quero que as respostas inseridas há minha questão sejam corrigidas pelo sistema

Tarefa	Dificuldade
Criar tabela no banco para respostas	2
Criar algoritmo para analisar existência de termos das frases	5
Criar algoritmo para analisar melhor aderência	8
Criar algoritmo para calcular nota	5
Criar tela para cadastrar respostas	2

Tabela 9 – Eu, como Aluno, quero responder a questões para que eu possa ser avaliado.

Tarefa	Dificuldade
Criar tela para responder questões	2

Tabela 10 – Eu, como Tutor, quero poder mudar a nota dada pelo sistema para corrigir a divergência de nota dada.

Tarefa	Dificuldade
Criar algoritmo genético para balancear os pesos	13
Criar tela de correção de questões	2

Tabela 11 – Eu, como Tutor, quero poder disponibilizar as notas no sistema para meus alunos.

Tarefa	Dificuldade
Criar tela de disponibilizar notas	2

Tabela 12 – Eu, como Aluno, quero ver minha correção para que eu possa entender como minha resposta foi avaliada.

Tarefa	Dificuldade
Criar tela de visualizar correção	2

4.4 Definição das tecnologias

Para o processamento de linguagem natural, citado por (ZHU; LAN, 2013) e (CROFT et al., 2013), temos a biblioteca *Natural Language Toolkit*(NLTK)(BIRD; KLEIN; LOPER, 2009). Desenvolvida em *python*, ela suporta diversas funcionalidades necessárias ao processamento de linguagem natural, sendo licenciada pela Apache License Version 2.0, tendo seu uso livre para desenvolvimento e comercialização. Outra biblioteca usado foi a *Gensim*, sendo licenciada pela GNU LGPLv2.1 license, tento seu uso livre para uso comercial, mas alterações na biblioteca devem ser abertas.

Para o desenvolvimento do sistema web, será usado o padrão de desenvolvimento MVC e o paradigma de programação Orientação a Objetos. Com a necessidade de se usar a biblioteca do *Natural Language Toolkit*(NLTK)(BIRD; KLEIN; LOPER, 2009), o desenvolvimento web será feito em linguagem Python, com o *framework Django*(DJANGO..., 2013), que nos permite o desenvolvimento web com Python.

Para o banco de dados será usado o PostgreSQL.

Para o controle de versão de código será usado o Git, de forma local.

4.5 Cronograma de desenvolvimento

Para o TCC1 foi montado o cronograma na tabela 13, nele estão às atividades realizadas.

A proposta de execução para o TCC2 está descrita no cronograma na tabela 14, temos a execução do desenvolvimento, aplicação do sistema em um estudo de caso, a análise dos dados e escrita da conclusão.

Tabela 13 – Cronograma de trabalho para o TCC1

Atividade	8/2015	9/2015	10/2015	11/2015
Criar 10 perguntas base	X			
Criar proposta de TCC1	X			
Pesquisa sobre artigos de *semantic text similarity*	X			
Definir linguagem formal		X		
Estudo dos algoritmos de similaridade		X		
Estudo dos algoritmos de *machine learning*		X		
Pesquisa de implementações de algoritmos genéticos			X	
Teste dos algoritmos (similaridade e genéticos)			X	
Organização das referências			X	
Escrita do TCC 1 Capítulo 3-Proposta				X
Escrita do TCC 1 Capítulo 2-Referencial teórico				X
Escrita do TCC 1 Capítulo 4-Desenvolvimento da Proposta				X
Escrita do TCC 1 Capítulo 1-Introdução				X

Tabela 14 – Cronograma de trabalho para o TCC2

Etapa	2/2016	3/2016	4/2016	5/2016	6/2016
Desenvolvimento do sistema	X	X	X		
Aplicação e testes do sistema em um caso de uso			X	X	
Análise dos dados			X	X	
Escrita do TCC2				X	X

Execução do desenvolvimento: Usando o formato de *sprint*, onde cada *sprint* teve duas semanas, temos a execução das *sprints* na tabela 15, as datas de início e fim da *sprint*, histórias desenvolvidas e a pontuação total da *sprint*:

Tabela 15 – Disposição das histórias nas *sprints*

Sprint	Início	Término	Histórias	Total de pontos
1	8 de fevereiro de 2016	19 de fevereiro de 2016	2	18
2	22 de fevereiro de 2016	4 de março de 2016	3	18
3	7 de março de 2016	18 de março de 2016	4,5	10
4	21 de março de 2016	1 de abril de 2016	1,6,7	9

Aplicação do sistema em um estudo de caso: Uma vez terminada a implementação do sistema, ele será aplicado a um estudo de caso na disciplina de Fundamentos de Arquitetura de Computadores(FAC).

Análise dos dados:

- Acurácia da nota, avaliação do sistema versus nota do tutor.
- Aperfeiçoamento do sistema para cada questão, com o algoritmo genético.

Escrita do TCC2: A escrita do TCC 2 inclui a sumarização dos dados de execução das *sprints*, os dados da aplicação do sistema em um estudo de caso e a análise dos dados, considerações finais e trabalhos futuros.

4.6 Desenvolvimento

É explicado o que ocorreu durante o desenvolvimento e implantação do sistema proposto, citando as alterações que ocorreram, implementação alcançada, os casos de teses, resultados e analise dos dados.

Foi seguido o cronograma da tabela 15, onde foram desenvolvidas as histórias da tabela 5. Foi priorizado fazer a parte de analise do texto, referente a figura 13, que consiste no foco desse trabalho, e depois uma interface web para uso nos testes que foram definidos.

Para a implementação dos algoritmos de similaridade foram utilizadas bibliotecas e implementações já existentes. Para o STASIS foi usado uma implementação já existente do artigo de LI (LI; BANDAR; MCLEAN, 2003), essa implementação foi modificada, a modificação feita foi no uso do corpus da língua inglesa, chamado brown, trocado para o corpus da língua portuguesa, chamado floresta. Para o LSA foi utilizado uma implementação com a biblioteca Gensim.

Quando o desenvolvimento do processo de analise do texto, da figura 13 foi concluído, foi criada uma questão teste e aplicado para um grupo seleto. Sendo coletadas 9 respostas, e manualmente, por meio de interface de terminal, foram inseridas no processo de analise do texto. Foram feitas 22 comparações de texto, na tabela 16 podemos ver a porcentagem de similaridade de cada uma, com esses valores, foi arbitrado o valor inicial de aderência de 0,3.

Na implementação do algoritmo genético foram arbitrados valores para a população inicial, igual a 16, e para a quantidade de gerações, igual a duas gerações. A iniciação da população se da por forma aleatória dentro de um nicho, como descrito por Silva (SILVA, 2005), após as duas gerações é escolhido o melhor valor dentro da população.

Esses valores arbitrários foram colocados para ter um algoritmo inicial, é esperado que, com a aplicação de testes do sistema, esses valores sejam alterados para valores que melhor evoluam a nota dada pelo sistema.

A parte de interface foi desenvolvida para acesso via web, utilizando *framework Django*. Inicialmente para acesso do sistema foi definido dois perfis, um de tutor e outro de aluno, o tutor poderá acessar o sistema para cadastrar questões, apagar questões e analisar questões, e o aluno responderá questões. O sistema foi chamado de MiLSA, podemos ver na figura 15 a tela inicial do sistema MiLSA e na figura 16 a tela de cadastro de questões.

Além das telas de cadastro de questões e inicial, existem as telas listadas abaixo,

Tabela 16 – Porcentagem de similaridade

Aderência
0.000000
0.000000
0.000000
0.044907
0.046836
0.051002
0.080735
0.119411
0.122161
0.135184
0.187743
0.219628
0.256074
0.264384
0.343579
0.357758
0.421830
0.426098
0.439905
0.457475
0.521325
0.651951

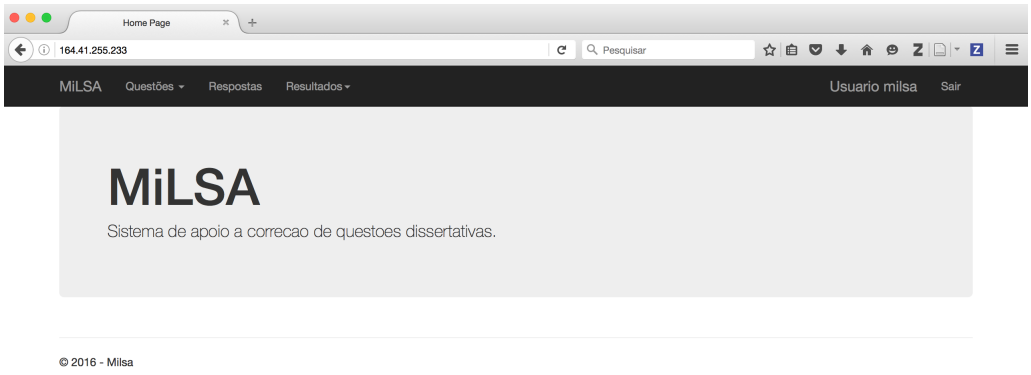


Figura 15 – Representa a tela inicial do sistema de apoio de correção de respostas dissertativas.

que podem ser vistas no apêndice C:

- Listar questões.
- Detalhar questão.
- Listar perguntas para serem respondidas.
- Responder questão.

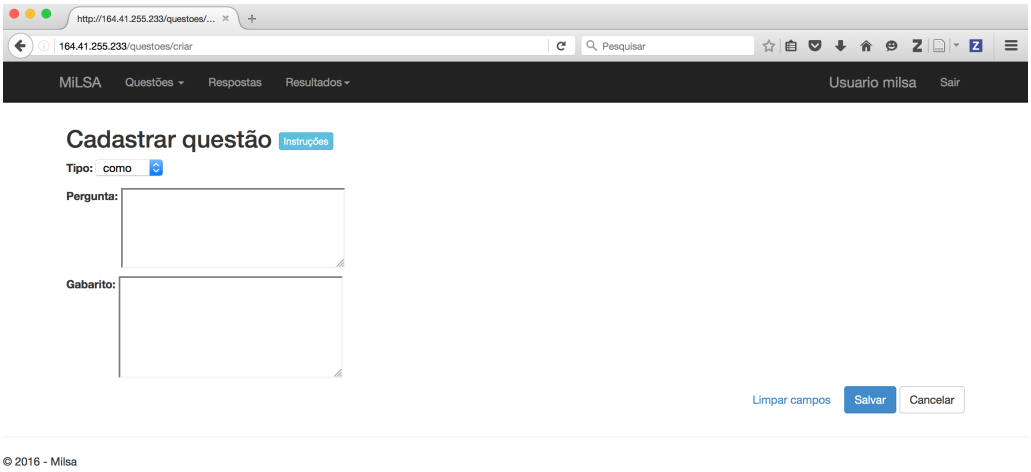


Figura 16 – Representa a tela de cadastro de questões do sistema de apoio de correção de respostas dissertativas.

- Listar respostas para serem avaliadas.
- Avaliar resposta.
- Tela de *login*.

4.6.1 Arquitetura

No *framework Django* a arquitetura é MVT(Model, View, Template), por comparação com o MVC(Model, View, Controller), que é o modelo proposto, a Model(MVT) é a Model(MVC), a View(MVT) é a Controller(MVC) e o Template(MVT) é o View(MVC).

Na model, existem os modelos de dados, vistos no banco de dados. Na view, existem os códigos de controle sobre as operações de criar, listar e apagar questões, responder questões, avaliar resposta, controle de acesso por períodos e distribuição de questões. Na url, existem os mapeamentos usados para uma url chamar uma função da view. No template, existem os códigos html da formação da estrutura da pagina web. No Apoio, existe o código de analise dos textos respostas com os textos gabaritos.

Podemos ver a arquitetura na figura 17 e o modelo do banco na figura 18.

Com o sistema MiLSA desenvolvido, foi implantado em um servidor para testes.

4.7 Aplicação Caso de Uso

Para o teste do sistema MiLSA, foi utilizado a disciplina de fundamentos de arquitetura de computadores(FAC), que atualmente está em processo de gamificação. Dentro desse processo de gamificação o MiLSA foi colocado como uma missão que os alunos deveriam executar, onde eles criaram questões, responderam questões e avaliaram questões.

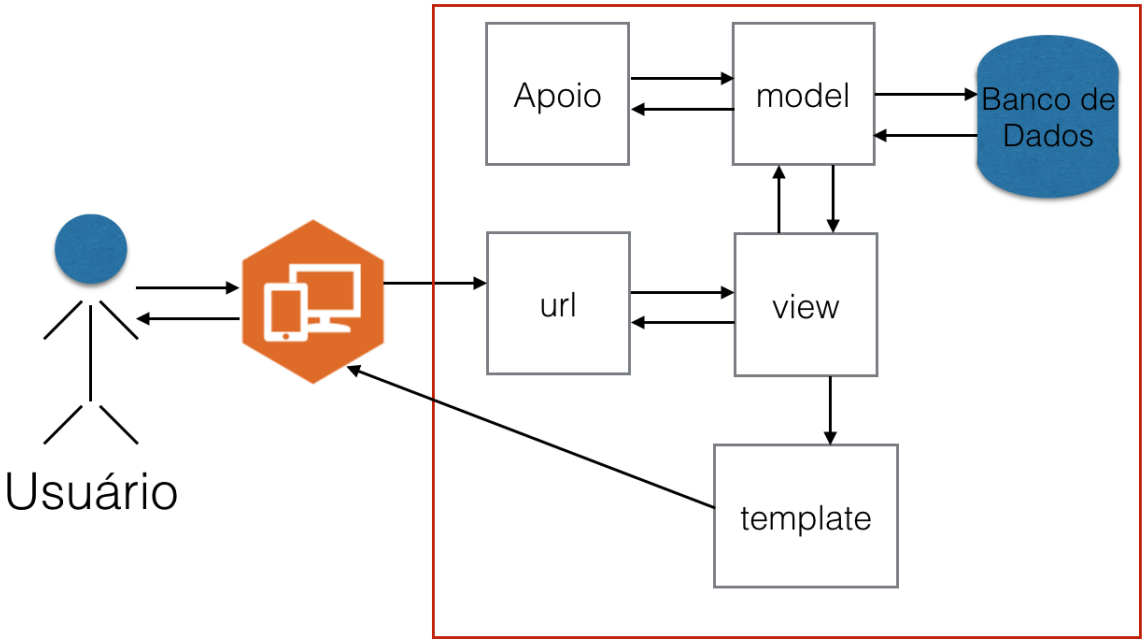


Figura 17 – Arquitetura do sistema.

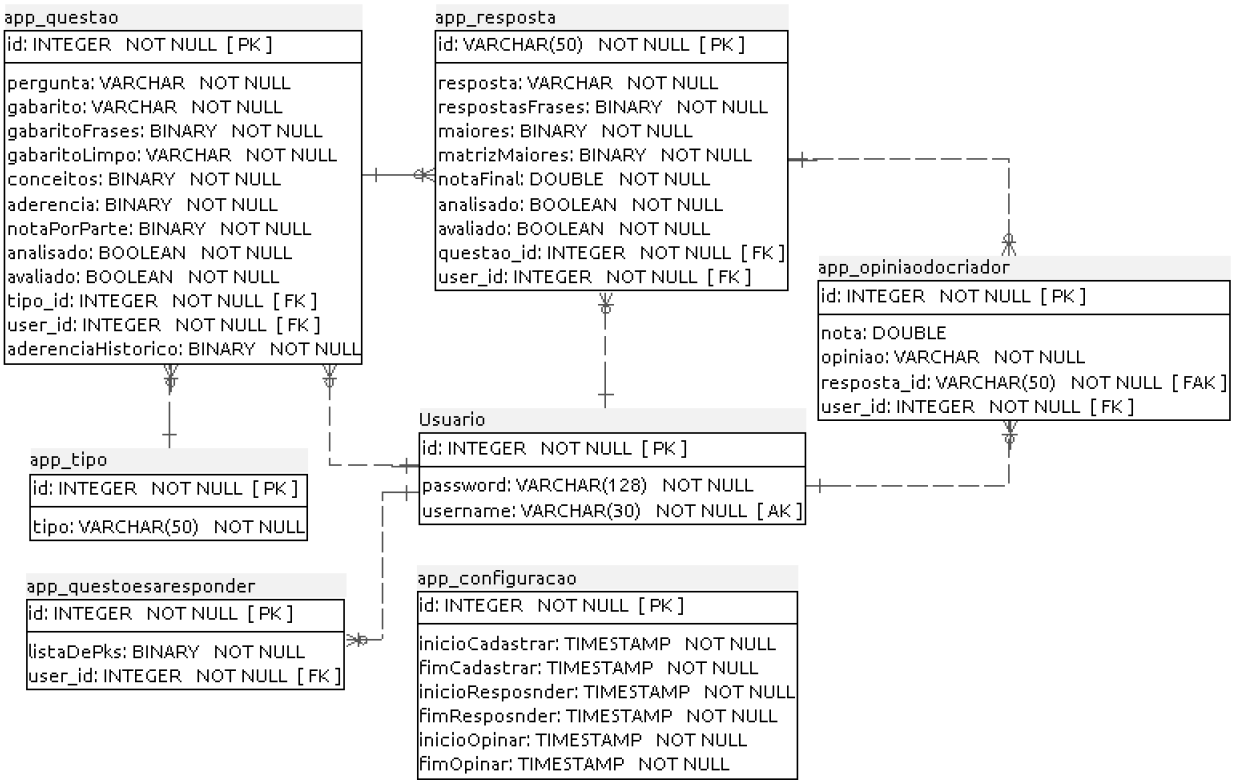


Figura 18 – Representação do modelo de dados.

Para execução da missão, foi criado um perfil de acesso para os alunos da disciplina, esse perfil continha o acesso tanto a cadastrar questões, apagar questões e analisar questões, que são do perfil do tutor, tanto a responder questões, que é o perfil do aluno.

Foram criados *logins* para cada aluno da disciplina de fundamentos de arquitetura de computadores, totalizando 39 *logins*.

4.7.1 A missão

A missão era dividida em três etapas:

- A primeira etapa, cadastro de questões, onde poderiam ser cadastradas um limite de questões por *login*.
- A segunda etapa, responder questões, onde o sistema selecionava aleatoriamente questões para o usuário responder, sendo que o número de questões a serem respondidas era igual ao número de questões que ele havia cadastrado.
- A terceira etapa, avaliar a nota dada pelo sistema, tendo respostas a sua questão, o usuário inserio a nota que ele achasse correta e uma justificativa.

Dentro da disciplina FAC(Fundamentos de arquitetura de computadores) o MiLSA tem três aplicações:

- Primeira aplicação, 04/04-07/04.
- Segunda aplicação, 30/05-03/06.
- Terceira aplicação, 27/06-01/07.

4.7.2 Primeira missão

A primeira missão foi aplicada seguindo o cronograma da tabela 17, essa missão foi limitada a duas perguntas por *login*, e seu objetivo era ter os dados da diferença da nota dada do sistema MiLSA e a nota dada pelo autor da questão.

Tabela 17 – Cronograma da primeira aplicação

Datas	etapa
04/04 - 00:00 a 05/04 - 11:55	1
05/04 - 12:00 a 06/04 - 23:55	2
07/04 - 12:00 a 09/04 - 23:55	3

Com essa aplicação prática do sistema MiLSA, conseguimos 64 questões cadastradas, 60 respostas e 35 opiniões.

Como o objetivo foi a diferença da nota dada do sistema MiLSA e a nota dada pelo autor da questão, usamos para analise as 35 opiniões.

Podemos ver na tabela 18 a diferença da média e desvio padrão e na tabela 19 a frequência das notas. A primeira coluna sendo o intervalo da nota, nas colunas seguintes temos a quantidade de notas para aquele intervalo.

Tabela 18 – Estatísticas de dispersão da primeira aplicação.

	Autor	MiLSA	Diferença
Média	8,60	7,03	1,37
Desvio padrão	2,80	3,66	-0,86

Tabela 19 – Frequência das notas.

[a [	Autor	MiLSA
0 a 1	1	4
1 a 2	0	0
2 a 3	2	2
3 a 4	1	3
4 a 5	0	0
5 a 6	2	3
6 a 7	1	2
7 a 8	0	2
8 a 9	1	1
9 a 10	1	0
10	26	18

4.7.3 Segunda missão

A segunda missão foi aplicada seguindo o cronograma da tabela 20, essa missão foi limitada a três perguntas por *login*. Seu objetivo era aplicar o algoritmo genético na avaliação das notas, e com isso comparar a diferença entre a nota sem a aplicação do algoritmo genético, a nota aplicando o algoritmo genético e a nota do autor.

A terceira etapa além da nota que o autor achasse correta e uma justificativa, foi pedido se aceitava ou não cada frase avaliada pelo sistema.

Tabela 20 – Cronograma da segunda aplicação.

Datas	etapa
30/05 - 00:00 a 31/05 - 11:55	1
31/05 - 12:00 a 01/06 - 23:55	2
02/06 - 12:00 a 03/06 - 23:55	3

Com essa segunda aplicação prática do sistema MiLSA, conseguimos 94 questões cadastradas, 89 respostas e 84 opiniões.

Os dados obtidos na segunda aplicação estão em analise, das 84 opiniões, apenas 28 apresentaram diferença de opinião do sistema.

Podemos dizer, inicialmente, que antes de aplicar o algoritmo genético no modelo de avaliação o erro médio foi 2,964. Porém quando aplicado o algoritmo de IA o erro reduziu para 2,141.

Abaixo, uma questão retirada da aplicação do MiLSA2.

Questão

- Tipo: Quais
- Pergunta: Quais portas lógicas são necessárias para a construção de um multiplexador de oito entradas com a solução mais otimizada?
- Gabarito: 3 inversoras, 8 ands de 3 entradas e uma or de 8 entradas.
- Frases: '3 inversoras', ' 8 ands de 3 entradas e uma or de 8 entradas.'
- Aderência: [0.3,0.3]
- Nota por frase: [5.0,5.0]

Resposta

- Resposta: Pode ser construído com 3 portas not , 8 portas and com 3 entradas e uma porta or com 8 entradas.
- Frases: 'Pode ser construído com 3 portas not ', ' 8 portas and com 3 entradas e uma porta or com 8 entradas.'
- Nota Final: 10.0

Podemos ver a análise entre questão e resposta nas tabelas 21 e 22. Mostrando a melhor similaridade.

Tabela 21 – Matriz de similaridade.

	FraseResposta1	FraseResposta2
FraseGabarito1	0.98319203	0.35901096
FraseGabarito2	0.27268839	0.99571729

Tabela 22 – Melhor similaridade.

	FraseResposta1	FraseResposta2
FraseGabarito1	0.98319203	0.35901096
FraseGabarito2	0.27268839	0.99571729

A nota da primeira frase do gabarito foi aceita, por sua similaridade ser maior que 0.3, a nota da segunda frase do gabarito foi aceita, por sua similaridade ser maior que 0.3, somando as notas de cada frase, 5.0+5.0, temos a nota 10.0.

4.7.4 Terceira missão

A terceira missão será seguindo o cronograma da tabela 23, ainda não realizado.

A terceira missão tem o mesmo objetivo da segunda, comparar a diferença entre a nota sem a aplicação do algoritmo genético, a nota aplicando o algoritmo genético e a nota do autor.

Esperamos ter com essa terceira aplicação, resultados que apontem melhorias para o algoritmo genético.

Tabela 23 – Cronograma da terceira aplicação.

Datas	etapa
27/06 - 00:00 a 28/06 - 11:55	1
28/06 - 12:00 a 29/06 - 23:55	2
20/06 - 12:00 a 01/07 - 23:55	3

5 Conclusão e trabalhos futuros

Com o foco em testar o sistema de correção, o escopo inicial do planejado foi adaptado e reduzido para focar na realização dos testes. Com a aplicação dos testes foram obtidos dados para análise.

É possível um sistema que usando LSA, STASIS e Algoritmo genético, faça a comparação de um gabarito definido, esse com sua sintaxe de marcação, e uma resposta, apontando uma nota para apoiar a correção do tutor.

O apontamento da nota pelo sistema teve uma distribuição ponderada dentro do intervalo de 0 a 10, não apresentando um estilo de correção 0 ou 10.

A correção apresentou em alguns casos uma alta cobertura, com baixa precisão. Tendo que a resposta contenha todo o conteúdo do gabarito, essa obteria uma nota ideal, mas a resposta pode conter algum conteúdo excedente ao gabarito sem afetar a nota final negativamente.

O algoritmo genético demostrou na sua primeira aplicação, uma melhoria na diferença média entre as notas dadas pelo autor e sistema.

Esse trabalho se mostrou bem sucedido ao seu objetivo, sendo desenvolvido e implantado para testes. Testado com usuários reais e obtendo dados que se demostraram significativos nas analises.

Os trabalhos futuros são estudos e melhorias ao sistema. Como proposta de trabalhos futuros, temos a lista abaixo.

- Criar a interface para suporte a múltiplas disciplinas.
- Testar o algoritmo genético com o retorno de aceitação das frases pelo autor, para aperfeiçoamento do sistema(Em andamento).
- Criar uma formar de evitar uma alta cobertura sem precisão.
- Analisar o impacto do tipo da questão na nota.
- Analisar o impacto da marcação do texto gabarito na nota.

Referências

ABRAEAD. *Anuário Brasileiro Estatístico de Educação Aberta e a Distância*. 2008. Citado 2 vezes nas páginas 15 e 27.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python*. 1st ed. ed. Beijing ; Cambridge [Mass.]: O'Reilly, 2009. ISBN 978-0-596-51649-9. Citado na página 63.

BOOCH, G. Object-oriented development. *IEEE Transactions on Software Engineering*, SE-12, n. 2, p. 211–221, fev. 1986. ISSN 0098-5589. Citado na página 54.

CARRIJO, I. B. *Extração de regras operacionais ótimas de sistemas de distrubuição de água através de algoritmos genéticos multiobjetivo e aprendizado de máquina*. Tese (Doutorado) — Universidade de São Paulo, Escola de Engenharia de São Carlos, 2004. Disponível em: <http://bases.bireme.br/cgi-bin/wxislind.exe/iah/online/?IsisScript=iah/iah.xis&src=google&base=LILACS&lang=p&nextAction=lnk&exprSearch=415521&indexSearch=ID>. Citado 14 vezes nas páginas 15, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48 e 49.

CROFT, D. et al. A fast and efficient semantic short text similarity metric. In: *2013 13th UK Workshop on Computational Intelligence (UKCI)*. [S.l.: s.n.], 2013. p. 221–227. Citado na página 63.

DAVIS, L. (Ed.). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, 1991. ISBN 978-0-442-00173-5. Citado 2 vezes nas páginas 44 e 48.

DEERWESTER, S. et al. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, v. 41, n. 6, p. 391–407, 1990. Citado na página 33.

DJANGO (Versão 1.5). 2013. Disponível em: <https://djangoproject.com>. Citado na página 63.

FILHO, A. R. P. *A Atividade Questionário em Moodle*. 2009. Disponível em: <http://giselebrugger.com/tutorial/questionarios_moodle_1_9_3.pdf>. Citado na página 27.

GALVAO, C. d. O. *Sistemas inteligentes aplicações a recursos hídricos e ciências ambientais*. Porto Alegre: UFRGS: ABRH, 1999. ISBN 978-85-7025-527-3. Citado na página 43.

GIL, A. C. *Como elaborar projetos de pesquisa*. São Paulo: Atlas, 2008. ISBN 978-85-224-3169-4. Citado na página 29.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 978-0-201-15767-3. Citado na página 43.

JUNIOR, J. R. C. *DESENVOLVIMENTO DE UMA METODOLOGIA PARA MINERAÇÃO DE TEXTOS*. Tese (TEXTO) — PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO, maio 2008. Citado 3 vezes nas páginas 31, 32 e 33.

KRASNER, G. E.; POPE, S. T. A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80. *J. Object Oriented Program.*, v. 1, n. 3, p. 26–49, ago. 1988. ISSN 0896-8438. Disponível em: <http://dl.acm.org/citation.cfm?id=50757.50759>. Citado 3 vezes nas páginas 15, 53 e 54.

LANDAUER, T. K.; FOLTZ, P. W.; LAHAM, D. An introduction to latent semantic analysis. *Discourse Processes*, v. 25, n. 2-3, p. 259–284, jan. 1998. ISSN 0163-853X, 1532-6950. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/01638539809545028>. Citado 4 vezes nas páginas 15, 33, 34 e 35.

LI, Y.; BANDAR, Z.; MCLEAN, D. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, v. 15, n. 4, p. 871–882, jul. 2003. ISSN 1041-4347. Citado 2 vezes nas páginas 34 e 65.

LI, Y. et al. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, v. 18, n. 8, p. 1138–1150, ago. 2006. ISSN 1041-4347. Citado na página 36.

MICHALEWICZ, Z.; ATTIA, N. Evolutionary Optimization of Constrained Problems. p. 98–108, 1994. Citado na página 43.

MICHALSKI, R. S.; BRATKO, I.; KUBAT, M. (Ed.). *Machine learning and data mining: methods and applications*. Reprint. Chichester: Wiley, 1998. ISBN 978-0-471-97199-3. Citado na página 37.

MITCHELL, T. M. *Machine learning*. International ed., [reprint.]. New York, NY: McGraw-Hill, 1997. (McGraw-Hill series in computer science). ISBN 978-0-07-115467-3. Citado na página 39.

PEREIRA, A. T. C.; SCHMITT, V.; DIAS, M. R. A. C. *Ambientes Virtuais de Aprendizagem: em diferentes contextos*. 2007. Rio de Janeiro: Ciência Moderna Ltda. Citado na página 27.

REZENDE, S. O. (Ed.). *Sistemas inteligentes: fundamentos e aplicações*. 1. ed. ed. Barueri, SP: Ed. Manole, 2003. ISBN 978-85-204-1683-9. Citado 10 vezes nas páginas 15, 17, 36, 37, 38, 39, 40, 41, 42 e 43.

RUSSELL, S.; NORVIG, P. *INTELIGÊNCIA ARTIFICIAL, 3E*. [S.l.: s.n.], 2013. ISBN 978-85-352-3701-6. Citado na página 31.

SCHWABER, K.; SUTHERLAND, J. *Guia do Scrum Um guia definitivo para o Scrum: As regras do jogo*. 2013. Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Citado 3 vezes nas páginas 51, 52 e 53.

SILVA, A. J. M. *IMPLEMENTAÇÃO DE UM ALGORITMO GENÉTICO UTILIZANDO O MODELO DE ILHAS*. Tese (Doutorado), 2005. Citado 9 vezes nas páginas 15, 17, 44, 45, 46, 47, 48, 49 e 65.

ZHU, T.-T.; LAN, M. Measuring short Text Semantic Similarity using multiple measurements. In: *2013 International Conference on Machine Learning and Cybernetics (ICMLC)*. [S.l.: s.n.], 2013. v. 02, p. 808–813. Citado na página 63.

APÊNDICE A – Primeiro Apêndice

[1]

An automatic essay correction for an active learning environment

Autor1, Autor2, and Autor3,

***Abstract*—In this paper, we describe and test a part of an active learning environment that is being used to engage and empower students' learning in a undergraduate engineering course. We developed a survey system, called Milsa, in order to insert questions and template answers, and to automatically correct the questions based on template answers. In Milsa, the students insert multiple questions and template answers, the questions alleatory distributed to multiple responsers. Milsa was used as an assessement in a gamified discipline [1]. After the automatic correction of the questions, the students evaluate the responses. The correction algorithm in Milsa is based on LSA [2], on STASIS [3] and on Michalski's genetic algoritm [4]. The template answer is a tagged text: hyperonymy, hyponymy, meronymy and synonymy are marked. By the end the tests on are showed and evaluated.**

Keywords*—Automatic essay correction, Active Learning, Genectic algorithm, STASIS, LSA.

I. Introduction

An interesting problem in the education is how someone can learn better. It seems that good teachers induce good students. However, is a better teaching the only way to learn better? Research in Education showed that there are many possibilities, in all students' ages. Active Learning (AL) [5] proposes a model of instruction that focuses on the responsibility of learning on learners. In AL, the students are immersed in a learning environment where the three learning domains - knowledge, skills and attitudes [6], [7] - are stimulated.

New generations make intensive use of technologies such as computer, mobile phones and video gamečitejan11. Those digital natives [8] do not care about reading instruction manuals or resort to technical expertise. They prefer to interact, to discover by themselves and to learn "immersed on the sea of information". Playing games are a common task to those generations.

Some researches [1], [8] suggest the use of games as a good methodology in engaging and motivating users. It was observed that such users can remain in a task for a long period of time, exclusively for game activity itself. In such virtual world, they experience a lot of sensations: adrenaline, adventure, mental challenge and the possibility of being in a fun activities alone or in the company of friends.

Gamification [9], [10], [11] is a new research area that entails the use of game-design elements and game principles in attempts to improve users' engagement, motivation and happiness on common tasks of everyday life. Gamification [9], [10] are being studied on different non-game context: organizational productivityčitejan11, learning [12], [13] and training, employee recruitment and evaluation [11].

Using Gamification in education of new generations seems a natural ideia, also in higher education. Gamification is, per se, an active learning methodology in which students become motivated self-directed learners. The challenge is to create a Gamification environment into a classroom space. Our gamification space is a game entitled "Battle of knowledge". It was implemented and tested on the Computer Architecture and Organization (CAO). CAO is a fourth semester discipline offered to both Software Engineering and Eletronic Engineering undergraduate courses at our University.

In the game, the assessments are called "missions". On a mission, every students must challenge their colleagues through a set of surveys. One of the CAO's learning outcomes is that students must learn how produce and evaluate knowledge about Computer Architecture. Missions are used to achieve such objective. A mission last a week and the student must produce questions, answer templates and evaluate/criticize answers. That's where comes up Milsa. The main component of Milsa is its automated correction system.

In the rest of the article, we detail the Milsa system as a part of the "Battle of knowledge" game. In Section 2 we present a brief summary of the CAO's and the Battle of Knowledge game. In section 3, we present the theoretical foundations for Milsa. In section 4, we present the Milsa correction algorithm. In section 5, we present the tests and evaluation. Finally, we have some conclusions and future research.

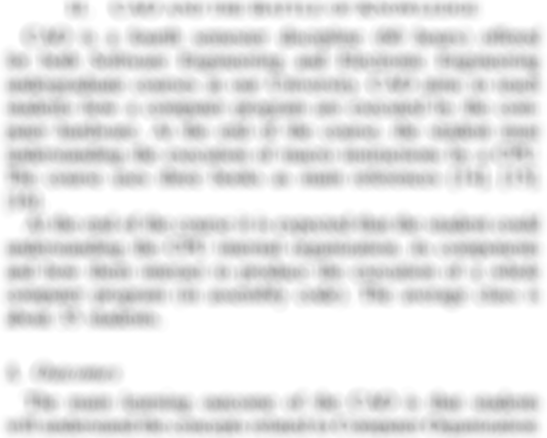


Figura 19 – Primeira página do artigo.

APÊNDICE B – Segundo Apêndice

An automatic correction and learning essay system

Abstract

In this paper, we propose and test an automatic correction and learning essay system. It was used to engage and empower students' learning in a gamified undergraduate engineering course. The survey system, called Milsa, is used to insert questions and template answers, to automatically correct the questions based on template answers and to learn from the users' feedback to an evaluate result. In Milsa, the students insert multiple questions and template answers, the questions aleatory distributed to multiple responders. Milsa was used as an assessment in a gamified [1]*. After the automatic correction of the questions, the students evaluate the responses. The correction algorithm in Milsa is based on LSA* [2] *on STASIS* [3] *and on Michalski's genetic algorithm* [4]*. The template answer is a tagged text: hypernym, hyponymy, meronym and synonymy are marked.*

1. Introduction

Active Learning (AL) [5] proposes a model of instruction that focuses on the responsibility of learning on learners. In AL, the students are immersed in a learning environment where the three learning domains - knowledge, skills and attitudes [6,7] - are stimulated.

New generations make intensive use of technologies such as computer, mobile phones and video game [1]. Those digital natives [8] do not care about reading instruction manuals or resort to technical expertise. They prefer to interact, to discover by themselves and to learn "immersed on the sea of information". Playing games are a common task to those generations.

Some researchers [1,8] suggest the use of games as a good methodology in engaging and motivating users. It was observed that such users can remain in a task for a long period of time, exclusively for game activity itself. In such virtual world, they experience a lot of sensations: adrenaline, adventure, mental challenge and the possibility of being in fun activities alone or in the company of friends.

Gamification [9,10,11] is a new research area that entails the use of game-design elements and game principles in attempts to improve users' engagement, motivation and happiness on common tasks of everyday life. Gamification [9,10] are being studied on different non-game context: organizational productivity [1], learning [12] and training, employee recruitment and evaluation [11].

Using Gamification in education of new generations seems a natural idea, also in higher education. Gamification is, per se, an active learning methodology in which students become motivated self-directed learners. The challenge is to create a Gamification environment into a classroom space. Our gamification space is a game entitled "Battle of knowledge". It was implemented and tested on the Computer Architecture and Organization (CAO). CAO is a fourth semester discipline offered to both Software Engineering and Electronic Engineering undergraduate courses at our University.

In the game, the assessments are called "missions". On a mission, every student must challenge their colleagues through a set of surveys. One of the CAO's learning outcomes is that students must learn how produce and evaluate knowledge about Computer Architecture. Missions are used to achieve such objective. A mission last a week and the student must produce questions, answer templates and evaluate/criticize answers. That's where comes up Milsa. The main component of Milsa is its automated correction system.

This article structure is: in section 2 we present a brief summary of the CAO's and the Battle of Knowledge game. In section 3, we present the theoretical foundations for Milsa. In section 4, we present the Milsa correction algorithm. In section 5, we present the tests and evaluation. Finally, we have some conclusions and future research.

Figura 20 – Primeira página do artigo.

APÊNDICE C – Terceiro Apêndice

Imagens das telas do sistema MiLSA

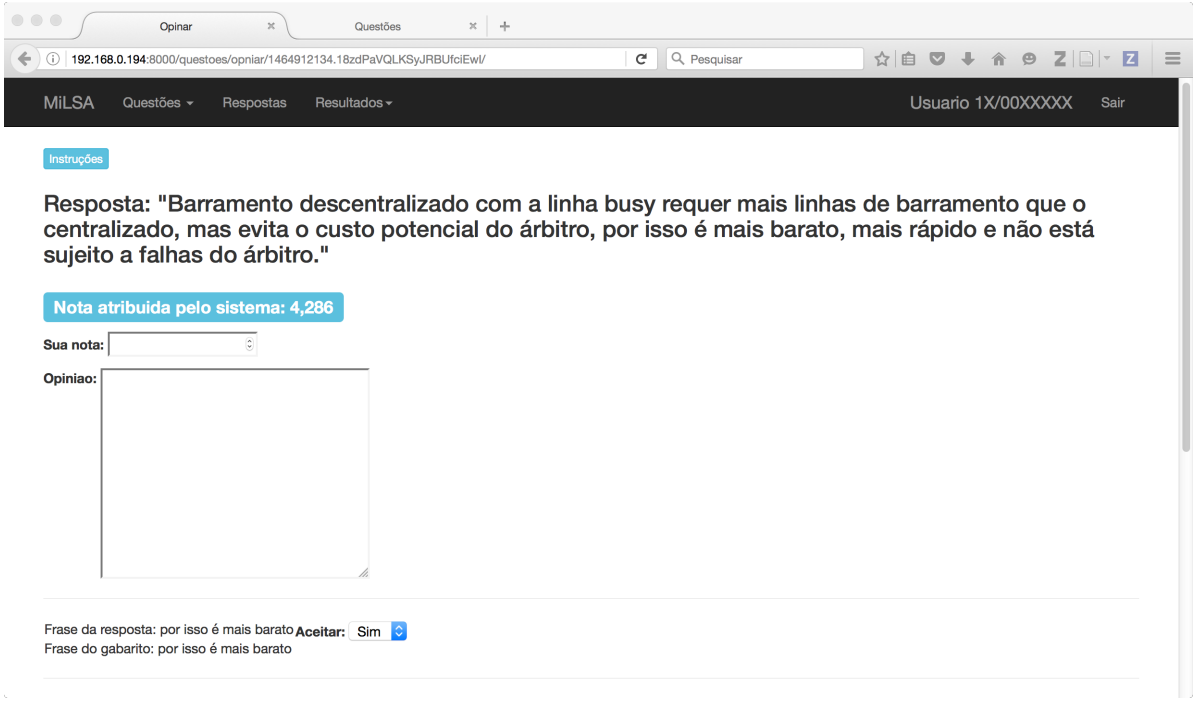


Figura 21 – Parte da tela de analisar uma resposta.

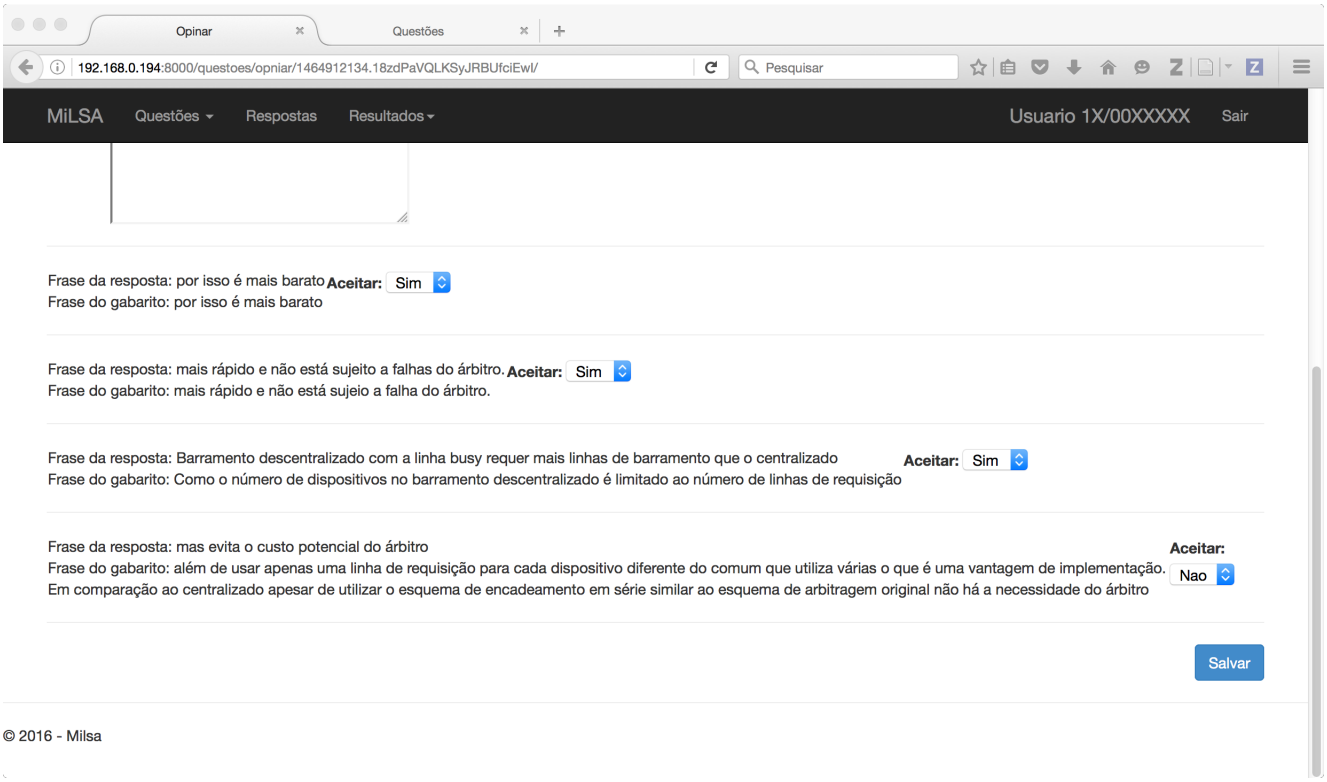


Figura 22 – Parte da tela de analisar uma resposta.

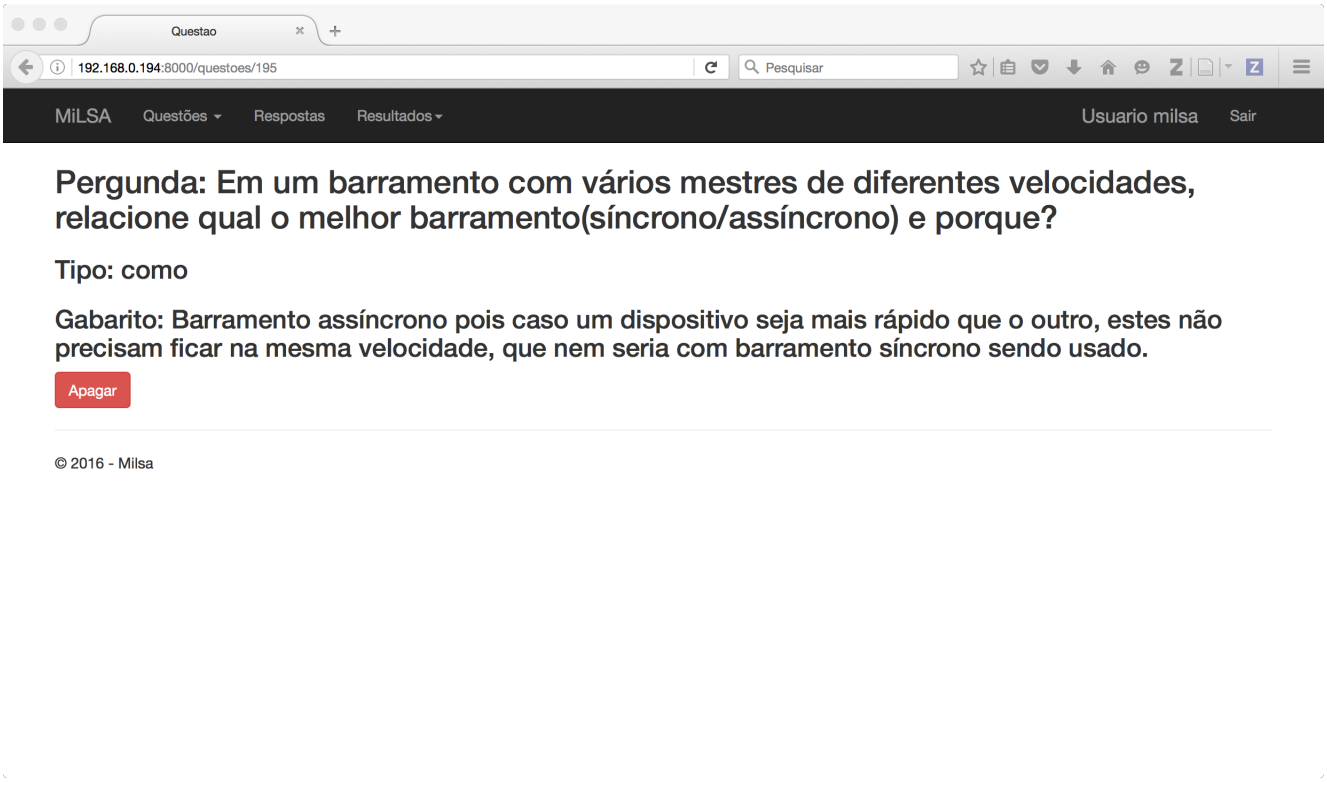


Figura 23 – Tela de detalhar questão.

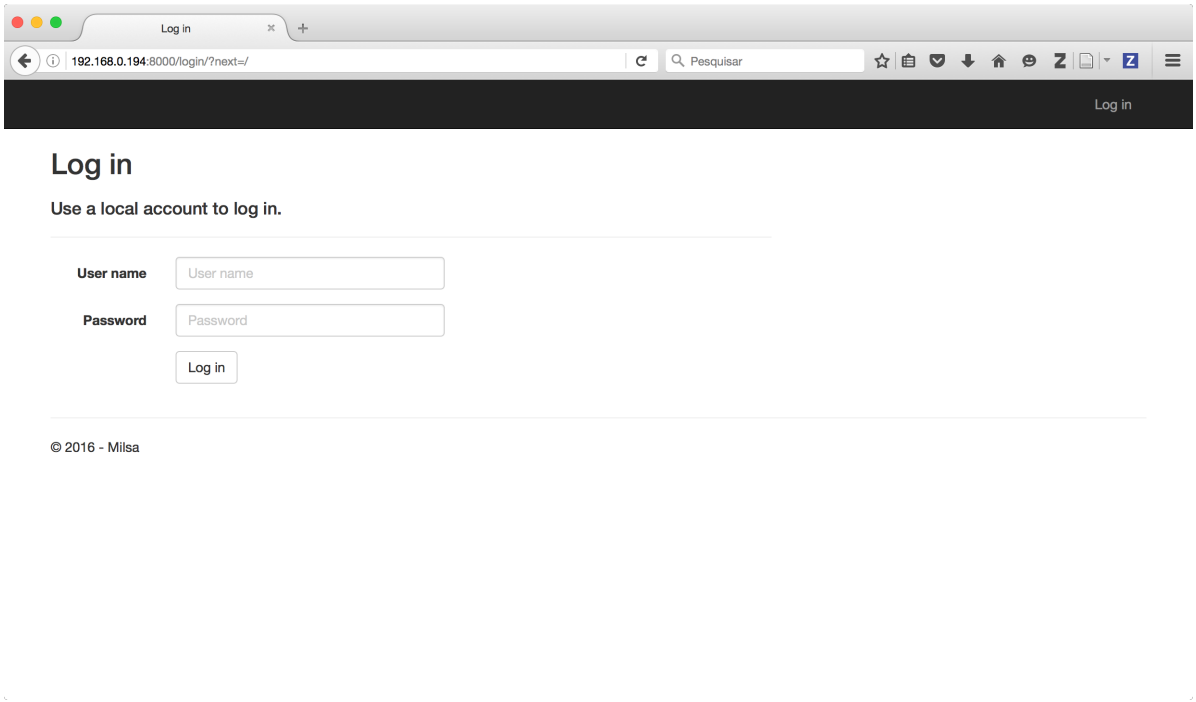


Figura 24 – Tela de *login*.

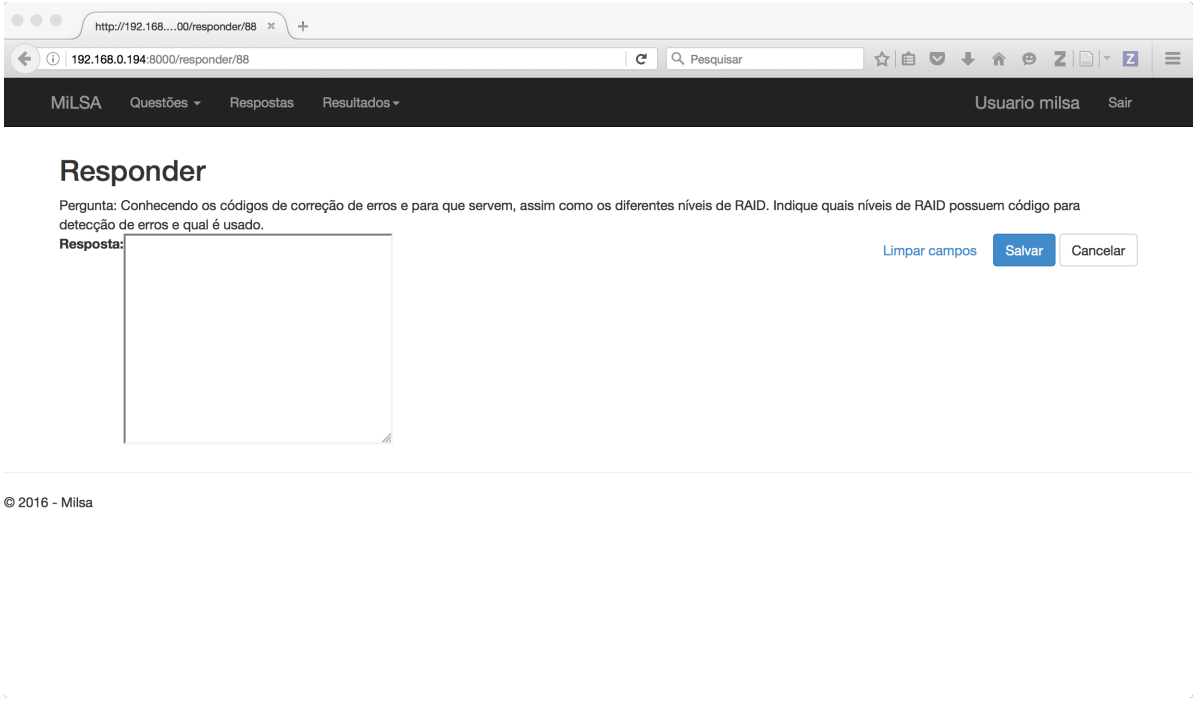


Figura 25 – Tela de responder questão.