UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE
ENGENHARIA DE COMPUTAÇÃO

Douglas Camilo de Oliveira

APLICAÇÃO DAS TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL COSINE SIMILARITY E WORD MOVER'S DISTANCE NA AUTOMATIZAÇÃO DA CORREÇÃO DE QUESTÕES DISCURSIVAS NO SISTEMA TUTOR INTELIGENTE MAZK

Araranguá
2019

Douglas Camilo de Oliveira

APLICAÇÃO DAS TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL COSINE SIMILARITY E WORD MOVER'S DISTANCE NA AUTOMATIZAÇÃO DA CORREÇÃO DE QUESTÕES DISCURSIVAS NO SISTEMA TUTOR INTELIGENTE MAZK

Trabalho Conclusão do Curso de Graduação em Engenharia de Computação do Centro de Ciências Tecnologias e Saúde da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia de Computação
Orientador: Profa. Eliane Pozzebon, Dra.

Araranguá

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Oliveira, Douglas Camilo de
APLICAÇÃO DAS TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL COSINE SIMILARITY E WORD MOVER'S DISTANCE NA AUTOMATIZAÇÃO DA CORREÇÃO DE QUESTÕES DISCURSIVAS NO SISTEMA TUTOR INTELIGENTE MAZK / Douglas Camilo de Oliveira ; orientadora, Eliane Pozzebon, 2019.
60 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2019.

Inclui referências.

1. Engenharia de Computação. 2. Correção Automática de Questões Discursivas. 3. Processamento de Linguagem Natural. 4. Cosine Similarity. 5. Word Mover's Distance. I. Pozzebon, Eliane. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Douglas Camilo de Oliveira

APLICAÇÃO DAS TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL COSINE SIMILARITY E WORD MOVER'S DISTANCE NA AUTOMATIZAÇÃO DA CORREÇÃO DE QUESTÕES DISCURSIVAS NO SISTEMA TUTOR INTELIGENTE MAZK

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Engenharia de Computação" e aprovado em sua forma final pelo Curso de Engenharia de Computação

Araranguá, 25 de novembro de 2019.

Documento assinado digitalmente
Mauricio Girardi
Data: 11/12/2019 11:35:26-0300
CPF: 947.553.619-15

Prof. Fabricio Ourique, Dr.
Coordenador do Curso

Banca Examinadora:

Profª Eliane Pozzebon

Assinado de forma digital por Profª Eliane Pozzebon
Dados: 2019.12.06 08:35:21 -03'00'

Profa. Eliane Pozzebon, Dra.
Orientadora
Universidade Federal de Santa Catarina

Assinado de forma digital por Luciana Bolan Frigo:91038979900
Dados: 2019.12.09 11:29:10 -03'00'

Profa. Luciana Bolan Frigo, Dra.
Avaliadora
Universidade Federal de Santa Catarina

Tatiana Nilson dos Santos
SIAPE 3017565

Assinado de forma digital por Tatiana
Dados: 2019.12.11 10:32:00 -03'00'

Profa. Tatiana Nilson Dos Santos, Ma.
Avaliadora
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus queridos pais, familiares, amigos e professores.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, por tudo que fez, e faz, tanto na minha vida acadêmica quanto fora dela.

Agradeço principalmente ao meus pais, Rosemar Camilo de Oliveira e Miguel dos Santos Oliveira, por todo amor, carinho e dedicação para que eu pudesse ter acesso à educação, sem medir esforços. Esse agradecimento se estende ao meu irmão Daverton Miguel Camilo de Oliveira, quis o destino que passássemos para o mesmo curso e com muito companheirismo e irmandade estamos chegando ao final. Agradeço a todos os meus familiares que me apoiaram de alguma forma, em especial ao meu padrinho Gilmar Camilo de Oliveira e toda sua família, além é claro da minha prima/irmã Leticia Camilo Custodio.

No meio acadêmico gostaria de agradecer aos professores, mestres e doutores, que se dispuseram a passar seu conhecimento, em especial as professoras Eliane Pozzebon e Luciana Bolan Frigo que me deram a oportunidade de fazer parte do laboratório LabTeC. Onde adquiri uma grande experiência e conheci colegas e amigos que tenho muita admiração, inclusive esse agradecimento se estende a toda a equipe do LabTeC.

Por fim agradeço aos demais amigos, amigos da família e colegas que de alguma forma me apoiaram e motivaram durante toda a graduação. Obrigado!

“A persistência é o caminho do êxito.”

(Charles Chaplin)

RESUMO

A avaliação dos alunos por meio de questões discursivas é uma das formas mais tradicionais do sistema de ensino, mas sua correção não é uma tarefa trivial. Em turmas com um número elevado de alunos essa tarefa ocupa boa parte do tempo de trabalho do professor. Este problema se agrava quando falamos de um Sistema Tutor Inteligente (STI) onde o número de alunos pode crescer consideravelmente e, automaticamente, o número de questões discursivas para correção cresce a cada exercício proposto para a turma. Realizando estudos na área de Processamento de Linguagem Natural (PLN) e utilizando as medidas de similaridade *Cosine similarity* e *Word Mover's Distance*, foi proposta uma solução para a criação de um sistema de correção automática de questões discursivas. Foram realizados testes para avaliar a eficiência de cada uma das medidas de similaridade e a *Cosine Similarity* se destacou, sendo assim escolhida para integrar a solução implementada no STI MAZK. Nos testes realizados, dentro do STI, o sistema de correção automática do MAZK obteve um erro relativo de 15% e uma acurácia de 88,7%, resultados interessantes e animadores, mas os testes também mostraram algumas deficiências que precisam ser avaliadas para melhoria dos seus resultados e para uma futura implantação definitiva.

Palavras-chave: Correção Automática de Questões Discursivas. Processamento de Linguagem Natural. Sistema Tutor Inteligente. *Cosine similarity. Word Mover's Distance*.

ABSTRACT

Student assessment through discursive questions is one of the most traditional forms of the education system, but correcting it is not a trivial task. In classes with a large number of students, this task takes up much of the teacher's work time. This problem gets worse when we talk about an Intelligent Tutor System (ITS) where the number of students can grow considerably and automatically the number of discursive questions for correction grows with each proposed exercise for the class. Conducting studies in the field of Natural Language Processing (PLN) and using the similarity measures Cosine similarity and Word Mover's Distance, a solution was proposed for the creation of an automatic correction system for discursive questions. Tests were performed to evaluate the efficiency of each of the similarity measures and Cosine Similarity stood out and was thus chosen to integrate the solution implemented in STI Mazk. In tests carried out within the STI the Mazk automatic correction system achieved a relative error of 15% and an accuracy of 88.7%, interesting and encouraging results, but the tests also showed some deficiencies that need to be evaluated to improve their results. and for future definitive deployment.

Keywords: Automatic Discursive Questions. Natural Language Processing. Intelligent Tutor System. Cosine similarity. Word Move’s Distance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BoW *Bag-of-Words*

CBOW *Continuous Bag-Of-Words*

EMD *Earth Mover's Distance*

PLN Processamento de Linguagem Natural

STI Sistemas Tutores Inteligentes

TF-IDF *Term Frequency - Inverse Document Frequency*

WMD *Word Mover's Distance*

SUMÁRIO

1 INTRODUÇÃO

Os Sistemas Tutores Inteligentes (STI) são sistemas computacionais voltados para o ensino e representam uma parte considerável da Inteligência Artificial na Educação (IAEd) (POZZEBON *et al.*, 2008). Uma das características dos STI's é serem capazes de representar conhecimento de um especialista em uma determinada área de conhecimento, tendo a capacidade de resolver os problemas apresentados aos estudantes. Outra característica, que está presente no seu próprio nome, é de simular as funções de um tutor adequando sua estratégia e modificando a forma de ensinar de acordo com às necessidades dos estudantes (ALVES *et al.*, 2018).

De Moura (2017) aponta os STI's como programas de aprendizagem educacional e enfatiza o uso de técnicas de inteligência artificial para adaptação das necessidades dos estudantes. Os STI's no meio de aprendizagem possuem conteúdos que estão à disposição do estudante bem como avaliações, através de exercícios e problemas para serem resolvidos, visando coletar informações sobre a aprendizagem do estudante para assim adaptar sua estratégia de ensino. O MAZK é um tutor inteligente para ensino e aprendizagem de diversos domínios. Neste aplicativo os professores poderão incluir os materiais e os estudantes poderão aprender sobre um determinado conteúdo exemplos, explanações e exercícios (MAZK, 2019).

A utilização de questões discursivas no processo de ensino permite ao professor, avaliar os processos cognitivos mais elevados quando comparado a aplicação de questões objetivas, além de incentivar as habilidades de comunicação e expressão do estudante (BURROWS; GUREVYCH; STEIN, 2015). A correção de questões discursivas por parte do professor não é uma tarefa simples, ainda mais em turmas com grandes quantidades de alunos. Essa tarefa demanda tempo do professor, se tornando comum os alunos esperarem dias ou mesmo semanas para ter acesso aos resultados (PASSERO; HAENDCHEN FILHO; DAZZI, 2016). Esse fato se agrava quando o assunto é a aplicação de questões discursivas em avaliações nos STI's, pois o número de alunos que tem acesso ao material não é limitado como uma turma em uma sala de aula. Uma solução para esse problema é a automatização da correção de questões discursivas. Todavia, produzir um algoritmo que consiga fazer a correção automática de questões discursivas é um problema complexo cuja solução completa é algo ainda não disponível. Por se tratar de informações em forma de textos, a correção de uma questão discursiva é um problema da área de Processamento de Linguagem Natural (PLN) (INDURKHYA; DAMERAU, 2010).

O Processamento de Linguagem Natural (PLN) é uma área da Ciência da Computação dedicada a encontrar formas de diminuir a dificuldade dos computadores processarem dados de linguagens naturais. O PLN estuda o desenvolvimento de programas de computadores capazes de analisar, reconhecer e/ou gerar textos em linguagens humanas, ou linguagens naturais (VIEIRA; LOPES, 2010). São várias as ferramentas que podem ser aplicadas nas áreas da tecnologia quando é possível transformar textos em dados estruturados, algumas delas são: classificação textual, linguística aplicada, assistentes pessoais, análise de mídias, robótica e inteligência artificial genérica (GOLDBERG, 2017). Sabendo que o PLN permite que o computador consiga processar dados em forma de textos, estudar suas técnicas para encontrar uma solução para automatização de questões discursivas se torna uma das opções mais viáveis.

1.1 PROBLEMÁTICA

A automatização de questões discursivas traz benefícios, mas encontrar uma solução para esse problema não é uma tarefa fácil. Um dos principais desafios é que em uma resposta discursiva o aluno tem várias possibilidades de uso da linguagem natural, para expressar seu conhecimento sobre o assunto. Tantas possibilidades diferentes de se falar sobre o mesmo assunto é o principal problema da área de PLN (PIRES; PIRES; PIRES, 2019).

Na PLN existem medidas ou funções que podem verificar a similaridade entre textos, cada uma com uma estratégia diferentes para análise de texto. Algumas se baseiam em dados estatísticos, outras no papel que uma palavra possui em uma sentença ou no seu significado (BARBOSA *et al.*, 2016). O *cosine similar*ity e *word mover's distance* são duas medidas de similaridade que possuem diferentes estratégias para identificar a semelhança entre os textos.

Tendo em vista os impactos positivos e as técnicas disponíveis atualmente o presente estudo estabelece como problema de pesquisa: É possível realizar a correção automática de questões discursivas utilizando as medidas de similaridade entre textos *Cosine Similarity* e *Word mover's distance*?

1.2 OBJETIVOS

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos deste TCC.

1.2.1 Objetivo Geral

Estudar, implementar e comparar a eficiência das medidas de similaridade entre textos *Cosine Similarity* e *Word mover's distance*, na realização da correção automática de questões discursivas.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral serão necessários atingir os seguintes objetivos específicos:

- Realizar estudo sobre o processamento de linguagem natural;
- Realizar estudos e implementar as medidas de similaridade entre textos *Cosine Similarity* e *Word mover's distance*;
- Elaborar e implementar uma solução para automatização da correção de questões discursivas no tutor inteligente MAZK utilizando as medidas de similaridade estudadas;
- Aplicar testes em turmas do MAZK e avaliar os resultados das medidas de similaridade utilizadas comparando com a correção dos professores.

1.3 JUSTIFICATIVA E MOTIVAÇÃO

A aplicação de questões discursivas para avaliação do aluno, permite ao professor analisar o desempenho do aluno de várias formas, mas como visto em seções anteriores, os tamanhos das turmas afetam diretamente no tempo que o professor terá que dispor para correção desse tipo de questão. Quando falamos em STIs esse problema toma proporções ainda maiores é o caso do aplicativo MAZK que segundo informações disponibilizadas pela equipe, possui cerca de 7.026 usuários e aproximadamente 2.413 questões discursivas para serem corrigidas. Tendo em vista que a plataforma tem o objetivo de continuar crescendo auxiliando alunos e professores esses números tendem a aumentar cada vez mais.

Com o objetivo de encontrar uma solução para auxiliar o processo de correção de questões discursivas, esse trabalho realizará estudos na área de processamento de linguagem natural, além da implementação e realização de testes utilizando duas medidas de similaridade

entre textos, *cosine similarity* e *word mover's distance*, podendo assim analisar o funcionamento e eficiência de ambos na automatização de questões discursivas.

1.4 TRABALHOS RELACIONADOS

Durante a pesquisa alguns trabalhos encontrados possuíam o enfoque na correção automática de questões discursivas/dissertativas, tendo similaridade na aplicação de PLN, mas com propostas diferentes para solução.

Em Frinhani (2016) é apresentada uma proposta de criação de um sistema onde o tutor pode cadastrar questões, gabaritos e disponibilizar os exercícios para os alunos, após a resolução do aluno o sistema sugere uma nota para o tutor, dinâmica parecida com a que foi proposta neste trabalho. A grande diferença está na forma de extrair e representar o contexto do texto utilizando *latent semantic analysis* (LSA) acompanhado do STASIS, além da utilização de um algoritmo genético para analisar as causas da diferença entre sistema e professor. Uma proposta interessante, que obteve bons resultados em alguns testes apresentados, tendo um erro médio de 2,964 e com a utilização do algoritmo genético esse erro caiu para 2,141.

Em Passero, Haendchen Filho e Dazzi (2016) é realizado um comparativo entre a utilização do *latent semantic analysis* (LSA) e algumas métricas de similaridade entre conceitos no *WordNet* na correção de questões discursivas. O *WordNet* segue um conceito diferente do *word embeddigns* apresentado neste trabalho, criando uma relação hierárquica entre as palavras e não um espaço vetorial. Na aplicação dos testes o melhor resultado foi obtido pelo método *Shortest path* obteve uma acurácia de 81,58%.

Em Pires, Pires e Pires (2019) é utilizado uma abordagem diferente das demais vistas até o momento, buscando realizar a correção automática de questões discursivas seguindo uma abordagem direcionada a contexto com linguagem limitada, tendo em comum o objetivo da aplicação em um STI e a comparação entre a resposta do aluno e do professor. A solução propõe ações para diminuir as possibilidades de expressões textuais e cria regras sintáticas. A representação de conhecimento utilizou os predicados lógicos e para cada regra sintática foi proposta um conjunto de predicados lógicos que conseguisse representar o conhecimento existente no trecho analisado e avaliado pela regra semântica. Logo em seguida são comparados os predicados lógicos encontrados na resposta do aluno e do professor gerando uma porcentagem de acerto. Os testes no STI não foram aplicados e a justificativa foi que seriam

necessários mais testes, mas o autor se mostrou satisfeito com os resultados, mostrando seu funcionamento na aplicação da correção automática da questão de um aluno.

1.5 PROCEDIMENTOS METODOLÓGICOS

A metodologia de desenvolvimento deste trabalho é dividida em algumas etapas, das quais, a ordem cronológica e os detalhes são apresentados na sequência.

- Etapa 1: Análise e definição do escopo do trabalho.
- Etapa 2: Levantamento bibliográfico sobre Processamento de Linguagem natural, Sistemas tutores inteligentes e técnicas de medidas de similaridade entre textos, além de avaliar a possibilidade da implementação dessas medidas.
- Etapa 3: Implementação dos algoritmos em Python, e realização de testes para definição de que medida será aplicada no STI MAZK.
- Etapa 4: Desenvolver e aplicar uma solução para a correção automática de questões discursivas na MAZK, utilizando a medida de similaridade selecionado na Etapa 3.
- Etapa 5: Criar turmas na plataforma MAZK como cenário para testar a solução proposta.
- Etapa 6: Avaliação e discussão dos resultados obtidos na Etapa 5.
- Etapa 7: Elaboração, escrita e apresentação do Trabalho de Conclusão de Curso (TCC).

1.6 ORGANIZAÇÃO DO TRABALHO

Este trabalho é fragmentado em seis capítulos. O **Capítulo 1** apresenta uma introdução, problemática do trabalho, os objetivos gerais e específicos, justificativa e motivação, trabalhos relacionados, procedimentos metodológicos e a organização do trabalho.

O **Capítulo 2** trata sobre conceitos da área de processamento de linguagem natural, onde serão apresentadas informações importantes para o entendimento completo desse trabalho. Bem como a apresentação das medidas de similaridade entre textos que serão estudadas *cosine similarity* e *word mover's distance*.

O **Capítulo 3** apresenta o sistema tutor inteligente MAZK, onde os testes estão sendo realizados.

O **Capítulo 4** apresenta a proposta para aplicação das medidas de similaridade estudadas na automatização da correção de questões discursivas da plataforma MAZK.

O **Capítulo 5** contém os testes realizados e a discussão dos resultados obtidos.

O **Capítulo 6** inclui as considerações finais e os trabalhos futuros.

2 PROCESSAMENTO DE LINGUAGEM NATURAL

O PLN vem sendo estudado e aplicado desde meados dos anos 50, quando houve a necessidade de se trabalhar com linguagem natural na comunicação com computadores. Alan Turing foi o principal precursor quando propôs o teste de Turing, basicamente neste teste a máquina tem que simular o comportamento humano ao ponto de não ser possível identificar se é uma máquina ou um ser humano, para isso é necessário entender e compreender a linguagem humana (SETZER, 2015).

O principal objetivo do PLN é estudar problemas de compreensão de linguagens naturais humanas. Umas das principais barreiras é a compreensão da língua humana de uma forma que os computadores consigam reconhecer e interpretar essa mesma linguagem (TERESO, 2019). Neste capítulo serão apresentadas técnicas que a PLN utiliza para trabalhar com esse tipo de dado.

2.1 PRÉ-PROCESSAMENTO

Para que os dados sejam processados é necessário que eles sejam tratados, o pré-processamento consiste na preparação desses dados (SILVA, 2014). Esse procedimento é necessário, pois a linguagem natural é muito complexa, dessa forma são extraídas dos textos apenas a informações relevantes, que são tratadas a ponto de o computador conseguir processá-las.

2.1.1 Normalização e Tokenização

A tokenização transforma os textos em sequências de tokens, é responsável por realizar a segmentação do texto identificando e separando em unidades, essas unidades são definidas com base na solução que procura, podendo ser por caractere, palavra, sentenças, entre outros. A normalização é utilizada para padronizar todos os dados de entrada, colocando todo o texto em letras minúsculas, removendo tokens indesejados como espaços em branco, caracteres especiais (ALVARENGA, 2019). Um exemplo para o melhor entendimento seria na frase: “Esta é uma segmentação por palavras.”, após a normalização e tokenização a frase ficaria da seguinte forma [“esta”, “é”, “uma”, “segmentação”, “por”, “palavras”,“.”].

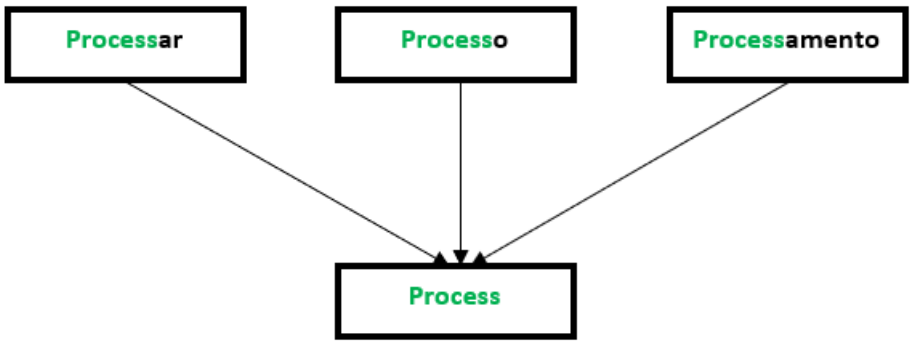
2.1.2 Remoção de *Stopwords*

A remoção de stopwords nada mais é do que a remoção de palavras consideradas irrelevantes, que não contribuem para o significado geral da frase (HARDENIYA *et al.*, 2016). Alguns exemplos de *stopwords* são: “de”, “a”, “o”, “que”, “para, “com”. Essas palavras são definidas em uma lista que serve de base para remoção.

2.1.3 Stemização e Lematização

A stemização se preocupa em reduzir a palavra para o seu radical (TEIXEIRA *et al.*, 2018). Por exemplo, as palavras, processar, processo e processamento, possuem o mesmo radical “process”, como mostrado na figura 1.

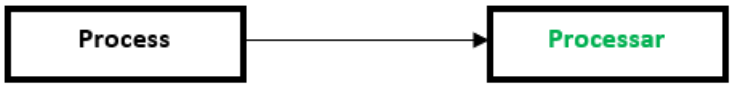
Figura 1 – Exemplo do processo de Stemização



Fonte: Elaborado pelo Autor

A stemização pode deixar a palavra sem sentido, com a Lematização esse problema é corrigido. Ela lida com a obtenção do "lema" de uma palavra, reduzindo a palavra à sua origem básica (TEIXEIRA *et al.*, 2018). No caso dos verbos se assume a forma infinitiva, já para outras palavras se assume a forma singular masculina. Seguindo com o exemplo anterior o radical “process”, após a lematização, assumiria a forma infinitiva do verbo, “processar” como se pode observar na figura 2. Então todas as palavras que possuem radical “process” assumiriam a forma “processar”.

Figura 2 – Exemplo do processo de Lematização



Fonte: Elaborado pelo Autor

2.2 VETORES DE CARACTERÍSTICAS

Após o pré-processamento é necessário representar o conjunto de dados de um documento de forma que seja possível extrair informações. Nessa seção serão apresentados alguns modelos que buscam realizar essa função.

2.2.1 Bag-of-Words

O modelo *bag-of-words* (BoW), é uma das maneiras de extrair informações de um texto e representá-las em uma forma vetorial. O BoW aprende o vocabulário baseado em todos os documentos pré-processados do conjunto de dados. Ele considera a contagem de palavras, observando quantas vezes cada uma aparece e armazena isso como uma característica. Assim cada documento é representado por um "saco de palavras", que não leva em consideração a ordem e posição das palavras (SANTOS, 2015). Como exemplo será utilizado um documento com duas frases, "Amanhã fará sol." e "Vou para a praia amanhã.", após aplicarmos alguns passos do pré-processamento as frases ficam da seguinte forma, ["amanhã", "fará", "sol"] e ["vou", "praia", "amanhã"]. Com as frases nesse formato já é possível aplicar a primeira parte do modelo BoW, que é realizar a contagem de quantas vezes as palavras aparecem, os resultados da primeira etapa do modelo estão na Tabela 1.

Tabela 1 – Exemplo primeira fase *Bag-of-Words*

Palavra	**Contador**
amanhã	2
fará	1
praia	1
sol	1
vou	1

Fonte: Elaborado pelo Autor

A segunda parte consiste em representar o documento como uma matriz, onde cada linha representa uma frase e cada coluna uma palavra. O resultado está na Tabela 2, nela é possível observar que o documento é representado de forma vetorial.

Tabela 2 – Exemplo segunda fase *Bag-of-Words*

Palavra	**Amanhã**	**fará**	**praia**	**Sol**	**Vou**
Frase 1	1	1	0	1	0
Frase 2	1	0	1	0	1

Fonte: Elaborado pelo Autor

Um ponto importante e já destacado nessa seção é que o BoW não leva em consideração a ordem e posição das palavras, consequentemente o contexto e seu significado semântico.

2.2.2 TF-IDF

O TF-IDF (*Term Frequency - Inverse Document Frequency*), diferente do BoW, busca dar mais importância para as palavras que aparecem com menos frequência no documento. Levando em consideração que a importância de uma palavra não se dá somente pela sua frequência (PINTO, 2018).

Introduzida por Salton e Yang (1973), o TD-IDF combina a frequência dos termos (TF) e a relevância do termo para o documento (IDF). Para isso é necessário realizar os seguintes cálculos.

TF = (Número de vezes que uma palavra aparece no documento)/(número total de palavras no documento)

Primeiro precisamos do número de vezes que uma palavra aparece no documento, essa primeira fase do cálculo é idêntica ao modelo BoW, então é possível utilizar a Tabela 1 e as frases do exemplo anterior. Logo em seguida é necessário analisar cada frase do documento, onde cada palavra assume o peso indicado na Tabela 1 dividido pelo número de palavras da frase em que ela está. No caso da palavra “amanhã” ela aparece 2 vezes no documento e a frase 1 possui um tamanho 3, assim o valor dela na frase 1 será 2 dividido por 3 esse cálculo é repetido para as demais palavras e frases como é possível observar na Tabela 3.

A segunda parte do cálculo é o IDF que é dado pela seguinte expressão.

IDF = log(Número total de frases/ Número de documentos que possuem a palavra analisada)

Tabela 3 – Exemplo primeira fase *TF-IDF*

Palavra	**Frase 1**	**Frase 2**
amanhã	0,66	0,66
Fará	0,33	0
praia	0	0,33
Sol	0,33	0
Vou	0	0,33

Fonte: Elaborado pelo Autor

Como sabemos o número de frases, que no caso é 2, e o número de vezes que cada palavra aparece no documento disponível na Tabela 1 podemos realizar a segunda parte do cálculo. A Tabela 4 mostra o cálculo do IDF para o exemplo estudado.

Tabela 4 – Exemplo segunda fase *TF-IDF*

Palavra	**Contador**
amanhã	log(2/2)
fará	log(2/1)
praia	log(2/1)
sol	log(2/1)
vou	log(2/1)

Fonte: Elaborado pelo Autor

Por fim o TF-IDF é o produto entre TF e IDF para um conjunto de palavras de um documento específico. O resultado final está na Tabela 5.

Nota-se que as palavras já começam a ganhar pesos diferentes mostrando as informações relevantes de cada frase, como na frase 1 em que dá mais importância para informação de que “fará sol” fornecendo um melhor contexto para classificação do texto, diferente do BoW.

Tabela 5 – Exemplo fase final TD-IDF

Palavra	**Amanhã**	**fará**	**praia**	**Sol**	**Vou**
Frase 1	0	0,09	0	0,09	0
Frase 2	0	0	0,09	0	0,09

Fonte: Elaborado pelo Autor

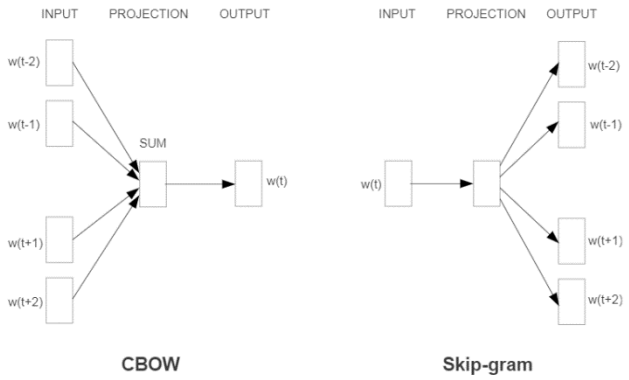
2.2.3 Word Embeddigns

Assim como o BoW e o TF-IDF um *word embedding* busca representar textos em forma de vetor. A forma que ele realiza esse processo é representando uma palavra dentro de um espaço vetorial, no qual palavras com sentidos similares estão em posições próximas umas das outras (CORDEIRO, 2019).

Para que o espaço vetorial possua essas características é necessário utilizar algumas técnicas, uma delas é o *Word2Vec*. O *Word2Vec* é um modelo proposto por (MIKOLOV *et al.*,2013a) sua proposta visa a representação vetorial para incorporação de palavras utilizando técnicas de redes neurais artificiais (AGUIAR; PRATI, 2015). Ele oferece dois tipos de arquiteturas o *continuous bag-of-words* (CBOW) e *Skip-gram* (MIKOLOV *et al.*,2013a).

O modelo CBOW, funciona da seguinte forma, ele aprende as representações de palavras prevendo a palavra central, de acordo com as suas palavras do contexto dentro de uma janela de opções próximas na sentença. O *Skip-gram* segue o caminho oposto, através de uma palavra ele prediz as palavras que estarão ao seu entorno. O funcionamento dos dois modelos é apresentado na figura 3 (DE ARAÚJO LIMA; GUERRA, 2018).

Figura 3 – Abordagem da representação de palavras CBOW e Skip-gram



Fonte: Mikolov *et al.* (2013a)

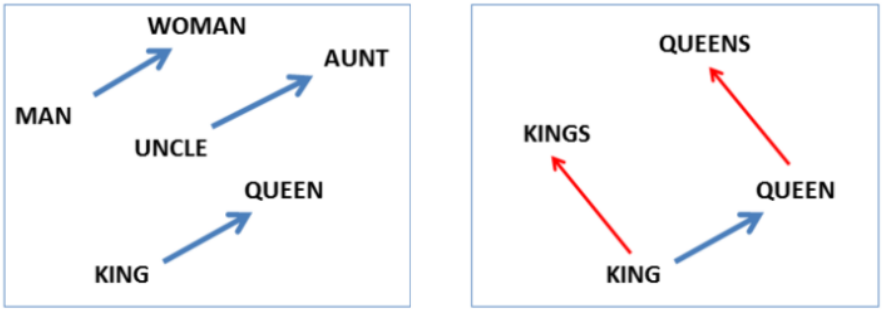
O treinamento se inicia com um arquivo de texto bruto, com uma enorme quantidade de sentenças, geralmente separadas pelo caractere predefinido \n. Todas as palavras que aparecem no documento formam um vocabulário e cada uma delas recebe um vetor que a representa. São realizadas várias iterações em uma rede neural para o cálculo do vetor que representa cada palavra (PELLE, 2019).

É produzida uma camada intermediária para cada iteração, que nada mais é que a projeção da palavra no espaço vetorial, tendo sua posição calculada com base no seu contexto. Dessa forma as palavras semanticamente relacionadas têm uma menor distância, esta relação é representada na figura 4 (PELLE,2019).

Um dos benefícios da utilização de uma *word embedding* são os valores semânticos que as palavras acabam mantendo entre elas, um exemplo clássico da captura dos valores semânticos pelo *word2vec* é que em conjuntos de dados bem treinados é possível realizar operações algébricas entre palavras (DE ARAÚJO LIMA; GUERRA, 2018).

vetor(“rei”) – vetor(“homem”) + vetor(“mulher”) = vetor(“rainha”)

Figura 4 – Exemplo Relação em um espaço vetorial treinado pelo *Word2Vec*



Fonte: Mikolov *et al.* (2013b)

Mostrando que a troca de gênero muda o substantivo, mas mantém a semântica correta, uma característica importante, trazendo vários benefícios na utilização de *word embeddings* (HARTMANN, 2016).

Na literatura a utilização de *word embeddigns* se mostra muito eficaz, principalmente com relação as informações semânticas do documento. Levando em consideração o contesto da problemática em questão a utilização de *word embeddigns* se mostra a mais qualificada. Até mesmo quando comparada a outras técnicas utilizadas em trabalhos relacionados, como a LSA, a utilização de *word embeddigns* treinadas com o *word2vec* se destaca. Em Edgar *et al*. (2016), foi realizada a comparação dos métodos e o modelo *Skip-gram* se mostrou mais eficiente

quando o córpus de treinamento era igual ou maior que 10 milhões de palavras. Atualmente é possível ter acesso a *word embeddigns* com essas proporções na língua portuguesa, viabilizando assim o estudo e aplicação neste trabalho.

2.3 MEDIDAS DE SIMILARIDADE

Nas subseções anteriores foram apresentados conceitos de como os dados em formato de linguagem natural são tratados e como é possível representar essas informações, preservando suas características. Após a realização desses passos é possível realizar o cálculo da similaridade entre documentos de texto. Segundo Russel (2011), em um espaço vetorial multidimensional, cujos vetores representam documentos, a similaridade entre eles é representada pela distância entre esses vetores. Nessa seção serão apresentados três métodos, que buscam realizar essa função de formas diferentes a *Euclidian Distance, cosine similarity* e o *word mover's distance*.

2.3.1 Distância Euclidiana

A Distância Euclidiana *(Euclidian Distance)* é uma métrica frequentemente utilizada para calcular o grau de similaridade, através do cálculo da diferença entre as coordenadas de dois objetos do espaço vetorial (VENI, 2009).

Como já foi comentado, para realizar o cálculo da distância Euclidiana, é necessário que os objetos analisados, estejam representados na forma de conjunto de vetores, partindo dessa premissa, é possível aplicar a função Distância Euclidiana (1) para encontrar o quão semelhante os objetos são. Essa função calcula o grau de similaridade utilizando a raiz quadrada da diferença das coordenadas dos pares de objetos (VENI, 2009).

$$D_{Euc}(x, y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \quad (1)$$

Na equação (1) as variáveis são os objetos x e y que são vetores n-dimensionais, respectivamente. N é o número de elementos dos valores. Para que a função alcance seu objetivo

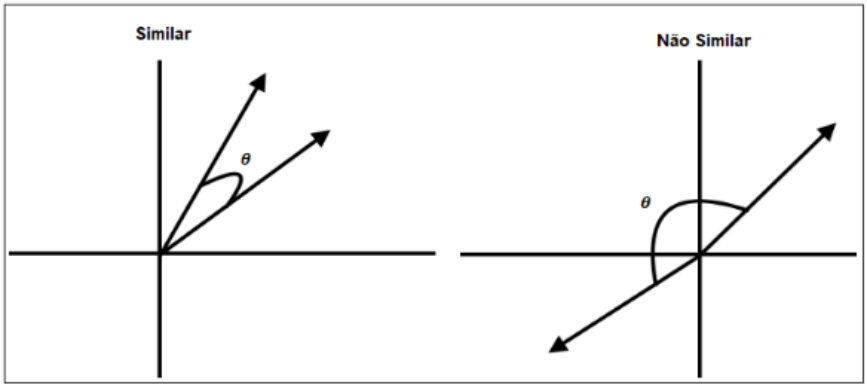
realizando o cálculo da distância entre os dois objetos, $D_{Euc}(x, y)$ deve seguir as seguintes propriedades:

- $D_{Euc}(x, y) \geq 0$, ou seja, deve retornar valores positivos, maiores ou iguais a 0.
- $D_{Euc}(x, y) = 0$, quando a função retornar 0 quer dizer que x=y, ou seja, os objetos que estão sendo comparados são iguais.

2.3.2 Cosine Similarity

O *Cosine Similarity*, traduzindo para o português similaridade do cosseno, é uma medida de similaridade comum na literatura do PLN. Ela utiliza o cosseno do ângulo entre vetores em uma representação vetorial realizando o cálculo, quanto maior for a similaridade entre os dois vetores que estão sendo comparados, menor será o ângulo e consequentemente maior será o cosseno (VILLAÇA *et al.*, 2013).

Figura 5 – Representação da influência do ângulo na similaridade do cosseno



Fonte: (VIANA, 2018)

Os *word embeddings* podem ser utilizadas para o cálculo de similaridade entre palavras, sendo a semelhança de cosseno a mais comumente, sendo constituída pelo cálculo da distância dos vetores baseado no ângulo (LEVY; GOLDBERG, 2014).

A equação (1) é utilizada para obter a distância entre os vetores:

$$SimCos(d_i, d_j) = \frac{\vec{d_i} \times \vec{d_j}}{\|\vec{d_i}\| \times \|\vec{d_j}\|} = \frac{\sum_{k=1}^{t} v_{i,k} \times v_{j,k}}{\sqrt{\sum_{k=1}^{t} v_{i,k}^2} \times \sqrt{\sum_{l=1}^{t} v_{j,k}^2}} \quad (1)$$

Onde, $\overrightarrow{d_i}$ é um vetor formado pelos valores $v_{i,k}$ atribuídos aos termos do documento de d_i, já $\overrightarrow{d_j}$ é um vetor formado pelos valores $v_{j,k}$ atribuídos aos termos documento d_j, $\|\overrightarrow{d_i}\|$ e $\|\overrightarrow{d_j}\|$ são as normas de seus respectivos vetores (SOUSA, 2018).

Na similaridade de cossenos, quando um ângulo de 90 graus se forma entre dois vetores, quer dizer que não existe nenhuma semelhança, já quando um ângulo de 0 graus é formado a similaridade assume seu valor máximo, que é 1. Junto com as *word embeddings* esta função dá origem a duas formas de obter alta similaridade, a primeira é se as palavras que aparecem com frequência em um vetor também aparecem no outro vetor, a segunda se as palavras comparadas estiverem próximas no espaço vetorial, ou seja, tenham valores semânticos parecidos (LEVY; GOLDBERG, 2014).

A similaridade de cossenos é uma medida conhecida na área e processamento de linguagem natural e junto com a utilização das *word embeddings* tem seus resultados potencializados, com a capacidade de, nos seus resultados, levar em consideração a semântica do documento.

2.3.3 Word Mover's Distance

A *word mover's distance* (WMD) faz o cálculo da distância entre os documentos baseado na *Earth Mover's Distance* (EMD), que é a distância calculada sobre o espaço vetorial de palavras, que será usada para determinar o quão custoso é transportar uma palavra até a outra. Buscando o custo mínimo para realizar essa ação (RUBNER; TOMASI; GUIBAS, 1998).

Com essas características o EMD, e consequentemente o WMD, é um caso especial do problema de transporte (*transportation problem*) que tem o objetivo de minimizar o custo de transporte e alocação de recursos para vários pontos de demanda a partir de um grupo de pontos de oferta (RUBNER; TOMASI; GUIBAS, 1998).

NORONHA (2019) detalha o cálculo do WMD da seguinte forma:

$$min_{\Phi} \|C \circ \Phi\|_2^2 t.q. \tag{1}$$

$$\sum_i \Phi_{ij} = q_j \tag{2}$$

$$\sum_j \Phi_{ij} = p_i \tag{3}$$

$$\Phi_{ij} \geq 0 \tag{4}$$

- ϕ: é a matriz que armazena os valores de quanto uma palavra que está na posição i da distribuição P, deve ser transportada para a posição j em Q. Essa matriz é conhecida como matriz de fluxo, nesse caso, entre os documentos P e Q, os elementos da matriz são representados por ϕ_{ij}.
- C: representa uma matriz de distâncias, entre uma palavra que está na posição i de P, representada por p_i e outra na posição j de Q, representada por q_j.
- $C \circ \phi$: o produto de Hadamard, que nada mais é do que a multiplicação de cada elemento entre as matrizes C e ϕ.

Para realizar o cálculo do WMD é necessário seguir três restrições:

- Na restrição (2) P só será considerado totalmente transformado em Q, se a quantidade do fluxo que sai de um ponto P e chega em j de Q, satisfaz o valor de q_j.
- Na restrição (3) O fluxo que sai de p_i e chega em um ponto Q deve se igualar ao valor original p_i.
- A última restrição (4), busca garantir que nenhum elemento deve ter seu valor negativo, tendo um fluxo sempre de P para o documento Q, o contrário não se aplica.

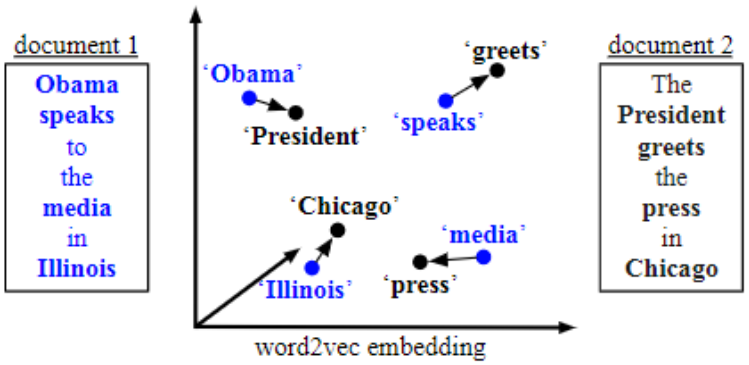
Para determinar o valor do custo de mover uma palavra para outra, é utilizada a distância euclidiana, que foi apresentada anteriormente, ela faz sua função realizando os cálculos com cada palavra representada como pontos no espaço vetorial. O custo c_{ij} de uma palavra *i* ser transportada para *j*, é dado por $\left\|\overrightarrow{w_i} - \overrightarrow{w_j}\right\|_2$. Onde w_i o vetor que está associado com à i-ésima palavra do documento P e w_j consequentemente está associado a j-ésima palavra do documento Q (NORONHA, 2019).

O WMD utiliza como base a distância euclidiana, e acaba "herdando" suas características com relação a similaridade entre as palavras. Como visto anteriormente, no cálculo da distância euclidiana, quanto menor a distância entre dois pontos, que representam palavras, mais próximos semanticamente eles estão. Para o WMD é muito importante que a matriz de custo esteja representando bem a similaridade semântica dos documentos, para isso é

utilizado novamente, o auxílio das *word embeddings* que é uma forma muito poderosa de representar palavras em um espaço vetorial.

Basicamente no WMD a similaridade entre os dois documentos é a distância acumulativa mínima que todas as palavras no P precisam percorrer para corresponder ao documento Q (KUSNER *et al.*, 2015). Na figura 6, é possível observar como o WMD se comporta dentro de um espaço vetorial *word embedding*. Ele consegue identificar que "Obama" está muito mais próximo de "President" do que de "Chicago", achando assim o seu correspondente no documento 2. Isso se repete para todas as palavras, nesse caso são apresentados dois documentos próximos semanticamente, então é esperado que o WMD retorne uma distância pequena entre os dois documentos, indicando uma alta similaridade semântica.

Figura 6 – Representação do *Word Mover's Distance* no espaço vetorial *Word Embeddings*



Fonte: (KUSNER *et al.*, 2015)

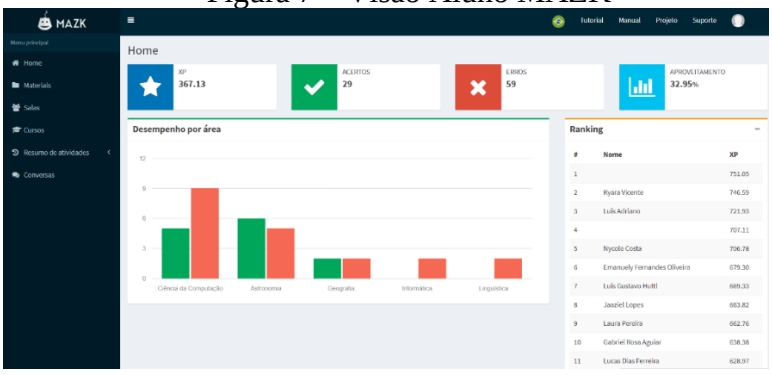
3 TUTOR INTELIGENTE MAZK

Como foi apresentado anteriormente o MAZK é um tutor inteligente para ensino e aprendizagem de diversos temas, onde o professor pode incluir materiais e os estudantes podem aprender sobre um determinado conteúdo, sendo avaliados através, explanações e exercícios com questões objetivas e discursivas. No MAZK a identificação de níveis de conhecimento do usuário, assim como o de dificuldade dos exercícios, são ajustados automaticamente a cada interação do aluno com o STI (MAZK, 2019).

O MAZK foi desenvolvido no Laboratório de Tecnologias Computacionais (LabTeC) da Universidade Federal de Santa Catarina (UFSC), onde participam alunos de graduação, pós-graduação e professores pesquisadores. O STI oferece quatro tipos de usuário: coordenador, o professor, o estudante e administrador. Sendo que em cada tipo de usuário serão disponibilizadas diferentes acessos e permissões, o aluno, por exemplo, poderá acessar a sala virtual, visualizar o conteúdo, responder questões, e visualizar seu desempenho com índices estatísticos e comparações com os demais usuários. Enquanto o professor, conta com o recurso de edição e inserção de novos conteúdos com diferentes estratégias pedagógicas. O Administrador autoriza o primeiro acesso do professor ou coordenador (CAMARGO *et al.*, 2018).

Segundo Vidotto *et al.* (2017) após a realização do seu cadastro o professor tem acesso ao sistema e terá permissão para criar e inserir explanações, exemplos e perguntas. O STI permite que o professor faça compilações com os itens criados, construindo assim seus materiais, voltados a área de conhecimento da turma que irá aplicar. Nestas perguntas cadastradas é necessário que o professor informe um nível de dificuldade, essa informação será utilizada para o balanceamento com o perfil de cada aluno. Para auxiliar na organização e recuperação de informações é possível a inserção de tags em cada item criado pelo professor.

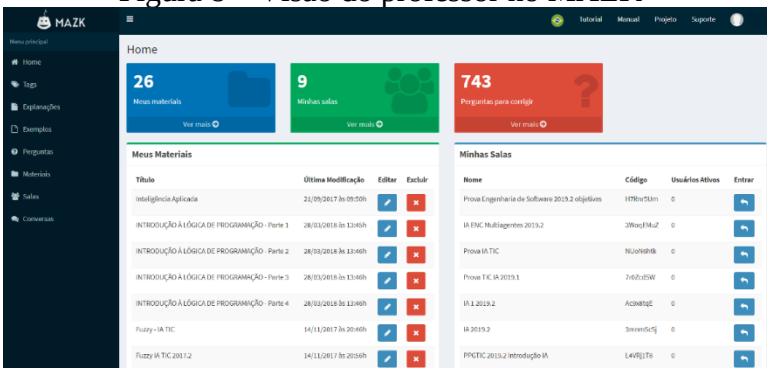
Figura 7 – Visão Aluno MAZK



Fonte: (MAZK,2019)

Como foi visto anteriormente a partir do momento em que as explanações, exemplos e perguntas foram cadastradas pelo professor, elas podem ser utilizadas na composição do material. O material será criado a partir da inserção de uma explanação e/ou perguntas na sua estrutura, após a criação do material o professor pode escolher se deseja deixar o material público ou privado. Essa opção visa a colaboração entre os professores dentro do MAZK, realizando o compartilhando de materiais, mas caso o professor considere que seu material não deve ser compartilhado pode escolher a opção privado, assim só a turma que ele deseja aplicar a aula terá acesso à informação (CAMARGO *et al.*, 2018).

Figura 8 – Visão do professor no MAZK

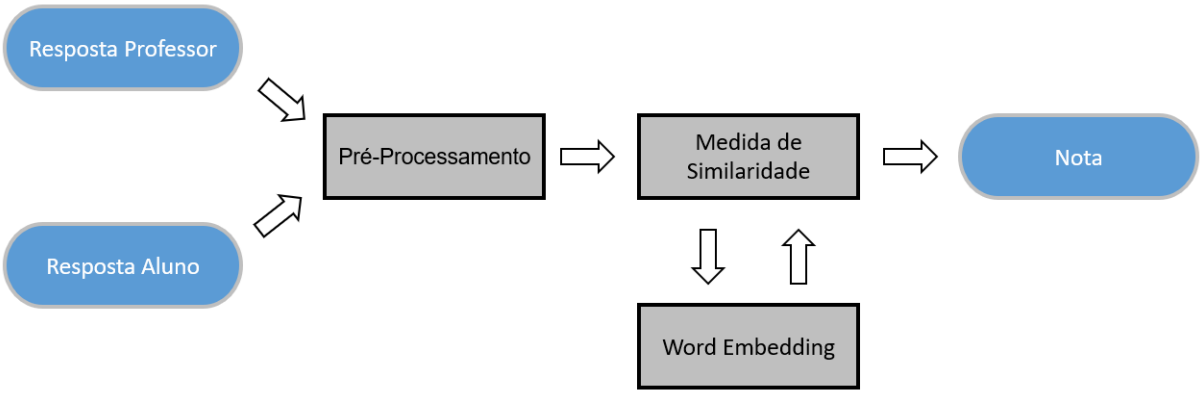


Fonte: (MAZK,2019)

4 SOLUÇÃO PROPOSTA

A proposta para realizar a correção automática de questões discursivas utiliza grande parte dos conceitos visto até o momento implementados na arquitetura do STI MAZK. Serão utilizadas as técnicas de processamento de linguagem natural, como pré-processamento, vetores de características e medidas de similaridade como é possível observar na figura 9.

Figura 9 – Fluxo da solução proposta



Fonte: Elaborado pelo Autor

Para que a solução proposta seja possível, será necessário que, quando o professor cadastrar seu material no MAZK, cadastre uma sugestão de resposta correta para cada questão discursiva presente nos exercícios. Isso será necessário para que a utilização das medidas de similaridade seja viável, pois ela servirá como base de comparação das respostas de cada aluno.

Basicamente a solução funciona da seguinte forma, aluno responderá à questão discursiva que junto com a resposta do professor irão passar por um pré-processamento, sendo aplicadas a normalização, tokenização e remoção de *stopwords*. Logo em seguida as respostas serão enviadas para uma função de medida de similaridade, que fará consultas a *Word Embedding* para obter os valores semânticos de cada palavra que está sendo comparada. No final dos cálculos a função retornará a nota para o aluno.

Como foi visto anteriormente os algoritmos de similaridade estudados foram o *cosine similarity* e o *word mover's distance*, mas apenas um deles poderá ser implementado no sistema MAZK, visando não afetar no tempo de resposta do sistema e atrapalhar a experiência dos usuários. Para isso foi realizado um pequeno pré-teste entre as medidas, para identificar qual delas obtém melhores resultados, e assim se tornar a medida oficial da solução proposta. Outro item que deve ser testado é sobre a *word embeddings* utilizada, como foi visto existem dois

modelos o CBOW e Skip-gram, a diferença entre os dois está na maneira de como são representadas as palavras no espaço vetorial. Para identificar a melhor solução também será aplicado um pequeno pré-teste para definição de como esses modelos irão se sair no problema da automatização da correção de questões discursivas.

4.1 IMPLEMENTAÇÃO

Para a implementação da solução foi utilizada a linguagem de programação Python, que é uma linguagem de programação que possui facilidade na integração de sistemas, além de ser uma linguagem de fácil entendimento. O Python tem suas atualizações e melhorias diretamente ligadas à sua comunidade de usuários, por ser de código aberto os usuários podem contribuir com código e documentação, tendo facilidade de corrigir erros e adicionar novos recursos, estando assim em constante atualização e com novas melhorias, bibliotecas e tutorias para feitos pelos próprios usuários (PYTHON, 2019).

As principais bibliotecas utilizadas foram a RE(*Regular expression operations*), NLTK (*Natural Language Toolkit*) e a Gensim. A RE possibilita realizar operações para o tratamento dos dados de linguagem natural, encontrando padrões nas strings e realizando as alterações, sendo muito utilizada para o pré-processamento (RE, 2019). Outra biblioteca importante foi a NLTK, é a plataforma líder para a criação de programas em Python quando o assunto é linguagem humana, fornecendo um grande número de bibliotecas que auxiliam em grande parte do trabalho relacionado a PLN (NLTK, 2019). A última biblioteca que teve grande importância na implementação da solução das medidas de similaridade e na utilização de *word embeddings* foi a gensim, tendo como alvo a comunidade de PLN, tem como uma das funcionalidades o trabalho com recuperação de similaridade e teve uma grande importância na solução proposta, pois trabalha com *Word2Vec* e como foi visto é uma das principais técnicas aplicadas (GENSIM, 2019).

A figura 10 mostra a função de pré-processamento, onde as bibliotecas RE e NLTK são utilizadas. Neste caso, a RE auxilia no tratamento dos dados na parte de normalização e tokenização, já a NLTK é utilizada para fazer a remoção e *Stopwords*. O pré-processamento é uma tarefa complexa, mas com o auxílio destes tipos de bibliotecas é resumida em poucas linhas de código.

Figura 10 – Função de Pré-processamento

```
import re
from nltk.corpus import stopwords
def preprocess(respostas):

    #Normalização e Tokenização
    respostas = re.sub(r'\W',' ',respostas)
    respostas= re.sub(r'\s+[a-zA-Z]\s+',' ', respostas)
    respostas= re.sub(r'\d',' ',respostas)
    respostas = re.sub(r'\^[a-zA-Z]\s+',' ', respostas)
    respostas = re.sub(r'\s+',' ', respostas)
    palavras = respostas.lower().split()

    #Remoção de stopwords
    aux_stopwords = set(stopwords.words("portuguese"))
    resposta_tratada = [p for p in palavras if p not in aux_stopwords]

    return resposta_tratada
```

Fonte: Elaborado pelo Autor.

4.2 NILC-EMBEDDINGS

Um dos pontos principais da solução proposta é a utilização de *word embeddings*, para isso é necessário encontrar algum repositório que ofereça os modelos CBOW e Skip-gram treinados pelo *Word2Vec* e principalmente que esteja na língua portuguesa e é isso que se encontra no NILC-Embeddings.

O NILC-Embeddings nada mais é que um repositório, que tem o objetivo de armazenar e compartilhar vetores de palavras gerados para Língua Portuguesa. Assim o acesso a recursos vetoriais prontos para serem utilizados nas tarefas de PLN fica mais fácil, além de incentivar os estudos voltados à área. O repositório traz vetores gerados a partir de um grande córpus do português do Brasil e português europeu, de variadas fontes e gêneros. No total foram utilizados dezessete córpus diferentes e 1.395.926.282 tokens (NILC, 2019). O NILC disponibiliza vetores gerados, vetores treinados nos modelos CBOW e Skip-gram e entre outras formas de treinamento, também disponibiliza vetores treinados com *Word2Vec*. Os vetores estão disponíveis em 50, 100, 300, 600 e 1000 dimensões, isso representa a quantidade de dimensões que uma palavra é representada, quanto maior a quantidade de dimensões mais preciso, mas os arquivos acabam aumentando muito seu tamanho. Como estamos trabalhando com uma aplicação web não é aconselhável utilizar arquivos muito pesados, por isso vamos trabalhar com vetores de 100 dimensões que possui um tamanho aceitável.

Figura 11 – Corpus NILC-Embeddings

Corpus	Tokens	Types	Genre	Description
LX-Corpus [Rodrigues et al. 2016]	714,286,638	2,605,393	Mixed genres	A huge collection of texts from 19 sources. Most of them are written in European Portuguese.
Wikipedia	219,293,003	1,758,191	Encyclopedic	Wikipedia dump of 10/20/16
GoogleNews	160,396,456	664,320	Informative	News crawled from GoogleNews service
SubIMDB-PT	129,975,149	500,302	Spoken language	Subtitles crawled from IMDb website
G1	105,341,070	392,635	Informative	News crawled from G1 news portal between 2014 and 2015.
PLN-Br [Bruckschen et al. 2008]	31,196,395	259,762	Informative	Large corpus of the PLN-BR Project with texts sampled from 1994 to 2005. It was also used by [Hartmann 2016] to train word embeddings models
Literacy works of public domain	23,750,521	381,697	Prose	A collection of 138,268 literary works from the Domínio Público website
Lacio-web [Aluísio et al. 2003]	8,962,718	196,077	Mixed genres	Texts from various genres, e.g., literary and its subdivisions (prose, poetry and drama), informative, scientific, law, didactic technical
Portuguese e-books	1,299,008	66,706	Prose	Collection of classical fiction books written in Brazilian Portuguese crawled from Literatura Brasileira website
Mundo Estranho	1,047,108	55,000	Informative	Texts crawled from Mundo Estranho magazine
CHC	941,032	36,522	Informative	Texts crawled from Ciência Hoje das Crianças (CHC) website
FAPESP	499,008	31,746	Science Communication	Brazilian science divulgation texts from Pesquisa FAPESP magazine
Textbooks	96,209	11,597	Didactic	Texts for children between 3rd and 7th-grade years of elementary school
Folhinha	73,575	9,207	Informative	News written for children, crawled in 2015 from Folhinha issue of Folha de São Paulo newspaper
NILC subcorpus	32,868	4,064	Informative	Texts written for children of 3rd and 4th-years of elementary school
Para Seu Filho Ler	21,224	3,942	Informative	News written for children, from Zero Hora newspaper
SARESP	13,308	3,293	Didactic	Text questions of Mathematics, Human Sciences, Nature Sciences and essay writing to evaluate students
Total	1,395,926,282	3,827,725		

Fonte: NILC (2019)

5 TESTES REALIZADOS E IMPLEMENTAÇÃO NO SISTEMA MAZK

Nesta seção serão apresentados os testes realizados, os cenários criados, além da discussão dos resultados obtidos.

5.1 TESTE DE DEFINIÇÃO

Após a implantação do pré-processamento dos dados, das medidas de similaridade *cosine similarity* e o *word mover's distance*, além da conexão com os documentos do repositório NILC-Embeddings utilizando a linguagem Python, a execução dos primeiros testes pode ser aplicada. Visando definir qual medida de similaridade e modelo de *word embeddings* será utilizado no teste na plataforma MAZK foi realizado um teste de definição.

Para realizar a definição se fazia necessário a criação de um pequeno cenário em que o as medidas de similaridade combinadas com os modelos de *word embeddings*, pudessem realizar a correção automática de uma questão discursiva aplicada em uma turma. Foi solicitado a um professor usuário do STI MAZK, que disponibilizasse uma questão e resposta esperada para essa pergunta, junto com as respostas dos alunos e a nota que ele mesmo atribuiu para cada aluno. O Cenário foi montado com a seguinte questão e resposta esperada do professor:

- Questão: Defina Inteligência Artificial.
- Resposta Esperada: A Inteligência Artificial é um campo que busca construir entidades inteligentes, ou seja, ela procura sistematizar e automatizar as tarefas intelectuais. Basicamente, ela procura simular em uma máquina o comportamento humano.

Com base nessas informações foram aplicadas as duas medidas de similaridades estudadas até o momento combinadas com os modelos CBOW e Skip-gram da NILC-Embeddings treinadas com *Word2Vec*. A turma possuía 17 alunos e as notas podem variar de 0 a 100, a tabela 5 mostra 6 dos 17 resultados, a tabela como todos os resultados está disponível no apêndice 1.

Tabela 6 – Resultado Teste de Definição

Aluno	**Resposta**	**CBOW**		**Skip-gram**		**Professor**
		WMD	Cosine	WMD	Cosine	
1	Inteligência Artificial é a capacidade de um sistema computacional de realizar tarefas de maneira inteligente, eficiente e lógica, costumeiramente usando de base a inteligência humana.	49,5	74,96	49,5	83,5	90
2	É a capacidade da máquina realizar tarefas como um ser humano.	49	66,25	49,25	80	90
3	É a parte da computação que estuda e tenta desenvolver algoritmos que tenham características de pensamentos humanos.	45	58	45,2	78	85
4	Um campo da computação, que procura um modo de resolver problemas com possibilidades diversas, implementando em diversas áreas, usando softwares integrando em hardwares.	47,5	68	47,67	80,5	70
5	É a capacidade de uma máquina resolver determinados problemas que lhe são propostos através de um conhecimento adquirido através de padrões.	44,88	58,5	45	70	60
6	Pode ser definida como sistema criado que produz resultados de forma lógica.	43	45	43,5	58	60

Fonte: Elaborado pelo Autor

No teste aplicado foi possível observar que a resposta com maior pontuação do professor, também foi a com maior pontuação das medidas de similaridade, bem como a com menor pontuação. Uma observação interessante é a resposta do segundo aluno, mesmo não tendo várias palavras parecidas com as utilizadas pelo professor as medidas deram uma maior classificação para ela, identificando que mesmo sem possuir o mesmo tamanho de resposta ou palavras parecidas elas estavam no mesmo contexto. Mas da mesma forma que temos uma observação positiva é possível identificar que quando utilizamos muitas palavras voltadas a área da computação as medidas acabam classificando com boas notas é o caso do aluno 4, que mesmo não sendo direto na resposta da questão, utilizou palavras da área tendo um boa pontuação, chegando a receber uma nota de 80,5 por uma das medidas.

Algumas conclusões podem ser avaliadas pelas informações encontradas, como a de que a medida de similaridade *word mover's distance*, não teve um bom desempenho no teste realizado, tendo uma pequena variação da utilização dos modelos CBOW e Skip-gram, já o *cosine similar* obteve melhores resultados se comparados com o WMD, chegando a ficar próximo das respostas do professor quando foi combinado com o modelo de treinamento de *word embeddings* Skip-gram.

Para avaliarmos qual a melhor combinação de medida de similaridade e modelo de *word embeddings*, foi calculado a exatidão das combinações através do erro relativo. A exatidão é a concordância entra o resultado do teste ou da medição e o valor verdadeiro (ISO, 2019). Essa concordância é encontrada no erro relativo, Salvador (2009) define o erro relativo como sendo o erro absoluto dividido pelo valor exato, sendo o erro absoluto a diferença entre o valor exato e o valor aproximado.

$$E_r = \frac{|V_e - V_a|}{V_e} \quad (1)$$

Onde:

- E_r é o erro relativo
- V_e o valor exato, no problema em questão a nota do professor
- V_a o valor aproximado, no problema em questão a nota das combinações propostas.

Os resultados dos erros relativos foram multiplicados por 100, para termos o seu valor percentual. Após a aplicação dos resultados na fórmula do erro relativo, tivemos os seguintes resultados.

Tabela 7 – Erro Relativo das Combinações Propostas

Combinações	**E_r**
WMD + CBOW	39%
Cosine + CBOW	16%
WMD + Skip-gram	38%
Cosine + Skip-gram	7%

Fonte: Elaborada pelo Autor.

Na tabela 8, fica clara a diferença da exatidão entre as combinações, sendo o modelo de *word embeddings* Skip -gram com o os menores erros relativos nas combinações com as duas medidas de similaridade. Nas medidas de similaridade, a que teve um melhor desempenho para o problema em questão foi o *cosine similar* tento resultados muito superiores aos do WMD, a diferença do erro relativo entre as duas medidas de similaridade é expressivo, enquanto na sua melhor combinação o WMD obteve um erro relativo de 38% o *cosine similar* obteve 7% quando combinado com o modelo Skip -gram.

Com base no cálculo do erro relativo dos resultados das combinações, a combinação da medida de similaridade *cosine similarity* e o modelo de *word embeddings* Skip -gram, se mostrou muito superior aos outros, sendo esse o escolhido para a implementação na plataforma MAZK e realização do teste principal no sistema.

5.2 TESTE PRINCIPAL

Tendo a escolha da medida de similaridade e modelo de *word embeddings* definidos, com base no teste de definição, o teste principal pode ser iniciado. Ele tem o objetivo de aplicar a solução diretamente no sistema e observar seu desempenho, não só dos resultados, mas também da experiência do professor utilizando a solução.

Para que o teste fosse viável foi necessário realizar um repasse de conhecimento para a equipe de desenvolvimento do MAZK, onde foi apresentada a solução e seu funcionamento no ponto de vista técnico, desde quais variáveis são necessárias para a solução, onde estariam disponíveis e como, até o tamanho do documento da NILC-Embedding Skip -gram e da

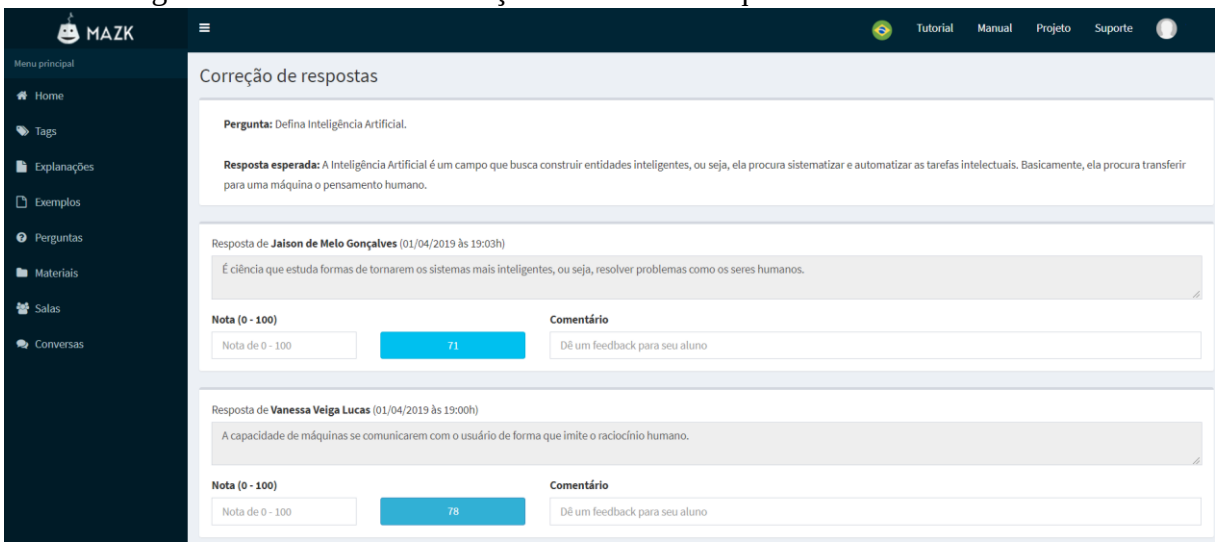
importância que teria para solução. Após o repasse de conhecimento, foi confirmada a viabilidade da aplicação e se deu início a implantação da solução no sistema para realização do teste.

A figura 8 mostra a aplicação da solução no sistema, é possível observar a visão do professor, mostrando a questão e resposta esperada, logo em seguida temos a resposta do aluno e, destacada no botão azul, a nota do sistema de correção automática de questões discursivas. Caso o professor concorde com a sugestão do sistema, só precisa clicar no botão e a nota será atribuída para o aluno, caso discorde na sugestão pode preencher a nota manualmente, a nota do professor e do sistema são armazenados no banco de dados, para a futura análise dos dados.

Com o sistema em funcionamento o cenário de teste foi criado, neste caso o cenário foi a aplicação real do sistema em uma sala no MAZK. A turma escolhida pelo professor, foi a da disciplina de inteligência artificial do curso de Tecnologia da Informação e Comunicação do semestre 2019/2, obtendo cerca de 80 questões respondidas.

Primeiro os alunos tiveram acesso ao material disponibilizado pelo professor na sala criada e logo em seguida uma série de exercícios foram aplicados, entre eles algumas questões discursivas. Após os alunos responderem as questões e o professor avaliar cada uma delas, as informações da nota do professor e do sistema foram extraídas do banco de dados para análise. Serão apresentadas as questões, e algumas respostas e notas selecionadas para discussão.

Figura 12 – Sistema de correção automática de questões discursivas no Mazk.



Fonte: (MAZK,2019)

- Questão 1: Qual é a diferença que existe entre aprendizado supervisionado e não supervisionado?

- Resposta Esperada: Aprendizado supervisionado: A rede neural recebe um conjunto de dados de entrada e seus padrões de saída correspondentes. Por isso ele é supervisionado. Aprendizado não supervisionado: Trabalha os dados de maneira a determinar algumas propriedades dos conjuntos de dados. Não existe para cada entrada uma saída desejada.

A tabela 8 mostra os cinco resultados selecionados para discussão. É possível analisar que os alunos 1 e 2 receberam nota máxima do professor e o sistema também os classificou com boas notas, mas com uma diferença considerável. O aluno 3 obteve nota 90 do professor e 91 do sistema, sendo um caso em que a diferença entre as duas notas foi muito pequena, o que não é o caso do aluno 4, que obteve 50 do professor e o sistema avaliou sua resposta como sendo 79 tendo uma diferença consideravelmente alta. Outro ponto para ser destacado é que a menor nota avaliada pelo professor, também foi a do sistema no caso do aluno 5 recebendo 49 do sistema e 50 do professor. Na primeira questão o sistema continua tendo características do teste de definição, acompanhando o professor e avaliando positivamente e negativamente os alunos, mas ampliando a quantidade de resposta algumas diferenças consideráveis começam a aparecer.

Tabela 8 – Respostas dos Alunos da Primeira Questão do Teste Principal

Aluno	**Resposta**	**Nota Sistema**	**Nota Professor**
1	Supervisionado é quando a rede neural recebe um conjunto de dados de entrada e seus padrões de saída correspondentes. O não supervisionado é quando não existe para cada entrada uma saída desejada.	91	100
2	Supervisionado: Quando há um "professor" intermediando o aprendizado, especificando se ele está correto ou errado. Não Supervisionado: é quando para fazer modificações nos valores das conexões sinápticas não se usa informações sobre se a resposta da rede foi correta ou não no caso não existe para cada entrada uma saída desejada.	86	100

3	No supervisionado, o professor indica o bom e o mal comportamento. A rede neural recebe conjunto de dados de entrada e padrões de saída correspondente. Não supervisionado, para fazer modificações nos valores das conexões sinápticos sem informações sobre resposta da rede correta ou não.	91	90
4	Supervisionado o professor indica explicitamente um comportamento bom ou ruim já o não supervisionado é quando para fazer modificações nos valores das conexões sinápticas não se usa informações sobre se a resposta da rede foi correta ou não.	79	50
5	Supervisionado: Onde um professor indica qual o resultado esperado ou saída esperada. Não supervisionado: Não se sabe-se se o resultado obtido é mesmo o correto.	49	50

Fonte: Elaborado pelo Autor

- Questão 2: O que são sinapses?
- Resposta Esperada: As sinapses são junções entre a terminação de um neurônio e a membrana de outro neurônio. São elas que fazem a conexão entre células vizinhas, dando continuidade à propagação do impulso nervoso por toda a rede neuronal.

A tabela 9 mostra as respostas de alguns alunos para a questão 2, nessa segunda questão alguns padrões continuam se repetindo, como a maior classificação e a menor, também ocorreu um acerto sem erro entre a nota do professor e do sistema. Outro ponto que se repetiu, esse de forma negativa, é a diferença entre as notas que aumentou em alguns casos, como a dos alunos 1,2 e 4, o caso do aluno 4 é um dos mais alarmantes, pois o sistema elevou de forma considerável a nota comparada com a do professor, isso é resultado da proximidade das palavras no espaço vetorial e um ponto negativo para ser observado.

Tabela 9 – Respostas dos Alunos da Segunda Questão do Teste Principal

Aluno	Resposta	Nota Sistema	Nota Professor
1	Sinapses é a parte do neurônio que conecta as extremidades do axônio com os dendritos de outros neurônios, que servem para memorização e armazenamento da informação.	85	100
2	São onde agem os neurotransmissores, que transmitem os impulsos entre neurônios, ou seja, por onde os neurônios conversam.	71	100
3	As sinapses têm um papel fundamental na memorização da informação e são principalmente as do córtex cerebral e algumas vezes de partes mais profundas do cérebro que armazenam esta informação.	80	80
4	Atividade inibitória que previne a excitação do neurônio	81	50
5	São conexões que memorizam informações e interconectam todos os neurônios	62	50

Fonte: Elaborado pelo Autor

- Questão 3: Cite as funções mais comumente utilizadas como funções de saída em neurônios?
- Resposta Esperada: A Função Linear, A Função Sigmoidal ou Logística e A Função Tangente Hiperbólica.

Continuando a apresentação dos resultados, na tabela 10 temos as respostas dos alunos referentes a questão 3. Esses resultados se diferenciam, pois nesse caso alguns alunos usaram fórmulas junto com a resposta que por serem consideradas como caracteres especiais são eliminadas no pré-processamento, levando em consideração apenas o texto. É o caso do aluno 3 que nesse caso utilizou fórmulas para complementar sua resposta, que foi considerada totalmente certa pelo professor e avaliado com uma nota alta também pelo sistema, mas com uma diferença de 13 pontos com relação a do professor. É importante ressaltar que o caso do aluno 4 da questão 2 se repete agora com o aluno 5 na questão 3, o aluno recebeu uma nota muito acima da avaliação do professor pelo mesmo motivo da questão anterior.

Tabela 10 – Respostas dos Alunos da Terceira Questão do Teste Principal

Aluno	Resposta	Nota Sistema	Nota Professor
1	A Função Linear, A Função Logística ou Sigmoidal e a Função Tangente Hiperbólica.	100	100
2	Função Linear, Função Sigmoidal e Função Tangente Hiperbólica.	98,6	100
3	Essencialmente, qualquer função contínua e monotômica crescente, tal que x E R e y(x) E [-1,1], pode ser utilizada como função de saída na modelagem neural. Alguns exemplos são: â€¢ Função Linear y(x) = ax, Função Sigmoidal ou Logí-stica y(x) = 1 / (1 + e-kx), Função Tangente Hiperbólica y(x) = tanh(kx) = (1 - e-kx) / (1 + e-kx).	87	100
4	Linear, Sigmoidal ou Logística e Tangente Hiperbólica.	89	90
5	Função continua ou monotônica crescente.	62	30

Fonte: Elaborado pelo Autor

- Questão 4: Quais são os elementos básicos de um Neurônio Artificial?
- Resposta Esperada: Entradas, A Combinação das Entradas - O "Net", Função de Ativação (fa) e Função de Saída (fs).

Na tabela 11, temos algumas respostas referentes a questão 4, o fato que diferencia um dos resultados dos demais vistos são as respostas dos alunos 1 e 2, mesmo sendo muito parecidas o sistema só considera a questão que está idêntica a esperada pelo professor com nota máxima.

Tabela 11 – Respostas dos Alunos da Quarta Questão do Teste Principal

Aluno	Resposta	Nota Sistema	Nota Professor
1	Entradas, a Combinação das Entradas - O "Net", Função de Ativação (fa), Função de Saída (fs)	100	100

2	Entradas, a combinação de entradas ou o "Net", a função de ativação e a função de saída	96	100
3	Entradas - Pesos Sinápticos - Função Soma - Função Transferência – Saída	88,61	100
4	Entradas, Pesos, Bias, somador, funções de ativação, Saídas.	71,75	100
5	Entrada, somatório, saída	60	50

Fonte: Elaborado pelo Autor

- Questão 5: Onde está armazenado o conhecimento de uma rede neural?
- Resposta Esperada: Nos valores das suas conexões sinápticas.

Na questão 5, todos os alunos que responderam a questão conseguiram obter nota máxima do professor e de forma geral foram classificados com notas acima de 84 pelo sistema, como é possível observar na tabela 12, mas novamente o problema com relação a posição das palavras no espaço vetorial afetou diretamente o aluno 5, mostrando que assim como uma palavra pode aproximar a resposta do aluno com a do professor, o contrário também pode ocorrer.

Tabela 12 – Respostas dos Alunos da Quinta Questão do Teste Principal

Aluno	**Resposta**	**Nota Sistema**	**Nota Professor**
1	Nos valores das suas conexões sinápticas.	100	100
2	Está armazenado nos valores das suas conexões sinápticas.	92,45	100
3	Está armazenando nos valores de suas conexões.	87,6	100
4	O conhecimento de uma rede neural está armazenado nos valores das suas conexões sinápticas.	84,5	100
5	Estão dispersados por toda a rede através dos neurônios e seus pesos sinápticos.	64	100

Fonte: Elaborado pelo Autor

O teste principal, busca analisar o comportamento da solução em uma escala maior, além do seu funcionamento dentro do STI MAZK, encontrando assim características que não ficaram visíveis no primeiro teste. O objetivo foi alcançado, vários comportamentos diferentes

foram encontrados, alguns bons outros ruins, mas todos necessários para avaliar a eficiência da solução.

Após a análise de todas as questões que os alunos tiveram que responder, é necessário calcular a exatidão desses valores. Para isso será utilizado a fórmula apresentada na subseção do teste de definição. Anteriormente o erro relativo da combinação do *cosine similarity* e *word embeddings* Skip -gram foi de 7%, com o aumento da coleta de informações, essa porcentagem subiu para mais que o dobro, indo para 15%. Resultado esperado, pois analisando as tabelas era possível observar que ao mesmo tempo que a solução classificava as notas da mesma forma que o professor, no sentido de maior para a menor, existia uma diferença entre as notas na grande parte das vezes, sendo umas menores, mas outras muito grandes.

Para futuras comparações com trabalhos relacionados outras métricas foram geradas a partir dos dados obtidos. A primeira foi o erro médio que no segundo teste foi de 11,3 em uma escala de 0 a 100. Outra métrica foi a acurácia, que foi calculada utilizando a fórmula de (DOS SANTOS; FAVERO, 2015), com o valor máximo da nota sendo 100.

$$acuracia = \frac{100 - erro\ médio}{100} * 100$$

Aplicando os valores dos testes na fórmula da acurácia foi obtida uma porcentagem de 88,7%, considerada alta, quando comparado com trabalhos relacionados.

5.3 ANÁLISE DOS RESULTADOS

O primeiro teste, denominado como teste de definição, teve a presença das duas principais medidas de similaridade estudadas neste trabalho, *cosine similarity* e *word mover's Distance*, além dos dois modelos de treinamento dos *word embeddings* CBOW e Skip -gram. O objetivo era criar um pequeno cenário com apenas uma questão discursiva, para analisar os comportamentos e resultados das combinações entre as medidas de similaridade e os modelos de *word embeddings* estudados. Buscando a combinação com a maior exatidão possível, foi calculado o erro relativo entre as avaliações do professor e as avaliações dos alunos. Nos resultados desses cálculos a combinação com menor erro foi a medida de similaridade *cosine similarity* utilizando o modelo de treinamento para *word embeddings* Skip -gram obtendo um valor percentual do erro relativo de 7%, já sua combinação com o modelo CBOW obteve um

percentual maior de 16%. Os resultados das combinações com a medida de similaridade WMD, acabaram ficando muito abaixo do esperado, tendo pouca variação de resultados entre o modelo de treinamento muito pequena e uma porcentagem de erro relativo muito alta, quando combinamos WMD com CBOW o erro foi de 39% já quando combinamos com Skip -gram essa porcentagem tem uma pequena variação para 38%. O primeiro teste mostrou que para o problema em questão a medida de similaridade WMD não atendeu as expectativas e não obteve bons resultados, o *cosine similarity* teve resultados melhores, principalmente quando combinado com o modelo de treinamento Skip -gram, sendo essa a combinação que escolhida para o segundo teste.

Já no segundo teste, a combinação escolhida no teste de definição se torna a proposta de solução e assim foi aplicada diretamente no STI, com auxílio da equipe de desenvolvimento do Mazk. A solução foi implementada e os testes realizados na própria plataforma, para além de realizar uma análise com um número maior de questões, obter avaliações da sua usabilidade pelo professor. No ponto de vista do professor, com relação a usabilidade e resposta do sistema, houve um retorno positivo, destacando que o sistema respondia de forma rápida e não afetava as outras funcionalidades do STI, além de demonstrar seu apoio aos estudos na busca da automatização da correção de questões discursivas. Já na análise dos dados o resultado não foi positivo, houve um aumento considerável no percentual do erro relativo, que no primeiro teste era de 7% e no segundo cenário de teste foi para 15%, mesmo tendo resultados positivos a diferença entre a avaliação do professor e do aluno, em alguns casos, era consideravelmente alta.

Quando são comparados os resultados obtidos neste trabalho com os resultados obtidos em alguns trabalhos relacionados é possível notar a sua eficiência. Na solução proposta por Frinhani (2016) obteve um erro médio de 2,96 e com a utilização do algoritmo genético esse erro caiu para 2,14 em uma escala de 0 a 10, já na solução aplicada no Mazk esse erro é de 11,3 em uma escala de 0 a 100. Na solução proposta por Passero, Haendchen Filho e Dazzi (2016) o melhor resultado nos testes foi obtido pelo método *Shortest path* obteve uma acurácia de 81,58%, na solução aplicada no Mazk a porcentagem da acurácia foi de 88,7%.

Em uma visão geral, os testes revelaram várias características com relação ao desempenho das medidas de similaridade e *word embeddings*. A primeira é que na maioria dos casos as avaliações do professor e do sistema convergiam quanto a classificação, das melhores para as piores. A segunda é que mesmo convergindo, existiam algumas diferenças entre as notas e que uma resposta só será classificada com a nota máxima se estiver idêntica a resposta

esperada pelo professor, sendo assim é necessário buscar a diminuição do valor do erro relativo para que essas diferenças sejam corrigidas por arredondamento. A última observação, é que a utilização de *word embeddings* foi positiva para a aplicação da solução, mas em alguns casos foram encontradas falhas, onde avaliava respostas com pouca similaridade com a do professor com notas altas e a situação contrária também foi encontrada, essas observações foram apresentadas com mais detalhes no capítulo 5 de testes.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver uma proposta de solução capaz de realizar a correção automática de questões discursivas no STI Mazk utilizando as medidas de similaridade entre textos *Cosine similarity* e *Word mover's distance*. A solução visa auxiliar os professores, usuário do Mazk, nesta atividade que muitas vezes acaba tomando muito do seu tempo, além de trazer para o aluno, em implementações futuras, um feedback quase que instantâneo dos seus resultados.

Para que o objetivo fosse alcançado foram realizados estudos na área de processamento de linguagem natural, desde o entendimento de como os computadores transformam dados no formato de linguagem natural para um formato que possa ser processado por eles, até como é possível medir a similaridade entre textos utilizando *Cosine similarity* e *Word mover's distance*.

A solução proposta utilizava a comparação entre as respostas do aluno com a resposta esperada do professor, que seria cadastrada junto com cada pergunta. Ambas as respostas passam pelo pré-processamento e em seguida é calculada a medida de similaridade entre elas, por fim é retornada a nota do aluno.

O *Cosine similarity* e *Word mover's distance* medem a similaridade entre os textos cada um de uma forma diferente e para auxiliá-los neste cálculo foram utilizados *word embeddings*, que são vetores de palavras pré-treinados com o objetivo de criar um espaço vetorial que leve em consideração a relação semântica entre as palavras. Existem dois modelos para o treinamento desses vetores, CBOW e Skip -gram, ambos buscam de forma diferentes representar este espaço vetorial. Apenas uma forma de medir a similaridade entre as respostas e um modelo *word embeddings* seriam implementados diretamente no Mazk, então foi necessário implementar as combinações, e realizar um comparativo da eficiência de cada uma, para isso foi utilizado a linguagem de programação Python e o repositório de *word embeddings* em português NILC-Embeddings.

No cenário do teste de definição o sistema teve que avaliar a respostas de uma questão discursiva aplicada em uma turma, logo em seguida foi medido a exatidão de cada combinação, sendo comparada a nota do sistema e do professor. Os resultados obtidos mostraram que o *Cosine similarity* se mostrou muito superior ao *Word mover's distance* e sua menor porcentagem do erro relativo, que foi 7%, foi obtido em combinação com o modelo Skip -gram, sendo assim escolhida como a combinação para ser implantada no Mazk para o teste principal.

No teste principal, o cenário foi uma turma criada diretamente no Mazk onde o professor aplicou um questionário com 5 questões discursivas. O *feedback* quanto a usabilidade do sistema pelo professor foi positivo e avaliando as notas, do sistema e do professor para cada respostas, foi possível encontrar várias características positivas no qual o sistema, mesmo com um certo erro, acompanhou as notas do professor, mas também foi possível observar alguns pontos negativos. Em algumas respostas que o professor classificou com notas baixas o sistema avaliou positivamente e a situação contraria, em que o professor avaliou com notas altas e o sistema negativamente, também aconteceu. Isso foi refletido no aumento da porcentagem do erro relativo, que no primeiro teste foi de 7% e no segundo subiu para 15%.

Ao final, foi possível atingir o objetivo geral deste trabalho, com a construção de um corretor automático de questões discursivas no STI Mazk, mostrando através de testes que a utilização do *Cosine similarity* se mostrou mais eficiente que o *Word mover's distance*. Porém sua eficiência precisa ser melhorada, o erro relativo ainda é alto e quando se trata de avaliar um aluno, uma pequena porcentagem de erro pode reprová-lo, mesmo merecendo a aprovação ou aprová-lo, mesmo merecendo a reprovação. Para diminuir o erro relativo é possível avaliar algumas soluções, como o aumento de dimensões dos *word embeddings*, que no nosso caso foi de 100, mas pode chegar até 1000, neste caso é necessário avaliar sua eficiência e os problemas que pode trazer para usabilidade do sistema em geral. Outra possibilidade é o estudo de outras medidas de similaridade, como o PLN é uma área em destaque atualmente, vários estudos estão sendo realizados e é possível encontrar outras medidas de similaridade que estão fora do escopo deste trabalho.

6.1 TRABALHOS FUTUROS

- Continuar a coleta de informações da eficiência da aplicação no STI Mazk.
- Avaliar a possibilidade de utilizar *word embeddings* com um maior número de dimensões.
- Implementar e testar novas medidas de similaridade.

REFERÊNCIAS

AGUIAR, Raul Freire; PRATI, Ronaldo Cristiano. Incorporação de representação vetorial distribuída de palavras e parágrafos na classificação de SMS SPAM. **ENIAC-Encontro Nacional de Inteligência Artificial e Computacional. Natal, Brasil**, 2015.

ALTSZYLER, Edgar et al. Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. **arXiv preprint arXiv:1610.01520**, 2016.

ALVARENGA, João Paulo Reis. Avaliação de métodos de transferência de aprendizado aplicados a problemas de processamento de linguagem natural em textos da língua portuguesa. 2019.

ALVES, Antônio et al. Melhorando a atenção dos estudantes através da tutoria de mindfulness em sistemas tutores inteligentes. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. 2018. p. 973.

BARBOSA, Ricardo Antunes et al. Análise de 'backlog' de estórias de usuário por meio de agrupamento de textos e similaridade semântica. 2016.

BURROWS, Steven; GUREVYCH, Iryna; STEIN, Benno. The eras and trends of automatic short answer grading. **International Journal of Artificial Intelligence in Education**, v. 25, n. 1, p. 60-117, 2015.

CAMARGO, Cintia Natalicio de et al. Estimular o Aprendizado para Exame Nacional do Ensino Médio Utilizando o Sistema Tutor Inteligente MAZK. 2018.

CORDEIRO, Bernardo Cardoso. **BERT E WORD2VEC: UMA ANÁLISE INFERENCIAL E COMPUTACIONAL NA CLASSIFICACAO DE TEXTOS COM REDES NEURAIS CONVOLUCIONAIS**. 2019. Tese de Doutorado. Universidade Federal do Rio de Janeiro.

DE ARAÚJO LIMA, Raul; GUERRA, Paulo T. An Analysis of the Sentiment Classification of Short Messages Using Word2Vec. In: **Anais do XV Encontro Nacional de Inteligência Artificial e Computacional**. SBC, 2018. p. 425-436.

DE MOURA, Janaína Varela. APLICABILIDADE DA TÉCNICA DE SISTEMAS TUTORES INTELIGENTES COMO MÉTODO DE ENSINO DE MATEMÁTICA. **Maiêutica-Tecnologias da Informação**, v. 2, n. 01, 2017.

DOS SANTOS, João Carlos Alves; FAVERO, Eloi Luiz. Practical use of a latent semantic analysis (LSA) model for automatic evaluation of written answers. **Journal of the Brazilian Computer Society**, v. 21, n. 1, p. 21, 2015.

FRINHANI, Cristóvão de Lima. Aplicação de processamento de linguagem natural: uma ferramenta de apoio à correção de questões dissertativas. 2016.

GOLDBERG, Yoav. Neural network methods for natural language processing. **Synthesis Lectures on Human Language Technologies**, v. 10, n. 1, p. 1-309, 2017.

GENSIM.docs.python.org. 2019. Acessado em 10/11/2019.<https://pypi.org/project/gensim/>.
HARDENIYA, Nitin et al. **Natural Language Processing: Python and NLTK**. Packt Publishing Ltd, 2016.

HARTMANN, Nathan Siegle. Solo Queue at ASSIN: Mix of a traditional and an emerging approaches. **Linguamática**, v. 8, n. 2, p. 59-64, 2016.

INDURKHYA, Nitin; DAMERAU, Fred J. **Handbook of natural language processing**. Chapman and Hall/CRC, 2010.

ISO.*iso.org*.2019. Acessado em 11/11/2019.< https://www.iso.org/obp/ui/#iso:std:iso:3534:-2:ed-2:v1:en >.

KUSNER, Matt et al. From word embeddings to document distances. In: **International conference on machine learning**. 2015. p. 957-966.

LEVY, Omer; GOLDBERG, Yoav. Linguistic regularities in sparse and explicit word representations. In: **Proceedings of the eighteenth conference on computational natural language learning**. 2014. p. 171-180.

MIKOLOV, Tomas et al. Efficient estimation of word representations in vector space. **arXiv preprint arXiv**:1301.3781, 2013.

MIKOLOV, T. et al. Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

MAZK.*mazk.labtec.ufsc.br*. Acessado em 11/11/2019.< https://mazk.labtec.ufsc.br/>.

NILC.*nilc.icmc.usp.br*.2019. Acessado em 10/11/2019.< http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc#>.

NLTK.*nltk.org.* 2019. Acessado em 10/11/2019.< http://www.nltk.org/>.

NORONHA, Victor Garritano. Relacionando geometricamente tweets e notícias utilizando a Word Mover's Distance. 2019.

PASSERO, Guilherme; HAENDCHEN FILHO, Aluizio; DAZZI, Rudimar. Avaliaçao do uso de métodos baseados em lsa e wordnet para correçao de questoes discursivas. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. 2016. p. 1136.

PELLE, Rogers Prates de. Identificação de comentários ofensivos da Web. 2019.

PINTO, Cleiton de Lima. Extração de características para identificação de discurso de ódio em documentos. 2018.

PIRES, Sandrerley Ramos; PIRES, Dulcinéia Gonçalves Ferreira; PIRES, Tobias Gonçalves. Correção Automática de Questões Discursivas–Uma Abordagem Direcionada a Contextos com Linguagem Limitada. **CIAIQ2019**, v. 1, p. 699-708, 2019.

POZZEBON, Eliane et al. Um modelo para suporte ao aprendizado em grupo em sistemas tutores inteligentes. 2008.

PYTHON.*Python.org.* 2019. Acessado em 10/11/2019.<https://www.python.org/>.

RE.docs.python.org. 2019. Acessado em 10/11/2019.<https://docs.python.org/3/library/re.html>.

ROQUE, Carolinne et al. Sistema de Apoio à Decisão por PLN para consultas de Pragas na Cultura da Soja. In: **Anais do XLVI Seminário Integrado de Software e Hardware**. SBC, 2019. p. 45-56.

RUBNER, Yossi; TOMASI, Carlo; GUIBAS, Leonidas J. A metric for distributions with applications to image databases. In: **Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)**. IEEE, 1998. p. 59-66.

RUSSELL, Matthew A. **Mining the social web: Analyzing data from Facebook, Twitter, LinkedIn, and other social media sites**. " O'Reilly Media, Inc.", 2011.

SALTON, Gerard; YANG, Chung-Shu. On the specification of term values in automatic indexing. **Journal of documentation**, v. 29, n. 4, p. 351-372, 1973.

SALVADOR, Manuel Bernardino Lino. Cálculo Numérico I. 2009.

SANTOS, Elisa Mussumeci Bianor dos. Dinâmica de disseminação de notícias em redes complexas. 2015.

SETZER, V. W. *Alan Turing e a Ciência da Computação*. Departamento de Ciência da Computação da USP. Disponível em: <http://www.ime.usp.br/~vwsetzer/Turing-teatro.html>. Acessado em: 09 de novembro de 2019.

SILVA, M. d AO. **Pré-Processamento em Mineração de Dados como método de suporte à modelagem algorítmica**. 2014. Tese de Doutorado. Dissertação (Mestrado)—Universidade Federal do Tocantins, Curso de Pós-Graduação em Modelagem Computacional de Sistemas da Universidade Federal do Tocantins, Palmas.

SOUSA, Priscila Sad de. Estimando similaridade entre entidades quando apenas seus nomes estão disponíveis. 2018.

TAGHVA, Kazem; VENI, Rushikesh. Effects of similarity metrics on document clustering. In: **2010 Seventh International Conference on Information Technology: New Generations**. IEEE, 2010. p. 222-226.

TEIXEIRA, Gabriel Morais et al. ITeligence: Sistema de Apoio à Análise de Intenções. 2018.

TERESO, Marco. ANÁLISE DE SENTIMENTO A COMPANHIAS AÉREAS NORTE AMERICANAS. **ISLA Multidisciplinary e-Journal**, v. 2, n. 1, p. 52-65, 2019.

VENI, Rushikesh. Effects of similarity metrics on document clustering. 2009.

VIANA, Luiz Henrique. Sistema de recomendação de imóveis utilizando filtragem colaborativa e mineração de textos. **Ciência da Computação-Tubarão**, 2018

VIDOTTO, Kajiana Nuernberg Sartor et al. Ambiente Inteligente de Aprendizagem MAZK com alunos do Ensino Fundamental II na disciplina de Ciências. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. 2017. p. 1367.

VIEIRA, Renata; LOPES, Lucelene. PROCESSAMENTO DE LINGUAGEM NATURAL E O TRATAMENTO COMPUTACIONAL DE LINGUAGENS CIENTÍFICAS. **EM CORPORA**, p. 183, 2010.

VILLAÇA, Rodolfo da Silva et al. Hamming DHTe HCube: arquiteturas distribuídas para busca por similaridade. 2013.

APÊNDICE A – DADOS TESTES DE DEFINIÇÃO

Tabela com os resultados do teste de definição:

Resposta	**CBOW**		**Skip-Gram**		**Professor**
	WMD	Cosine	WMD	Cosine	
Pode ser definida como sistema criado que produz resultados de forma lógica.	43	45	43,5	58	60
É a capacidade de uma máquina resolver determinados problemas que lhe são propostos através de um conhecimento adquirido através de padrões.	44,88	58,5	45	70	60
Sistema criado capaz de resolver problemas através da tomada de decisões.	44	50	44	58	65
Capacidade de uma máquina de tomar decisões como um humano toma, com base em seus conhecimentos sobre o assunto e sempre com certa incerteza.	47	67	47,5	71,5	70
Um campo da computação, que procura um modo de resolver problemas com possibilidades diversas, implementando em diversas áreas, usando softwares integrando em hardwares.	47,5	68	47,67	80,5	70
Capacidade de tomar decisões e adaptar-se a diferentes ambientes.	45	62	45	66,5	75
É ciência que estuda formas de tornarem os sistemas mais inteligentes, ou seja, resolver problemas como os seres humanos.	46	62	46	71	75
Existe uma definição de inteligência que é inteligência é aquilo que nós usamos quando não sabemos o que fazer, então inteligência artificial é quando uma máquina é inteligente, ou seja, consegue pensar.	49	64	48,5	73	75

Existem várias definições para inteligência artificial, uma delas pode ser considerada como a capacidade de adaptação do computador para situações desconhecidas.	47	64,5	47	75	75
É a capacidade de simular um pensamento inteligente e lógico considerando as circunstâncias e aprendizado armazenado.	46	69	46,5	76	80
I.A. é a inteligência das máquinas, a capacidade que um software tem de interagir com problemas e tomar certas decisões.	46	66,5	46	77	80
A capacidade de máquinas se comunicarem com o usuário de forma que imite o raciocínio humano.	46,5	72	46,5	79	80
É a tentativa de reprodução de ações inteligentes, comumente realizadas por seres humanos, porém pelas máquinas através do uso da tecnologia. Ações as quais consiga-se retirar uma previsão de porcentagem de acerto dessa atividade que será realizada pela máquina.	47	67	47	75	85
É a parte da computação que estuda e tenta desenvolver algoritmos que tenham características de pensamentos humanos.	45	58	45,2	78	85
É a tentativa de emular o cérebro humano nas máquinas, de forma que consiga pensar e agir como as pessoas e trabalhar para achar melhores soluções para um problema.	47	72,5	47	78	85
É a capacidade da máquina realizar tarefas como um ser humano.	49	66,25	49,25	80	90
Inteligência Artificial é a capacidade de um sistema computacional de realizar tarefas de maneira inteligente, eficiente e lógica,	49,5	74,96	49,5	83,5	90

costumeiramente usando de base a inteligência humana.