UNIVERSIDADE FEEVALE

GABRIEL LAMBRECHT

DESENVOLVIMENTO DE UM PROTÓTIPO PARA ANÁLISE DE FEEDBACK CORPORATIVO ATRAVÉS DO USO DE PROCESSAMENTO DE LINGUAGEM NATURAL

Novo Hamburgo

2021

GABRIEL LAMBRECHT

DESENVOLVIMENTO DE UM PROTÓTIPO PARA ANÁLISE DE FEEDBACK CORPORATIVO ATRAVÉS DO USO DE PROCESSAMENTO DE LINGUAGEM NATURAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

Orientador: Marta Rosecler Bez

Novo Hamburgo

2021

GABRIEL LAMBRECHT

DESENVOLVIMENTO DE UM PROTÓTIPO PARA ANÁLISE DE FEEDBACK CORPORATIVO ATRAVÉS DO USO DE PROCESSAMENTO DE LINGUAGEM NATURAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

APROVADO EM: ___ / ___ / _____

MARTA ROSECLER BEZ
Orientador – Feevale

GABRIEL DA SILVA SIMÕES
Examinador interno – Feevale

PAULO RICARDO MUNIZ BARROS
Examinador interno – Feevale

Novo Hamburgo
2021

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial: Aos amigos e às pessoas que convivem comigo diariamente, minha gratidão, pelo apoio emocional - nos períodos mais difíceis do trabalho.

RESUMO

O mercado cada vez mais competitivo incentiva as empresas a buscarem formas de aumentar sua eficiência e seus resultados. Diante disso, muitas empresas têm investido na gestão de pessoas como um dos fatores decisivos para garantir um bom desempenho do negócio e alcançar os objetivos desejados. Uma das áreas importantes da gestão de pessoas é a avaliação e *feedback*, responsável por fornecer à empresa e ao funcionário uma análise do desempenho do colaborador. Esses registros são feitos principalmente de forma textual, o que permite a aplicação de técnicas computacionais como o Processamento de Linguagem Natural (PLN) para qualificação do conteúdo. Diante disso, o objetivo desse trabalho é o desenvolvimento de um protótipo para análise de *feedbacks* textuais aplicando estratégias de PLN. Para a condução desse trabalho será usada a metodologia *Desing Science Research* (DSR), que define etapas que vão desde a identificação do problema/motivação, desenvolvimento, avaliação e divulgação dos resultados, entre outras. Diante disso, foi desenvolvido um protótipo para classificação de sentenças quanto a sua positividade, usando como base um modelo BERTimbau. O protótipo atingiu eficácia de 78% segundo a métrica *f1-score* nos testes realizados, compatível com os resultados encontrados nos trabalhos correlatos.

Palavras-chave: Gestão de Pessoas. Processamento de Linguagem Natural. Avaliação e *Feedback*.

ABSTRACT

The increasingly competitive market encourages companies to seek ways to increase their efficiency and their results. In face of that, many companies have invested in people management as one of the decisive factors to ensure a good business performance and achieve the desired goals. One of the important areas in people management is evaluation and feedback, responsible for providing the company and the employee with an analysis of the employee's performance. These records are made mainly in textual form, which allows the application of computational techniques such as Natural Language Processing (NLP) to qualify the content. Therefore, the objective of this work is to develop a prototype to analyze textual feedbacks by applying NLP strategies. To conduct this work the Desing Science Research (DSR) methodology will be used, which defines steps from problem identification/motivation, development, evaluation, and dissemination of results, among others. In light of this, a prototype for classifying sentences as to their positivity was developed, using a BERTimbau model as a basis. The prototype reached an efficiency of 78% according to the f1-score metric in the tests performed, compatible with the results found in related works.

Keywords: People Management. Natural Language Processing. Evaluation and Feedback.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CMM	*Conditional Markov Model*
CRF	*Conditional Random Fields*
IA	*Inteligência Artificial*
KNN	*K — Nearest Neighbors*
NLP	*Natural Language Processing*
PLN	Processamento de Linguagem Natural
SNL	*Semantic Role Labeling*
StArt	*State of the Art through Systematic Review*

SUMÁRIO

1 INTRODUÇÃO

Cada vez mais as empresas têm procurado agregar diferenciais em seus negócios, buscando se destacar no atual mercado econômico que é altamente competitivo. De acordo com Knapik (2008), são as pessoas que compõem a estrutura na qual as organizações se apoiam, uma vez que são elas que, com seus conhecimentos e habilidades, trabalham para gerar valor e atingir os objetivos desejados pela empresa. É nesse contexto que Malheiros, Rocha e Ramal (2014) reforçam o papel decisivo que a gestão de pessoas alcançou, tornando-se essencial para que as empresas sejam eficientes e possam executar suas estratégias de negócio.

A avaliação e *feedback* é uma das subáreas da gestão de pessoas e é definida por Chiavenato (2008) como uma análise sistemática de cada funcionário, considerando suas atividades realizadas, os resultados alcançados, o potencial demonstrado e as competências oferecidas. O processo de avaliação, segundo Malheiros, Rocha e Ramal (2014), é definido pelo registro e pela divulgação dos aspectos positivos e os pontos de melhoria, permitindo ao funcionário seu desenvolvimento profissional. A avaliação pode ser usada como justificativa para a concessão de retornos financeiros, como aumento salarial ou recebimento de bônus.

O *feedback* é estabelecido por Missel (2017) como a troca de informações entre o gestor e o funcionário, no qual ambos fazem observações sobre os resultados obtidos pela avaliação de desempenho do funcionário. O autor ainda afirma que o *feedback* é composto tanto pela comunicação verbal (palavras ditas ou escritas) quanto pelo aspecto não verbal como gestos, expressões, tom da voz, postura, entre outras.

O *feedback* pode ser composto tanto de aspectos positivos quanto negativos. O *feedback* positivo se aplica quando o funcionário atinge as necessidades e expectativas da empresa com o seu desempenho. Nesse caso, o *feedback* vem para tornar o funcionário ciente do seu bom desempenho, para que ele possa manter o que está fazendo. Já o *feedback* negativo, também chamado de *feedback* de desenvolvimento, ocorre quando há necessidade de correção de algum comportamento desempenhado pelo colaborador. O *feedback*, nesse caso, vem para que o funcionário entenda quais características do seu trabalho são deficientes, para que ele possa pensar em soluções que tornem seu desempenho mais eficiente. É necessária muita consideração ao passar um *feedback* negativo ao colaborador, pois existe o risco dele ser interpretado de forma errada, o que apenas irá piorar a situação ao invés de resolvê-la (VASCONCELOS; ALBUQUERQUE, 2016).

Considerando o papel da avaliação de desempenho e *feedback* para o correto andamento da empresa, torna-se importante que os documentos utilizados ou registrados

durante o processo sejam capazes de fornecer informações de uma forma simples, ajudando a empresa na tomada de decisões estratégicas. Considerando que a maioria dos registros se encontram em formato textual, geralmente campos de texto livre, pensou-se em aplicar técnicas de processamento de linguagem natural (PLN) como forma de analisá-los, processá-los e ao final extrair informações organizadas, facilitando assim a compreensão de seu conteúdo.

Processamento de Linguagem Natural (PLN) é explicado por Gonzalez e Lima (2003) como a área responsável em permitir o entendimento computacional dos diversos aspectos da comunicação humana, como sons, palavras, sentenças e discursos, levando em conta formatos e referências, estruturas e significados, contextos e usos. Para isso, as sentenças passam por diversas técnicas que as analisam considerando sua estrutura léxica e sintática, para enfim considerar sua estrutura semântica, na qual o significado da frase é compreendido de forma computacional.

Diante de todas as informações apresentadas, surge a problemática deste trabalho: como desenvolver uma aplicação (protótipo) utilizando Processamento de Linguagem Natural capaz de analisar documentos de *feedback* corporativo?

Esse trabalho tem como objetivo desenvolver um protótipo para análise de *feedbacks* corporativos através do uso de processamento de linguagem natural. Os objetivos específicos realizados para atingir esse objetivo são:

- Caracterizar *feedback* e avaliação de desempenho no contexto corporativo;
- Realizar uma pesquisa teórica sobre processamento de linguagem natural (PLN);
- Propor uma estratégia para avaliação das informações de *feedback* utilizando processamento de linguagem natural;
- Desenvolver um protótipo utilizando a estratégia proposta;
- Verificar a aderência do protótipo desenvolvido aplicando em um cenário corporativo real.

Este trabalho foi organizado em seis capítulos, tendo a introdução como primeiro capítulo. O segundo capítulo apresenta uma contextualização de conceitos básicos de linguística, importantes para a compreensão de PLN. O terceiro aborda o Processamento de Linguagem Natural, descrevendo as principais técnicas utilizadas para o tratamento dos textos. No capítulo 4 encontra-se a revisão sistemática realizada em busca do estado da arte na área em questão. O capítulo 5 aborda a metodologia utilizada para o desenvolvimento deste trabalho, além de contextualizar algumas ferramentas utilizadas durante o desenvolvimento. No capítulo 6 é encontrado o desenvolvimento, que descreve o processo de criação do *dataset* e do desenvolvimento do modelo responsável pela classificação

das sentenças. Por fim, se encontra a conclusão, na qual estão descritos os resultados alcançados neste trabalho e as possibilidades de futuros trabalhos.

2 LINGUÍSTICA

Neste capítulo serão contextualizados conceitos de linguística importantes para a compreensão do próximo capítulo, sobre Processamento de Linguagem Natural (PLN), no qual os aspectos linguísticos da linguagem natural são tratados de um ponto de vista computacional.

2.1 LEXICOLOGIA

A lexicologia é a área de estudos dentro da gramática responsável por estudar o léxico. O léxico, como definido por Vilela (1997), é o conjunto de palavras usadas pelos membros de uma comunidade linguística para se comunicar. Neto e Infante (2003) explicam que léxico é a palavra com que se costuma denominar o conjunto de palavras que integra uma língua.

Já o vocabulário é caracterizado como um subconjunto de palavras pertencentes ao léxico. Enquanto o léxico é único por linguagem, uma vez que engloba todas as palavras da língua, os vocabulários são infinitos. Assim, tem-se o vocabulário de um determinado autor, de uma determinada área do conhecimento, de uma determinada região, entre outros (VILELA, 1997).

O léxico é composto principalmente por palavras, mas é possível considerar que ele contém também os lexemas. A palavra é a unidade linguística principal usada na comunicação, enquanto o lexema ou morfema é a parte da palavra que serve como base para o significado por ela expresso, muitas vezes descrita como raiz da palavra. As palavras lindo, linda, lindos, lindas, lindinha e lindíssima, apesar de serem todas diferentes, pertencem ao mesmo lexema, ou seja, a um mesmo significado (DUARTE, 2004).

2.2 MORFOLOGIA

A Morfologia, segundo Neto e Infante (2003), analisa a estrutura, a formação e os mecanismos de flexão das palavras, além de dividi-las em classes gramaticais. A seguir tem-se as descrições das 10 classes gramaticais da língua portuguesa (CEGALLA, 2008):

1. **Substantivo**: Designam os seres, coisas, idéias, ações, estados, qualidades etc. Exemplos: menino, menina, Brazil, beleza;

2. **Artigo**: Utilizada antes dos substantivos para dar um sentido determinado ou indeterminado. Indicam gênero e número. Exemplos: o, os, a, as, um, uma, uns.

3. **Adjetivo**: Expressam qualidades ou características do ser. Exemplos: arrogante, azul, bonito, velho.

4. **Numeral**: Demonstra número, ordem numérica, múltiplo ou fração. Exemplos: seis, primeiro, dobro, meio.

5. **Pronome**: Palavras que substituem os substantivos ou os determinam, indicando a pessoa do discurso. Exemplo: Encontrei **seu** celular, porém não **o** desbloqueei.

6. **Verbo**: Exprime ação, estado, fato ou fenômeno. Exemplos: ser, estar, é, estava, abriu, nevou.

7. **Advérbio**: Modifica o sentido do verbo, adjetivo ou do próprio advérbio. Exemplo: Luiza cozinhou **bem**.

8. **Preposição**: Palavra invariável que liga um termo dependente a um termo principal, estabelecendo uma relação entre ambos. Exemplos: Olhei **para** ela, A televisão **de** Suelem era nova.

9. **Conjunção**: Palavra invariável que liga orações ou palavras da mesma oração. Exemplos: Os amigos comeram **e** estudaram, Chegamos em casa **quando** escurecia.

10. **Interjeição**: Palavra ou locução que exprime um estado emotivo. Exemplos: Eita!, Meu Deus!, Nossa!

As seis primeiras classes são variáveis, ou seja, tem sua forma alterada pois se flexionam, enquanto as outras quatro são invariáveis. Porém, em alguns casos, os advérbios podem ganhar sufixos visando dar a ideia de grau, como no seguinte exemplo: forte, fortíssimo e fortinho. É importante destacar que uma mesma palavra pode pertencer a diferentes classes gramaticais, cabendo a semântica identificar qual deles a palavra representa na sentença.

2.3 SINTAXE

A sintaxe é a área da linguística que trata a estrutura das frases das línguas naturais, tendo em vista especificar a sua estrutura interna e funcionamento (VIOTTI, 2008).

Cada palavra realiza uma determinada função sintática em suas respectivas frases. Uma sentença, para estar completa, precisa ter pelo menos um sujeito e um predicado. A seguir estão descritos todas as funções sintáticas conforme Bechara (2018):

- **Sujeito**: Indica a pessoa ou coisa de que afirmamos ou negamos uma ação ou qualidade. Exemplo: **Os meninos** jogaram futebol;

- **Predicado**: É tudo que se diz sobre o sujeito. Exemplo: Os meninos **jogaram futebol**;
- **Objeto direto**: Completa o sentido de um verbo transitivo direto ou de um verbo transitivo direto e indireto, sem precisar de preposição. Exemplo: Comprei **um telefone**;
- **Objeto indireto**: Completa o sentido de um verbo transitivo indireto ou transitivo direto e indireto, necessariamente por meio de uma preposição. Exemplo: Entreguei **a ela** meu trabalho;
- **Predicativo do sujeito**: Característica ou qualidade do sujeito dentro de um predicado nominal. Exemplo: Maria é **simpática**;
- **Predicativo do objeto**: Característica ou qualidade do objeto dentro de um predicado verbo-nominal. Exemplo: O marido suspeito deixou a esposa **desconfiada**;
- **Complemento nominal**: Completa o sentido de substantivos, adjetivos e advérbios por meio de uma preposição. Exemplo: Tenho vergonha **da minha irmã**;
- **Agente da passiva**: Quando a oração estiver na voz passiva analítica, isto é, for formada por um verbo auxiliar (ser ou estar) e um verbo principal, o termo que pratica a ação expressa pelo verbo principal é o agente da passiva. Exemplo: Marcos foi convidado por **Paulo**;
- **Adjunto adverbial**: Indica uma circunstância, que pode ser de lugar, finalidade, tempo, intensidade, causa ou modo. O adjunto adverbial pode mudar o sentido de um verbo, de um adjetivo, de um advérbio ou da oração inteira. Exemplo: Faz **calor** em Santa Catarina;
- **Adjunto adnominal**: Termo associado a um substantivo que o caracteriza ou complementa seu significado. Exemplo: A menina **responsável** estudou para a prova;
- **Aposto**: É o termo capaz de ampliar, explicar ou resumir o significado de outro termo ao qual aparece associado. O aposto geralmente aparece entre vírgulas ou antecedido por dois pontos na oração. Exemplo: Meu amigo, **que trabalha no shopping**, adora ir ao cinema;
- **Vocativo**: Expressa chamamento ou convocação, geralmente usado para invocar a atenção do interlocutor. Exemplo: **Luis**, não vá por aí!

2.4 SEMÂNTICA

A semântica, como definido por Cançado (2005), é a área da linguística responsável pela determinação do significado das frases. É o conhecimento semântico que faz com que

os falantes da língua saibam que duas frases sintaticamente diferentes se referem ao mesmo fato, como as duas frases a seguir:

Maria acredita, até hoje, que Papai Noel existe.

Maria ainda pensa, atualmente, que Papai Noel existe.

Toda frase carrega significado, assim como todas as palavras que a compõem. Porém, para chegar ao significado da frase não basta acumular os significados das palavras. Se fosse assim, *gatos perseguem ratos* e *ratos perseguem gatos* teriam o mesmo significado. O processo de compreender o significado das frases exige que sejam levados em consideração todos os aspectos da sentença, em especial as relações semânticas presentes. Veja a seguir algumas das relações semânticas mais comuns, conforme Cançado (2005) e Stock (2010):

Sinonímia: Ocorre quando um par de palavras ou sentenças diferentes apresentam o mesmo significado. Nesse contexto, ter o mesmo significado implica que elas possam ser trocadas mantendo o significado original da sentença onde estão aplicadas. Exemplos: *careca e calvo*, *forte e rápido* e *rápido e forte*.

Antonímia: Ocorre quando um par de palavras ou sentenças diferentes apresenta o significado oposto. Existem três tipos básicos de antonímia: binária, inversa e gradativa. Na binária, se uma das palavras for verdadeira, a outra necessariamente deve ser falsa, como em morto ou vivo, igual ou diferente e homem ou mulher. Na inversa, uma das palavras define uma determinada relação e a outra define a relação inversa, como em pai e filho e menor que e maior que. Na gradativa, as palavras estão em terminais opostos de uma escala contínua de valor, como quente e frio. Para esse tipo, a negação de um dos termos não significa a afirmação de outro, pois o fato de não estar quente não indica que está frio.

Meronímia: Ocorre quando uma das palavras representa uma parte e a outra representa o todo que compõe essa parte, como em perna $\rightarrow$ corpo.

Hiponímia: Ocorre quando uma das palavras tem seu significado abrangido dentro do significado da outra, como em cachorro $\rightarrow$ mamífero $\rightarrow$ animal.

No próximo capítulo serão apresentados conceitos de Processamento de Linguagem Natural, assim como técnicas que tratam de forma computacional os conceitos vistos neste capítulo.

3 PROCESSAMENTO NATURAL DE LINGUAGEM

A Linguagem Natural (LN) pode ser definida como a linguagem utilizada pelos seres humanos para se comunicarem, seja de forma falada ou escrita. Essa definição abrange inclusive expressões, gírias, entre outros padrões informais popularmente utilizados na comunicação (LOPES, 2002). Ela é categorizada como natural porque sua concepção se deu através de um processo de evolução natural, sendo o oposto das linguagens artificiais, como matemática, lógica e linguagens de programação, cuja criação foi planejada por uma ou mais pessoas com um determinado objetivo em mente (JACKSON; MOULINIER, 2007).

O Processamento de Linguagem Natural (PLN) é uma sub-área da Inteligência Artificial (IA), responsável por tratar computacionalmente os aspectos da linguagem natural, tanto na forma escrita quanto na falada (GONZALEZ; LIMA, 2003). Um dos principais focos de estudo nessa área é o entendimento da linguagem humana, que visa permitir que o computador possa compreender as informações presentes na linguagem da mesma forma que o ser humano (JACKSON; MOULINIER, 2007).

3.1 TÉCNICAS DE PLN

Para que o texto em linguagem natural seja entendido computacionalmente, e possa ser trabalhado conforme as diferentes aplicações, é necessário a aplicação de diversas técnicas, com responsabilidades e intenções diferentes. Neste item serão apresentadas as principais técnicas aplicadas nesse sentido.

3.1.1 Tokenização

A tokenização é a primeira etapa para o pré-processamento de um texto. Nela, o texto que inicialmente é representado como uma sequência de caracteres é agrupado em um primeiro nível segundo fronteiras delimitadas por caracteres primitivos como espaço (“ ”), vírgula, ponto etc. Cada grupo de caracteres estabelecidos no primeiro nível é chamado de *token*. Nessa etapa somente o espaço em branco é descartado. (ARANHA; VELLASCO, 2007).

O aluno, que está de camisa azul, estudou para a prova.

[O] [aluno] [,] [que] [está] [de] [camisa] [azul] [,] [estudou] [para] [a] [prova] [.]

3.1.2 Tokenização em N-grama

A tokenização em N-grama, ou *N-Gram* em inglês, é uma técnica na qual são testados agrupamentos de *tokens*, conhecidos como N-gramas, em dicionários de signifi-

cados, buscando encontrar agrupamentos maiores que sejam mais significativos. Na área de linguística computacional, uma N-grama é uma sequência contínua de N itens de uma determinado texto. Esses itens podem ser caracteres, sílabas, frases, etc. As sequências de n-gram de tamanho um, dois e três são chamadas respectivamente de unigramas, bigramas e trigramas (BARBOSA *et al.*, 2017)

No exemplo a seguir temos uma frase, suas separações em unigramas, bigramas e trigramas e o resultado final no qual se considera a maior combinação que tenha significado:

Unigramas: [Ele] [fez] [frango] [assado] [no] [forno] [elétrico]

Bigramas: [Ele fez] [fez frango] [frango assado] [assado no] [no forno] [forno elétrico]

Trigramas: [Ele fez frango] [fez frango assado] [frango assado no] [assado no forno] [no forno elétrico]

Resultado: [Ele] [fez] [frango assado] [no] [forno elétrico]

3.1.3 Normalização

Segundo Arampatzis *et al.* (2000) a normalização linguística se divide em três categorias distintas: morfológica, sintática e léxico-semântica.

A normalização morfológica é responsável por agrupar palavras que representam o mesmo conceito, evitando assim, a necessidade de tratar todas as variações existentes de palavras associadas a um mesmo conceito. Esse processo normalmente é feito através de algoritmos de conflação (*conflation*). É importante destacar que as palavras não são exatamente iguais, mas carregam grande semelhança em termos de significado (GONZALEZ; LIMA, 2003). Existem diversos tipos de processo de normalização morfológica. Alguns deles são:

Lematização: Substitui-se as diversas formas de representação da palavra pela forma primitiva. Normalmente se reduz os verbos no infinitivo e os adjetivos e substantivos à forma masculina singular. As formas “cantor”, “cantora”, “cantores” e “cantoras” seriam substituídas pelo lexema “cantor”, enquanto “cantaríamos” viraria “cantar” (ARAMPATZIS *et al.*, 2000).

Stemização: Reduz todas as palavras com mesmo radical a uma forma denominada *stem* (similar ao próprio radical), sendo eliminados os afixos (prefixos e sufixos). Em alguns casos, somente os sufixos são retirados. Nesse processo é possível que seja perdida a categoria morfológica original, uma vez que ambas as palavras “cantores” e “cantaríamos” seriam substituídas por “cant” (ORENGO; HUYCK, 2001).

A normalização sintática ocorre quando há a normalização de frases semanticamente equivalentes, mas sintaticamente diferentes, em apenas uma forma de representa-

ção, como em “processo eficiente e rápido” e “processo rápido e eficiente” (GONZALEZ; LIMA, 2003).

A normalização léxico-semântica ocorre quando são utilizadas as relações semânticas entre os itens lexicais para criar um agrupamento de similaridades semânticas, identificado por um item lexical que representa um conceito único.

Sendo assim, existem duas formas distintas para a normalização dos itens lexicais. Uma delas é a stemização, que explora a morfologia das palavras inferindo proximidades conceituais, e a outra é a normalização léxico-semântica, que busca sinônimos em dicionários de ideias (*thesaurus*) (GONZALEZ; LIMA, 2003).

3.1.4 Remoção dos *stopwords*

Um importante aspecto de todas as tarefas de *machine learning* envolvendo processamento de texto é a remoção de *stopwords*. Os textos em linguagem natural possuem uma grande proporção de palavras que não agregam ao significado da frase e são usadas somente para cumprir algum papel sintático. Alguns *stopwords* em português são: de, a, o, para, com, que, como, etc.

A remoção de *stopwords* melhora a performance das tarefas através do aumento na velocidade de execução, cálculos e também na precisão. Elas são removidas na etapa de pré-processamento do texto, o que ajuda a diminuir o tamanho do texto analisado (KAUR; BUTTAR, 2018).

3.1.5 Etiquetagem

A etiquetagem morfossintática, muito conhecida como *Part-of-speech (POS) Tagging*, tem como objetivo categorizar e etiquetar as palavras na sentença conforme sua classe gramatical (substantivo, adjetivo, verbos, advérbios, etc) e conforme sua função sintática (sujeito, predicado, objeto direto, etc). É uma das tarefas fundamentais do PLN, pois as atividades subsequentes beneficiam-se desta marcação das partes do discurso (MARQUEZ; PADRO; RODRIGUEZ, 2000).

3.1.6 Reconhecimento de Entidades Nomeadas

O reconhecimento de entidades nomeadas, do inglês *Named Entity Recognition (NER)*, busca identificar no texto as entidades nomeadas e classificá-las. Por entidades deve-se entender pessoas, números, expressão de tempo, lugares, instituições, etc. Porém, para reconhecer essas entidades de forma eficiente, faz-se necessário o reconhecimento de todos os objetos do texto (FONSECA *et al.*, 2015).

Na frase “Gabriel Lambrecht reside em Novo Hamburgo e trabalha na Rech Informática”, seriam encontradas três entidades: “Gabriel Lambrecht”, “Novo Hamburgo”

e “Rech Informática”.

Essa tarefa, apesar de natural para seres humanos, se torna complexa para tratar computacionalmente. Um exemplo disso pode ser visto em “Marta R. Bez”, pois há uma abreviação no meio com a presença de ponto, que muitas vezes é considerado como finalizador de sentença. Outro exemplo é “Professor Vandersílvio da Silva” no qual aparece o termo *da*, utilizado normalmente para separar dois termos (GONZALEZ; LIMA, 2003).

No capítulo a seguir, será apresentada uma revisão sistemática realizada para determinar qual o estado da arte sobre o assunto deste trabalho, buscando encontrar os objetos de estudos, técnicas utilizadas e formas de validação de trabalhos acerca de PLN.

4 TRABALHOS CORRELATOS

Esse capítulo apresenta uma revisão sistemática sobre a aplicação de Processamento de Linguagem Natural na extração de informação de documentos, buscando determinar qual o estado da arte desse assunto. Para isso, montou-se um protocolo composto por uma *string* de busca, as etapas executadas durante o desenvolvimento, os critérios de inclusão e exclusão dos artigos e as questões de interesse que essa revisão visa responder.

A base de dados escolhida para a execução da busca pelos artigos foi a *Web of Science* (WoS), devido ao grande volume de artigos contidos nela e também por abranger diversas áreas do conhecimento.

Como palavras-chave para a busca dos artigos foram definidos os termos *Natural language processing* ou NLP, *extration*, *document* e *sematinc*. A escolha dos termos foi feita através da realização de diversos testes na base de dados até que se chegou nesses termos que trouxeram os artigos mais relevantes com o objetivo desta revisão sistemática. Em sequência foi elaborada a seguinte *string* de busca: TS=(((*Natural language processing*) OR NLP) AND *extraction* AND *document* AND *semantic*).

Os seguintes critérios foram definidos para a inclusão ou exclusão dos artigos durante a aplicação do protocolo:

- (Inclusão) O artigo deve ter sido publicado nos últimos 5 anos;
- (Inclusão) O artigo deve estar escrito em inglês ou português;
- (Inclusão) O foco do artigo deve ser a extração de informação de textos não estruturados;
- (Inclusão) O artigo deve compreender o processo inteiro de interpretação de texto, iniciando com a informação natural e finalizando com a informação tratada;
- (Exclusão) O artigo não deve focar em um único idioma.

Foi definido que a execução do protocolo se daria em 5 etapas. Na primeira foram excluídos os artigos conforme a data de publicação. Em seguida, leu-se o título, palavras-chave e *abstract* para aplicação dos critérios de inclusão/exclusão. Após, foram lidos a introdução e conclusão, aplicando novamente os critérios de inclusão/exclusão. Na próxima etapa foi realizada a busca dos artigos completos, excluindo os que não estavam disponíveis. Por último, foram lidos de forma completa os artigos e excluídos os que não atendessem a todos os critérios.

Tabela 1 – Questões de interesse

Referência	Questão
Q1	Sobre quais textos houve a aplicação de PLN?
Q2	Quais técnicas de PLN foram utilizadas para o tratamento dos textos?
Q3	Como as aplicações foram validadas?

Fonte: do autor

Como questões de interesse definiu-se as três perguntas encontradas na Tabela 1 para que juntas pudessem determinar o estado da arte na aplicação de PLN à documentos textuais.

Tendo o protocolo sido definido, seguiu-se para a etapa de execução do mesmo. Para isso, aplicou-se a *string* de busca na base de dados *Web of Science*, retornando 75 registros. Os resultados foram exportados da base de dados no formato BibTex e importados na ferramenta stArt, que auxilia na execução da revisão sistemática. As etapas do processo foram executadas e os critérios de inclusão/exclusão aplicados, como pode ser visto na Figura 1. Ao final do processo, 4 artigos foram selecionados para responder às perguntas motivadoras da revisão.

Figura 1 – Exclusão de artigos conforme etapas

Fonte: do autor

4.1 SOBRE QUAIS TEXTOS HOUVE A APLICAÇÃO DE PLN?

Narouei, Takabi e Nielsen (2020) tiveram como foco os documentos de políticas de controle de acesso, nos quais são descritas regras de controle sobre quais pessoas ou papéis tem acesso a quais informações, entre outros aspectos de segurança que empresas que lidam com informações sensíveis devem definir para garantir a segurança dos dados.

Já Maree, Kmail e Belkhatir (2019) focaram seu trabalho nas descrições de vagas e de currículos. Descrições de vagas e currículos possuem características semelhantes pois listam, mesmo que em estruturas gramaticais diferentes, os mesmos grupos de informações, como escolaridade, conhecimentos, requisitos e etc.

Al-Hroob, Imam e Al-Heisa (2018) tiveram como foco os documentos de requisitos para o desenvolvimento de software. Esses documentos, muitas vezes escritos em forma de *User Story*, descrevem as rotinas que o software deve realizar.

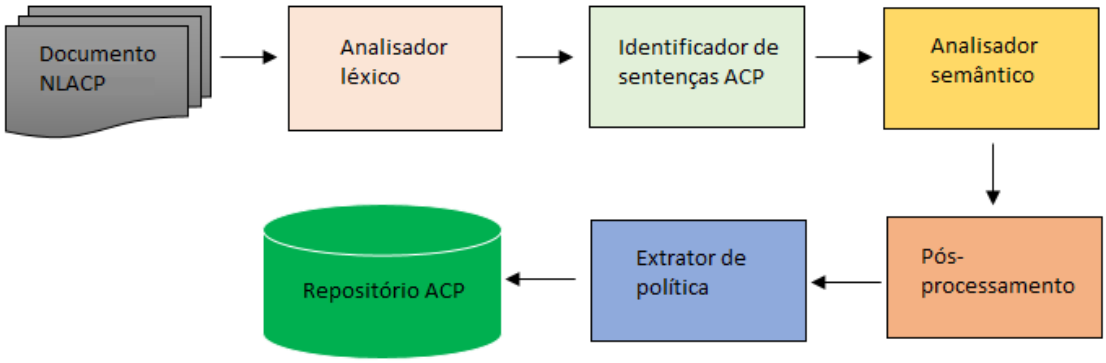
Por último, Hassanpour e Langlotz (2016) trabalhou com os laudos radiológicos, documentos nos quais contam o parecer do radiologista sobre as imagens obtidas pelo raio-x e compõem uma fonte muito importante de informação sobre o estado do paciente.

4.2 QUAIS TÉCNICAS DE PLN FORAM UTILIZADAS PARA O TRATAMENTO DOS TEXTOS?

Narouei, Takabi e Nielsen (2020) propuseram um *framework* composto de várias etapas e técnicas empregadas para o processamento das políticas de acesso, como pode ser visto na Figura 2. Inicialmente o texto é preparado separando as sentenças em linhas e é aplicado o analisador léxico, responsável pela segmentação e tokenização das sentenças. Para isso, eles utilizaram o *CoreNLP tool kit*. Em seguida é realizada a etapa de resolução de correferência, na qual são identificados termos e expressões que indicam a mesma entidade, utilizando o algoritmo descrito por Charniak e Elsner (2009). Na etapa seguinte, como nem todas as sentenças eram de política de controle de acesso (ACP), foi utilizada uma abordagem K-vizinhos mais próximos (KNN) usando modelos específicos de ACP já conhecidos da literatura. Em seguida foi realizada a etapa do analisador sintático, na qual foi usado *Semantic Role Labeling* (SNL) para identificar automaticamente a estrutura de predicados e argumentos. Na etapa de pós-processamento são tratados detalhes para a extração das ACP das sentenças, considerando a possibilidade de diversas ACP estarem descritas em uma mesma sentença. Por último, na etapa da extração de políticas, as palavras foram organizadas no formato de sujeito, objeto e ação.

O processamento de textos descritos por Maree, Kmail e Belkhatir (2019) iniciava com a separação do documento em unidades (sentenças ou parágrafos) cujo processamento é feito de forma separada. Em seguida era realizada a tokenização em *N-gram* considerando unigramas, bigramas e trigramas e submetida às bases semânticas *WordNet* e YAGO3. Em seguida, realizou-se a remoção dos *stopwords* e etiquetagem morfossintática (*part-of-speech tagging*), na qual foram consideradas somente os pronomes, verbos e advérbios. O processo se encerra com a realização do reconhecimento de entidades nomeadas usando o *RegexNER Stanford CoreNLP*, na qual, além das categorias normalmente utilizadas, o autor também adicionou categorias específicas da sua aplicação, como formação, experiência profissional, etc.

Figura 2 – *Framework* proposto em Narouei, Takabi e Nielsen (2020)



Fonte: traduzido de Narouei, Takabi e Nielsen (2020)

Hassanpour e Langlotz (2016) fizeram uso de diversas técnicas de PLN. Uma delas é a *part-of-speech tagging* utilizando o *Stanford Part of Speech Tagger*. Para a normalização, foi feita a stemização através do algoritmo *Porter stemmer*. Outra técnica utilizada foi a tokenização em *N-gram*. Além dessas, também foi aplicada a técnica *word shape* que classifica os *tokens* conforme seu formato (letras maiúsculas, minúsculas, números, pontuação, etc.) utilizando o *Stanford CoreNLP toolkit*. Para tratar a negação de palavras foi usado o *NegEx*, que busca descobrir quais palavras são afetadas pela negação. Para o reconhecimento de entidades, foram utilizados três modelos: baseado em dicionário, modelo CMM, também conhecido como *maximum entropy Markov models* ou MEMMs, e modelo de CRF. Para o modelo baseado em dicionário foi utilizado o *RadLex lexicon*, uma base de lexicon específicos para a área de radiologia, em conjunto com o *cTAKES*: um sistema de análise de textos médicos e extração de conhecimento. Tanto para o modelo CMM quanto para o CRF foi utilizada a rotina de treinamento presente no *Stanford Named-Entity Recognizer toolkit* para construir um modelo de reconhecimento de entidades para anotar e extrair informação dos laudos radiológicos.

O processo de extração presente em Al-Hroob, Imam e Al-Heisa (2018) era constituído pelas seguintes etapas: Análise Linguística, Pré-Processamento e Mapeamento para semânticas SRS (*Software Requirements Specification*). Ao final, era gerada uma lista de atores e ações presentes nos casos de uso. O processo utilizou uma abordagem semi-automática, pois o usuário precisa interagir durante o processamento. Na etapa de análise linguística, o arquivo de texto é importado no *general architecture for text engineering* ou GATE, um conjunto de ferramentas de NLP que possibilita o uso de *plugins*. A análise sintática foi feita usando o *plugin* ANNIE (*A Nearly-New Information Extraction System)* que extrai os pronomes e verbos com seus respectivos tipos. Em seguida, era utilizado o *plugin Stanford-CoreNLP* para a análise semântica, etapa na qual eram determinadas as

dependências de sujeito ou complemento. O resultado dessa etapa é uma tabela contendo a palavra e suas características: *token*, tamanho, formato (maiúsculo ou minúsculo), tipo de *token* e categoria da palavra. Na etapa de pré-processamento, são preparadas a entrada para a próxima etapa, que é um mapeamento *ANN (Artificial Neural Network)*. Para isso, essa etapa fez uso de um *corpus*, no qual cada palavra era substituída por um código correspondente. O algoritmo procura se a palavra já está na lista. Se sim, utiliza o valor já atribuído, senão, registra a palavra no próximo código disponível e utiliza esse código. Além disso, também foi feito um de-para dos código de POS (*Part-of-Speech*) e de dependência. Na etapa de mapeamento para semânticas SRS (*Software Requirements Specification*), foi utilizado o resultado das etapas anteriores aplicadas em uma rede neural de retropropagação, em inglês conhecidas como BPNN (*Back-Propagation Neural Networks*)). Essa rede possui três camadas. A camada de entrada é composta de 3 neurônios, correspondentes ao código da palavra, código do POS e o código da dependência. A camada escondida era composta de 3 camadas de 10 neurônios cada e a camada de saída por 3 neurônios para o autor, ação e outros tipos.

4.3 COMO AS APLICAÇÕES FORAM VALIDADAS?

Narouei, Takabi e Nielsen (2020), Al-Hroob, Imam e Al-Heisa (2018) e Hassanpour e Langlotz (2016) descrevem o processo de avaliação considerando três métricas: precisão (*precision*), recordação (*recall*) e *f1-score*.

Maree, Kmail e Belkhatir (2019) apresentam o uso de duas validações diferentes. A primeira delas buscou validar se o modelo de extração de *features* proposto era eficaz. Usando uma amostra de 3 vagas e 6 currículos, foi definido por um especialista um *score* de relevância para cada combinação e também anotado o *score* calculado pelo sistema fazendo uso ou não do modelo de extração de *features*. A outra validação visava determinar a precisão do *score* definido pelo sistema em comparação com o especificado pelo especialista. Para isso, foram utilizadas 3 vagas e 4 currículos, para os quais um especialista atribuiu um valor de *score* de relevância para cada combinação e essa foi comparada com o *score* automático gerado pelo sistema, chegando a um percentual de precisão calculado pela fórmula

$$P = \frac{\mid Vmanual - Vautomatico \mid}{\frac{Vmanual+Vautomatico}{2}} * 100\%$$

4.4 ARTIGOS CORRELATOS

Um dos artigos analisados foi o de Narouei, Takabi e Nielsen (2020), que tinha como objetivo facilitar a implantação de modelos de políticas de controle de acesso, uma vez que muitas empresas possuem essas informações de acesso em documentos de texto livre,

necessitando apenas formalizá-las. Para isso, foi desenvolvido um *framework* utilizando PLN que realizava a leitura desses documentos de linguagem natural e formalizava as políticas de controle de acesso encontradas neles. O *framework* atingiu um *f1-score* de 75% na identificação das políticas.

Outro trabalho analisado foi o de Maree, Kmail e Belkhatir (2019), que buscava aumentar a efetividade da sugestão de candidatos que atendessem a determinada vaga. Para isso, foi desenvolvido um *framework* utilizando PLN que transformava os currículos e as vagas escritos em linguagem natural em dados estruturados para então calcular a relevância do currículo conforme a vaga. A validação comprovou que o *framework* proposto foi capaz de melhorar o *score* de relevância do candidato na vaga, mas também apresentou que o *framework* proposto ainda possui diversas limitações.

O trabalho de Al-Hroob, Imam e Al-Heisa (2018) teve como objetivo a identificação automática de atores e ação presentes em documentos de descrição de requisitos funcionais no desenvolvimento de software, que muitas vezes são descritos na forma de *user case*. Tendo esse objetivo, foi desenvolvido um *framework* utilizando PLN e redes neurais em uma abordagem semi-automática, na qual o usuário precisa interagir durante o processo, para a interpretação. O artigo atingiu seu objetivo, tendo como resultados precisão de 17% a 63%, recordação de 56% a 100% e *f1-score* de 29% a 71%.

Por último, o trabalho de Hassanpour e Langlotz (2016) buscava a interpretação de laudos radiológicos para extração e destaque dos termos relevantes. Para isso foi desenvolvido um sistema de extração de informação que classificava os conteúdos dos documentos em 5 categorias: Anatomia, Modificador de Anatomia, Observação, Modificador de observação e Incerteza. No desenvolvimento foram investigados dois métodos de aprendizado de máquina, CMM e CRF, e um método baseado em dicionário popularmente usado. Através da validação usando um *dataset* de cerca de 150 laudos radiológicos foi possível constatar que ambas as propostas (CMM e CRF) eram melhores que a baseada em dicionário, apresentando *f1-score* médio de 85%.

O próximo capítulo apresenta o desenvolvimento do protótipo para análise das principais características dos *feedback* cooporativos.

5 METODOLOGIA

Foi escolhida como metodologia a ser seguida neste trabalho o *Design Science Research* (DSR). Essa metodologia é caracterizada em Junior *et al.* (2017) como um rigoroso processo para projetar artefatos visando a resolução de problemas, avaliação do que foi projetado e comunicação dos resultados obtidos.

Segundo Peffers *et al.* (2007) a DSR é desenvolvida através de seis etapas. A descrição dessas etapas, e como foram realizadas neste trabalho, se encontram logo abaixo:

Identificação do problema e sua motivação: Se define o problema de pesquisa a ser abordado, assim como a justificativa para a investigação. Para este trabalho foram realizadas pesquisas bibliográficas sobre Linguística e Processamento de Linguagem Natural, presentes nos capítulos 2 e 3, respectivamente. Os resultados dessa pesquisa também serviram como base para a escrita do capítulo da Introdução, na qual é possível encontrar o problema de pesquisa e a sua motivação.

Definição dos objetivos para a solução: Considerando que a solução é viável e factível, se definem os objetivos da solução a ser desenvolvida. Neste trabalho a definição dos objetivos se encontra também no capítulo de Introdução. Eles foram definidos após a pesquisa bibliográfica indicar que seria viável utilizar PLN para resolver o problema levantado na etapa anterior.

Projetar e desenvolver: São definidas as funcionalidades necessárias para o artefato, qual arquitetura será usada e é realizado o desenvolvimento do artefato, neste caso o protótipo proposto. Para atender a essa etapa neste trabalho se iniciou fazendo uma pesquisa por trabalhos correlatos, presente no capítulo 4. Com base nas informações de funcionalidades e arquiteturas dos trabalhos correlatos que essa pesquisa forneceu, foi definido quais ferramentas e arquiteturas seriam utilizadas, as quais podem ser encontradas neste capítulo. Referente a parte da execução do desenvolvimento, ele pode ser encontrado no capítulo 6, de Desenvolvimento.

Demonstração: São escolhidos um ou mais contextos do problema para a aplicação do artefato tendo como objetivo uma simulação dos resultados entregues pelo protótipo. Neste trabalho, essa etapa foi executada durante o desenvolvimento. A rotina desenvolvida para o treinamento do modelo BERTimbau possui um método de validação, no qual são utilizados alguns registros do *dataset* para validar o comportamento do modelo. Essa rotina foi muito importante, pois durante o desenvolvimento ajudou a avaliar o impacto das alterações e ajustes realizadas ao modelo em busca de uma maior eficácia.

Avaliação: Nessa etapa é medido e observado se o artefato foi capaz de solucionar o problema, comparando os resultados alcançados com os objetivos propostos. Nesse traba-

lho, essa etapa se encontra no capítulo 6, de desenvolvimento, em específico na sub-seção Resultados/análise e discussão. Foi utilizado um *dataset* de teste, reservado do *dataset* original, para medição dos resultados do modelo, comparando com os valores observados nos trabalhos correlatos.

Comunicação: O trabalho será publicado e o artefato desenvolvido será divulgado. Para isso foi realizada a inscrição e apresentação deste trabalho na Feira de Iniciação Científica da Universidade Feevale. Além disso, este trabalho também conta como divulgação da pesquisa desenvolvida.

5.1 FERRAMENTAS UTILIZADAS

A linguagem utilizada para o desenvolvimento deste trabalho foi Python. O motivo para essa escolha é porque Python é uma das linguagens mais utilizadas do momento para se trabalhar com PLN devido a grande quantidade de bibliotecas já existentes para essa finalidade.

Outra ferramenta muito importante é o PyTorch, um projeto *open-source* desenvolvido pela equipe de pesquisa em Inteligência Artificial do Facebook e lançado em 2016. Ele foi escrito usando as linguagem Python, C++ e CUDA, sendo pensado como uma ferramenta para Python. O PyTorch é o principal concorrente da biblioteca TensorFlow, do Google, já que ambos visam a oferecer rotinas auxiliares para o trabalho com *Machine Learning*. Um dos módulos básicos dessa ferramenta e que foi utilizado neste trabalho é o Tensor. Esse módulo implementa a classe dos tensores, com todos os seus atributos e métodos. O tensor é muito similar aos arrays, sendo que é possível a fácil conversão entre eles. A memória física dos tensores pode ser armazenada tanto em CPU como em GPU.

O ambiente utilizado para execução dos códigos desenvolvidos foi o Google Colaboratory, ou Colab, um produto desenvolvido pela equipe de pesquisa da Google. Ele permite que qualquer pessoa possa escrever e executar códigos Python através do navegador, sendo especialmente apropriado para rotinas de *machine learning*. A característica mais importante é que o Colab fornece um ambiente de execução gratuito, inclusive com o acesso a GPU, o que é indispensável para se trabalhar com *machine learning* e rotinas de treinamento que podem levar muito tempo se utilizado somente CPU.

CUDA é um modelo de programação paralela desenvolvido pela NVIDIA para computação em GPUs. Utilizando o CUDA, desenvolvedores podem diminuir drasticamente o tempo de execução das aplicações ao utilizar o poder computacional das GPUs. Nas aplicações que utilizam essa estratégia, a parte sequencial da aplicação é executada no CPU, enquanto a parte da aplicação que lida com intensa computação é enviada para a GPU para rodar em paralelo usando os milhares de processadores da GPU. Além disso, essa biblioteca permite, de modo simples, o envio de execuções para a GPU, sendo neces-

sário somente alguns simples comandos.

5.2 BERT E BERTIMBAU

Para a classificação de sentença buscou-se um modelo em que fosse possível realizar um ajuste fino (*fine-tuning*). Entre algumas opções encontradas, foi escolhido o modelo BERT a ser utilizado no Desenvolvimento.

O modelo BERT foi desenvolvido em 2019 pela equipe de Inteligência Artificial da Google e foi publicado em Devlin *et al.* (2019). Desde a sua publicação, o modelo tem sido utilizado em diversos trabalhos, sendo visto como um modelo mais influente no estado da arte atual sobre Processamento de Linguagem Natural. BERT é um acrônimo de *Bidirectional Encoder Representations from Transformers*.

O modelo utiliza uma estratégia de aprendizagem profunda (*deep learning*) na qual redes neurais profundas (*deep neural networks*) utilizam modelos bidirecionais usando representação de linguagem natural. A maioria dos modelos para PLN consideravam somente um sentido, seja da direita para esquerda ou da esquerda para a direita. Já o modelo BERT, por ser bidirecional, leva em conta todas as palavras (tanto as anteriores quanto posteriores) para identificar o contexto de um determinado termo na frase.

Utilizando um modelo pré-treinado é possível ajustá-lo conforme os diferentes tipos de aplicação na qual ele pode ser utilizado. Isso é possível através da implementação de uma camada de *fine-tuning* do modelo.

Como modelo pré-treinado, optou-se pelo uso do BERTimbau, um modelo BERT pré-treinado para o português publicado por Souza, Nogueira e Lotufo (2020).

5.3 VALIDAÇÃO

Para a validação do protótipo foram utilizadas três métricas comuns para a avaliação de redes neurais: precisão (*precision*), recordação (*recall*) e *f1-score*.

A recordação (R) é a proporção de argumentos que são encontrados ou previstos pelo sistema, enquanto precisão (P) é a proporção de argumentos que o sistema prevê corretamente. O *f1-score* é uma métrica de análise estatística que utiliza a recordação e precisão.

Para apurar essas métricas as predições são classificadas em 4 categorias: positivos verdadeiros (TP), negativos verdadeiros (TN), positivos falsos (FP) e negativos falsos (FN).

Considerando isso, a precisão é definida pela fórmula

$$P = \frac{TP}{TP + FP}$$

enquanto a recordação é definida como

$$R = \frac{TP}{TP + FN}$$

e o f1-score se define como

$$f1 - score = 2 * \frac{P * R}{P + R}$$

6 DESENVOLVIMENTO

Neste capítulo serão descritos todos os procedimentos realizados durante o desenvolvimento deste trabalho. Será descrito como foi criado o *dataset* a ser utilizado para treinamento, o funcionamento do protótipo desenvolvido, assim como detalhes sobre a rotina de classificação de sentenças e quais resultados foram atingidos.

6.1 GERAÇÃO DO DATASET

Um dos requisitos levantados para o protótipo é a classificação das sentenças encontradas no *feedback* entre positivas e negativas. Para atender a esse requisito, faz-se necessário o desenvolvimento de uma classe de avaliação da sentença, na qual será utilizado um algoritmo de Inteligência Artificial, mais especificamente um de *Machine Learning*. Esses algoritmos necessitam ser treinados para que, baseados nas entradas e saídas recebidas, seja construído um modelo analítico a ser utilizado para a predição de novas entradas. Diante disso, para o desenvolvimento da classe de classificação de sentenças, foi necessário um conjunto de dados, ou *dataset* como é popularmente chamado, para treinar a classe com sentenças e sua classificação como positiva ou negativa.

Inicialmente foi pesquisado por um *dataset* já pronto que pudesse atender a necessidade do protótipo. Para isso pesquisou-se nas principais plataformas colaborativas de dados, https://www.kaggle.com/ e https://huggingface.co/, por um *dataset* que pudesse ser usado para treinar o classificador de sentenças de *feedback*, porém não foi encontrado. Um dos fatores que contribuiu para que não tenham sido encontrados *dataset* foi que para aplicação em algoritmos de Processamento de Linguagem Natural (PLN) o idioma é fundamental, ou seja, não seria possível utilizar um *dataset* que não estivesse em português, uma vez que a entrada para os modelos de *machine learning* são as palavras presentes na sentença.

Outro fator que contribuiu para o insucesso da busca é o fato de que as informações de *feedback* do funcionário são informações sensíveis. Em um ambiente corporativo, caso um funcionário que não deveria ter acesso entre em contato com o *feedback* de outro funcionário, seria uma violação grave da privacidade das informações do funcionário, além de poder gerar grande constrangimento e desconforto no ambiente de trabalho. Porém, o fato de ser uma informação sensível não impede o seu compartilhamento em conjuntos de dados, desde de que seja feita a anonimização dos dados, removendo qualquer informação como nomes e outras características que possam identificar o funcionário.

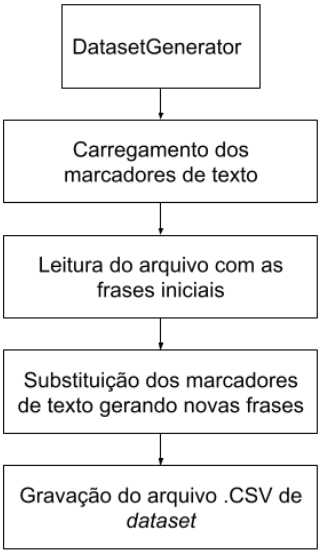
Como não foi encontrado um *dataset* pronto para utilização na classe de classificação de sentenças, fez-se necessária a criação de um *dataset* próprio. Para um *dataset* de

treinamento para uma *Machine Learning* é necessário um grande volume de registros, a fim de que o modelo possa interpretar diversas possibilidades de sentença. Para atender isso, foi elaborada uma estratégia de geração fazendo o uso de marcadores para a substituição de texto. Utilizando esses marcadores é possível substituir uma parte da frase, gerando assim, de uma frase original, diversas frases com sentidos diferentes. Por exemplo, uma das sentenças utilizadas foi "O desempenho do funcionário é [avaliacao_masculina]" que gera frases variadas como "O desempenho do funcionário é abaixo do esperado" e "O desempenho do funcionário é ótimo". Para a classificação da polaridade da frase, escolheu-se cinco níveis, sendo que cada nível é representado por um valor numérico e uma descrição:

- Muito negativo (-2)
- Negativo (-1)
- Neutro (0)
- Positivo (1)
- Muito positivo (2)

A implementação da classe de geração do *dataset* foi feita na linguagem Python, por ser a mesma linguagem utilizada no desenvolvimento do restante do protótipo, e pode ser encontrado em `https://github.com/gabriellambrecht/DatasetGenerator`. O fluxo de execução da classe *DatasetGenerator* pode ser observado na Figura 3 e será detalhado a seguir.

Figura 3 – Fluxograma do funcionamento da classe *DatasetGenerator*



Fonte: do autor

Tabela 2 – Lista de marcadores de texto para substituição

Marcador	Quantidade de opções	Afeta polaridade
[quantificador_masculino]	5	Sim
[quantificador_masculino_plural]	5	Sim
[quantificador_feminino]	5	Sim
[necessidade]	5	Sim
[avaliacao_masculina]	11	Sim
[avaliacao_feminina]	11	Sim
[avaliacao_masculina_plural]	11	Sim
[negacao]	2	Sim
[competencia_feminina]	25	Não
[competencia_masculina]	11	Não

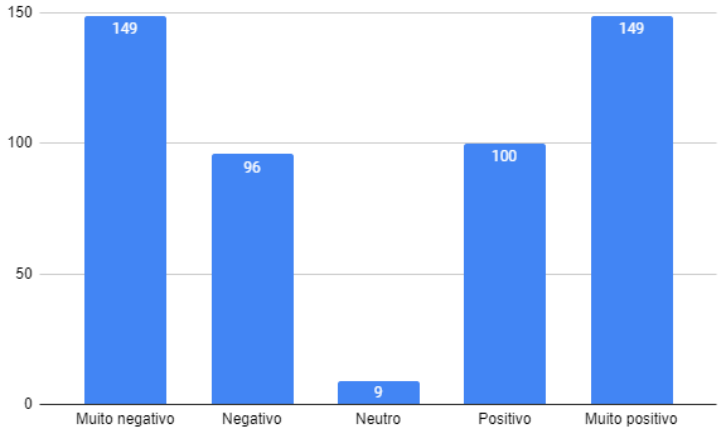
Fonte: do autor

No método de inicialização da classe são carregadas as listas de substituição para cada marcador de texto. Os marcadores que não contribuem para a mudança de polaridade da frase são do tipo *list*, pois tem somente o texto, enquanto os marcadores que mudam a polaridade são do tipo *dict*, pois são elementos chave e valor contendo o texto e a polaridade do texto correspondente. Além dos marcadores de texto, o método de inicialização também já carrega a lista de frases para substituição. Isso é feito através da leitura de um arquivo com a extensão .txt que contém 24 registros. Cada linha desse arquivo representa uma estrutura de frase com um ou mais marcadores de substituição. A substituição é feita através de um método recursivo no qual, para cada marcador encontrado, é percorrida a lista de termos para substituição e para cada termo é realizada a substituição, ajustado a polaridade da frase conforme o termo e encaminhada a frase resultante para o mesmo método, de forma recursiva, para caso ainda reste outro marcador na frase. Quando a frase não possui mais nenhum marcador a ser substituído, ela é registrada em um lista do tipo *dict* junto com o valor da polaridade calculado durante a substituição dos marcadores. Por último, com a lista de frases geradas, é possível chamar três métodos para o tratamento de saída. Um dos métodos registra a lista em arquivo .csv, sendo esse o método utilizado para a geração do *dataset* para treino da classe de classificação de sentença. Outro método simplesmente imprime a lista no console do Python e o último método imprime a lista no console, porém, imprimindo a descrição da polaridade.

A Tabela 2 mostra a lista de marcadores de texto existentes, assim como quantos termos para substituição cada marcador possui e se o marcador altera ou não a polaridade da frase.

O *dataset* gerado para utilização na classe de classificação de sentenças possui 503 linhas. Além disso, foi realizado um histograma do *dataset*, apresentado na Figura 4, para análise de quantos registros existem de cada tipo.

Figura 4 – Histograma de polaridade das frases do *dataset*



Fonte: do autor

6.2 AVALIADOR DE SENTENÇAS

O avaliador de sentença é a classe principal do protótipo. Essa classe recebe uma sentença, ou seja, uma frase e retorna qual a polaridade da frase, conforme as 5 classes pré-estabelecidas: Muito negativo, Negativo, Neutro, Positivo, Muito positivo.

Como modelo de inteligência artificial foi realizado um *fine-tuning* do modelo BERTimbau, um modelo BERT pré-treinado para o Português. Para o treinamento desse modelo foi utilizado como referência o código encontrado no artigo de Joshi (2020), sobre classificação de e-mail em *spam* ou não *spam* utilizando um modelo BERT.

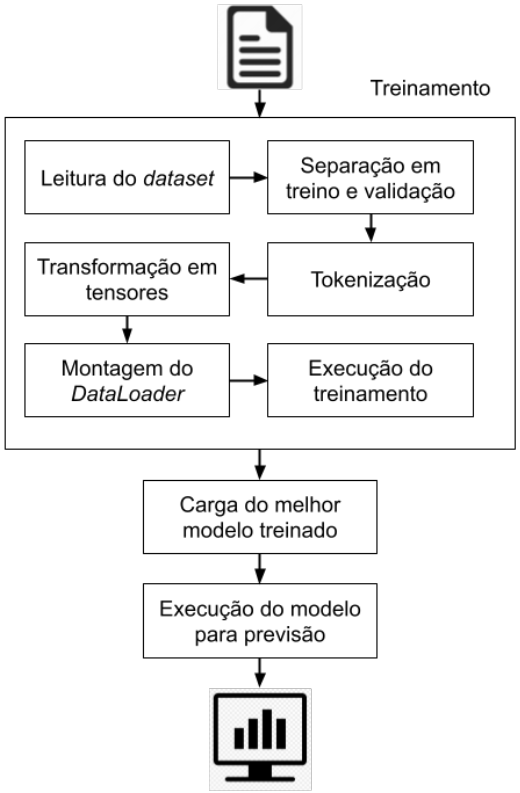
O código-fonte utilizado nessa seção está disponível em `https://github.com/gabriellambrecht/SentenceClassification`.

6.2.1 Fluxograma

A Figura 5 mostra o fluxo do processo de classificação de sentença.

Para a realização da classificação de sentenças quanto a sua polaridade, é necessário o treinamento do modelo BERTimbau. Esse treinamento foi realizado utilizando o *dataset* gerado na seção anterior. A leitura do *dataset* é a primeira etapa desse treinamento. Em seguida é executada a rotina de separação do *dataset* em três, sendo 70% para treinamento, 15% para validação e 15% para os testes. As sentenças presentes nesses *dataset* são então tokenizadas usando o tokenizador do BERTimbau, que retorna a lista de *id* e a máscara correspondente a cada sentença. Em seguida é realizada a criação do *DataLoader* para separação dos *datasets* de treinamento e validação em *batch* para realização do treinamento iterativo. Por último, é executada a rotina de treinamento, que utiliza o modelo de *fine-tuning* baseado no modelo BERTimbau desenvolvido neste trabalho, assim como a função de treinamento e validação.

Figura 5 – Fluxograma da classificação de sentença



Fonte: do autor

Com o modelo treinado é possível então realizar a previsão de sentenças utilizando o modelo. Para isso é carregado o estado interno do modelo treinado que apresente a menor perda entre as 10 épocas utilizadas no treinamento. Depois de carregar o melhor modelo, as sentenças de teste são tokenizadas e aplicadas no modelo, que retorna as previsões de polaridade para cada sentença.

6.2.2 Definindo dispositivo de execução

Antes de iniciar o treinamento é necessário definir alguns objetos que serão utilizados durante o desenvolvimento. Um desses objetos é o dispositivo que irá executar o treinamento. Essa informação é importante porque o treinamento da uma rede neural é um processo que pode levar bastante tempo dependendo de estrutura do modelo e do tamanho do *dataset*. Por isso, recomenda-se que seja executado através de processos paralelos, o que pode ser alcançado através do uso do ambiente CUDA, que permite o processamento em paralelo usando a GPU.

O serviço Google Colab utilizado neste trabalho permite tanto o uso de ambiente CPU quanto GPU. Diante disso, foi executado o comando exibido na Figura 6 para definir se a execução está sendo feita no ambiente CUDA ou CPU. Esse objeto será utilizado durante o treinamento quando for necessário enviar algum objeto para o dispositivo

apropriado para a execução.

Figura 6 – Define o dispositivo utilizado para execução

```
] import torch

  # Se disponível usa o ambiente CUDA, senão usa o CPU
  device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Fonte: do autor

6.2.3 Definição do tamanho para sentenças

Como o modelo o BERTimbau utiliza internamente uma rede neural, é preciso que a quantidade de entradas seja fixo. Nos modelos BERT, a quantidade de entradas é correspondente a quantidade de palavras presente nas sentenças, após ser realizada a tokenização.

Figura 7 – Histograma da quantidade de palavras no *dataset*

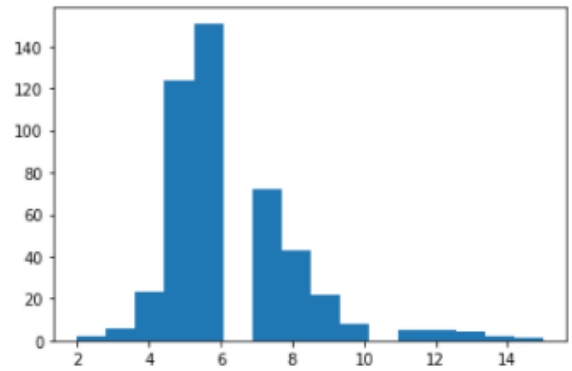
```
import pandas as pd

# Executa a leitura do dataset
dataset = pd.read_csv('dataset_feedback.csv', sep=',',header=None)
dataset.columns = ['text', 'label']

# Armazena em uma lista o tamanho de cada sentença
seq_len = [len(i.split()) for i in dataset['text']]

# Monta e exibe o histograma do tamanho de cada sentença
pd.Series(seq_len).hist(bins = 16, grid=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f920da982d0>
```



Fonte: do autor

Por isso, para determinar um tamanho fixo da quantidade de palavras, foi realizada a leitura do *dataset* em uma estrutura de dados utilizando a biblioteca pandas e realizado o cálculo de quantas palavras cada sentença possui através de uma simples separação utilizando o espaço como delimitador. O resultado dessa apuração é registrado em uma

lista que é transformada num objeto *Series* do pandas para que seja possível realizar o gráfico de histograma da quantidade de palavras presente nas sentenças, como pode ser visto na Figura 7.

Analisando o histograma apresentado, é possível observar que a sentença mais longa no *dataset* é de 15 palavras, sendo que os resultados se concentram principalmente no intervalo de 4 a 9 sentenças. Baseado nos resultados obtidos pelo histograma, foi definida a utilização do tamanho 17 como tamanho fixo das sentenças.

6.2.4 Carga do *dataset*

O processo de treinamento da classe e avaliação de sentenças começa com a leitura do *dataset*, como pode ser visto na Figura 8.

Figura 8 – Carga do *dataset* com ajuste na coluna de resultado

```
import numpy as np
import pandas as pd

dataset = pd.read_csv('dataset_feedback.csv', sep=',', header=None)
dataset.columns = ['text', 'label']
dataset.label = dataset.label+2
dataset.head()
```

	text	label
0	O funcionário possui pouquíssima perspicácia	0
1	O funcionário possui pouquíssima visão gerencial	0
2	O funcionário possui pouquíssima influência na...	0
3	O funcionário possui pouquíssima clareza	0
4	O funcionário possui pouquíssima qualidades	0

Fonte: do autor

O modelo BERT espera que as classes de saída sejam codificadas de forma sequencial com início em 0. Como o modelo utilizado neste trabalho possui 5 classes (Muito negativo, Negativo, Neutro, Positivo, Muito Positivo) é necessário que a codificação dele seja de 0 a 4. Para isso, é somado 2 na coluna do *dataset* que contém o valor da classificação, transformando assim o intervalo -2 a +2 em 0 a 4 e atendendo as instruções do modelo BERT.

6.2.5 Separação do *dataset*

Em seguida foi executada a separação do *dataset* em 3 grupos: treino, validação e teste. Para isso foi utilizada a função *train_test_split* da biblioteca *scikit-learn*, uma biblioteca de auxílio para se trabalhar com aprendizado de máquina (*machine learning*) em Python. Primeiramente foi executada a separação do *dataset* em um *dataset* de trei-

namento e um temporário, que posteriormente foi separado entre validação e teste, como pode ser vista na Figura 9.

Figura 9 – Separação do *dataset* em treino, validação e teste

```
from sklearn.model_selection import train_test_split

# Separa dataset original em dataset de treino e temporário
train_text, temp_text, train_labels, temp_labels = train_test_split(dataset['text'], dataset['label'],
                                                                    random_state=2021,
                                                                    test_size=0.3,
                                                                    stratify=dataset['label'])
# Separa dataset temporário em dataset de validação e teste
val_text, test_text, val_labels, test_labels = train_test_split(temp_text, temp_labels,
                                                                random_state=2021,
                                                                test_size=0.5,
                                                                stratify=temp_labels)
```

Fonte: do autor

Um dos parâmetros utilizados na rotina foi o *random_state*, um valor inteiro que é repassado a rotina internamente responsável por ordenar aleatoriamente os registros do *dataset*. Dessa forma, ao passar esse parâmetro tem-se a garantia de que independentemente de quantas execuções sejam realizadas, a separação será a mesma. Isso é muito importante pois durante o desenvolvimento são testadas diferentes configurações e parâmetros para o modelo e observadas as métricas para saber o resultado, necessitando neste caso de estabilidade nas informações de entrada.

O outro parâmetro opcional muito importante utilizado foi o *stratify*. Esse parâmetro ao ser utilizado recebe uma lista de valores e passa a respeitar a distribuição da lista recebida na separação dos *dataset*. Por exemplo, em um *dataset* que tenha 3 valores distintos distribuídos nas proporções 40%, 25% e 35%, esse parâmetro garante que os dois *dataset* separados também respeitem essa proporção. Isso é muito importante para o *dataset* utilizado nesse trabalho, uma vez que, como visto no capítulo anterior, a quantidade de registros para cada valor não é balanceada, possuindo muito mais registros nos extremos (Muito negativo e Muito positivo) do que no meio (Negativo, Neutro e Positivo).

Outra configuração utilizada foi o *test_size*, que recebe um valor entre 0 e 1 correspondente a porcentagem em forma decimal do *dataset* que será dividido do *dataset* original. Para este trabalho foi definido 0.3, o que significa um *dataset* de 70% para treinamento e 30% para o temporário. Esse temporário então é dividido usando 0.5, o que divide ele igualmente entre validação e teste. Dessa forma, ao final da rotina o *dataset* original foi dividido em 70% para treinamento, 15% para validação e 15% para teste.

6.2.6 Tokenização

A próxima etapa de execução do treinamento é a tokenização das sentenças para utilizar como entrada para o modelo BERTimbau. Para isso, a primeira atividade realizada foi a carga do tokenizador e do modelo Bertimbau pré-treinado, que será usado como base para o treinamento de *fine-tuning* para classificação das sentenças. Essa carga pode ser vista na Figura 10.

Figura 10 – Carga do modelo e tokenizador Bertimbau

```
import transformers
from transformers import AutoModel, AutoTokenizer

# Importa modelo Bertimbau pré-treinado em português
bert = AutoModel.from_pretrained('neuralmind/bert-large-portuguese-cased', num_labels=5)

# Importa tokenizer Bertimbau pré-treinamento em português
tokenizer = AutoTokenizer.from_pretrained('neuralmind/bert-large-portuguese-cased')
```

Fonte: do autor

A próxima etapa é tokenizar as sentenças para alimentar o modelo Bertimbau.

Com essa definição foi possível realizar a tokenização dos três *dataset* utilizados nessa rotina, como pode ser visto na Figura 11.

Figura 11 – Tokenização dos *dataset* de treinamento, validação e teste

```
# Define tamanho fixo das sentenças
max_seq_len = 17

def tokeniza_sentencas (sentencas):
  return tokenizer.batch_encode_plus(
          sentencas.tolist(),
          max_length = max_seq_len,
          padding='max_length',
          truncation=True,
          return_token_type_ids=False
  )

# Tokeniza as sentenças dos datasets de treinamento, validação e teste
tokens_train = tokeniza_sentencas(train_text)
tokens_val = tokeniza_sentencas(val_text)
tokens_test = tokeniza_sentencas(test_text)
```

Fonte: do autor

Foi definido o método tokeniza_sentencas que recebe a lista de sentenças a tokenizar e retorna a lista de sentenças tokenizadas. Para a tokenização foi utilizada a função *bacth_encode_plus* do tokenizador pré-carregado do BERTimbau. Foi utilizada essa função pois ela recebe toda a lista para a tokenização e também faz o *encode*, que nesse

caso é adicionar as *tags* delimitadoras de início e final de frase.1 Além da lista de sentenças, foram utilizados alguns parâmetros para garantir que todas as sentenças tenham o mesmo tamanho. A propriedade *max_length* recebe o valor de comprimento máximo das sentenças enquanto a propriedade *padding* é configurada para adicionar zeros ao final das sentenças menores para que elas fiquem com o tamanho correto. Já a propriedade *truncation* é ativada para que sentenças que sejam maiores que o limite sejam cortadas. Por último, a propriedade *return_token_type_ids* é desativada, pois as informações referente a tipagem dos *ids* não são necessárias.

Após a realização da tokenização é necessário transformar as listas em objetos tensores, usando o módulo *torch.tensor*, como pode ser visto na Figura 12.

Figura 12 – Transformação em tensores

```
import torch

# Transforma em tensores o conjunto de treinamento
train_seq = torch.tensor(tokens_train['input_ids'])
train_mask = torch.tensor(tokens_train['attention_mask'])
train_y = torch.tensor(train_labels.tolist())

# Transforma em tensores o conjunto de validação
val_seq = torch.tensor(tokens_val['input_ids'])
val_mask = torch.tensor(tokens_val['attention_mask'])
val_y = torch.tensor(val_labels.tolist())

# Transforma em tensores o conjunto de teste
test_seq = torch.tensor(tokens_test['input_ids'])
test_mask = torch.tensor(tokens_test['attention_mask'])
test_y = torch.tensor(test_labels.tolist())
```

Fonte: do autor

Foram criados três tensores para conjunto de informações. Os dois primeiros vêm do resultado da tokenização, sendo a propriedade *input_ids*, que contém os *ids* correspondentes a sentença e o *attention_mask*, uma lista que indica para todos os *ids* se ele corresponde a uma palavra ou é *padding*. O terceiro tensor é dos valores que correspondem a avaliação da frase, variando de 0 a 4 após realizado o ajuste na leitura do *dataset*.

6.2.7 Criação do *DataLoader*

Como o processo de treinamento da rede neural pode consumir muita memória do computador que estiver executando, é recomendado utilizar uma estratégia de treinamento iterativo, através de um *DataLoader*. A primeira coisa que devemos definir é o tamanho do lote (*batch size*). Esse número pode variar de 1 até o tamanho do *dataset*. Neste trabalho foi utilizado o valor 32, que é o tamanho padrão utilizado para a rotina. Isso quer dizer que

a cada iteração de treinamento a rede neural recebe 32 registros, até que todo o conjunto de treinamento tenha sido usado. A criação do *DataLoader* pode ser vista na Figura 13.

Figura 13 – Criação do *DataLoader*

```
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler

# Define o batch size (tamanho de registro para cada iteração do treinamento)
batch_size = 32

# Monta um DataLoader do dataset de treinamento
train_data = TensorDataset(train_seq, train_mask, train_y)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)

# Monta um DataLoader do dataset de validação
val_data = TensorDataset(val_seq, val_mask, val_y)
val_sampler = SequentialSampler(val_data)
val_dataloader = DataLoader(val_data, sampler = val_sampler, batch_size=batch_size)
```

Fonte: do autor

6.2.8 Definição do modelo com o *fine-tuning*

A próxima etapa é configurar o *fine-tuning* do modelo Bertimbau pré-treinado que foi importado. Para realizar esse ajuste fino é necessário definir uma classe de módulo para a rede neural e fazer a definição de dois métodos: *___init___* e *forward*, como visto na Figura 14.

Ambos os métodos são semelhantes, uma vez que no método de inicialização é definida a estrutura do modelo e no método de avanço é programada a execução dessa estrutura definida.

No método de inicialização um dos detalhes a ser observado é que o modelo BERT será recebido por parâmetro para a classe. Além disso, esse método configura algumas funções que serão utilizadas no método de avanço.

A primeira delas é o uso do *dropout* de 0.1 (10%) para a rede neural. Essa função faz com que, a cada execução, o modelo torne oculta uma determinada porcentagem dos neurônios (10% no caso desse trabalho) para forçar o treinamento de modelos diferentes, evitando assim o *overfitting*.

Como função de ativação foi carregada a função ReLU, utilizada recentemente como padrão em muitas redes neurais. Além dessa função, também foi carregada a função de ativação *LogSoftmax*, utilizada ao final do modelo para ajustar o vetor através da representação como probabilidades na escala logarítmica que somadas resultam em 1. Para essa rotina foi passado o parâmetro *dim* como 1 para indicar qual dimensão dos

Figura 14 – Configuração do modelo de *fine-tuning* do BERT

```
import torch.nn as nn

class BERT_Arch(nn.Module):

    # Define o método de inicialização
    def __init__(self, bert):
      super(BERT_Arch, self).__init__()
      self.bert = bert
      self.dropout = nn.Dropout(0.1)
      self.relu =  nn.ReLU()
      self.fc1 = nn.Linear(1024, 512)
      self.fc2 = nn.Linear(512,5)
      self.softmax = nn.LogSoftmax(dim=1)

    # Define o método de avanço
    def forward(self, sent_id, mask):
      cls_hs = self.bert(sent_id, attention_mask=mask, return_dict=False)[1]
      x = self.fc1(cls_hs)
      x = self.relu(x)
      x = self.dropout(x)
      x = self.fc2(x)
      x = self.softmax(x)
      return x
```

Fonte: do autor

tensores deve utilizar, sendo 0 a dimensão onde fica cada registro e 1 a dimensão onde fica a probabilidade de cada classe de saída por registro.

Além dessas funções, foram definidas duas camadas de transformações lineares. A primeira foi definida para transformar a dimensão de entrada (1024) para a dimensão interna utilizada no modelo BERT (512). A segunda transforma a dimensão interna (512) na dimensão de saída (5).

No método de avanço foi configurada a ordem em que cada função deve ser aplicada. Primeiro é chamado o modelo BERT passando os *ids* e a máscara proveniente da tokenização feita na frase. Esse modelo retorna um tensor de dimensão 1024 que é encaminhado na primeira camada de transformação linear e retorna um tensor de dimensão 512. Em seguida é aplicada a função de ativação, o *dropout* e, por fim, é aplicada a segunda camada de transformação linear, que retorna um tensor de saída com dimensão 5. Por último, é calculada a função de perda utilizando a função *LogSoftmax*, carregada na inicialização.

6.2.9 Definição dos pesos de classe

O próximo passo é a apuração dos pesos de cada classe da saída, ou seja, qual valor de peso deve ser utilizado para balancear a distribuição de resultados presente no *dataset* utilizado. A apuração destes pesos foi realizada através do uso de uma função encontrada na biblioteca *scikit-learn*, como pode ser visto na Figura 15.

Figura 15 – Apuração dos pesos das classes de saída

```
from sklearn.utils.class_weight import compute_class_weight

#Calcula os pesos de cada classe
class_wts = compute_class_weight('balanced', np.unique(train_labels), train_labels)
print(class_wts)
```

```
[0.62884615 1.52093023 2.10967742 1.45333333 0.62884615]
```

Fonte: do autor

Essa função recebe como parâmetro o tipo de cálculo, no caso desse trabalho sendo o cálculo balanceado, quais as opções de resultados, e a lista com os resultados utilizados para o treinamento. Analisando os pesos encontrados, é coerente com o histograma de valores do *dataset* visto no capítulo anterior, uma vez que o peso dos valores de extremidade são menores, já que compõem a maioria do *dataset* enquanto o peso da classe intermediária é maior, uma vez que é o resultado com menos ocorrência no *dataset*.

Com os pesos de classe definidos é possível configurar a classe de perda, como visto na Figura 16. Foi iniciado transformando o vetor em um objeto tensor do tipo *float* e então enviado para armazenar esse objeto no dispositivo que estiver rodando o treinamento. Esse objeto tensor é utilizado como parâmetro para configurar a classe *CrossEntropyLoss*, que será utilizada para calcular a perda (*loss*) do modelo.

Figura 16 – Configuração da classe de perda

```
import torch.nn as nn

# Converte pesos das classes em tensor e armazena no dispositivo de execução
weights= torch.tensor(class_wts,dtype=torch.float)
weights = weights.to(device)

# Configura classe de cálculo de perda usando os pesos
cross_entropy  = nn.CrossEntropyLoss(weight=weights)
```

Fonte: do autor

6.2.10 Funções de treino e avaliação

Tendo definida a classe do modelo com *fine-tuning* é possível então instância-lo, como pode ser visto na Figura 17.

Primeiramente é feito o congelamento dos parâmetros não utilizados do modelo BERTimbau importado, a fim de agilizar o processamento, uma vez que os gradientes de cada etapa não serão analisados neste trabalho.

Feito isso, é instanciado o objeto do modelo de *fine-tuning* passando o modelo BERTimbau como parâmetro. Esse objeto é então armazenado no dispositivo de execução

Figura 17 – Instância modelo e método de otimização

```
from transformers import AdamW

# Congela parâmetros não utilizados
for param in bert.parameters():
    param.requires_grad = False

# Instância o modelo com a fine-tuning e armazena no dispositivo de execução
model = BERT_Arch(bert)
model = model.to(device)

# Define a função de otimização
optimizer = AdamW(model.parameters(), lr = 1e-3)
```

Fonte: do autor

visando acelerar o processamento.

Por último, foi configurada a função de otimização Adam, passando como parâmetro a taxa de aprendizado de 0,001. Foi utilizado esse valor por ser o padrão ao se trabalhar com o Torch.

Em seguida, é definida a função de treinamento a ser utilizada, conforme pode ser visto na Figura 18.

Figura 18 – Função de treinamento

```
def train():
  model.train()
  total_loss = 0
  for batch in train_dataloader:
    batch = [r.to(device) for r in batch]
    sent_id, mask, labels = batch
    model.zero_grad()
    preds = model(sent_id, mask)
    loss = cross_entropy(preds, labels)
    total_loss = total_loss + loss.item()
    loss.backward()
    optimizer.step()
    preds=preds.detach().cpu().numpy()
  avg_loss = total_loss / len(train_dataloader)
  return avg_loss
```

Fonte: do autor

A função de treinamento se inicia com a execução do método *train* do modelo, que indica o início da etapa de treinamento, e com a inicialização da variável responsável por armazenar o valor total da perda durante o treinamento.

Em seguida é percorrido o objeto *DataLoader* carregado na etapa anterior, processando cada *batch* separadamente. A primeira coisa a ser feita com o *batch* é passar o seu armazenamento para o dispositivo de execução, visando aumentar o desempenho

quando rodando em uma GPU. Em seguida são extraídos do *batch* a lista de *ids*, máscaras e a classificação da sentença e executada uma função do modelo para que ele inicialize os gradientes internos, para que a execução anterior não interfira nessa nova execução. Após isso, é executado o modelo passando como parâmetros a tabela de *ids* e máscaras e retornando uma lista com a previsão de saída de todas as sentenças do *batch*. Com a lista da previsão de saída e com a lista das saídas original é executada a rotina para calcular a perda do treinamento. O cálculo dessa perda é feito através do método de *cross entropy* (entropia cruzada) que foi configurada anteriormente com os pesos de cada classe. Essa perda é acumulada na variável que totaliza a perda entre todos os *batch* do *dataloader*. Após isso é executada a rotina *backwards* da função de perda para que sejam atualizados os gradientes das etapas do modelo utilizado. Por precaução é chamado um método que limita os valores dos gradientes para que não ultrapassem o limite numérico após diversas execuções. Após isso, é executada a função de otimização indicando que foi executado um passo de treinamento.

O retorno dessa função é a perda média, calculada como sendo a perda total dividida pelo tamanho do *dataset* de treinamento.

Além da função de treinamento, também é necessário definir uma função de avaliação. Essa função, como pode ser vista na Figura 19, é muito semelhante à função de treinamento. As principais diferenças entre elas são o uso da função *eval* para indicar para o modelo que está na etapa de validação e que é percorrido o *dataloader* de validação ao invés do treinamento. Através do método *no_grad* é indicado que o uso do modelo não precisa gerar os gradientes, mesmo motivo para o qual não são utilizadas a função de *optmizer* e do método *backwards* da função de perda.

Figura 19 – Função de validação

```
def evaluate():
  model.eval()
  total_loss = 0
  for batch in val_dataloader:
    batch = [t.to(device) for t in batch]
    sent_id, mask, labels = batch
    with torch.no_grad():
      preds = model(sent_id, mask)
      loss = cross_entropy(preds,labels)
      total_loss = total_loss + loss.item()
      preds = preds.detach().cpu().numpy()
  avg_loss = total_loss / len(val_dataloader)
  return avg_loss
```

Fonte: do autor

6.2.11 Execução do treinamento

Após a execução das rotinas anteriores é possível então executar o treinamento do modelo, como é possível ver na Figura 20.

Figura 20 – Execução do treinamento

```
best_valid_loss = float('inf')
epochs = 10

for epoch in range(epochs):
    print('\n Epoch {:} / {:}'.format(epoch + 1, epochs))

    train_loss = train()
    valid_loss = evaluate()

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model.state_dict(), 'saved_weights.pt')

    print(f'\nTraining Loss: {train_loss:.3f}')
    print(f'Validation Loss: {valid_loss:.3f}')
```

Fonte: do autor

Devido a vários fatores, desde a separação dos *dataset* em *batch* quanto a função *dropout* que torna oculta alguns neurônios de forma aleatória para treinar diferentes cenários, cada execução do treinamento trará um resultado diferente. O treinamento e validação são executados um determinado número de vezes de forma completa, que é denominado de época (*epoch*). Neste trabalho foi definido a execução de 10 épocas, pois durante o desenvolvimento esse foi o melhor entre os valores testados que não apresentavam tanto *overfitting*.

Para cada época é executado o treinamento e validação do modelo registrando no console as informações de perda de ambas as funções. Além disso, como cada execução do treinamento possui um valor de perda diferente, são salvas as informações do modelo que apresenta melhor perda na etapa de treinamento. Esse modelo salvo é o modelo que será utilizado para a previsão na etapa de teste.

Na Tabela 3 são apresentados os resultados de cada época registrados no console durante a execução do treinamento do modelo. É possível observar na tabela que, durante a execução, o modelo apresentou a menor perda durante a validação para a época 40, sendo esse modelo que seria utilizado como o modelo oficial. Apesar de que nesse caso o melhor modelo tenha sido na última época executada, isso não necessariamente é uma regra. A perda realmente vai diminuindo conforme o modelo é treinado em diferentes cenários, mas é possível ver que entre a época 5 e 7 houve um aumento na perda de validação, ou seja, nem sempre mais execução garantirá uma perda melhor.

Tabela 3 – Perda de treinamento e validação através das épocas

Época	Perda de treinamento	Perda de validação
1	1.606	1.286
2	1.315	1.137
3	1.170	1.018
4	0.965	0.809
5	0.914	0.724
6	0.768	0.812
7	0.712	0.649
8	0.682	0.606
9	0.652	0.618
10	0.535	0.561

Fonte: do autor

Com isso, tem-se o modelo BERT treinado, usando o BERTimbau como base, e é possível executar o processo de teste, no qual serão coletados os resultados.

6.3 RESULTADOS/ANÁLISE E DISCUSSÃO

Para a apuração dos resultados, foi utilizado o *dataset* de teste reservado durante a etapa de treinamento. Esse *dataset* de teste corresponde a 15% do *dataset* original.

Como cada execução do treinamento fornece um modelo diferente, conforme fatores explicados na seção anterior, é importante registrar que para a obtenção dos resultados presentes nessa seção foi utilizado um modelo que registrou menor perda na época 10/10, tendo perda de treinamento de 0.535 e perda de validação de 0.561.

A Figura 21 mostra o processo para execução dos testes e a montagem do relatório de classificação.

Figura 21 – Execução do relatório de classificação

```
from sklearn.metrics import classification_report

# Carrega o estado do melhor modelo treinado
path = 'saved_weights.pt'
model.load_state_dict(torch.load(path))

# Busca previsões para o dataset de teste
with torch.no_grad():
  preds = model(test_seq.to(device), test_mask.to(device))
  preds = preds.detach().cpu().numpy()

# Imprime o relatório de classificação comparando as previsões e os valores reais
preds = np.argmax(preds, axis = 1)
print(classification_report(test_y, preds))
```

Fonte: do autor

Tabela 4 – Resultados do relatório de classificação

	Precisão	Recordação	*f1-score*	Registros
Muito negativo	0,66	0,83	0,73	23
Negativo	0,80	0,86	0,83	14
Neutro	0,50	1,00	0,67	1
Positivo	0,93	0,87	0,90	15
Muito positivo	0,88	0,61	0,72	23
Acurácia			0,78	76
Média geral	0,75	0,83	0,77	76
Média ponderada	0,80	0,78	0,78	76

Fonte: do autor

Primeiramente foi carregado o estado do modelo escolhido com menor perda na etapa de treinamento executada anteriormente. Em seguida é executada a rotina para previsão do modelo passando os *ids* e máscara do registro do *dataset* de teste e retornado os valores previstos do dispositivo de execução para o CPU. É importante notar que foi usado o método *no_grad()* pois na previsão não é necessário gerar novos gradientes.

Em seguida é chamado o método *classification_report*, da biblioteca *scikit-learn*, passando a lista de saída prevista e a lista de saída original. O resultado da execução do método pode ser visto na Tabela 4.

O relatório de classificação apresenta 3 métricas: precisão (*precision*), recordação (*recall*) e *f1-score* e uma coluna detalhando quantos registros foram utilizados para o teste. Essas informações são apuradas para as 5 classes de resultado (Muito negativo a muito positivo). Além dessas informações por classe, também é impresso a acurácia total (*accuracy*) que é composta pelo *f1-score* (uma vez que não existe precisão e recordação para múltiplas classes). Para cada uma das 3 métricas também são exibidas duas médias. A média geral faz a média simples dos 5 valores, enquanto a média ponderada leva em consideração o peso das classes, ou seja, as classes que têm mais registros irão influenciar mais na média.

O relatório de classificação apresenta que a acurácia do modelo é de 78% levando em consideração a métrica *f1-score*. Esse valor é bem próximo aos valores encontrados nos artigos correlatos, como pode ser observado na Tabela 5

Para entender o comportamento do modelo, além do relatório de classificação, também foi realizado a montagem de uma tabela de referência cruzada (*crosstab*), como visto na Tabela 6. Para isso, foi utilizado o método *crosstab* da biblioteca pandas que recebe duas listas (valores originais e valores previstos) e realiza a montagem da matriz representando a frequência dos valores presentes.

Esse comando da biblioteca No caso da Tabela 6, as linhas representam os valores

Tabela 5 – Métrica de eficácia *f1-score* comparada a trabalhos correlatos

Trabalho	*f1-score*
Al-Hroob, Imam e Al-Heisa (2018)	29% a 71%
Narouei, Takabi e Nielsen (2020)	75%
Gabriel (2021)	78%
Hassanpour e Langlotz (2016)	85%

Fonte: do autor

Tabela 6 – Tabela cruzada de resultados

	Muito negativo	Negativo	Neutro	Positivo	Muito Positivo
Muito negativo	19	2	0	0	2
Negativo	1	12	0	1	0
Neutro	0	0	1	0	0
Positivo	0	1	1	13	0
Muito Positivo	9	0	0	0	14

Fonte: do autor

originais, enquanto as colunas são os valores previstos. Por exemplo, a primeira linha pode ser interpretada da seguinte maneira: de todos os registros Muito negativo presentes no *dataset* de teste, 19 foram classificados como Muito negativo, 2 foram classificados como Negativo e 2 foram classificados como Muito positivo.

Essa tabela cruzada possibilita a realização de diversas avaliações sobre o comportamento do modelo treinado. Um dos comportamentos que se pode observar é que nas extremidades, Muito negativo e Muito positivo, apesar de apresentar uma precisão de 73% e 72% levando em consideração a métrica *f1-score*, os registros previstos incorretamente pelo modelo se localizam na extremidade oposta. No caso dos registros Muito positivo, 14 foram classificados como Muito positivo e 9 como Muito negativo. Isso é comportamento muito prejudicial para o modelo, uma vez que o resultado previsto de uma sentença pode ser o completo oposto.

Esse comportamento pode ter ocorrido devido ao fato de que o *dataset* que foi utilizado apresenta uma distribuição desbalanceada, tendo muito mais registros nas extremidades do que no centro. Dessa forma, o modelo pode não ter aprendido corretamente as nuances para classificar os registros em um classe intermediária (Negativo, Neutro e Positivo).

O próximo capítulo apresenta as conclusões das atividades desenvolvidas nesse trabalho até o presente momento.

7 CONCLUSÃO

Nos capítulos anteriores foram estudados conceitos relacionados a Linguística e Processamento de Linguagem Natural. Esses estudos possibilitaram um entendimento sobre o fluxo do processo de aplicação de PLN para extração de conhecimento, que se inicia com o texto como um fluxo de caracteres até a compreensão semântica por meios computacionais. Os conhecimentos adquiridos foram aplicados no desenvolvimento do protótipo que integra o objetivo deste trabalho.

Através da revisão sistemática, descrita no capítulo 4, foram identificados os contextos nos quais PLN está sendo aplicado, as técnicas e ferramentas utilizadas nos tratamentos dos textos e os critérios de validação das aplicações. Essas informações serviram de base para a etapa de desenvolvimento.

O objetivo desse trabalho, que era desenvolver um protótipo para análise de *feedbacks* corporativos através do uso de processamento de linguagem natural, foi concluído. Através do protótipo desenvolvido foi demonstrado que é possível classificar as sentenças presentes em um feedback quanto a sua positividade, permitindo assim um entendimento computacional sobre o *feedback* e possibilitando o desenvolvimento de uma ferramenta para auxílio ao profissional de gestão de pessoas na análise de *feedback*.

Quanto aos objetivos específicos, o primeiro que era caracterizar *feedback* e avaliação de desempenho no contexto corporativo foi realizado através de uma pesquisa teórica realizada e os resultados constam na Introdução, ajudando a contextualizar o cenário do problema de pesquisa. O segundo objetivo, realizar uma pesquisa teórica sobre processamento de linguagem natural (PLN), foi realizado e resultou no capítulo 3. Já o terceiro e quarto objetivo, propor uma estratégia para avaliação das informações de *feedback* utilizando PLN e desenvolver um protótipo utilizando a estratégia proposta, foram realizados e compõem o capítulo 6. Por fim, o último objetivo, verificar a aderência do protótipo desenvolvido aplicando em um cenário corporativo real, não foi realizado, pois iria demandar mais tempo e por isso consta como um possível trabalho futuro deste trabalho.

O protótipo desenvolvido neste trabalho atingiu a eficácia de 78% considerando a métrica *f1-score*, utilizada para a apuração de eficiência em modelos de classificação. Essa métrica é semelhante aos resultados encontrados nos trabalhos correlatos pesquisados, comprovando assim o resultado obtido neste trabalho.

Como trabalho futuro, uma das possibilidades a serem exploradas é a diversificação e complementação do *dataset*. Como visto nos resultados obtidos no capítulo de desenvolvimento, foi levantado a hipótese de que a distribuição irregular do *dataset* pode ter sido a causa do comportamento do modelo de, para as extremidades (Muito negativo

e Muito positivo), prever o resultado oposto nos casos em que o modelo não acerta. Além disso, como o *dataset* foi gerado a partir de substituição de uma lista pequena de frases é provável que o ele não tenha diversidade na estrutura das frases, o que pode diminuir o a precisão durante aplicação de sentenças não encontradas no *dataset*.

Outra oportunidade futura é a aplicação do protótipo em um ambiente corporativo através da criação de uma ferramenta para o auxílio do profissional de gestão de pessoas da empresa. Com esse trabalho o modelo seria exposto às necessidades reais encontradas na empresa, o que pode tornar o modelo ainda mais potente na classificação de sentenças no contexto de avaliação.

REFERÊNCIAS

AL-HROOB, A.; IMAM, A. T.; AL-HEISA, R. The use of artificial neural networks for extracting actions and actors from requirements document. *INFORMATION AND SOFTWARE TECHNOLOGY*, 101, p. 1–15, SEP 2018. ISSN 0950-5849. Citado 5 vezes nas páginas 24, 25, 26, 27 e 50.

ARAMPATZIS, A. *et al.* Linguistically-motivated information retrieval. In: _____. [S.l.: s.n.], 2000. v. 69, p. 201–222. Citado na página 19.

ARANHA, C. N.; VELLASCO, M. Uma abordagem de pré-processamento automático para mineração de textos em português: sob o enfoque da inteligência computacional. *Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ*, p. 33–34, 2007. Citado na página 18.

BARBOSA, J. *et al.* Introdução ao processamento de linguagem natural usando python. *III Escola Regional de Informatica do Piauí*, v. 1, p. 336–360, 2017. Citado na página 19.

BECHARA, E. *Lições de português pela análise sintática*. [S.l.]: Nova Fronteira, 2018. Citado na página 15.

CANÇADO, M. Manual de semântica. *Belo Horizonte: Editora UFMG*, 2005. Citado 2 vezes nas páginas 16 e 17.

CEGALLA, D. P. Novíssima gramática da língua portuguesa. Companhia Ed. Nacional, 2008. Citado na página 14.

CHARNIAK, E.; ELSNER, M. Em works for pronoun anaphora resolution. In: *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. USA: Association for Computational Linguistics, 2009. (EACL '09), p. 148–156. Citado na página 24.

CHIAVENATO, I. *Gestão de pessoas*. [S.l.]: Elsevier Brasil, 2008. Citado na página 11.

DEVLIN, J. *et al.* BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://aclanthology.org/N19-1423>. Citado na página 30.

DUARTE, P. M. T. Do lexema e termos afins na terminologia gramatical. Revista Philologus, 2004. Citado na página 14.

FONSECA, E. B. *et al.* Reconhecimento de entidades nomeadas para o português usando o opennlp. *Anais do ENIAC 2015, 2015, Brasil.*, 2015. Citado na página 20.

GONZALEZ, M.; LIMA, V. L. S. de. Recuperação de informação e processamento da linguagem natural. *XXIII Congresso da Sociedade Brasileira de Computação*, p. 347–395, 2003. Citado 5 vezes nas páginas 12, 18, 19, 20 e 21.

HASSANPOUR, S.; LANGLOTZ, C. P. Information extraction from multi-institutional radiology reports. *ARTIFICIAL INTELLIGENCE IN MEDICINE*, 66, p. 29–39, JAN 2016. ISSN 0933-3657. Citado 5 vezes nas páginas 24, 25, 26, 27 e 50.

JACKSON, P.; MOULINIER, I. *Natural Language Processing for Online Applications: Text retrieval, extraction and categorization. Second revised edition*. John Benjamins, 2007. 3 p. Disponível em: <https://www.jbe-platform.com/content-/books/9789027292445>. Citado na página 18.

JOSHI, P. *Transfer Learning NLP: Fine Tune Bert For Text Classification*. 2020. Disponível em: <https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/>. Citado na página 35.

JUNIOR, V. *et al.* Design science research methodology as methodological strategy for technological research. *Espacios*, v. 38, p. 25, 02 2017. Citado na página 28.

KAUR, J.; BUTTAR, P. K. A systematic review on stopword removal algorithms. *International Journal on Future Revolution in Computer Science & Communication Engineering*, v. 4, n. 4, p. 207–210, 2018. Citado na página 20.

KNAPIK, J. *Gestão de pessoas e talentos*. Ibpex, 2008. ISBN 9788587053657. Disponível em: <https://books.google.com.br/books?id=K-CGLor1j90C>. Citado na página 11.

LOPES, I. L. Uso das linguagens controlada e natural em bases de dados: revisão da literatura. *Ciência da Informação*, scielo, v. 31, p. 41 – 52, 01 2002. ISSN 0100-1965. Citado na página 18.

MALHEIROS, B.; ROCHA, A.; RAMAL, A. *Avaliação e gestão de desempenho*. Grupo Gen - LTC, 2014. (MBA : gestâo de pessoas). ISBN 9788521626244. Disponível em: <https://books.google.com.br/books?id=BD4NswEACAAJ>. Citado na página 11.

MAREE, M.; KMAIL, A. B.; BELKHATIR, M. Analysis and shortcomings of e-recruitment systems: Towards a semantics-based approach addressing knowledge incompleteness and limited domain coverage. *JOURNAL OF INFORMATION SCIENCE*, 45, n. 6, p. 713–735, DEC 2019. ISSN 0165-5515. Citado 3 vezes nas páginas 24, 26 e 27.

MARQUEZ, L.; PADRO, L.; RODRIGUEZ, H. A machine learning approach to pos tagging. *Machine Learning*, Springer, v. 39, n. 1, p. 59–91, 2000. Citado na página 20.

MISSEL, S. *FEEDBACK CORPORATIVO - Como saber se está indo bem*. Saraiva Educação S.A., 2017. ISBN 9788557170322. Disponível em: <https://books.google.com-.br/books?id=ckNnDwAAQBAJ>. Citado na página 11.

NAROUEI, M.; TAKABI, H.; NIELSEN, R. Automatic Extraction of Access Control Policies from Natural Language Documents. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 17, n. 3, p. 506–517, MAY-JUN 2020. ISSN 1545-5971. Citado 4 vezes nas páginas 23, 24, 26 e 50.

NETO, P. C.; INFANTE, U. *Gramática da língua portuguesa*. [S.l.]: Scipione São Paulo, 2003. Citado na página 14.

ORENGO, V. M.; HUYCK, C. A stemming algorithmm for the portuguese language. In: . [S.l.: s.n.], 2001. p. 186– 193. ISBN 0-7695-1192-9. Citado na página 19.

PEFFERS, K. *et al.* A design science research methodology for information systems research. *J. Manage. Inf. Syst.*, M. E. Sharpe, Inc., USA, v. 24, n. 3, p. 45–77, dez. 2007. ISSN 0742-1222. Disponível em: <https://doi.org/10.2753/MIS0742-1222240302>. Citado na página 28.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: pretrained BERT models for Brazilian Portuguese. In: *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*. [S.l.: s.n.], 2020. Citado na página 30.

STOCK, W. Concepts and semantic relations in information science. *JASIST*, v. 61, p. 1951–1969, 10 2010. Citado na página 17.

VASCONCELOS, V. L. S. de; ALBUQUERQUE, E. A. de. Feedback e sua contribuição para o desenvolvimento profissional. *Revista Cientefico*, v. 16, n. 33, p. 223–252, 2016. Disponível em: <https://cientefico.emnuvens.com.br/cientefico/article/view/202>. Citado na página 11.

VILELA, M. O léxico do português: perspectiva geral. *Filologia e Linguística Portuguesa*, n. 1, p. 31–50, ago. 1997. Disponível em: <https://www.revistas.usp.br/flp/article/view-/59644>. Citado na página 14.

VIOTTI, E. Introdução aos estudos linguísticos. *Universidade Federal De Santa Catarina. Florianópolis*, 2008. Citado na página 15.

Apêndices

APÊNDICE A – *DATASET* DE SENTENÇAS CLASSIFICADAS

Sentença	Classificação
O funcionário possui pouquíssima capacidade técnica	-2
O funcionário possui pouquíssima perspicácia	-2
O funcionário possui pouquíssima educação	-2
O funcionário possui pouquíssima inovação	-2
O funcionário possui pouquíssima experiência	-2
O funcionário possui pouquíssima confiabilidade	-2
O funcionário possui pouquíssima qualidades	-2
O funcionário possui pouquíssima visão gerencial	-2
O funcionário possui pouquíssima liderançavisão estratégica	-2
O funcionário possui pouquíssima desenvoltura	-2
O funcionário possui pouquíssima responsabilidade	-2
O funcionário possui pouquíssima clareza	-2
O funcionário possui pouquíssima assiduidade	-2
O funcionário possui pouquíssima criatividade	-2
O funcionário possui pouquíssima performance	-2
O funcionário possui pouquíssima inteligência social e emocional	-2
O funcionário possui pouquíssima facilidade com os problemas encontrados	-2
O funcionário possui pouquíssima competências comportamentais	-2
O funcionário possui pouquíssima organização	-2
O funcionário possui pouquíssima competências técnicas	-2
O funcionário possui pouquíssima dedicação	-2
O funcionário possui pouquíssima influência na empresa	-2
O funcionário possui pouquíssima ética	-2
O funcionário possui pouquíssima comunicação	-2
O funcionário não possui pouquíssima capacidade técnica	2
O funcionário não possui pouquíssima perspicácia	2
O funcionário não possui pouquíssima educação	2
O funcionário não possui pouquíssima inovação	2
O funcionário não possui pouquíssima experiência	2
O funcionário não possui pouquíssima confiabilidade	2
O funcionário não possui pouquíssima qualidades	2
O funcionário não possui pouquíssima visão gerencial	2
O funcionário não possui pouquíssima liderançavisão estratégica	2
O funcionário não possui pouquíssima desenvoltura	2

O funcionário não possui pouquíssima responsabilidade	2
O funcionário não possui pouquíssima clareza	2
O funcionário não possui pouquíssima assiduidade	2
O funcionário não possui pouquíssima criatividade	2
O funcionário não possui pouquíssima performance	2
O funcionário não possui pouquíssima inteligência social e emocional	2
O funcionário não possui pouquíssima facilidade com os problemas encontrados	2
O funcionário não possui pouquíssima competências comportamentais	2
O funcionário não possui pouquíssima organização	2
O funcionário não possui pouquíssima competências técnicas	2
O funcionário não possui pouquíssima dedicação	2
O funcionário não possui pouquíssima influência na empresa	2
O funcionário não possui pouquíssima ética	2
O funcionário não possui pouquíssima comunicação	2
O funcionário possui pouca capacidade técnica	-2
O funcionário possui pouca perspicácia	-2
O funcionário possui pouca educação	-2
O funcionário possui pouca inovação	-2
O funcionário possui pouca experiência	-2
O funcionário possui pouca confiabilidade	-2
O funcionário possui pouca qualidades	-2
O funcionário possui pouca visão gerencial	-2
O funcionário possui pouca liderançavisão estratégica	-2
O funcionário possui pouca desenvoltura	-2
O funcionário possui pouca responsabilidade	-2
O funcionário possui pouca clareza	-2
O funcionário possui pouca assiduidade	-2
O funcionário possui pouca criatividade	-2
O funcionário possui pouca performance	-2
O funcionário possui pouca inteligência social e emocional	-2
O funcionário possui pouca facilidade com os problemas encontrados	-2
O funcionário possui pouca competências comportamentais	-2
O funcionário possui pouca organização	-2
O funcionário possui pouca competências técnicas	-2

O funcionário possui pouca dedicação	-2
O funcionário possui pouca influência na empresa	-2
O funcionário possui pouca ética	-2
O funcionário possui pouca comunicação	-2
O funcionário não possui pouca capacidade técnica	2
O funcionário não possui pouca perspicácia	2
O funcionário não possui pouca educação	2
O funcionário não possui pouca inovação	2
O funcionário não possui pouca experiência	2
O funcionário não possui pouca confiabilidade	2
O funcionário não possui pouca qualidades	2
O funcionário não possui pouca visão gerencial	2
O funcionário não possui pouca liderançavisão estratégica	2
O funcionário não possui pouca desenvoltura	2
O funcionário não possui pouca responsabilidade	2
O funcionário não possui pouca clareza	2
O funcionário não possui pouca assiduidade	2
O funcionário não possui pouca criatividade	2
O funcionário não possui pouca performance	2
O funcionário não possui pouca inteligência social e emocional	2
O funcionário não possui pouca facilidade com os problemas encontrados	2
O funcionário não possui pouca competências comportamentais	2
O funcionário não possui pouca organização	2
O funcionário não possui pouca competências técnicas	2
O funcionário não possui pouca dedicação	2
O funcionário não possui pouca influência na empresa	2
O funcionário não possui pouca ética	2
O funcionário não possui pouca comunicação	2
O funcionário possui capacidade técnica	1
O funcionário possui perspicácia	1
O funcionário possui educação	1
O funcionário possui inovação	1
O funcionário possui experiência	1
O funcionário possui confiabilidade	1
O funcionário possui qualidades	1
O funcionário possui visão gerencial	1
O funcionário possui liderançavisão estratégica	1

O funcionário possui desenvoltura	1
O funcionário possui responsabilidade	1
O funcionário possui clareza	1
O funcionário possui assiduidade	1
O funcionário possui criatividade	1
O funcionário possui performance	1
O funcionário possui inteligência social e emocional	1
O funcionário possui facilidade com os problemas encontrados	1
O funcionário possui competências comportamentais	1
O funcionário possui organização	1
O funcionário possui competências técnicas	1
O funcionário possui dedicação	1
O funcionário possui influência na empresa	1
O funcionário possui ética	1
O funcionário possui comunicação	1
O funcionário não possui capacidade técnica	-1
O funcionário não possui perspicácia	-1
O funcionário não possui educação	-1
O funcionário não possui inovação	-1
O funcionário não possui experiência	-1
O funcionário não possui confiabilidade	-1
O funcionário não possui qualidades	-1
O funcionário não possui visão gerencial	-1
O funcionário não possui liderançavisão estratégica	-1
O funcionário não possui desenvoltura	-1
O funcionário não possui responsabilidade	-1
O funcionário não possui clareza	-1
O funcionário não possui assiduidade	-1
O funcionário não possui criatividade	-1
O funcionário não possui performance	-1
O funcionário não possui inteligência social e emocional	-1
O funcionário não possui facilidade com os problemas encontrados	-1
O funcionário não possui competências comportamentais	-1
O funcionário não possui organização	-1
O funcionário não possui competências técnicas	-1
O funcionário não possui dedicação	-1
O funcionário não possui influência na empresa	-1
O funcionário não possui ética	-1

O funcionário não possui comunicação	-1
O funcionário possui muita capacidade técnica	2
O funcionário possui muita perspicácia	2
O funcionário possui muita educação	2
O funcionário possui muita inovação	2
O funcionário possui muita experiência	2
O funcionário possui muita confiabilidade	2
O funcionário possui muita qualidades	2
O funcionário possui muita visão gerencial	2
O funcionário possui muita liderançavisão estratégica	2
O funcionário possui muita desenvoltura	2
O funcionário possui muita responsabilidade	2
O funcionário possui muita clareza	2
O funcionário possui muita assiduidade	2
O funcionário possui muita criatividade	2
O funcionário possui muita performance	2
O funcionário possui muita inteligência social e emocional	2
O funcionário possui muita facilidade com os problemas encontrados	2
O funcionário possui muita competências comportamentais	2
O funcionário possui muita organização	2
O funcionário possui muita competências técnicas	2
O funcionário possui muita dedicação	2
O funcionário possui muita influência na empresa	2
O funcionário possui muita ética	2
O funcionário possui muita comunicação	2
O funcionário não possui muita capacidade técnica	-2
O funcionário não possui muita perspicácia	-2
O funcionário não possui muita educação	-2
O funcionário não possui muita inovação	-2
O funcionário não possui muita experiência	-2
O funcionário não possui muita confiabilidade	-2
O funcionário não possui muita qualidades	-2
O funcionário não possui muita visão gerencial	-2
O funcionário não possui muita liderançavisão estratégica	-2
O funcionário não possui muita desenvoltura	-2
O funcionário não possui muita responsabilidade	-2
O funcionário não possui muita clareza	-2

O funcionário não possui muita assiduidade	-2
O funcionário não possui muita criatividade	-2
O funcionário não possui muita performance	-2
O funcionário não possui muita inteligência social e emocional	-2
O funcionário não possui muita facilidade com os problemas encontrados	-2
O funcionário não possui muita competências comportamentais	-2
O funcionário não possui muita organização	-2
O funcionário não possui muita competências técnicas	-2
O funcionário não possui muita dedicação	-2
O funcionário não possui muita influência na empresa	-2
O funcionário não possui muita ética	-2
O funcionário não possui muita comunicação	-2
O funcionário possui muitíssima capacidade técnica	2
O funcionário possui muitíssima perspicácia	2
O funcionário possui muitíssima educação	2
O funcionário possui muitíssima inovação	2
O funcionário possui muitíssima experiência	2
O funcionário possui muitíssima confiabilidade	2
O funcionário possui muitíssima qualidades	2
O funcionário possui muitíssima visão gerencial	2
O funcionário possui muitíssima liderançavisão estratégica	2
O funcionário possui muitíssima desenvoltura	2
O funcionário possui muitíssima responsabilidade	2
O funcionário possui muitíssima clareza	2
O funcionário possui muitíssima assiduidade	2
O funcionário possui muitíssima criatividade	2
O funcionário possui muitíssima performance	2
O funcionário possui muitíssima inteligência social e emocional	2
O funcionário possui muitíssima facilidade com os problemas encontrados	2
O funcionário possui muitíssima competências comportamentais	2
O funcionário possui muitíssima organização	2
O funcionário possui muitíssima competências técnicas	2
O funcionário possui muitíssima dedicação	2
O funcionário possui muitíssima influência na empresa	2
O funcionário possui muitíssima ética	2
O funcionário possui muitíssima comunicação	2

O funcionário não possui muitíssima capacidade técnica	-2
O funcionário não possui muitíssima perspicácia	-2
O funcionário não possui muitíssima educação	-2
O funcionário não possui muitíssima inovação	-2
O funcionário não possui muitíssima experiência	-2
O funcionário não possui muitíssima confiabilidade	-2
O funcionário não possui muitíssima qualidades	-2
O funcionário não possui muitíssima visão gerencial	-2
O funcionário não possui muitíssima liderançavisão estratégica	-2
O funcionário não possui muitíssima desenvoltura	-2
O funcionário não possui muitíssima responsabilidade	-2
O funcionário não possui muitíssima clareza	-2
O funcionário não possui muitíssima assiduidade	-2
O funcionário não possui muitíssima criatividade	-2
O funcionário não possui muitíssima performance	-2
O funcionário não possui muitíssima inteligência social e emocional	-2
O funcionário não possui muitíssima facilidade com os problemas encontrados	-2
O funcionário não possui muitíssima competências comportamentais	-2
O funcionário não possui muitíssima organização	-2
O funcionário não possui muitíssima competências técnicas	-2
O funcionário não possui muitíssima dedicação	-2
O funcionário não possui muitíssima influência na empresa	-2
O funcionário não possui muitíssima ética	-2
O funcionário não possui muitíssima comunicação	-2
O empregado possui pouquíssimo comprometimento	-2
O empregado possui pouquíssimo esforço	-2
O empregado possui pouquíssimo trabalho colaborativo	-2
O empregado possui pouquíssimo compromisso com o trabalho	-2
O empregado possui pouquíssimo desempenho	-2
O empregado possui pouquíssimo raciocínio lógico	-2
O empregado possui pouquíssimo conhecimento técnico	-2
O empregado possui pouquíssimo planejamento das atividades	-2
O empregado possui pouquíssimo planejamento pessoal	-2
O empregado possui pouquíssimo respeito	-2
O empregado possui pouquíssimo perfil de gestor	-2
O empregado não possui pouquíssimo comprometimento	2

O empregado não possui pouquíssimo esforço	2
O empregado não possui pouquíssimo trabalho colaborativo	2
O empregado não possui pouquíssimo compromisso com o trabalho	2
O empregado não possui pouquíssimo desempenho	2
O empregado não possui pouquíssimo raciocínio lógico	2
O empregado não possui pouquíssimo conhecimento técnico	2
O empregado não possui pouquíssimo planejamento das atividades	2
O empregado não possui pouquíssimo planejamento pessoal	2
O empregado não possui pouquíssimo respeito	2
O empregado não possui pouquíssimo perfil de gestor	2
O empregado possui pouco comprometimento	-2
O empregado possui pouco esforço	-2
O empregado possui pouco trabalho colaborativo	-2
O empregado possui pouco compromisso com o trabalho	-2
O empregado possui pouco desempenho	-2
O empregado possui pouco raciocínio lógico	-2
O empregado possui pouco conhecimento técnico	-2
O empregado possui pouco planejamento das atividades	-2
O empregado possui pouco planejamento pessoal	-2
O empregado possui pouco respeito	-2
O empregado possui pouco perfil de gestor	-2
O empregado não possui pouco comprometimento	2
O empregado não possui pouco esforço	2
O empregado não possui pouco trabalho colaborativo	2
O empregado não possui pouco compromisso com o trabalho	2
O empregado não possui pouco desempenho	2
O empregado não possui pouco raciocínio lógico	2
O empregado não possui pouco conhecimento técnico	2
O empregado não possui pouco planejamento das atividades	2
O empregado não possui pouco planejamento pessoal	2
O empregado não possui pouco respeito	2
O empregado não possui pouco perfil de gestor	2
O empregado possui comprometimento	1
O empregado possui esforço	1
O empregado possui trabalho colaborativo	1
O empregado possui compromisso com o trabalho	1
O empregado possui desempenho	1
O empregado possui raciocínio lógico	1

O empregado possui conhecimento técnico	1
O empregado possui planejamento das atividades	1
O empregado possui planejamento pessoal	1
O empregado possui respeito	1
O empregado possui perfil de gestor	1
O empregado não possui comprometimento	-1
O empregado não possui esforço	-1
O empregado não possui trabalho colaborativo	-1
O empregado não possui compromisso com o trabalho	-1
O empregado não possui desempenho	-1
O empregado não possui raciocínio lógico	-1
O empregado não possui conhecimento técnico	-1
O empregado não possui planejamento das atividades	-1
O empregado não possui planejamento pessoal	-1
O empregado não possui respeito	-1
O empregado não possui perfil de gestor	-1
O empregado possui muito comprometimento	2
O empregado possui muito esforço	2
O empregado possui muito trabalho colaborativo	2
O empregado possui muito compromisso com o trabalho	2
O empregado possui muito desempenho	2
O empregado possui muito raciocínio lógico	2
O empregado possui muito conhecimento técnico	2
O empregado possui muito planejamento das atividades	2
O empregado possui muito planejamento pessoal	2
O empregado possui muito respeito	2
O empregado possui muito perfil de gestor	2
O empregado não possui muito comprometimento	-2
O empregado não possui muito esforço	-2
O empregado não possui muito trabalho colaborativo	-2
O empregado não possui muito compromisso com o trabalho	-2
O empregado não possui muito desempenho	-2
O empregado não possui muito raciocínio lógico	-2
O empregado não possui muito conhecimento técnico	-2
O empregado não possui muito planejamento das atividades	-2
O empregado não possui muito planejamento pessoal	-2
O empregado não possui muito respeito	-2
O empregado não possui muito perfil de gestor	-2

O empregado possui muitíssimo comprometimento	2
O empregado possui muitíssimo esforço	2
O empregado possui muitíssimo trabalho colaborativo	2
O empregado possui muitíssimo compromisso com o trabalho	2
O empregado possui muitíssimo desempenho	2
O empregado possui muitíssimo raciocínio lógico	2
O empregado possui muitíssimo conhecimento técnico	2
O empregado possui muitíssimo planejamento das atividades	2
O empregado possui muitíssimo planejamento pessoal	2
O empregado possui muitíssimo respeito	2
O empregado possui muitíssimo perfil de gestor	2
O empregado não possui muitíssimo comprometimento	-2
O empregado não possui muitíssimo esforço	-2
O empregado não possui muitíssimo trabalho colaborativo	-2
O empregado não possui muitíssimo compromisso com o trabalho	-2
O empregado não possui muitíssimo desempenho	-2
O empregado não possui muitíssimo raciocínio lógico	-2
O empregado não possui muitíssimo conhecimento técnico	-2
O empregado não possui muitíssimo planejamento das atividades	-2
O empregado não possui muitíssimo planejamento pessoal	-2
O empregado não possui muitíssimo respeito	-2
O empregado não possui muitíssimo perfil de gestor	-2
O funcionário é dispensável para [organizacao]	-1
O funcionário não é dispensável para [organizacao]	1
O funcionário é necessário para [organizacao]	1
O funcionário não é necessário para [organizacao]	-1
O funcionário é importante para [organizacao]	1
O funcionário não é importante para [organizacao]	-1
O funcionário é essencial para [organizacao]	2
O funcionário não é essencial para [organizacao]	-2
O funcionário é indispensável para [organizacao]	2
O funcionário não é indispensável para [organizacao]	-2
Um dos destaques do empregado é sua capacidade técnica	1
Um dos destaques do empregado é sua perspicácia	1
Um dos destaques do empregado é sua educação	1
Um dos destaques do empregado é sua inovação	1
Um dos destaques do empregado é sua experiência	1
Um dos destaques do empregado é sua confiabilidade	1

Um dos destaques do empregado é sua qualidades	1
Um dos destaques do empregado é sua visão gerencial	1
Um dos destaques do empregado é sua liderançavisão estratégica	1
Um dos destaques do empregado é sua desenvoltura	1
Um dos destaques do empregado é sua responsabilidade	1
Um dos destaques do empregado é sua clareza	1
Um dos destaques do empregado é sua assiduidade	1
Um dos destaques do empregado é sua criatividade	1
Um dos destaques do empregado é sua performance	1
Um dos destaques do empregado é sua inteligência social e emocional	1
Um dos destaques do empregado é sua facilidade com os problemas encontrados	1
Um dos destaques do empregado é sua competências comportamentais	1
Um dos destaques do empregado é sua organização	1
Um dos destaques do empregado é sua competências técnicas	1
Um dos destaques do empregado é sua dedicação	1
Um dos destaques do empregado é sua influência na empresa	1
Um dos destaques do empregado é sua ética	1
Um dos destaques do empregado é sua comunicação	1
Um dos destaques do empregado não é sua capacidade técnica	-1
Um dos destaques do empregado não é sua perspicácia	-1
Um dos destaques do empregado não é sua educação	-1
Um dos destaques do empregado não é sua inovação	-1
Um dos destaques do empregado não é sua experiência	-1
Um dos destaques do empregado não é sua confiabilidade	-1
Um dos destaques do empregado não é sua qualidades	-1
Um dos destaques do empregado não é sua visão gerencial	-1
Um dos destaques do empregado não é sua liderançavisão estratégica	-1
Um dos destaques do empregado não é sua desenvoltura	-1
Um dos destaques do empregado não é sua responsabilidade	-1
Um dos destaques do empregado não é sua clareza	-1
Um dos destaques do empregado não é sua assiduidade	-1
Um dos destaques do empregado não é sua criatividade	-1
Um dos destaques do empregado não é sua performance	-1

Um dos destaques do empregado não é sua inteligência social e emocional	-1
Um dos destaques do empregado não é sua facilidade com os problemas encontrados	-1
Um dos destaques do empregado não é sua competências comportamentais	-1
Um dos destaques do empregado não é sua organização	-1
Um dos destaques do empregado não é sua competências técnicas	-1
Um dos destaques do empregado não é sua dedicação	-1
Um dos destaques do empregado não é sua influência na empresa	-1
Um dos destaques do empregado não é sua ética	-1
Um dos destaques do empregado não é sua comunicação	-1
Os colegas apontam que a principal qualidade do empregado é seu comprometimento	1
Os colegas apontam que a principal qualidade do empregado é seu esforço	1
Os colegas apontam que a principal qualidade do empregado é seu trabalho colaborativo	1
Os colegas apontam que a principal qualidade do empregado é seu compromisso com o trabalho	1
Os colegas apontam que a principal qualidade do empregado é seu desempenho	1
Os colegas apontam que a principal qualidade do empregado é seu raciocínio lógico	1
Os colegas apontam que a principal qualidade do empregado é seu conhecimento técnico	1
Os colegas apontam que a principal qualidade do empregado é seu planejamento das atividades	1
Os colegas apontam que a principal qualidade do empregado é seu planejamento pessoal	1
Os colegas apontam que a principal qualidade do empregado é seu respeito	1
Os colegas apontam que a principal qualidade do empregado é seu perfil de gestor	1
Os colegas apontam que a principal qualidade do empregado não é seu comprometimento	-1
Os colegas apontam que a principal qualidade do empregado não é seu esforço	-1

Os colegas apontam que a principal qualidade do empregado não é seu trabalho colaborativo	-1
Os colegas apontam que a principal qualidade do empregado não é seu compromisso com o trabalho	-1
Os colegas apontam que a principal qualidade do empregado não é seu desempenho	-1
Os colegas apontam que a principal qualidade do empregado não é seu raciocínio lógico	-1
Os colegas apontam que a principal qualidade do empregado não é seu conhecimento técnico	-1
Os colegas apontam que a principal qualidade do empregado não é seu planejamento das atividades	-1
Os colegas apontam que a principal qualidade do empregado não é seu planejamento pessoal	-1
Os colegas apontam que a principal qualidade do empregado não é seu respeito	-1
Os colegas apontam que a principal qualidade do empregado não é seu perfil de gestor	-1
O desempenho do empregado é péssimo	-2
O desempenho do empregado é ruim	-1
O desempenho do empregado é abaixo do esperado	-1
O desempenho do empregado é baixo	-1
O desempenho do empregado é mediano	0
O desempenho do empregado é aceitável	0
O desempenho do empregado é na média	0
O desempenho do empregado é acima do esperado	1
O desempenho do empregado é bom	1
O desempenho do empregado é alto	1
O desempenho do empregado é ótimo	2
Alcançou níveis péssimos de desempenho e performance	-2
Alcançou níveis ruins de desempenho e performance	-1
Alcançou níveis abaixo do esperado de desempenho e performance	-1
Alcançou níveis medianos de desempenho e performance	0
Alcançou níveis aceitáveis de desempenho e performance	0
Alcançou níveis na média de desempenho e performance	0
Alcançou níveis acima do esperado de desempenho e performance	1
Alcançou níveis bons de desempenho e performance	1
Alcançou níveis altos de desempenho e performance	1

Alcançou níveis ótimos de desempenho e performance	2
É pouquíssimo respeitado pelos colaboradores por compartilhar preocupações, problemas e oportunidades	-2
É pouco respeitado pelos colaboradores por compartilhar preocupações, problemas e oportunidades	-2
É respeitado pelos colaboradores por compartilhar preocupações, problemas e oportunidades	1
É muito respeitado pelos colaboradores por compartilhar preocupações, problemas e oportunidades	2
É muitíssimo respeitado pelos colaboradores por compartilhar preocupações, problemas e oportunidades	2
Incentiva a colaboração com a equipe	1
Não incentiva a colaboração com a equipe	-1
Compartilha ideias e técnicas	1
Não compartilha ideias e técnicas	-1
Constrói bons relacionamentos	1
Não constrói bons relacionamentos	-1
Demonstra espírito harmonioso e cooperativo	1
Não demonstra espírito harmonioso e cooperativo	-1
Gosta de compartilhar sua experiência	1
Não gosta de compartilhar sua experiência	-1
Busca soluções inovadoras	1
Não busca soluções inovadoras	-1
Fomenta a curiosidade por soluções inovadoras	1
Não fomenta a curiosidade por soluções inovadoras	-1
Promove a inovação através [exemplo]	1
Não promove a inovação através [exemplo]	-1
Estabelece relações de trabalho efetivas	1
Não estabelece relações de trabalho efetivas	-1
Busca sinergia	1
Não busca sinergia	-1
Promove a cultura empresarial entre seus colegas de trabalho	1
Não promove a cultura empresarial entre seus colegas de trabalho	-1
Promove a cultura de aprendizado	1
Não promove a cultura de aprendizado	-1
Responde rapidamente a novas instruções, situações, métodos e procedimentos	1

Não responde rapidamente a novas instruções, situações, métodos e procedimentos	-1
Finalizou pouquíssimos projetos	-2
Finalizou poucos projetos	-2
Finalizou projetos	1
Finalizou muitos projetos	2
Finalizou muitíssimos projetos	2
Finalizou projetos com resultados péssimos	-2
Finalizou projetos com resultados ruins	-1
Finalizou projetos com resultados abaixo do esperado	-1
Finalizou projetos com resultados medianos	0
Finalizou projetos com resultados aceitáveis	0
Finalizou projetos com resultados na média	0
Finalizou projetos com resultados acima do esperado	1
Finalizou projetos com resultados bons	1
Finalizou projetos com resultados altos	1
Finalizou projetos com resultados ótimos	2
Cumpre todos os prazos	1
Não cumpre todos os prazos	-1
Respeita o tempo dos outros	1
Não respeita o tempo dos outros	-1
Utiliza o tempo de forma eficaz	1
Não utiliza o tempo de forma eficaz	-1