UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

Witalo Pietler Pimenta de Sousa

RECONHECIMENTO DE ENTIDADES NOMEADAS COMUNS NA COMUNICAÇÃO VIA RÁDIO

Vitória-ES

Dezembro, 2023

Witalo Pietler Pimenta de Sousa

RECONHECIMENTO DE ENTIDADES NOMEADAS COMUNS NA COMUNICAÇÃO VIA RÁDIO

Parte manuscrita do Projeto de Graduação do aluno Witalo Pietler Pimenta de Sousa, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Dezembro, 2023

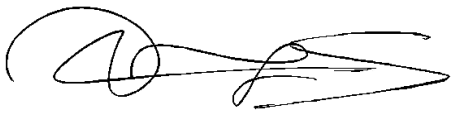
Witalo Pietler Pimenta de Sousa

RECONHECIMENTO DE ENTIDADES NOMEADAS COMUNS NA COMUNICAÇÃO VIA RÁDIO

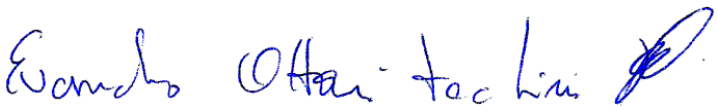
Parte manuscrita do Projeto de Graduação do aluno Witalo Pietler Pimenta de Sousa, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 18 de Dezembro de 2023.

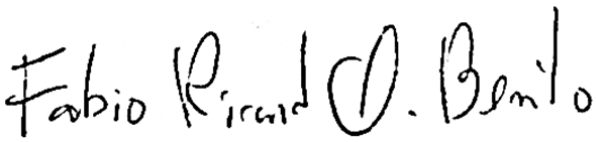
COMISSÃO EXAMINADORA:



Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Orientador



Prof. Dr. Evandro Ottoni Teatini Salles
Universidade Federal do Espírito Santo
Examinador



Prof. Dr. Fabio Ricardo Oliveira Bento
Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo
Examinador

Vitória-ES

Dezembro, 2023

Aos meus pais, pelo apoio incondicional, pelo amor generoso e pelo exemplo honroso que sempre demonstraram.

AGRADECIMENTOS

Quero expressar minha profunda gratidão a todas as pessoas que estiveram presentes ao longo da minha jornada. Em especial, reconheço o apoio incondicional e a dedicação exemplar dos meus pais, Sebastião Candido de Sousa e Acrisiana Pimenta de Sousa, que moldaram minha formação como ser humano. Destaco meu irmão, Iarlen Lucas Pimenta de Sousa, como meu parceiro constante na vida.

Aos meus avós, Ana Maria Ayres Pimenta e Acrisio Vitorino Pimenta, agradeço pelo exemplo de superação, persistência e honestidade.

Aos meus tios Ester Butke Pimenta, Graciana Pimenta Spalenza e Wanderson Brás Pimenta, que sempre me trataram como um filho, expresso minha gratidão por todo o carinho e suporte.

Minha madrinha, Miriana Lucia, tem sido uma fonte constante de inspiração. Por isso, expresso minha profunda gratidão por sua presença significativa em minha vida.

À minha afilhada Érika Pimenta e minha prima Mirela Luiza, agradeço por todo amor e carinho.

Aos amigos que compartilharam esta jornada comigo — Danilo Louzada, Eric Carvalho, Felipe Audibert, Lucas Follador, Luiz Bananal, Pedro Soares, entre outros — agradeço por tornarem essa caminhada mais leve com sua amizade e apoio.

Aos membros do laboratório Visio, especialmente à coordenadora Prof. Dra. Raquel Frizera Vassallo e ao meu orientador Dr. Jorge Leonid Aching Samatelo, agradeço por me guiarem e despertarem meu interesse no mundo da pesquisa acadêmica.

Expresso minha gratidão ao Marcelo de Lima Soares e ao Prof. José Luiz Borba, que foram extremamente solícitos ao contribuir para o trabalho com suas experiências profissionais.

À banca examinadora, agradeço pela aceitação do convite e pelo tempo dedicado à leitura e avaliação deste trabalho.

À Universidade Federal do Espírito Santo, expresso meu agradecimento pela oportunidade de formação acadêmica e pelas incríveis pessoas que conheci ao longo dessa trajetória.

RESUMO

O Reconhecimento de Entidades Nomeadas (*Named Entity Recognition* - NER) representa uma subárea essencial do Processamento de Linguagem Natural, desempenhando um papel crucial na interação entre humanos e computadores. A tarefa de NER é particularmente valiosa ao extrair informações significativas de texto, contribuindo para avanços em diversas áreas, como comunicação, automação de tarefas e análise de dados. Embora o NER já tenha aplicação consolidada no contexto empresarial, há uma lacuna quando se trata de sua utilização em comunicações via rádio, especialmente no contexto específico das ferrovias, objeto de estudo deste trabalho.

A proposta central deste trabalho é realizar o treinamento de um modelo neural dedicado à tarefa de NER no contexto da comunicação via rádio da empresa Vale, especializada em transporte de cargas ferroviárias. O treinamento será conduzido por meio da técnica de *Fine Tuning* utilizando o modelo BERTimbau, empregando conjuntos de dados rotulados manualmente e sintéticos gerados a partir de padrões textuais identificados no conjunto de dados real.

Experimentos foram conduzidos para validar a eficácia do uso de dados sintéticos na melhoria do desempenho do modelo. Ao final, concluiu-se que a incorporação moderada desses dados artificiais foi benéfica para o modelo, resultando em um escore micro F1 de $90,54\%$ na configuração mais eficiente, uma melhora de $2,47\%$ comparado a utilização exclusiva de dados reais. Os resultados alcançados foram considerados satisfatórios, destacando a eficácia da abordagem proposta.

Palavras-chave: Reconhecimento de Entidades Nomeadas; Processamento de Linguagem Natural; Conjunto de dados sintéticos; Comunicação via rádio.

ABSTRACT

Named Entity Recognition (NER) stands as a vital subfield in Natural Language Processing, playing a crucial role in human-computer interaction. The NER task is particularly useful in extracting meaningful information from text, contributing to advancements in various fields such as communication, task automation, and data analysis. While NER has already found consolidated applications in the business context, there is a gap in its use in radio communications, especially in the specific context of railways, the subject of this study.

The central proposal of this work is to train a neural model dedicated to the NER task in the context of radio communication at Vale, a company specializing in railway cargo transport. The training will be conducted using the Fine-Tuning technique with the BERTimbau model, employing manually labeled datasets and synthetic datasets generated from patterns identified in the real dataset.

Experiments were conducted to validate the effectiveness of using synthetic data in improving the model's performance. In conclusion, we found that the moderate incorporation of these artificial data was beneficial for the model, resulting in a micro F1 score of 90.54% in the most efficient configuration, a 2.47% improvement compared to the exclusive use of real data. The achieved results were considered satisfactory, highlighting the effectiveness of the proposed approach.

Keywords: Named Entity Recognition; Natural Language Processing; Synthetic Datasets; Radio Communication

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BERT	*Bidirectional Encoder Representations from Transformers*
DL	*Deep Learning*
IEEE	*Instituto de Engenheiros Eletricistas e Eletrônicos*
IE	*Information Extraction*
ML	*Machine Learning*
NPL	*Natural Language Processing*
NE	*Named Entity*
NER	*Named Entity Recognition*
RNN	*Recurrent Neural Networks*
UFES	Universidade Federal do Espírito Santo

SUMÁRIO

1 INTRODUÇÃO

A tarefa de Reconhecimento de Entidades Nomeadas (*Named Entity Recognition* - NER) surgiu como uma subtarefa do problema de Extração de Informações (*Information Extraction* - IE) e, ao longo do tempo, se tornou uma área essencial no Processamento de Linguagem Natural (*Natural Language Processing* - NLP), com aplicações em uma ampla gama de domínios, como análise de sentimentos, resumo automático, tradução automática, entre outras.

O NER a nível empresarial pode ser de utilidade para a automação de tarefas de arquivamento e busca quando há a necessidade de guardar os históricos de comunicações textuais de maneira estruturada. Nesse contexto, esse projeto será desenvolvido baseado no ambiente de trabalho da Vale S.A, uma empresa de mineração e transporte de minérios por vias férreas que, por ser um ambiente de trabalho sensível a acidentes, tem a necessidade de efetuar auditorias internas das comunicações de radio efetuadas entre os operadores das locomotivas e os controladores das vias. Atualmente, esse processo é realizado manualmente, mas é possível automatizá-lo, parcialmente, com a aplicação de tecnologias de processamento de voz e NLP. Nesse cenário, a Vale S.A. forneceu um conjunto de dados constituído por áudios de comunicação interna realizada via rádio. Esses áudios foram transcritos e transformados em um conjunto de dados textuais. A partir daqui o objetivo do projeto é a construção de um modelo neural orientado para a tarefa NER que permita a detecção automatizada das entidades nomeadas presentes nas transcrições relacionadas à comunicação via rádio efetuada no sistema ferroviário da Vale S.A.

1.1 Objetivos

Objetivo Geral

- O desenvolvimento de um modelo neural orientado à detecção de entidades nomeadas no domínio da comunicação via rádio dentro da empresa Vale S.A.

Objetivos Específicos

- Estudar o problema de NER;
- Realizar a anotação do conjunto de dados fornecido;
- Encontrar modelos para a tarefa NER treinados para a língua portuguesa e que estejam disponíveis publicamente, de forma que seja possível realizar um processo de retreinamento do modelo para a tarefa proposta;
- Criar um conjunto de dados sintéticos baseado em padrões identificados no conjunto de dados reais, visando aprimorar a robustez e o desempenho do modelo.
- Validar e testar o modelo.

1.2 Estrutura do Texto

O presente trabalho está estruturado da seguinte maneira:

- **Introdução**: Este capítulo inicial tem como objetivo contextualizar o problema de NER na comunicação via rádio e apresentar os objetivos deste trabalho;
- **Marco Teórico**: Ao longo deste capítulo, são expostos os principais conceitos utilizados ao longo do trabalho;
- **Proposta**: Neste capítulo, é demonstrado como o problema em estudo será abordado, detalhando o processo de construção dos conjuntos de dados e as características do modelo neural a ser usado;
- **Resultados**: Aqui, são apresentados detalhes sobre os recursos utilizados e os experimentos realizados, assim como os resultados obtidos;
- **Conclusão**: No capítulo final deste trabalho, são discutidas as conclusões e perspectivas futuras a serem abordadas.

2 MARCO TEÓRICO

2.1 Características da Comunicação Via Rádio

Como indicado em (VáZQUEZ et al., 2010), a comunicação via rádio é o processo de transmitir sinais por meio da modulação de ondas eletromagnéticas com frequências abaixo do espectro de luz visível. Essa comunicação é possível em várias frequências e faixas, e cada faixa de frequência possui características distintas que determinam sua adequação para diferentes tipos de comunicação, para isso deve-se levar em consideração a finalidade, a banda de frequência disponível e também os equipamentos necessários, como os transmissores e os receptores, além de outras variáveis.

No contexto empresarial a comunicação via rádio é utilizada por diversos motivos. Entre eles, o alcance abrangente, a capacidade de comunicação em tempo real, a confiabilidade dos sistemas, que podem ser projetados para operar em condições adversas, como por exemplo, durante fortes tempestades ou falhas de energia, e também a confidencialidade, que é assegurada graças à possibilidade de criptografar as informações transmitidas, o que é de importância em setores que possuem troca de informações sensíveis, além disso, os equipamentos necessários são portáteis e relativamente simples de usar, exigindo um treinamento mínimo, além de serem economicamente acessíveis comparados a outras tecnologias.

A comunicação visa ser o mais clara e concisa possível, para esse fim, normalmente é adotado algum protocolo de comunicação de forma que as mensagens sejam padronizadas. Esses protocolos podem variar de acordo com o uso da comunicação em cada instituição, mas algumas características são comuns em várias situações, como por exemplo, o padrão de mensagem pode exigir a identificação do orador emissor e nomear o destino da mensagem, como por exemplo, “Estação X chamando Estação Y”. Outra característica bastante comum é o uso de palavras de controle, que são palavras ou frases específicas como “câmbio”, “roger” e “over” que podem assumir funções específicas dentro da fala, sinalizando por exemplo o início de uma fala ou o final dela, algo especialmente importante quando levamos em conta sistemas mais simples de comunicação que apenas suportam que um usuário do canal se comunique por vez. Ademais, nota-se a preferência pela utilização de siglas e códigos específicos, como códigos de emergência, siglas para posições geográficas, abreviações de termos técnicos, entre outros, visando o aumento na velocidade que a informação é passada.

2.2 Sobre o Conceito de Entidades Nomeadas

O termo "Entidade Nomeada" também conhecido como *Named Entity* (NE) foi usado pela primeira vez na *Sixth Message Understanding Conference* (MUC-6) (GRISHMAN; SUNDHEIM, 1996), na tarefa de identificar nomes de organizações, pessoas e lugares, bem como expressões numéricas, como valores monetários, datas e percentuais. O foco da conferência era em IE, e nesse contexto, a tarefa de reconhecimento e classificação das NE se mostrou uma importante etapa de pré processo, como indicado em Li et al. (2020).

Uma definição para as NEs dada por Petasis et al. (2000) é "Uma NE é um nome próprio usado para nomear algo ou alguém". Como é afirmado por Nadeau e Sekine (2007), a palavra "nomeada" limita as NE a entidades referenciadas por um ou mais designadores-rígidos, conceito utilizado por (KRIPKE, 1985) que define um designador-rígido como um termo que sempre se refere ao mesmo objeto, independentemente do contexto ou das circunstâncias. Dessa forma, designadores-rígidos incluem nomes próprios e termos de tipo natural, como nomes científicos de espécies biológicas e substâncias químicas, um exemplo seria o nome "Luiz Inácio Lula da Silva". Enquanto os designadores-não-rígidos são termos que podem denotar diferentes objetos de acordo com o contexto ou as circunstâncias. Um exemplo de designador-não-rígido é o termo "presidente do Brasil", onde o referente desse termo pode mudar de acordo com quem ocupa o cargo no momento.

Normalmente são categorizadas como NE genéricas e NE específicas. As NE genéricas são entidades amplamente aplicáveis em diferentes contextos e domínios. Essas entidades incluem pessoas, localização, valores, entre outros. As NE específicas do domínio são entidades relacionadas a um campo específico ou a um domínio particular, como por exemplo doenças, medicamentos e procedimentos dentro do contexto médico ou atletas, premiações e campeonatos dentro do contexto esportivo.

2.3 Sobre a Tarefa de NER

A tarefa de NER surgiu na década de 1990 e, desde então, vários métodos foram desenvolvidos com o objetivo de obter melhores resultados. Algumas abordagens, conhecidas como métodos clássicos, utilizavam dicionários pré-definidos de palavras ou expressões associadas a entidades específicas, ou então utilizavam conjuntos de regras criadas manualmente, geralmente baseadas em padrões linguísticos e heurísticas específicas do domínio. Embora funcionais, esses métodos apresentavam várias limitações, como dificuldades de generalização, escalabilidade, atualização e manutenção.

Com o avanço das aplicações baseadas em Aprendizado Profundo (*Deep Learning* - DL), o uso de arquiteturas de redes neurais profundas se tornou o estado da arte nessa área, solucionando vários dos problemas mencionados. Entre os modelos de maior interesse em NLP tem-se:

- **As Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNN)**. Utilizadas principalmente devido à sua capacidade de considerar o contexto em que as palavras estão inseridas em uma sentença. No entanto, apesar do progresso trazido pelas RNNs, ainda existem alguns problemas ao trabalhar com essa arquitetura. Por exemplo, a perda de contexto que ocorre quando uma sentença extensa é fornecida como entrada, fazendo com que o modelo dê mais "atenção" às partes finais da sentença e menos ao início. Além disso, o processo de treinamento de uma RNN é custoso, pois a natureza recorrente da arquitetura limita a capacidade de paralelização do processo de treinamento

- **A arquitetura *Transformers***. Proposta em Vaswani et al. (2017), baseada no uso de mecanismos de atenção[1], mostrou um bom desempenho em tarefas de NLP e até mesmo fora desse campo, alcançando o estado da arte em algumas áreas da Visão Computacional por exemplo. Modelos derivados dessa arquitetura foram capazes de mitigar o problema de perda de contexto, aumentando significativamente a quantidade de sentenças que os modelos podem receber de entrada sem comprometer os resultados de saída, além disso, a arquitetura *Transformer* é paralelizável, o que viabiliza o treinamento com grandes volumes de dados. Por ser uma arquitetura relevante para o trabalho, nos aprofundaremos na Seção 2.4

Apesar dos animadores avanços na área, podemos destacar alguns desafios ainda presentes:

- **Desafio 1 - Idiomas menos difundidos.** A adaptação de um modelo de linguagem para uma tarefa específica tende a ser mais fácil quando o modelo já possui um entendimento do idioma alvo, exigindo menos tempo e dados rotulados no processo. No entanto, isso pode ser problemático ao lidar com línguas menos difundidas, uma vez que a maioria dos conjuntos de dados disponíveis e modelos de linguagem foram desenvolvidos para o inglês, como exemplos, temos o GPT (RADFORD, 2018), BERT (DEVLIN et al., 2019) e GPT2 (RADFORD et al., 2019). Atualmente, já existem modelos de linguagem especializados em outras línguas, como o BETO (CANETE

[1] O mecanismo de atenção é um mecanismo adaptativo que permite a um modelo neural concentrar sua atenção em partes específicas dos dados de entrada, melhorando seu desempenho em tarefas de processamento sequencial.

et al., 2020) para o espanhol, AlBERTo (POLIGNANO et al., 2019) para o italiano, e o BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020) para português brasileiro, todos esses modelos são adaptações do modelo BERT.

- **Desafio 2 - Anotação de dados.** O ajuste de um modelo de linguagem é realizado por meio de aprendizado supervisionado, exigindo, portanto, um conjunto de dados rotulados. No entanto, o processo de anotação desses dados é demorado e muitas vezes requer especialistas na área do domínio, tornando-o custoso.

  Além disso, existem outras dificuldades, como a falta de consistência na anotação, que é frequentemente causada pela ambiguidade da língua. Por exemplo, a NE "São Paulo" pode se referir a um local (cidade ou estado), uma organização ("São Paulo Futebol Clube") ou uma pessoa ("Apóstolo Paulo"), dependendo do contexto. Essa divergência pode estar presente até mesmo em conjuntos de dados populares. Um exemplo destacado por Li et al. (2020) mostra que o termo "*Baltimore*" na frase "*Baltimore defeated the Yankees*" é rotulado como `Local` no MUC-7 e como `Organização` no CoNLL03, o que causa confusão nos limites da entidade. Portanto, um modelo treinado em um conjunto de dados pode não ter um bom desempenho em outro, mesmo que sejam do mesmo domínio.

- **Desafio 3 - Texto informal.** O trabalho de Li et al. (2020) apresenta os resultados obtidos por diferentes modelos em conjuntos de dados variados. Observa-se que melhores métricas são alcançadas em conjuntos de dados com textos formais (artigos jornalísticos, livros, etc.) em comparação com conjuntos de dados gerados por usuários, como *tweets*, comentários em redes sociais e discussões em fóruns públicos. Isso ocorre devido a vários fatores, incluindo a presença de abreviações, uso de gírias, erros ortográficos e palavras malformadas, além das dificuldades relacionadas ao contexto, já que os textos podem pertencer a domínios específicos e frequentemente contêm entidades desconhecidas, ou seja, NE que não estavam presentes no conjunto de dados de treinamento.

2.4 Arquitetura *Transformers*

Como visto em (VASWANI et al., 2017), o modelo original dos *Transformers* é baseado na arquitetura *encoder-decoder*, amplamente utilizado em tarefas do tipo *sentence-to-sentence*, onde uma sentença é recebida como entrada e a saída também é uma sentença. Essa arquitetura consiste em dois componentes:

- ***Encoder***. Os *encoders* convertem a sentença de entrada em uma sequência de *embeddings*, também chamados de *hidden states*.
- ***Decoder***. Os *decoders* utilizam os *embeddings* gerados pelos *encoders* para produzir uma sequência de saída.

A Figura 1 apresenta um diagrama da arquitetura do modelo *encoder-decoder*, a imagem foi retirada de (VASWANI et al., 2017) que pode ser consultado para mais detalhes.

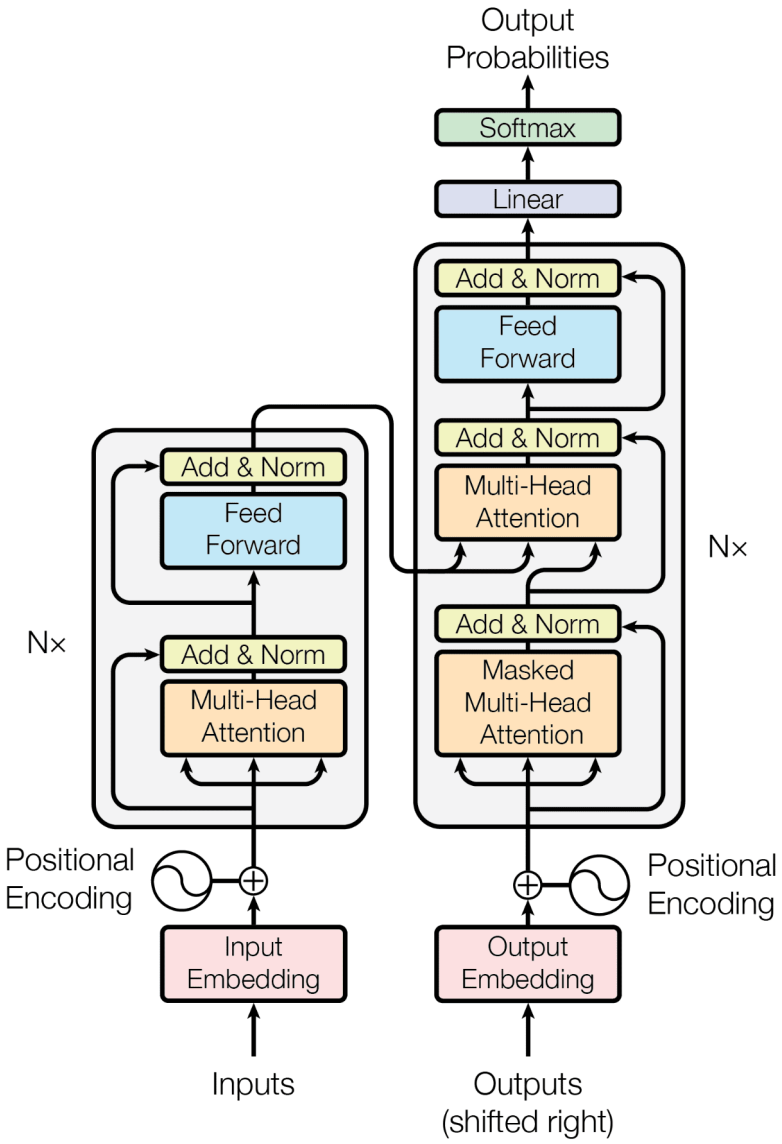


Figura 1 – Diagrama de uma arquitetura encoder-decoder.

Apesar dessa arquitetura ter sido desenvolvida originalmente para que os blocos *encoder* e *decoder* funcionassem de maneira conjunta, ao longo do tempo, tanto o componente *encoder* quanto o *decoder* foram adaptados como modelos independentes. Um desses modelos é o *Bidirectional Encoder Representations from Transformers* - BERT.

2.5 Arquitetura *Modelo BERT*

A arquitetura do modelo BERT foi proposta por Devlin et al. (2019)). Sua estrutura é do tipo E*ncoder-only*. Esses modelos especializam-se em converter sentenças de texto em uma representação numérica rica que, posteriormente, pode ser utilizada em tarefas de classificação, como o Reconhecimento de Entidades Nomeadas (NER). Na Figura 2, apresentamos um diagrama do modelo BERT, incluindo o processo de tokenização, o qual é essencial para o funcionamento do modelo e será discutido de forma mais detalhada na subseção 2.5.1.

O modelo BERT é composto por duas etapas fundamentais: a Camada de Embedding, abordada na subseção 2.5.2, e a Pilha de Encoders, discutida em 2.5.3. A seguir, abordaremos o treinamento do modelo na subseção 2.5.4 para uma compreensão abrangente do funcionamento do BERT.

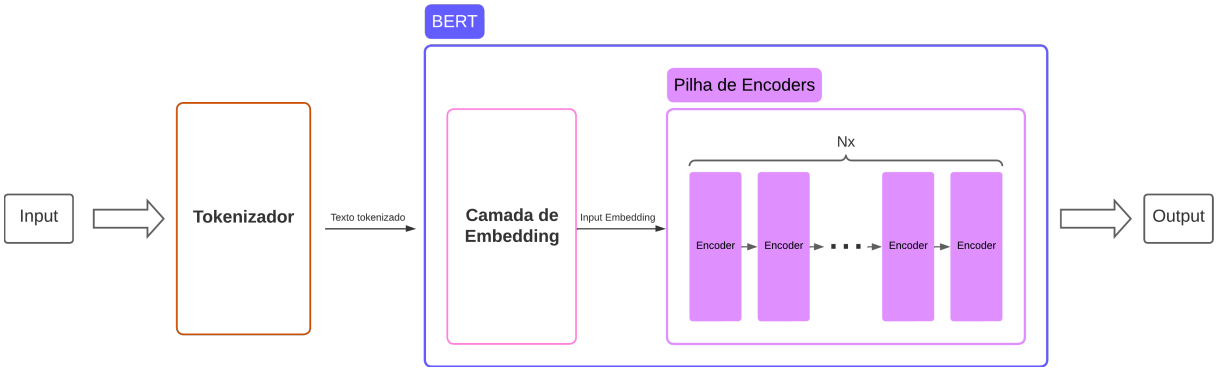


Figura 2 – Diagrama da arquitetura BERT.

2.5.1 Tokenização

Inicialmente, ocorre o processo de tokenização da sentença de entrada, detalhado no Anexo 6. Esse processo implica na divisão da sentença em unidades menores chamadas *tokens*, aos quais são adicionados marcadores especiais, a saber:

- **`[CLS]`:** *token* adicionado no início de cada sentença de entrada durante o treinamento do BERT;
- **`[SEP]`:** *token* usado para separar duas sentenças em uma única entrada;
- **`[MASK]`:** *token* usado durante o pré-treinamento do BERT. Ele substitui aleatoriamente algumas palavras na sentença de entrada para que o modelo aprenda a prever as palavras ocultas. Durante a inferência, esse *token* não é usado;

- **`[UNK]`:** *token* que representa palavras desconhecidas ou fora do vocabulário. É usado quando o modelo encontra palavras que não estão presentes no vocabulário durante a inferência;
- **`[PAD]`:** *token* usado para preencher as sentenças com *tokens* extras, a fim de garantir que todas as sentenças tenham o mesmo tamanho. Isso é necessário porque o BERT processa lotes de sentenças de entrada com dimensões fixas.

Após esse processo, cada *token* é mapeado para um código de identificação em um vocabulário, incluindo os *tokens* especiais. O método de tokenização utilizado no treinamento do modelo BERT foi a tokenização em subpalavras, e o tamanho do vocabulário utilizado originalmente é aproximadamente 30.000. A Figura 3 ilustra o processo de tokenização e mapeamento de cada *token* par aum código do vocabulário.

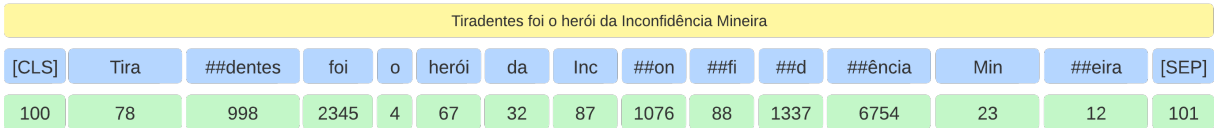


Figura 3 – Diagrama do processo de tokeniação de uma frase.

2.5.2 Camada de Embedding

Na arquitetura do BERT, o *embedding* de entrada representa a codificação inicial dos *tokens* de entrada. O BERT adota uma abordagem bidirecional para processar sequências de *tokens*, considerando o contexto tanto à esquerda quanto à direita de cada palavra ao calcular suas representações. Cada *token* de entrada corresponde a um vetor de saída com comprimento d_k. Os *embeddings* de entrada no BERT são formados pela combinação de três tipos principais de *embeddings*:

- **Token Embedding:** Cada token é inicialmente representado por um vetor. Esses vetores são aprendidos durante o treinamento do modelo.
- **Segment Embeddings:** Para possibilitar que o modelo compreenda a relação entre diferentes partes de uma entrada, como em pares de sentenças, é adicionado um *token* especial de segmento ("[SEP]") entre as sentenças. Cada *token* de entrada é atribuído a um segmento (0 ou 1), e um vetor de segmento é associado a cada *token*. Embora o uso de Segment Embeddings não seja essencial para NER, em que todos os segmentos podem ser atribuídos como 0, essa técnica é particularmente útil em tarefas que envolvem múltiplos segmentos de texto, como no contexto da tarefa de de *Question Answering* (Q&A).

- **Positional Embeddings:** O BERT não leva em consideração a ordem das palavras na entrada. Para superar essa limitação, são adicionados *embeddings* posicionais que codificam a posição relativa de cada *token* na sequência. Esse componente desempenha um papel crucial, permitindo que a arquitetura BERT processe palavras de uma sequência de forma paralela. Em comparação com RNNs que mantêm o contexto de ordem das palavras por meio da recorrência temporal, o *positional embedding* possibilita um treinamento mais rápido e eficiente em hardware moderno, como GPUs.

A soma desses três tipos de *embeddings* forma a representação inicial de um *token* na entrada do modelo BERT. Durante o treinamento, esses *embeddings* são ajustados para capturar relações semânticas e contextuais na linguagem, contribuindo para a eficácia do BERT em diversas tarefas, como compreensão de texto, tradução automática e resposta a perguntas. Um diagrama detalhando esse processo para um *token* de entrada está explicitado na figura 4, enquanto a figura 5 ilustra a saída para uma frase de entrada, onde a entrada com N_t *tokens* resulta em uma matriz $Nt \times d_k$.

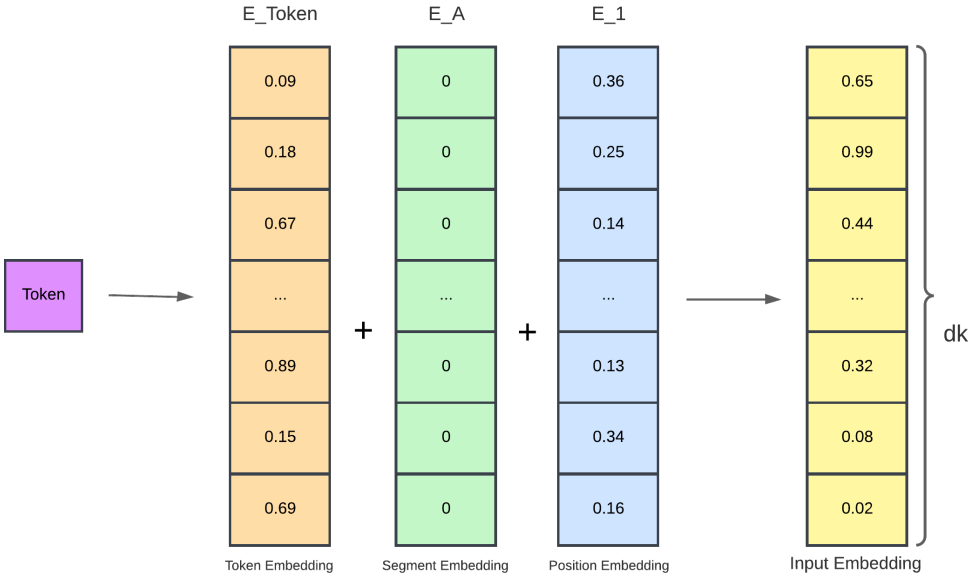


Figura 4 – Diagrama do processo de construção do *embedding* de entrada a partir de um *token*.

2.5.3 Pilha de Encoders

Uma pilha de *encoders* consiste em empilhar vários blocos *encoder* uns sobre os outros. Cada camada da pilha recebe as representações de saída da camada anterior. A ideia por trás dessa pilha é permitir que o modelo aprenda representações hierárquicas e complexas da entrada, capturando informações em diferentes níveis de abstração. A Figura 6 detalha

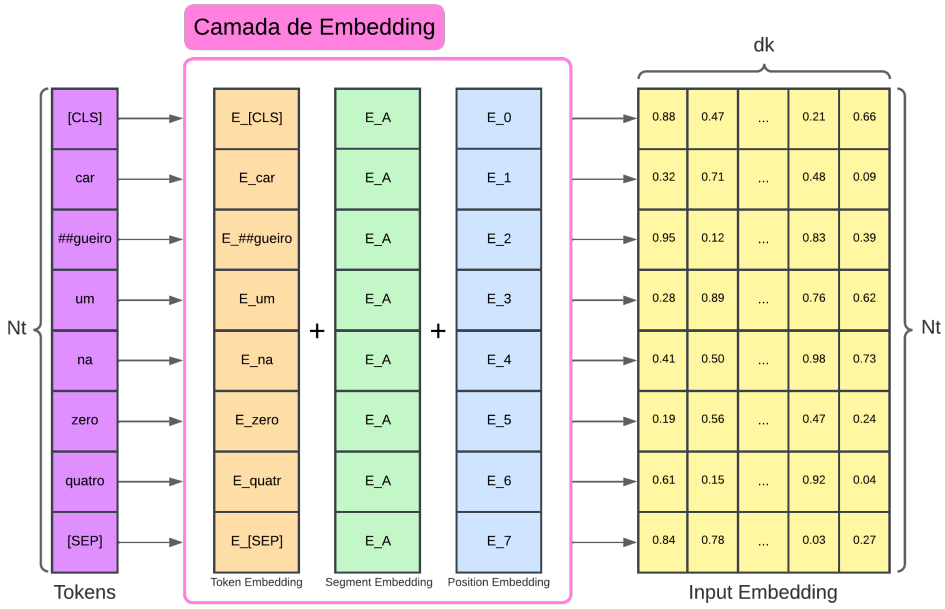


Figura 5 – Diagrama do processo de construção do *embedding* de entrada a partir de uma sequência de *tokens*.

a arquitetura encontrada em cada bloco de *encoder*, composta por três partes: *Multi-Head Attention*, *Feed Forward* e *Add and Norm*, detalhados a seguir.

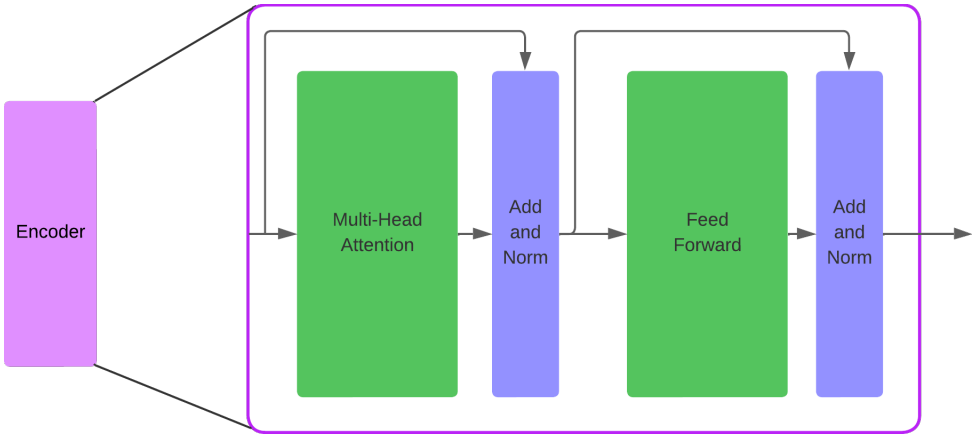


Figura 6 – Representação das subcamadas de um bloco *encoder*

- **Multi-Head Attention**

  A camada de *Multi-Head Attention* (Atenção Multi-Cabeça) é um componente central nas arquiteturas de *transformers*, como o BERT. Essa camada permite que o modelo dê atenção a diferentes partes da entrada em paralelo, através de várias "cabeças" de atenção. Vamos explorar detalhadamente como funciona a camada *Multi-Head Attention*:

  1. **Entrada Q (Consulta), K (Chave), V (Valor):** Cada *token* na sequência de entrada é representada como um vetor em três projeções lineares: consulta

(Q), chave (K) e valor (V), cada um com as dimensões $N_t \times d_k$. Estas projeções são aprendidas durante o treinamento do modelo.

2. **Divisão em cabeças:** A divisão em cabeças no contexto de projeções (Q, K, V) envolve a separação simultânea dessas projeções em múltiplas "cabeças"ou conjuntos. Para um número h de cabeças, teremos conjuntos individuais representados por $(Q_1, K_1, V_1), ..., (Q_n, K_n, V_n)$, onde cada projeção possui as dimensões $N_t \times \frac{d_k}{h}$. Essa abordagem de múltiplas cabeças possibilita que o modelo aprenda diferentes representações da entrada de forma paralela.

   Essa estratégia é fundamental para melhorar a capacidade do modelo de capturar nuances e complexidades nos dados. Ao dividir as projeções em cabeças distintas, o modelo pode processar informações de maneira mais eficiente e aprender representações mais ricas e especializadas. Essa abordagem paralela contribui para a eficácia do modelo ao lidar com tarefas complexas de processamento de linguagem natural e otimizar a extração de padrões nos dados, como é dito em (VASWANI et al., 2017).

3. **Atenção Escalonada:** Dentro de cada cabeça, o cálculo da atenção é realizado utilizando a seguinte equação:

$$\text{Attention} = \text{Softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{2.1}$$

   O resultado é uma matriz de atenção para cada cabeça, refletindo a importância relativa de cada *token* em relação aos outros. Essa matriz de atenção destaca quais *tokens* na sequência são mais relevantes para a compreensão do contexto, proporcionando uma abordagem eficaz para a captura de relações semânticas e a extração de informações importantes. A operação Softmax normaliza os valores, atribuindo pesos adequados a cada *token* com base na sua relevância no contexto da atenção. Essa abordagem escalonada, realizada em paralelo em múltiplas cabeças, é fundamental para melhorar a capacidade do modelo de processar informações de maneira abrangente e eficiente.

4. **Saída:** As saídas de todas as cabeças de atenção são concatenadas e multiplicadas por uma matriz de pesos aprendidos (projeção linear). Isso cria uma representação combinada que captura informações de todas as cabeças.

A Figura 7 ilustra um diagrama que apresenta o funcionamento dos processos envolvidos na camada de atenção multi-cabeça. Este diagrama fornece uma representação visual dos diferentes passos e interações que ocorrem dentro dessa camada, destacando como as projeções (Q, K, V) são divididas em múltiplas cabeças e como a atenção é calculada de forma paralela para capturar informações contextuais em diversas perspectivas.

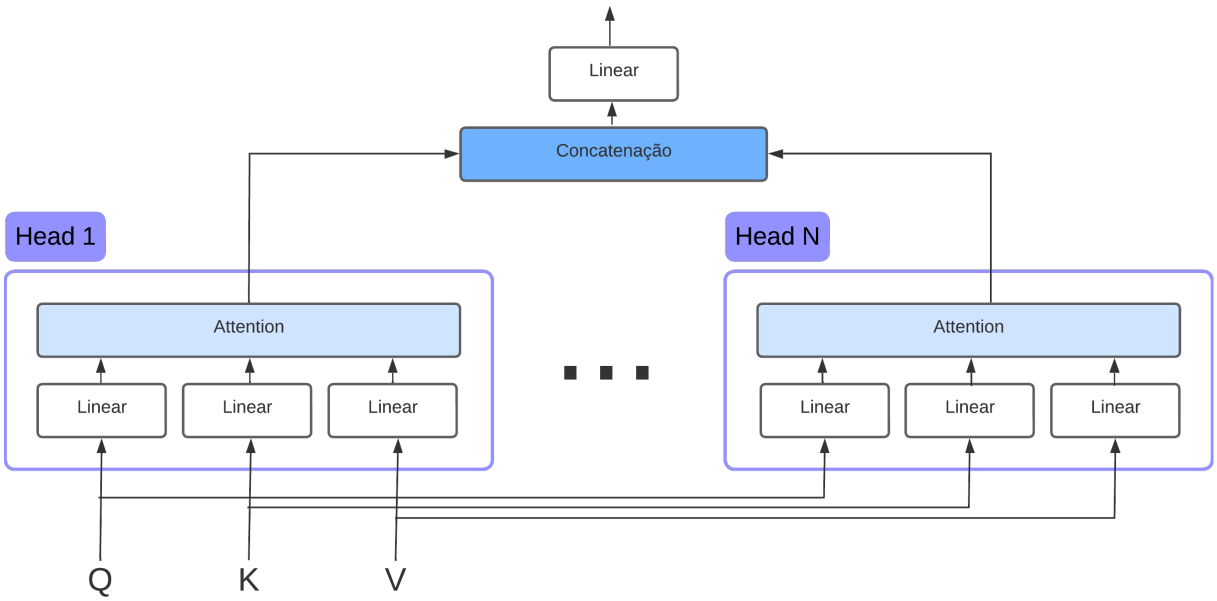


Figura 7 – Diagrama do funcionamento dos processos envolvidos na camada *multi-head attention*

- **Feed Forward**

  A camada *Feed forward* consiste em uma ou mais camadas totalmente conectadas (também conhecidas como camadas lineares) com funções de ativação não lineares entre elas. Cada posição na sequência é processada independentemente, ou seja, não há interdependência entre as diferentes posições da sequência durante essa etapa.

- **Norm and Add**

  A camada de "Add & Norm" (Adição e Normalização) em um modelo *Transformer*, como o BERT, é uma combinação de duas operações principais:

  1. **Adição (Add):** A saída da camada anterior é somada à entrada original. Essa adição é uma forma de conexão residual, preservando a informação original e facilitando o fluxo do gradiente durante o treinamento. Essa técnica ajuda a mitigar o problema de desaparecimento de gradientes em redes profundas.
  2. **Normalização (Normalization):** Após a adição, é aplicada uma operação de normalização, geralmente usando *Layer Normalization*. Essa normalização ajusta as ativações para ter média zero e desvio padrão unitário, o que auxilia no treinamento estável e na generalização do modelo.

  Portanto, a função da camada "Add & Norm" é melhorar a estabilidade do treinamento, facilitando o fluxo de informações residuais e normalizando as ativações para evitar problemas como a explosão ou desaparecimento de gradientes. Essa camada é comumente encontrada após cada subcamada em uma pilha de *encoders* ou *decoders* em arquiteturas *Transformer*.

Em suma, a função principal dessa pilha de camadas *encoders* é "aperfeiçoar" o *embedding* de entrada para produzir uma representação numérica que carregue o sentido contextual da sentença. Um exemplo seria a palavra "rio", que ao passar pela pilha de *encoders*, terá uma representação mais relacionada à "cidade" e menos relacionada à "corrente de água", se os termos "Ipanema" ou "Cristo Redentor" estiverem próximos a ela.

Na Figura 8, temos um diagrama diagrama que mostra o peso dado ao relacionamento do token *cross_* com os outros tokens da mesma sentença, a imagem está considerando a saída de um sistema com duas *heads*, representadas pelas cores laranja e rosa, quanto mais grossa a linha, e mais forte cor, maior é o peso dada para a relação entre os dois tokens. Nota-se que os resultados obtidos por cada cabeça são ligeiramente diferentes.

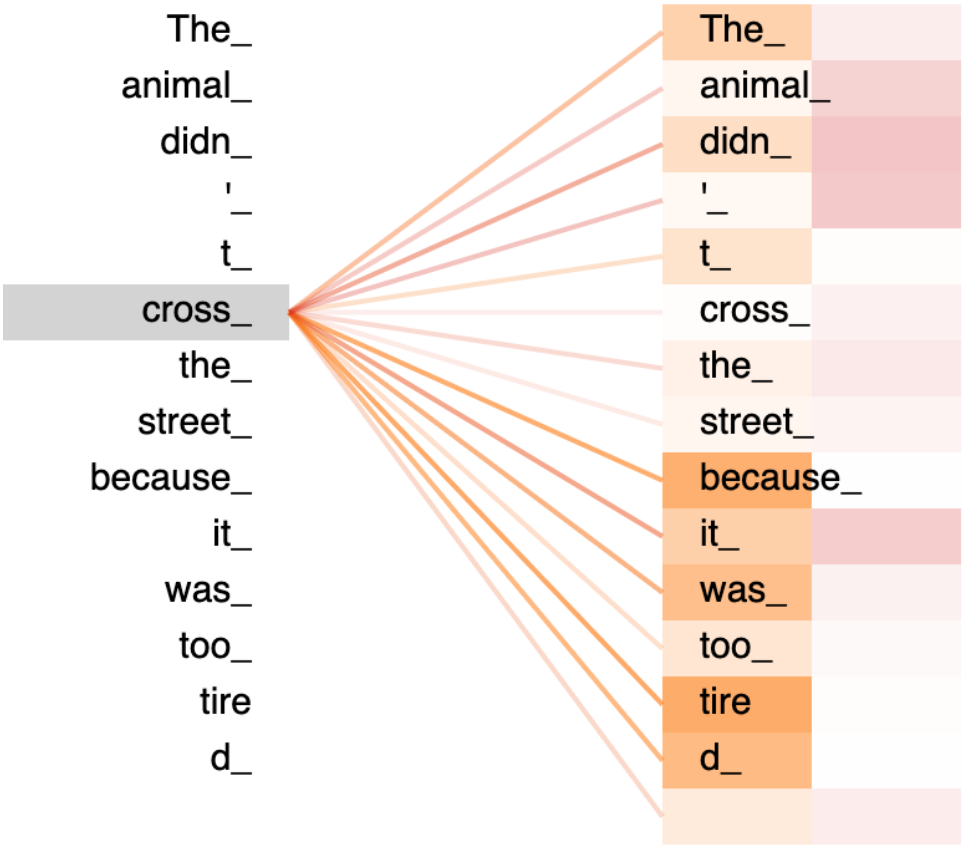


Figura 8 – Diagrama mostrando o peso das relações entre o *token cross_* e os outros tokens dentro da sentença, considerando os resultados de duas *heads* do *encoder*. Imagem retirada de (PREOTEASA, 2020)

A Figura 9, retirada de (TUNSTALL, 2022), demonstra como, após passar pela pilha de *encoders*, a mesma palavra "*flies*" passa a ter diferentes *embeddings*, pois de acordo com o contexto, nota-se que o significado nas duas situações é diferente.

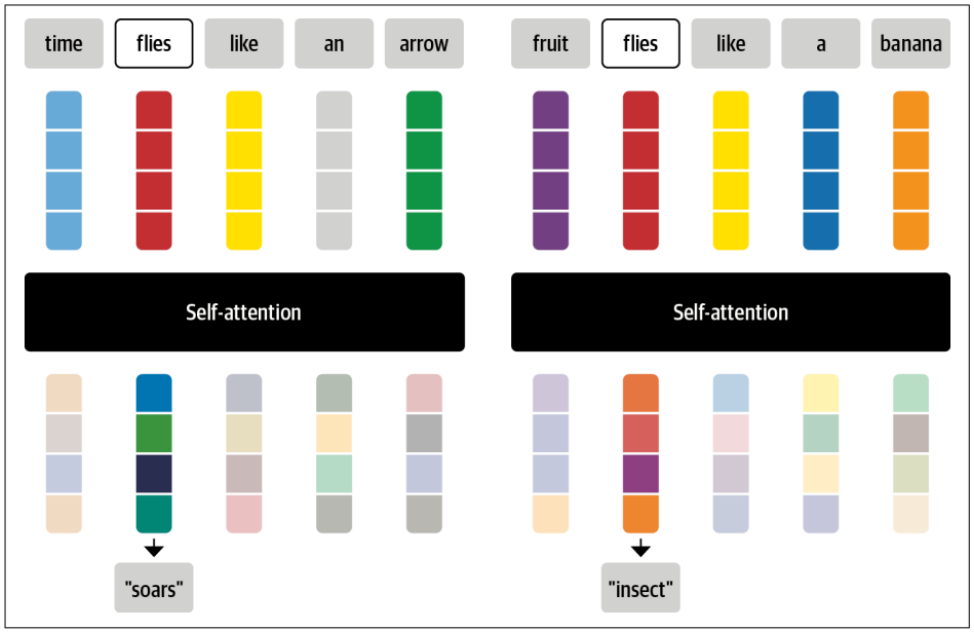


Figura 9 – Diagrama demonstrando como o *embedding* é aprimorado ao longo da pilha de *encoders* para criar uma representação numérica que carregue informações do contexto.

2.5.4 Treinamento do modelo BERT

O treinamento do modelo BERT, detalhado em (DEVLIN et al., 2019) e apresentado de maneira didática em (RASCHKA, 2022), ocorre em dois estágios: pré-treinamento e *fine-tuning*, de forma similar a outros modelos de linguagem.

O pré-treinamento do BERT ainda dividido em duas etapas: *Masked Language Model (MLM)* e *next-sentence prediction*, especificamente:

- ***Masked Language Mode***. O modelo é alimentado com uma grande quantidade de dados não rotulados, onde uma porcentagem desses *tokens* é mascarada (substituída pelo *token* especial `[MASK]`). A rede tem como tarefa prever essas palavras mascaradas, considerando o contexto fornecido pelas palavras vizinhas.
- ***Next-sentence prediction***. O modelo é treinado para prever se duas frases são adjacentes em um corpus. Pares de frases são criados a partir do conjunto de dados, onde metade são frases adjacentes (positivas), enquanto a outra metade consiste em frases aleatórias que não são adjacentes (negativas). O modelo recebe como entrada a concatenação das duas frases e tenta prever se elas são adjacentes ou não.

Ao ser pré-treinado com essas duas etapas, o BERT aprende representações contextuais profundas para palavras e frases, capturando relações semânticas e estruturais. Após o

pré-treinamento, o modelo pode ser ajustado, via um processo de *fine-tunning*, para realizar tarefas específicas de NLP, como o NER, através de um treinamento supervisionado.

2.6 Modelo BERTimbau

Para este trabalho, o modelo mais relevante é o BERTimbau, uma adaptação do modelo BERT para o português brasileiro, desenvolvido por Souza, Nogueira e Lotufo (2020). O BERTimbau passou por um processo de pré-treinamento utilizando o conjunto de dados BrWaC - Brazilian Portuguese Web as Corpus, proposto por Wagner et al. (2018). Em um estudo comparativo conduzido por Souza, o BERTimbau demonstrou resultados superiores ao realizar ajuste fino para a tarefa de NER em textos em português, quando comparado ao modelo BERT Multilingual, sob as mesmas condições de treinamento.

Além disso, conforme mencionado em (SOUZA; NOGUEIRA; LOTUFO, 2020), um vocabulário em português contendo 30.000 subpalavras foi gerado utilizando SentencePiece e o algoritmo *Byte-Pair Encoding* (BPE). Esse vocabulário foi convertido para ser compatível com a arquitetura BERT, incluindo os *tokens* especiais ([CLS], [MASK], [SEP] e [UNK]).

No repositório oficial do modelo BERTimbau (NEURALMIND, 2023), estão disponíveis duas versões diferentes: *base* e *large*. As principais diferenças entre os dois modelos residem no número de *encoders* (N_x) na pilha de *encoders*, no tamanho do vetor de *embedding* (d_k), no número de *heads* em cada encoder e no número de parâmetros. A Tabela 1 exibe os valores dos atributos listados para cada modelo:

Atributo	BERTimbau *Base*	BERTimbau *Large*
Número de *Encoders* (N_x)	12	24
Tamanho do Vetor de *Embedding* (d_k)	768	1024
Número de *heads*	12	16
Número de Parâmetros (milhões)	110	335

Tabela 1 – Comparação entre os Modelos *Base* e *Large* do Bertimbau

2.7 Avaliação de Desempenho dos modelos NER

A análise de desempenho é uma etapa importante na avaliação de modelos neurais, sendo essencial tanto para avaliar e metrificar melhorias do próprio modelo quanto para comparar os resultados obtidos com outros modelos disponíveis, portanto, é importante que as métricas escolhidas não só façam sentido dentro da tarefa como também sejam bem adotadas pela comunidade, facilitando a comparação de resultados.

Dentro desse aspecto, para modelos de classificação que envolvem múltiplas classes, como a maioria dos trabalhos orientados à tarefa de NER, são de utilidade as métricas de *Precision*, *Recall* e *F1-score*, que são calculadas com base nos valores de verdadeiro positivo (VP), falso negativo (FN) e falso positivo (FP) calculados para cada classe, especificamente:

- ***Precision***. A precisão mede a exatidão das previsões positivas. Calculando a proporção de exemplos classificados corretamente como positivos em relação ao total de exemplos classificados como positivos.

$$Precision = \frac{VP}{VP + FP} \tag{2.2}$$

- ***Recall***. Também conhecido como taxa de verdadeiros positivos, mede a proporção de exemplos positivos corretamente classificados em relação ao total de exemplos rotulados como positivos.

$$Recall = \frac{VP}{VP + FN} \tag{2.3}$$

- **F1-*score***. O *score* F1 é uma medida de desempenho que combina as métricas de *precision* e *recall* em um único valor. Ela é calculada como a média harmônica entre a *precision* e o *recall*.

$$\text{F1-}score = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

Uma peculiaridade da avaliação de um modelo NER é que, para que uma predição seja considerada correta, todas as palavras de uma NE devem ser previstas corretamente. Por exemplo, se apenas a palavra “Janeiro” da NE “Rio de Janeiro” for classificada como a NE `local`, a predição é considerada incorreta.

É comum também realizar a diferenciação entre as métricas macro e micro de *recall*, *precision* e *F1-score*, que se diferenciam na forma como elas tratam os exemplos e classes de NE.

- **Uma métrica macro** calcula a média das métricas individuais de cada classe, atribuindo o mesmo peso a todas as classes. Isso é útil para avaliar o desempenho global do modelo, sem levar em consideração o desequilíbrio na distribuição de classes.

- **Uma métrica micro** calcula as métricas globalmente, considerando a contribuição de todas as amostras de forma igualitária. Isso é especialmente útil quando há um desequilíbrio significativo na distribuição de classes, pois cada amostra tem o mesmo peso na métrica final.
- **Uma métrica ponderada** calcula as métricas individuais de cada classe, e em seguida, pondera essas métricas com base na proporção de ocorrências de cada classe no conjunto de dados de teste. Isso tende a dar mais importância às classes mais representadas.

Para simplificar o processo de avaliação, é comum utilizar o *framework* `seqeval` (NAKAYAMA, 2018), que, por padrão, fornece as métricas mencionadas anteriormente como saída da função `classification_report`. Isso torna a avaliação mais fácil e conveniente. A Figura 10 exibe um exemplo de funcionamento do código em um conjunto de dados de teste fictício, e exibe as métricas calculadas, o exemplo foi adaptado de (TUNSTALL, 2022).

```
from seqeval.metrics import classification_report

y_label = [['B-PER', 'I-PER', 'O', 'O', 'B-ORG', 'I-ORG', 'I-ORG', 'O', 'B-PER', 'I-PER']]
y_pred =  [['B-PER', 'I-PER', 'O', 'O', 'B-ORG', 'I-ORG', 'I-ORG', 'O', 'O'    ,'O'     ]]
print(classification_report(y_label, y_pred))
```

```
              precision    recall  f1-score   support

         ORG       1.00      1.00      1.00         1
         PER       1.00      0.50      0.67         2

   micro avg       1.00      0.67      0.80         3
   macro avg       1.00      0.75      0.83         3
weighted avg       1.00      0.67      0.78         3
```

Figura 10 – Exemplo do cálculo de métricas em um conjunto de teste fictício.

Além das métricas citadas, uma ferramenta interessante para avaliação é a matriz de confusão, que exibe de forma intuitiva os resultados para cada classe. Ela ajuda a identificar facilmente confusões entre classes semelhantes e a investigar os limites entre as classes.

Em Li et al. (2020) ainda é citado outro tipo de avaliação, conhecido como avaliação de correspondência relaxada. Esse método considera correspondências parciais ou aproximadas entre as NE identificadas pelo modelo e as NE de referência (rótulos). Isso permite avaliar a capacidade geral do modelo em identificar NE, mesmo que haja variação na correspondência. No entanto, os autores ressaltam que a variação na forma de avaliação torna essa métrica menos intuitiva e dificulta a comparação de resultados com outros trabalhos. Como

resultado, ela não é amplamente utilizada. Além disso, outras formas de avaliação mais complexas foram propostas, mas são limitadas a trabalhos específicos.

3 PROPOSTA

Neste capítulo, apresenta-se a proposta desenvolvida para abordar o problema da classificação de NEs nos textos transcritos da comunicação interna por rádio da empresa Vale S.A. Tal proposta utiliza o modelo pré-treinado Bertimbau e um classificador neural, ambos treinados com conjuntos de dados reais rotulados manualmente, assim como um conjunto de dados sintéticos.

O capítulo está estruturado em duas seções. Na Seção 3.1, detalha-se a construção dos conjuntos de dados, e na Seção 3.2, descreve-se a arquitetura do modelo de classificação desenvolvido.

3.1 Construção dos conjuntos de dados

Os dados utilizados no treinamento originam-se de duas fontes distintas: dados reais provenientes da empresa Vale S.A e dados sintéticos gerados para replicar padrões identificados no conjunto de dados reais. Uma vantagem significativa dos dados sintéticos é a facilidade com que podem ser ampliados.

O procedimento de treinamento do modelo é conduzido de maneira supervisionada, o que implica que os dados precisam ser rotulados previamente. Nesse contexto, as NEs de interesse foram categorizadas em duas classes:

- `TREM`. A definição de "trem" utilizada pela empresa VALE em sua comunicação interna transcende o conceito convencional, definido como:

  > Conjunto de vagões ou carruagens engatadas umas nas outras e puxadas por uma locomotiva (PRIBERAM, 2023).

  Para além dessa definição padrão, a empresa considera como "trem" não apenas a formação clássica, mas também locomotivas sem vagões, equipamentos de manutenção da via e até mesmo outros tipos de veículos que, embora não sejam locomotivas, são capazes de se movimentar pelos trilhos. Em resumo, considera-se como "trem":

  > Qualquer veículo ou conjunto de veículos que se locomova sobre trilhos.

Durante o processo de rotulação, essa definição será adotada como parâmetro.

Todos os trens recebem um identificador único que contém informações relevantes, composto por duas partes: o Prefixo e o Identificador Numérico.

O prefixo representado por uma letra, geralmente representa o tipo de carga ou função associada a esse trem. Alguns exemplos comuns incluem:

- `M`: Minério
- `C`: Cargueiro
- `B`: Turmas de Manutenção Via Permanente
- `P`: Passageiro
- `L`: Locomotivas Escoteiras

Com o objetivo de mitigar o risco de perda de informações devido a ruídos comuns ou interrupções na comunicação via rádio, em alguns casos, opta-se por falar o prefixo por completo em vez de apenas utilizar a letra correspondente. Por exemplo, utiliza-se "Passageiro" e "Cargueiro" em vez de simplesmente `P` e `C`, respectivamente. Pelo mesmo motivo, ocasionalmente, é empregado um alfabeto fonético, que consiste em palavras associadas a cada letra do alfabeto. Exemplos dessa prática incluem o uso da palavra "Viriato" para se referir à letra `V` e "Laurindo" para a letra `L`.

O identificador numérico é composto por uma sequência de três dígitos, abrangendo o intervalo de 0 a 9. Cada dígito dessa sequência contém informações específicas sobre o trem. Essas informações podem englobar diversos aspectos, como o sentido da viagem, tipo de carga transportada, dia da semana, condição do trem, origem, destino, entre outros detalhes relevantes.

A seguir, apresentam-se algumas frases nas quais as NEs foram rotuladas como `TREM`:

- p zero um TREM na rh zero oito LOCAÇÃO cco cambio [...]
- cco passageiro um TREM na zero um LOCAÇÃO cambio
- cco viriato cento e dezoito TREM na sete LOCAÇÃO cambio

Alguns exemplos de NEs que não foram nomeadas e o motivo:

- *bom dia bom trabalho aqui e o cargueiro rancando de camara pra livrar a cinco sete* LOCAÇÃO *brigado cambio*

  No caso do trem com o prefixo "cargueiro", ele é mencionado, mas o identificador numérico não é especificado, tornando a identificação abrangente e dependente do contexto de data, hora e local para a compreensão adequada. Por esses motivos os trens identificados apenas com o prefixo não são rotulados. No

entanto, os trens referenciados apenas com o identificador numérico ainda serão rotulados, uma vez que essa forma de comunicação é comum, estando presente em cerca de 30% das amostras.

– *[...] o v trocando abaixo da tres quatro* LOCAÇÃO *cargueiro por enquanto foi com composicao de ibc mesmo [...]*

Da mesma forma, ocorre com os trens referenciados pelo prefixo `V` e pela palavra "cargueiro".

- **`LOCAÇÃO`**

As Locações, conhecidas como "*RH's*" ou "*Houses*", representam edificações estrategicamente localizadas ao longo da via, próximas aos circuitos de chaves, onde os equipamentos de sinalização e comunicação estão instalados. Essas construções são numeradas de maneira sequencial e servem como referência de localização ao longo da ferrovia. Embora essa numeração sequencial não seja uma regra geral em toda a extensão da ferrovia, ela é amplamente adotada em grandes trechos, tornando-se uma referência comum. A Figura 11 mostra a RH de número 80.

Figura 11 – Foto da RH 80 (VALE, 2013)

A Figura 12, apresentada publicamente durante o IV Encontro de Ferrovias da Associação Nacional dos Transportadores Ferroviários em 2013 (VALE, 2013), é um mapa que destaca pontos importantes da ferrovia, indicando o número de suas locações.

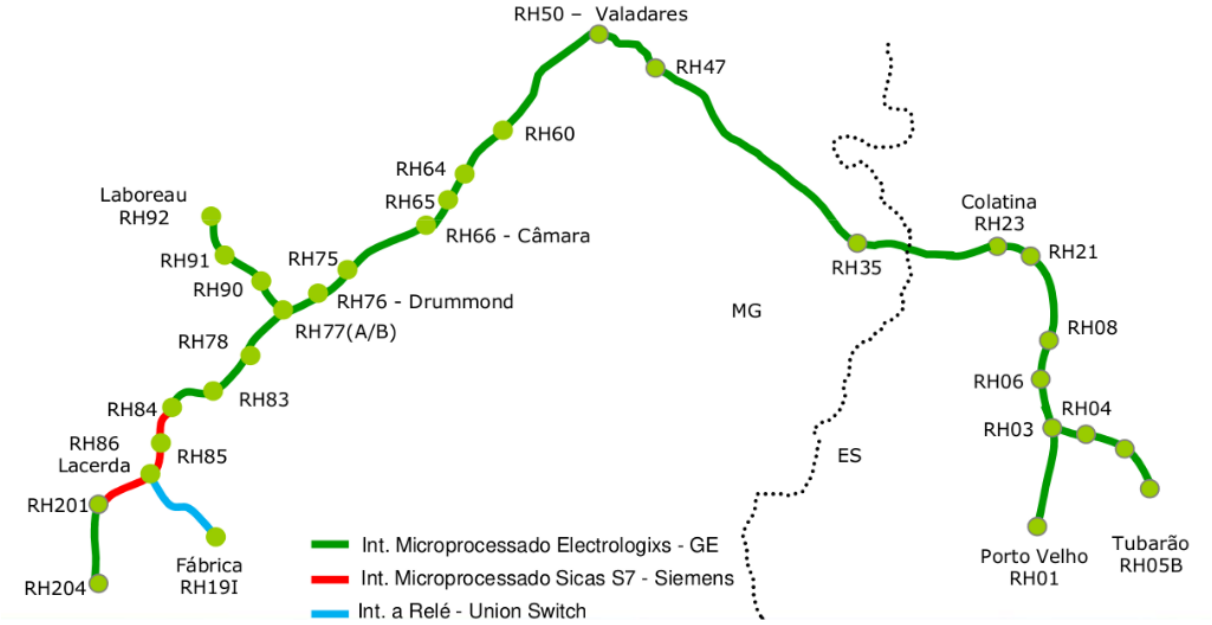


Figura 12 – Mapa exibindo as principais "*RH's*" ao longo da Estrada de Ferro Vitória Minas (VALE, 2013)

A seguir, apresentam-se alguns exemplos de frases onde as NEs foram rotuladas como `LOCAÇÃO`:

- v dois oito oito TREM meia sete LOCAÇÃO cco cambio cco atende o viriato dois oito oito TREM na meia sete LOCAÇÃO [...]
- cco atendendo socorro de resplendor na entrehouse de tres duas LOCAÇÃO tres tres LOCAÇÃO linha um cambio
- cco minerio duzentos e um TREM na rh cinquenta uma LOCAÇÃO cambio

Algumas NEs podem sugerir a ideia de localização, porém, não devem ser erroneamente interpretadas como locações e, portanto, não devem ser rotuladas como tal. Exemplos incluem nomes de cidades, pátios de manobra, travadores, *SBs* e linhas. Abaixo, em negrito, apresentamos alguns exemplos nos quais essas NEs estão presentes:

- passageiro em **colatina**
- to com um r nove sete TREM agora no **travador tres dois** de **valadares**[...]
- a intermediaria sete tres LOCAÇÃO sete duas LOCAÇÃO ate a **sb dois** da sete duas LOCAÇÃO dentro do tunel

3.1.1 Conjunto de dados reais

O conjunto de dados reais original fornecido consiste em 1.925 trechos transcritos no trabalho de (RIBEIRO, 2020). Dentre esses, 600 foram rotulados manualmente utilizando a ferramenta *NER Text Annotator* (ARUNMOZHI, 2023). Contudo, algumas amostras rotuladas foram descartadas, e os principais motivos foram:

- **Falhas nos Dados:** Essas irregularidades comprometiam significativamente a compreensão das transcrições, podendo ter origens diversas, como cortes excessivos no áudio, ocorrência de ruídos intensos, presença de sons conflitantes ou amostras duplicadas. Um total de 80 amostras foi eliminado por esses motivos.
- **Escassez de Informação:** Alguns trechos originam-se de processos e rotinas totalmente ou parcialmente desconhecidas, o que implica a falta de conhecimento sobre a situação, tornando desafiador identificar com precisão as NEs.
- **Ausência de Contexto Específico:** Algumas mensagens exigiriam informações contextuais para que as NEs fossem identificadas com segurança. Aspectos relacionados ao emissor e receptor, bem como dados de localidade e o histórico da conversa, seriam cruciais para a rotulação de determinados trechos e, portanto, a ausência desses requisitos prejudica a confiabilidade da rotulação.

Ao final do processo, têm-se um conjunto com 432 trechos.

3.1.2 Conjunto de dados sintéticos

Com o objetivo de aprimorar a robustez e desempenho do *NER* desenvolvido neste estudo, é implementada uma técnica de aumento de dados. Essa abordagem visa enfrentar os desafios associados à limitação do tamanho do conjunto de dados rotulados, promovendo a expansão artificial da diversidade das amostras disponíveis para o treinamento.

A estratégia adotada consistiu na identificação de padrões textuais de comunicação comuns no conjunto de dados original e na reprodução desses padrões, diversificando as NEs rotuladas por outras possíveis. Para isso, foram criados padrões textuais que servirão como "molde" para as frases geradas.

Além do texto normal, esses padrões são compostos por palavras especiais, representadas

em maiúsculas, que serão substituídas por possíveis NEs. As palavras especiais existentes são as seguintes:

- **TREM**: Será substituída pelo identificador único de um trem. Exemplos: "*c cento e oitenta e quatro*", "*minerio três cinto*", etc...
- **LOCA_SIMPLES**: Será substituída por uma locação possível. Exemplos: "*rh cento e quatro*", "*house cinco*", etc...
- **LOCA_DUPLA**: Representa a ideia de espaço entre duas locações, mencionando normalmente duas locações sequenciais. Exemplos: "*trinta e um ida trinta e dois volta*" ou simplesmente "*tres um tres dois*".

A seguir, apresenta-se um exemplo de um desses padrões textuais:

TREM [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] chamando cco [*null*/cambio]

Observe que, as palavras dentro dos colchetes, serão selecionadas aleatoriamente para compor a frase final. Ressalta-se que que a palavra "*null*" indica que nenhuma palavra será adicionada. A continuação três frases geradas a partir do padrão textual apresentado:

- "*t oito cinco oito na rh sete dois volta linha um chamando cco cambio*"
- "*r dois nove oito sinalizada tres quatro ida chamando cco*"
- "*r quatrocentos e sete house oito dois volta linha dois chamando cco cambio*"

Todos os padrões textuais utilizados para a geração dos dados sintéticos estão listados no Anexo 7.

3.2 Arquitetura do modelo para classificação de NE

As características das versões disponíveis do modelo BERTimbau podem ser vistas na Tabela 1, ao comparar a arquitetura das duas versões, observa-se que o modelo *Large* possui uma maior capacidade de abstração, no entanto, o treinamento é mais custoso.

Devido às restrições de tempo e processamento computacional, foi usada apenas a versão *Base*.

O vetor de saída do BERTimbau é utilizado para classificar cada *token* em algum das NEs de interesse (`TREM` ou `LOCAÇÃO`). Para tal finalidade, empregamos a saída do modelo Bertimbau como entrada para uma rede neural com uma única camada totalmente conectada.

A rede neural tem como tamanho de entrada o número de elementos do vetor de saída do modelo BERTimbau e como número de saídas o total de classes atribuíveis a cada *token*, incluindo as classes `B-TREM`, `I-TREM`, `B-LOCAÇÃO`, `I-LOCAÇÃO` e `O`.

A Figura 13 mostra em detalhes a camada responsável pela classificação, denominada **classificador**, enquanto a Figura 14 ilustra a arquitetura final do modelo que será utilizado no treinamento.

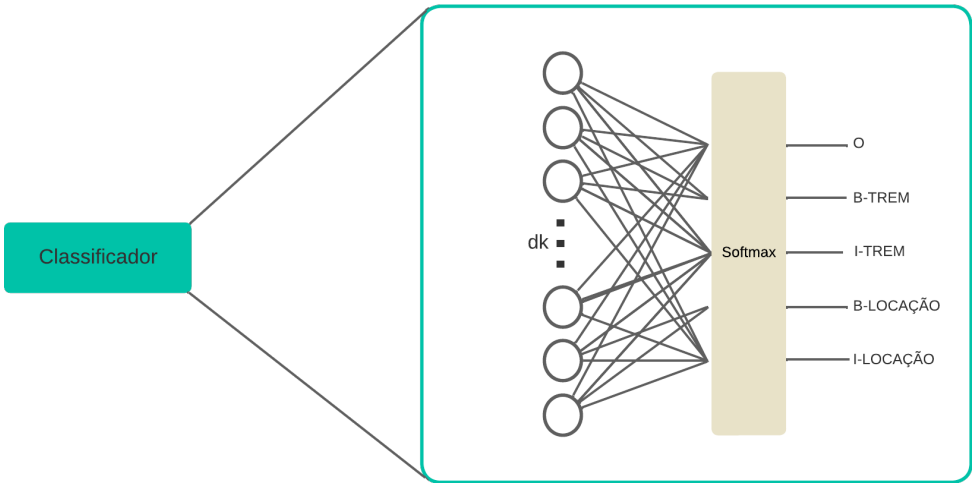


Figura 13 – Representação detalhada do **classificador**

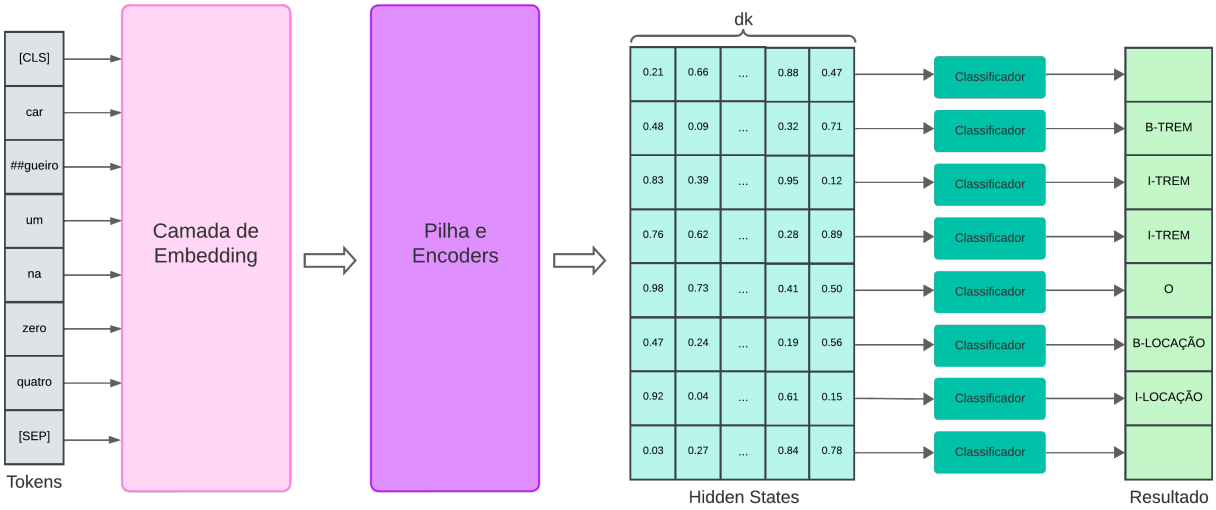


Figura 14 – Estrutura do modelo final utilizado no processo de classificação dos *tokens*

É importante destacar que, para o treinamento do modelo foram feitas as seguintes consi-

derações: (*i*) a função de custo empregada é a Entropia Cruzada Categórica, amplamente usada em problemas de classificação com mais de duas classes, conforme destacado em (RASCHKA, 2022) e (GéRON, 2023); (*ii*) todos os pesos da rede são ajustados no processo de *fine-tuning*, não se limitando apenas aos pesos do classificador.

4 RESULTADOS

Neste capítulo, apresentaremos os resultados obtidos. O capítulo inicia descrevendo os recursos computacionais utilizados em 4.1, em seguida, são apresentados detalhes dos experimentos realizados em 4.2, ressaltando as métricas utilizadas e os resultados obtidos em cada uma delas e, por fim, é elaborado um resumo dos resultados obtidos.

4.1 Recursos Computacionais

Recursos de *Hardware*

O processo de treinamento e avaliação dos modelos desenvolvidos será executado na plataforma *Google Colab*. Essa ferramenta oferece acesso gratuito, porém limitado, a recursos de computação, como CPU, GPU e armazenamento. A configuração pode variar de acordo com a disponibilidade desses recursos ou políticas internas da empresa *Google*. No momento de escrita desse trabalho a configuração a ser usada é:

- **Sistema Operacional:** Linux version 5.15.107+;
- **Processador:** Intel Xeon CPU @ 2.20GHz;
- **Memória RAM:** 12,7GB;
- **Unidade de armazenamento:** 78,2GB;
- **Placa de vídeo:** Tesla T4;
- **Versão do driver da GPU:** 525.85.12.

Recursos de *Software*

Entre os recursos de Software utilizados podemos destacar: (*i*) a linguagem *Python*, utilizada no desenvolvimento de todos os códigos utilizados no trabalho, a biblioteca *Pytorch*, utilizada em todo o *pipeline* do aprendizado de máquina; (*ii*) a biblioteca *Transformers* da empresa *Hugging Face*, que oferece funcionalidades para criar e usar modelos compartilhados; (*iii*) a ferramenta *open-source NER Text Annotator*, utilizada para rotular os dados textuais.

4.2 Experimentos

4.2.1 Métricas

Durante o treinamento e a avaliação dos modelos, três métricas são utilizadas:

- **Train Loss e Eval Loss**. Estas métricas correspondem à saída da função de custo aplicada aos conjuntos de treino e validação, respectivamente. A monitorização desses valores é crucial para verificar se há um comportamento típico de subajuste ou sobreajuste do modelo aos dados.
- **Overall Micro-F1**. A avaliação global do desempenho do modelo está sendo feita utilizando a variação micro da métrica F1, que considera o peso de cada amostra de forma equitativa. Isso é especialmente útil quando não há uma preocupação em balancear o conjunto de dados, como foi o caso, embora não haja um desequilíbrio significativo, conforme pode ser observado nas Tabelas 3 e 4. Além disso, há uma preferência por ser a métrica geral mais utilizada, facilitando a comparação com outros trabalhos.

4.2.2 Conjunto de Dados

Para o treinamento de modelos de aprendizado, é essencial dividir o conjunto de dados em três subconjuntos distintos, a dizer: treino, validação e teste. Com o objetivo de garantir um desempenho satisfatório do modelo em situações do mundo real, os conjuntos de validação e teste, dos quais extrairemos as métricas de desempenho, consistem exclusivamente em dados reais, cada um contendo 43 amostras, o que representa 10% do total de dados reais rotulados.

No conjunto de dados de treino, são apresentadas quatro configurações distintas, permitindo a avaliação da influência dos dados artificiais no treinamento do modelo. Os números de amostras em cada configuração estão detalhados na Tabela 2.

A Tabela 3 exibe a quantidade de *tokens* de cada uma das classes de interesse presentes nos diferentes conjuntos de treino, enquanto a Tabela 4 exibe as mesmas informações para os conjuntos de validação e teste. Cada conjunto de dados foi gerado três vezes, variando as sementes aleatórias.

Configuração do Conjunto de treino	Amostras reais	Amostras artificiais
I	346	0
II	346	500
III	346	1000
IV	346	2000

Tabela 2 – Composição de cada configuração do conjunto de treino

Config.	**Quantidade NEs**		**Quantidade Tokens**			
	TREM	LOCAÇÃO	B-TREM	I-TREM	B-LOCAÇÃO	I-LOCAÇÃO
I	397 ± 6	397 ± 13	397 ± 6	981 ± 11	397 ± 13	481 ± 17
II	897 ± 6	822 ± 18	897 ± 6	2376 ± 42	822 ± 18	1300 ± 17
III	1397 ± 6	1248 ± 18	1397 ± 6	3735 ± 15	1248 ± 18	2124 ± 13
IV	2397 ± 6	2101 ± 21	2397 ± 6	6510 ± 35	2101 ± 21	3765 ± 20

Tabela 3 – Quantidade de NEs e *tokens* em cada configuração do conjunto de treino

Conjunto	**Qtd NE**		**Qtd Tokens**			
	TREM	LOCAÇÃO	B-TREM	I-TREM	B-LOCAÇÃO	I-LOCAÇÃO
Validação	53 ± 7	54 ± 12	53 ± 7	129 ± 18	54 ± 12	67 ± 14
Teste	43 ± 2	41 ± 7	43 ± 2	105 ± 10	41 ± 7	49 ± 10

Tabela 4 – Quantidade de NEs e *tokens* nos conjuntos de validação e teste

Cabe ressaltar que tanto a geração dos dados sintéticos quanto a divisão dos conjuntos de treino, validação e teste ocorrem de maneira aleatória, assegurando assim a imparcialidade e a representatividade do modelo em relação à diversidade dos dados disponíveis.

4.2.3 Treinamento

O treinamento do modelo foi conduzido por meio da *API Trainer* disponibilizada pelo *Hugging Face*, e todos os hiperparâmetros empregados estão detalhados no Anexo 8. Contudo, devido à sua relevância no processo de treinamento, a seguir serão descritos tanto o otimizador quanto o número de épocas e o critério de Parada considerados:

- **Otimizador**. O otimizador utilizado é o AdamW, uma variação do otimizador Adam (KINGMA; BA, 2015). O AdamW foi proposto em (LOSHCHILOV; HUTTER, 2018) e apresenta melhorias significativas na abordagem da regularização de decaimento de peso, proporcionando uma abordagem mais robusta e eficiente durante o treinamento do modelo.
- **Número de Épocas e Critério de Parada**. O número de épocas utilizado em todos os treinamentos foi fixado em 100. No entanto, também foi estabelecido um critério de parada responsável por encerrar o treinamento caso não haja me-

lhora em uma métrica específica, medida no conjunto de dados de validação, ao longo de um determinado número de épocas. Tanto a métrica observada quanto a quantidade de épocas monitoradas são configuráveis pelos parâmetros `best_metric` e `early_stopping_patience`, respectivamente. Os valores adotados em todos os treinamentos são uniformes e estão detalhados na Tabela 5.

Parâmetros	Valores
`best_metric`	Overall Micro F1
`early_stopping_patience`	10

Tabela 5 – Configuração dos parâmetros de *best_metric* e *early_stopping_patience*

4.2.4 Avaliação

Para avaliar os modelos, realizamos cada experimento três vezes, variando as sementes aleatórias. Os resultados obtidos foram consolidados na Tabela 6, onde cada coluna representa uma configuração do conjunto de treinamento e os melhores resultados estão destacados.

	I	**II**	**III**	**IV**
F1 - `LOCAÇÃO` (%)	83,34 ± 1,33	**86,72 ± 0,92**	85,75 ± 4,58	82,84 ± 4,59
F1 - `TREM` (%)	92,87 ± 4,11	**94,28 ± 1,28**	93,61 ± 2,57	91,54 ± 1,38
Overall Micro F1	88,06 ± 2,33	**90,53 ± 0,73**	89,45 ± 2,70	87,16 ± 2,98

Tabela 6 – Compilação dos resultados obtidos com os modelos treinados

Nota-se que as configurações de treinamento II e III alcançaram melhores resultados quando comparadas com a configuração I, que é composta exclusivamente por dados reais. Isso sugere que a adição de dados sintéticos teve um impacto positivo no desempenho do modelo. No entanto, a configuração IV, que apresentou um desempenho inferior, pode ser associada à grande quantidade de amostras sintéticas em relação às amostras reais, possivelmente resultando em um viés do modelo em direção aos dados sintéticos.

É interessante observar que o uso parcimonioso de dados sintéticos, como evidenciado nos resultados de II e III, contribuiu positivamente. O melhor desempenho foi alcançado em II, onde os dados sintéticos correspondem a $59,10\%$ do conjunto de dados, como podemos observar na Tabela 7. Esta constatação sugere que a inclusão de dados sintéticos pode ser uma estratégia eficaz para melhorar os resultados do modelo, principalmente considerando que esses dados podem ser facilmente reproduzidos sem a necessidade de rotulação manual. Essa característica torna a abordagem com dados sintéticos uma opção promissora para cenários nos quais a criação de dados rotulados manualmente é desafiadora ou dispendiosa.

Configuração	Proporção dos dados sintéticos (%)
I	0.00
II	59.10
III	74.29
IV	85.25

Tabela 7 – Proporção dos dados sintéticos em relação ao conjunto

A seguir, serão apresentados alguns exemplos de predições incorretas feitas pelo modelo, acompanhados de comentários sobre suas possíveis causas:

- (a)
  PREDIÇÃO: [...] inspetor ja foi pra dois zero cinco TREM testar o cargueiro meia dois cinco TREM cambio [...]

  RÓTULO: [...] inspetor ja foi pra dois zero cinco LOCAÇÃO testar o cargueiro meia dois cinco TREM cambio [...]

- (b)
  PREDIÇÃO: confirma pra mim shunt fixo circuito de chave linha dois rh vinte sete LOCAÇÃO cambio confirmado shunt fixo circuito de chave da linha dois dois zero cinco LOCAÇÃO o senhor vai trabalhar quanto tempo programado na ldl oito quatro quatro nove LOCAÇÃO cambio

  RÓTULO: confirma pra mim shunt fixo circuito de chave linha dois rh vinte sete LOCAÇÃO cambio confirmado shunt fixo circuito de chave da linha dois dois zero cinco TREM o senhor vai trabalhar quanto tempo programado na ldl oito quatro quatro nove cambio

- (c)
  PREDIÇÃO: camara ana to com laurindo sete cinco dois TREM passando pelo travador tres tres LOCAÇÃO pra socorrer o minerio zero setenta TREM

  RÓTULO: camara ana to com laurindo sete cinco dois TREM passando pelo travador tres tres pra socorrer o minerio zero setenta TREM

- (d)
  PREDIÇÃO: torre d t meia vinte TREM um no silva cambio

  RÓTULO: torre d t meia vinte um TREM no silva cambio

Visualizando as predições e seus respectivos rótulos, notam-se dois padrões:

- Nos exemplos (*a*) e (*b*), observa-se a mesma NE, "*dois zero cinco*", em dois contextos distintos. Enquanto uma foi rotulada como `LOCAÇÃO`, a outra foi rotulada como `TREM`. É comum encontrar situações em que os trens são mencionados sem o uso do prefixo, podendo assim ser confundidos com `LOCAÇÕES`. Este equivoco ocorre majoritariamente com locações com três dígitos, já que essa quantidade de dígitos é normalmente associada a trens. Além disso, o número de locações com três dígitos é reduzido em comparação ao total de locações, o que resulta em poucas variações de situações para o treinamento do modelo. Ademais, em algumas ocasiões, a diferenciação só é possível com o auxílio do histórico das conversas como contexto.
- Nos exemplos (*b*), (*c*), e (*d*) observa-se o comportamento do modelo em situações menos comuns, como a presença de códigos numéricos que não pertencem a nenhuma NE de interesse. Por exemplo, no exemplo (*b*), é mencionado o código de uma LDL (Documento para Liberação e Devolução de Linha), e no exemplo (*c*), o código de um travador. Nota-se uma predição incorreta do modelo também na situação do exemplo (*d*), onde as letras `d` e `t` são erroneamente previstas como prefixo. Vale ressaltar que é bastante incomum essas letras aparecerem fora desse contexto.

5 CONCLUSÕES E PROJETOS FUTUROS

5.1 Conclusões

O objetivo central deste estudo foi apresentar uma abordagem eficaz para classificar as entidades nomeadas no conjunto de dados fornecido pela VALE, proveniente das comunicações internas via rádio da empresa. A técnica proposta fundamenta-se principalmente na criação de conjuntos de dados para treinamento de um modelo neural, utilizando a técnica de *Fine Tuning* do modelo Bertimbau. Os conjuntos de dados para treinamento foram gerados tanto por meio do processo de rotulagem manual dos dados originais quanto pela criação de amostras sintéticas baseadas em padrões textuais. Nesse contexto, diferentes versões do modelo foram treinados com diversas quantidades de dados sintéticos, permitindo a realização de comparações entre os resultados obtidos. A métrica micro F1 foi utilizada para avaliação, revelando que os melhores resultados foram alcançados com uma utilização parcimoniosa dos dados sintéticos, resultando em um desempenho satisfatório de $90,53\%$.

Este trabalho pode oferecer contribuições em diversas aplicações no contexto empresarial da Vale, destacando-se especialmente para o arquivamento e busca estruturada nos históricos de comunicação. Essa abordagem facilita e agiliza processos de auditoria, proporcionando uma gestão mais eficiente e assertiva.

5.2 Temas a serem pesquisados

Com o intuito de aprimorar o desempenho do modelo desenvolvido, futuras pesquisas explorarão a expansão da rotulagem de dados, ampliando o conjunto de treinamento para capturar a diversidade das entidades nomeadas. Além disso, será considerada a expansão dos padrões de texto para a geração dos dados sintéticos, visando aumentar a variedade e representatividade do conjunto de treinamento.

Outra linha de investigação consistirá na repetição dos experimentos utilizando a versão *large* do Bertimbau. Tal versão, com sua maior capacidade de abstração, pode aprimorar ainda mais o desempenho do modelo.

Ao analisar as amostras que apresentaram dificuldades na classificação das NE de interesse, uma hipótese para contornar o problema é ampliar a janela de contexto. Tal extensão

pode beneficiar a classificação das entidades que são ambíguas ou apresentam desafios específicos.

Por fim, visando atender às necessidades específicas da Vale, é sugerida a inclusão de novas classes de NEs. A adoção de categorias como `LOCOMOTIVA`, `VAGÃO`, `TRAVADOR`, entre outras, pode enriquecer a capacidade do modelo em reconhecer entidades relevantes para o contexto empresarial da Vale.

REFERÊNCIAS

ARUNMOZHI. NER Text Annotator. 2023. Disponível em: <https://tecoholic.github.io/ner-annotator>. 35

CANETE, J. et al. Spanish pre-trained bert model and evaluation data. In: PML4DC at ICLR 2020. [S.l.: s.n.], 2020. 17

DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. 16, 19, 26

GRISHMAN, R.; SUNDHEIM, B. Message Understanding Conference- 6: A Brief History. 1996. 15

GéRON, A. Hands-on machine learning with Scikit-Learn, Keras and TensorFlow. 2ª ed.. ed. [S.l.]: O'Reilly, 2023. ISBN 9781492032649. 38

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR), 2015. Disponível em: <https://arxiv.org/abs/1412.6980>. 41

KRIPKE, S. A. Naming and Necessity. 1985. 15

LI, J. et al. A Survey on Deep Learning for Named Entity Recognition. 2020. 15, 17, 29

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. international conference on learning representations, 2018. 41

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. 2007. 15

NAKAYAMA, H. seqeval: A Python framework for sequence labeling evaluation. 2018. Software available from https://github.com/chakki-works/seqeval. Disponível em: <https://github.com/chakki-works/seqeval>. 29

NEURALMIND. BERTimbau - Portuguese BERT. 2023. Disponível em: <https://github.com/neuralmind-ai/portuguese-bert>. 27

PETASIS, G. et al. Automatic adaptation of proper noun dictionaries through cooperation of machine learning and probabilistic methods. 2000. 15

POLIGNANO, M. et al. AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets. 2019. 17

PREOTEASA, I. F. Self-attention. Towards Data Science, 2020. Disponível em: <https://towardsdatascience.com/self-attention-5b95ea164f61#c2e8>. 7, 25

PRIBERAM. "Trem"in Dicionário Priberam da Língua Portuguesa. 2023. Disponível em: <https://dicionario.priberam.org/trem>. 31

RADFORD, A. Improving Language Understanding by Generative Pre-Training. 2018. 16

RADFORD, A. et al. Language Models are Unsupervised Multitask Learners. 2019. 16

RASCHKA, S. Machine Learning with PyTorch and Scikit-Learn. 1ª ed.. ed. [S.l.]: Packt Publishing, 2022. ISBN 9781801819312. 26, 38

RIBEIRO, D. J. Desenvolvimento de um Dataset de Sinais de Áudio e um Modelo de Rotulação Orientado a Aplicações de Reconhecimento de Fala. 2020. 35

SOUZA, F. A.; NOGUEIRA, R.; LOTUFO, R. d. A. BERTimbau: Pretrained BERT Models for Brazilian Portuguese. 2020. 17, 27, 51

TUNSTALL, L. Natural Language Processing with Transformers. 2ª ed.. ed. [S.l.]: O'Reilly Media, Inc., 2022. ISBN 9781098136796. 25, 29

VALE. Sistemas de Licenciamento Ferroviário da EFVM. 2013. IV Encontro de Ferrovias da Associação Nacional dos Transportadores Ferroviários. Disponível em: <https://www.antf.org.br/_uploads/2017/08/Sistema-de-Licenciamento-Ferrovi%C3%A1rio-da-EFVM-Modo-de-Compatibilidade.pdf>. 7, 33, 34

VASWANI, A. et al. Attention is All You Need. 2017. 16, 17, 18, 23

VáZQUEZ, R. A. et al. Description and Analysis of an Indoor Positioning System That Uses Wireless ZigBee Technology. 2010. 14

WAGNER, J. et al. The brwac corpus: A new open resource for brazilian portuguese. In: . [S.l.: s.n.], 2018. 27

Appendices

6 TOKENIZAÇÃO E MAPEAMENTO

O processo de tokenização consiste em dividir um texto em unidades menores, chamadas de *tokens*. Após essa divisão, cada *token* é mapeado para um código de identificação em um vocabulário, incluindo os *tokens* especiais, quando existentes. Existem três principais métodos de tokenização, que serão brevemente discutidos a seguir:

- **Tokenização em Palavras**. A tokenização em palavras é o método mais tradicional, onde cada *token* corresponde a uma palavra individual no texto. As principais vantagens desse método são a simplicidade e a interpretabilidade, já que é mais fácil compreender o contexto de cada *token*, uma vez que cada unidade carrega um significado. No entanto, para obter uma representação satisfatória do texto, é necessário um grande número de *tokens* únicos no vocabulário, o que pode resultar em um vocabulário extenso, aumentando a complexidade computacional e podendo gerar dificuldades no treinamento. Além disso, esse método pode ter dificuldades em lidar com palavras desconhecidas ou fora do vocabulário, exigindo a atribuição de um *token* especial para essas palavras, o que, dependendo da frequência com que ocorre, pode prejudicar a representação do texto. A Figura 15 exibe um exemplo de tokenização em palavras.

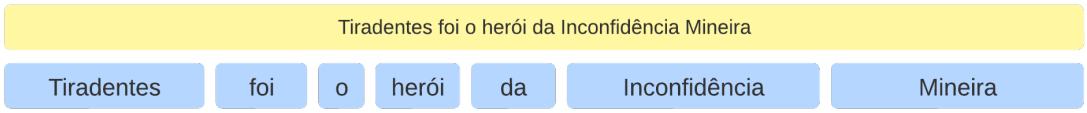


Figura 15 – Exemplo de tokenização de uma sentença em palavras.

- **Tokenização em Caracteres**. Essa técnica lida com o texto em seu nível mais fundamental, dividindo-o em caracteres individuais. Entre as vantagens, podemos citar a flexibilidade linguística, uma vez que qualquer palavra do idioma escolhido pode ser representada nessa técnica, e a redução significativa do vocabulário. Entre os problemas enfrentados por essa técnica, temos o aumento da sequência de entrada, pois cada caractere individual é tratado como um *token*, o que pode aumentar a complexidade computacional, e a perda de interpretabilidade, já que o caractere isolado não tem significado contextual ou semântico. A Figura 16 exibe um exemplo de tokenização em caracteres.

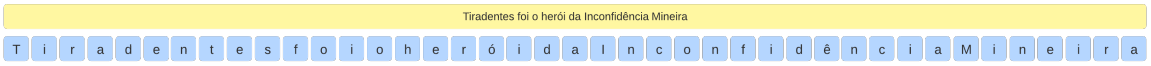


Figura 16 – Exemplo de tokenização de uma sentença em caracteres.

- **Tokenização em Subpalavras**. Essa é uma abordagem mais avançada que divide as palavras em partes menores, chamadas subpalavras. Essa técnica combina vantagens e desvantagens das técnicas mencionadas anteriormente, como a redução do vocabulário e flexibilidade linguística da tokenização em caracteres e a maior interpretabilidade da tokenização em palavras. Essa técnica tem se mostrado uma abordagem equilibrada entre os benefícios e desvantagens dos outros modelos e, portanto, tem sido amplamente utilizada em trabalhos recentes.

  Essa é a técnica utilizada em (SOUZA; NOGUEIRA; LOTUFO, 2020), onde é explicado como o vocabulário foi desenvolvido usando a biblioteca SentencePiece com o algoritmo BPE. Posteriormente, o vocabulário gerado é convertido para o formato WordPiece para manter a compatibilidade com o modelo BERT original. Na Figura 17, é apresentado um exemplo de como um texto de entrada passa pelo processo de tokenização em subpalavras.

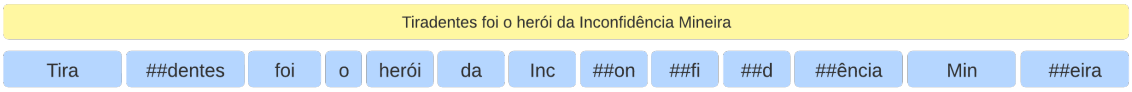


Figura 17 – Exemplo de tokenização de uma sentença em subpalavras.

7 PADRÕES TEXTUAIS UTILIZADOS NA CRIAÇÃO DOS DADOS SINTÉTICOS

Todos os padrões textuais utilizados na criação do conjunto de dados sintéticos, no processo explicado em 3.1.2, são listadas a seguir:

- **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] [*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] [*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] chamando cco [*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] chamando cco [*null*/cambio]
- **TREM** atende cco [*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] atende cco[*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] atende cco[*null*/cambio]
- cco chamando **TREM** [*null*/ cambio]
- cco chamando **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] [*null*/cambio]
- cco chamando **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] [*null*/cambio]
- cco atende **TREM** [*null*/ cambio]
- cco atende **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] [*null*/cambio]
- cco atende **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] [*null*/cambio]
- cco chamando **TREM** [*null*/ cambio] **TREM** atende cco [*null*/cambio]

- cco chamando **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] [*null*/cambio] **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] atende cco[*null*/cambio]
- cco chamando **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] [*null*/cambio] **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] atende cco[*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] chamando cco [*null*/cambio] cco atende **TREM** [*null*/ cambio]
- **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] chamando cco [*null*/cambio] cco atende **LOCA_SIMPLES** [*null*/ cambio]
- **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] chamando cco [*null*/cambio] cco atende **TREM** [*null*/na/pra] [*null*/singela/sinalizada] **LOCA_SIMPLES** [*null*/linha um/linha dois] [*null*/cambio]
- **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] chamando cco [*null*/cambio] cco atende **TREM** [*null*/na/pra] [*null*/intermediária/entrehouse] **LOCA_DUPLA** [*null*/linha um/linha dois] [*null*/cambio]

8 HIPERPARÂMETROS UTILIZADOS NO TREINAMENTO

Os hiperparâmetros usados nos treinamentos dos modelos implementados neste trabalho são mostrados na Tabela 8, Cabe indicar que todos os modelos tiveram a mesma configuração.

Hipérmetro	**Valor**
attention_probs_dropout_prob	0,1
hidden_act	gelu
hidden_dropout_prob	0,1
hidden_size	768
initializer_range	0,02
intermediate_size	3072
layer_norm_eps	10^{-12}
max_position_embeddings	512
num_attention_heads	12
num_hidden_layers	12
pad_token_id	0
pooler_fc_size	768
pooler_num_attention_heads	12
pooler_num_fc_layers	3
pooler_size_per_head	128
pooler_type	first_token_transform
position_embedding_type	absolute
type_vocab_size	2
use_cache	true
vocab_size	29794

Tabela 8 – Configurações do Modelo