Allison de Souza
Vinicios Camello

BERT PolREN: Análise da precisão sobre a inferência textual de reconhecimento de entidades nomeadas do BERTimbau no cenário político

Florianópolis – SC

2023

Allison de Souza
Vinicios Camello

BERT PolREN: Análise da precisão sobre a inferência textual de reconhecimento de entidades nomeadas do BERTimbau no cenário político

Proposta de Trabalho Conclusão do Curso de Graduação em Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal de Santa Catarina – UFSC

Departamento de Informática e Estatística – INE

Graduação em Sistemas de Informação

Orientador: Carina Friedrich Dorneles

Florianópolis – SC

2023

Resumo

Atualmente enfrentamos diversos problemas relacionados à sobrecarga de informações, principalmente no âmbito político. Diante dessa situação, estamos em busca de uma solução que facilite a identificação de pessoas, cargos políticos, locais e eventos mencionados nos textos, a fim de filtrar e compreender melhor os acontecimentos. Nesse contexto, é proposto realizar análises utilizando o BERTimbau, uma versão em português do modelo BERT, aplicado a dados coletados por meio de web scraping, com o parâmetro "política Brasil". Nosso objetivo é avaliar a precisão da inferência de Reconhecimento de Entidades Nomeadas (NER) em um cenário seletivo, dividido em cinco classes: PESSOA/CARGO, ABSTRAÇÃO/IDEIA, ACONTECIMENTO, LOCAL/HUMANO e FÍSICO/REGIÃO. Além disso, pretendemos comparar a precisão entre fontes de dados do Twitter e da Wikipedia. Para validar nossa abordagem, adotamos dois métodos distintos, o primeiro consiste em uma validação manual, no qual analisamos individualmente a saída do modelo BERTimbau para a classe "PESSOA/CARGO", gerando gráficos que demonstram a precisão dos resultados obtidos; O segundo método envolve o uso de *scripts* de validação, que podem ser aplicados posteriormente, para validar uma grande quantidade de dados de forma automatizada. O primeiro método apresenta uma precisão maior, enquanto o segundo método oferece perspectivas promissoras para validações em escala. Com base nas análises e nas precisões obtidas, foi possível verificar a eficiência do modelo BERT PolREN no que diz respeito à resolução do problema identificado.

Palavras-chave: político, *scrapings*, NER, classes, BERTimbau, validação, análises, BERT PolREN.

Lista de ilustrações

Lista de abreviaturas e siglas

BERT	Bidirectional Encoder Representations from Transformers
HAREM	Harmonia, Anotação e Recuperação de Entidades Mencionadas
NER	Named-Entity Recognition
NLP	Natural Language Processing
JSON	JavaScript Object Notation
API	Application Programming Interface
CRF	Conditional Random Field
URL	Uniform Resource Locator
CoNLL	Conference on Natural Language Learning

Sumário

1 Introdução

A solução de Reconhecimento de Entidades Nomeadas (NER, na sigla em inglês) é uma técnica de processamento de linguagem natural que tem como objetivo identificar e classificar entidades específicas em um texto, tais como nomes de pessoas, organizações, locais, datas, valores monetários, entre outros. Essa abordagem é amplamente utilizada em diversas áreas, como extração de informações, busca e recuperação de informações, análise de sentimentos e opiniões, *chatbots* e assistentes virtuais, análise de dados e mineração de texto, e classificação de documentos.

No entanto, o treinamento de um modelo de NER eficaz apresenta algumas dificuldades. É necessário dispor de um conjunto de dados rotulados, nos quais as entidades nomeadas estejam devidamente anotadas. A criação manual desses dados pode ser um processo trabalhoso, considerando a variação de tipos de entidades, a presença de nomes ambíguos ou incomuns, além da limitação de generalização para diferentes domínios. Por exemplo, um modelo treinado em notícias pode ter dificuldades para reconhecer entidades em textos políticos.

O modelo do Google BERT tem demonstrado um desempenho excepcional em diversas tarefas de processamento de linguagem natural, incluindo classificação de texto, reconhecimento de entidades nomeadas, análise de sentimento, entre outras. Porém a versão original do BERT não funciona para língua portuguesa, tornando necessário a utilização do BERTimbau, versão do BERT para português.

Com base nisso, propomos realizar uma análise dos dados coletados das fontes Wikipedia e Twitter, no contexto político. Pretendemos aplicar o algoritmo do BERTimbau para extrair informações valiosas de forma resumida. Para avaliar a eficácia do BERT, adotamos duas abordagens: uma análise manual dos dados caso a caso, a fim de medir a precisão com maior eficácia, e uma abordagem automatizada, utilizando *scripts* que utilizam dados públicos do governo contendo nomes de políticos para avaliar sua precisão.

Espera-se que essa metodologia proporcione uma avaliação completa da eficácia do BERTimbau na análise de dados no contexto político, proporcionando insights valiosos para compreender e extrair informações relevantes a partir dessas fontes de dados.

1.1 Objetivo Geral

O objetivo é verificar a precisão e acurácia da inferência textual NER do BERTimbau, em diferentes fontes e tamanhos de dados, para um cenário político. Para isso é necessário coletar dados do Twitter(textos curtos) e Wikipedia(textos longos), aplicar

a inferência textual NER do BERTimbau sobre esses dados, e na sequência analisar a acurácia e precisão de forma manual para os dados do Twitter e de forma automatizada para Wikipedia dessa inferência para um cenário político, Nesse processo é necessário a criação de dois web Scrapings para coletar os dados, treinar um modelo de inferência NER do BERTimbau, aplicar essa inferência sobre os dados e na sequência rodar os *scripts* de validação sobre os resultados da inferência, para o modelo automatizado do Wikipedia e aplicar a validação feita por uma pessoa análise no modelo manual do Twitter.

1.2 Objetivos Específicos

Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

i Coletar dados da Wikipedia e do Twitter através de scrapers, tendo como parâmetro política Brasil;

ii Normalização e armazenamento dos dados em um arquivo *JSON*;

iii Treinar um modelo com NER no BERTimbau utilizando o *First HAREM Dataset*;

iv Analisar a precisão e acurácia do modelo utilizando o *script conlleval*;

v Efetuar a inferência textual NER do algoritmo BERTimbau no conjunto de dados;

vi Efetuar a análise manual do desempenho da inferência sobre os dados do Twitter;

vii Efetuar a análise automatizada do desempenho da inferência sobre os dados do Wikipedia.

2 Fundamentação Teórica

Neste capítulo detalhamos os conceitos e definições teóricas que utilizamos para implementação deste projeto. Vamos detalhar o que é Scraping, CRF, NLP, o conceito de NER, como funciona o BERT e a descrição do HAREM *Dataset*.

2.1 *Scraping*

É o processo de coletar dados de um site de forma automatizada, envolve a extração de informações estruturadas ou não estruturadas de páginas da web.

> *Web scraping* é o processo de coleta de dados da web de maneira automatizada. Também é chamado de extração de dados da *web*. Alguns dos principais casos de uso do web scraping incluem monitoramento de preços, inteligência de preços, monitoramento de notícias, geração de *leads* e pesquisa de mercado, entre muitos outros. Em geral, a extração de dados da web é usada por pessoas e empresas que desejam usar a vasta quantidade de dados da web disponíveis publicamente para tomar decisões mais inteligentes. (KENNY, 2020, p. 1)

> Se você já copiou e colou informações de um site, você executou a mesma função que qualquer "raspador" da *web*, apenas em uma escala microscópica e manual. Ao contrário do processo mundano e entorpecedor de extrair dados manualmente, o web scraping usa automação inteligente para recuperar centenas, milhões ou até bilhões de pontos de dados da fronteira aparentemente infinita da Internet. (KENNY, 2020, p. 1)

Esse método é utilizado nesse trabalho, para coletar dados para executar a inferência textual NER e análises.

2.2 CRF

Conditional Random Fields é um *framework* para construir modelos probabilísticos utilizado em tarefas de processamento de linguagem natural e visão computacional para a tarefa de segmentar e rotular dados sequenciais. CRF oferece várias vantagens em relação a modelos ocultos de Markov e gramáticas estocásticas para essas tarefas, incluindo a capacidade de relaxar as fortes suposições de independência feitas nesses modelos. Os campos aleatórios condicionais também evitam uma limitação fundamental dos modelos de Markov de entropia máxima (MEMMs) e outros modelos de Markov discriminativos baseados em modelos gráficos direcionados, que podem ser tendenciosos em relação a estados com poucos estados sucessores (LAFFERTY; MCCALLUM; PEREIRA, 2001).

O CRF é amplamente aplicado em várias tarefas, como reconhecimento de entidades nomeadas, marcação de partes do discurso, segmentação de sequências de fala, entre outros. Ele modela as relações contextuais entre as observações de entrada e os rótulos de saída por meio de probabilidades condicionais (SUTTON; MCCALLUM et al., 2012).

No treinamento do CRF, o algoritmo de otimização mais comumente utilizado é o algoritmo de forward-backward (SUTTON; MCCALLUM et al., 2012).Este algoritmo é responsável por ajustar os parâmetros do CRF com base nos dados de treinamento e anotações das sequências rotuladas.

O CRF foi amplamente adotado e demonstrou forte desempenho em várias tarefas de visão computacional e processamento de linguagem natural. Sua capacidade de modelar dependências de longo alcance em sequências o torna uma escolha poderosa para analisar dados de sequência.

2.3 NLP

Também conhecido em português como PLN, sigla para Processamento de Linguagem Natural, é um subcampo da inteligência artificial criado para auxiliar robôs na compreensão do processo orgânico da comunicação humana, isso envolve desde a compreensão de palavras únicas até a avaliação de frases e textos completos.

De acordo com Jurafsky e Martin (2019), o NLP é um tópico em constante mudança e tem se tornado mais conhecido nos últimos anos devido à necessidade de automatizar operações relacionadas à linguagem e à quantidade crescente de dados disponíveis online. Tradução de máquina, análise de sentimento em redes sociais e resumo de textos são algumas das aplicações do NLP. Para realizar essas tarefas, utiliza métodos como análise morfológica, sintética e semântica, além de algoritmos de aprendizado de máquina. Esses algoritmos são treinados com grandes quantidades de dados para encontrar conexões e padrões entre palavras e frases, permitindo que a máquina compreenda a linguagem natural de forma eficaz.

Lidar com a ambiguidade e variedade da linguagem humana é um dos principais desafios enfrentados pelo NLP, como afirmado por Bird, Klein e Loper (2009). O significado das palavras pode mudar dependendo do contexto em que são usadas, e os falantes nem sempre seguem as regras gramaticais. Apesar dessas dificuldades, avançou consideravelmente nos últimos anos, em parte devido a novas estratégias e algoritmos, bem como à maior quantidade e qualidade de dados disponíveis. Como resultado, espera-se que o NLP continue desempenhando um papel fundamental em diversos campos, como marketing, saúde e educação, entre outros.

Ferramentas de escuta social, *chatbots* e sugestões de palavras em *smartphones* são

alguns outros exemplos de avanços tecnológicos possíveis graças ao NLP. Para mecanismos de busca, o NLP não é algo novo, no entanto, o BERT, que utiliza treinamento bidirecional e é o foco da pesquisa neste trabalho, representa um grande avanço no processamento de linguagem natural.

2.4 NER

Também conhecido como REN em português, que significa Reconhecimento de Entidade Nomeada.

> É uma técnica de NLP que consiste em identificar e categorizar informações-chave (entidades) em textos. Uma entidade pode ser qualquer palavra ou série de palavras que se referem ao mesmo tema. Cada entidade detectada é classificada em uma categoria predeterminada. Por exemplo, as entidades podem ser nomes de pessoas, organizações, locais, horários, quantidades, valores, entre outros. (ELINT, 2021, p. 1)

Segundo Elint (2021), em algoritmos de NER normalmente você precisa de uma grande quantidade de dados classificados manualmente para treinamento. Consequentemente, uma abordagem semi-controlada é recomendada para reduzir o esforço manual, no caso deste trabalho é utilizado um dataset conhecido como *Fisrt HAREM Dataset*.

Durante o treinamento, é importante evitar que o algoritmo simplesmente memorize os exemplos. Em vez disso, o objetivo é capacitá-lo a identificar e classificar as palavras-chave em um contexto semelhante, permitindo a automação do processo de extração de informações.

2.5 BERT

O Google é o principal buscador utilizado atualmente e passou por uma atualização do seu algoritmo, denominado BERT, que é um algoritmo de *Deep Learning* para NLP, que ajuda computadores a entender a linguagem como humanos. Ou seja, o BERT pode ajudar o Google a entender melhor o significado das palavras nas consultas no mecanismo de busca, pois essa nova atualização melhorou a capacidade de traduzir a linguagem humana para os computadores.

Segundo Devlin et al. (2018), Ao contrário de modelos recentes de representação de linguagem, o BERT foi projetado para pré-treinar representações profundas bidirecionais a partir de texto não rotulado, condicionando conjuntamente o contexto à esquerda e à direita em todas as camadas. Como resultado, o modelo pré-treinado BERT pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração para uma ampla gama de tarefas, como resposta a perguntas e inferência de linguagem, sem modificações substanciais na arquitetura específica da tarefa.

BERT é um modelo de aprendizado profundo que o Google *AI Language* desenvolveu em 2018 e é baseado na arquitetura do Transformer. Ele alcançou resultados inovadores em tarefas NLP, como classificação de sentimentos, tradução automática e funções de perguntas e respostas. Um dos principais benefícios dele é a sua capacidade de reconhecer o contexto bidirecional de palavras em uma frase, o que lhe permite entender a relação entre uma palavra e seus predecessores e sucessores, para isso, o BERT emprega um procedimento de pré-treinamento no qual é alimentado com grandes quantidades de texto não rotativo, aprendendo a representar palavras de forma mais eficaz como resultado.

> Um ponto importante de diferença entre o BERT e outros modelos de NLP é que é a primeira tentativa do Google de um modelo pré-treinado que é profundamente bidirecional e faz pouco uso de qualquer outra coisa além de um corpo de texto simples. Como é um modelo de código aberto, qualquer pessoa com conhecimento sólido de algoritmos de aprendizado de máquina pode usá-lo para desenvolver um modelo de NLP sem ter que integrar conjuntos de dados diferentes para treinamento de modelo, economizando recursos e dinheiro. (ACADEMY, 2022, p. 1).

O BERT tem sido a base para o desenvolvimento de outros modelos de PLN mais avançados, como o RoBERTa, o ALBERT e especificamente neste trabalho foi utilizado o BERTimbau, um modelo adaptado para língua portuguesa.

2.6 HAREM Dataset

O conjunto de dados Harem é uma coleção de anotações linguísticas criada para fins de pesquisa e desenvolvimento no campo de processamento de linguagem natural (PLN) e aprendizado de máquina. O nome "Harem"é um acrônimo para "Harmonia, Anotação e Recuperação de Entidades Mencionadas". Foram utilizados como base para escrever os seguintes parágrafos, os autores Mota e Santos (2008), Souza (2023) e Martins (2021)

O Harem dataset foi desenvolvido com o objetivo de fornecer um conjunto de dados anotados manualmente que representasse uma variedade de tipos de entidades e relações presentes na língua portuguesa. Essas anotações são particularmente relevantes para tarefas de extração de informações, análise de sentimentos, sumarização de texto, tradução automática e outras aplicações relacionadas à compreensão de linguagem natural.

O conjunto de dados Harem contém textos em português provenientes de diversas fontes, como notícias, artigos acadêmicos, blogs e páginas da *web*. Os documentos são anotados com informações sobre entidades mencionadas, como nomes de pessoas, organizações, locais e datas. Além disso, também são fornecidas anotações de relações entre essas entidades, como a filiação de uma pessoa a uma organização ou o papel de uma pessoa em um evento.

As anotações no conjunto de dados Harem são realizadas por anotadores humanos especializados, que seguem diretrizes claras e criteriosas para garantir a consistência e

qualidade das anotações. Além disso, o conjunto de dados passa por revisões e verificações adicionais para garantir sua precisão e utilidade para a comunidade de pesquisa.

O Harem *dataset* tem sido amplamente utilizado como referência em tarefas de PLN e aprendizado de máquina para a língua portuguesa. Pesquisadores e desenvolvedores podem treinar e avaliar algoritmos de processamento de linguagem natural usando esse conjunto de dados, permitindo avanços nas áreas de reconhecimento de entidades, análise de sentimentos, tradução automática e muito mais.

3 Trabalhos Relacionados

Trouxemos a avaliação de quatro trabalhos relacionados a implementação da nossa ferramenta, que trazem muitas informações sobre extração de dados e treinamento de conjuntos de dados, facilitando muito o entendimento sobre o assunto e auxiliando no desenvolvimento do projeto.

3.1 qFex: um Crawler para Busca e Extração de Questionários de Pesquisa em Documentos HTML

Segundo Mathias et al. (2017), o qFex que é um conjunto entre um extrator com base em heurísticas pré-definidas e um web crawler focado, ou seja um programa que varre a Web fazendo o download de páginas conforme a necessidade do usuário, a ideia é que juntos sejam capazes de fazer a coleta dos questionários e a extração de seus dados para uma base de dados. O objetivo do trabalho é coletar questionários de pesquisa na Web e extrair os dados para construir um banco de dados centralizado, os dados extraídos servirão como uma base de conhecimento que pode ser usada como ponto de partida para a construção de novos questionários ou para a análise das características presentes visando extrair algum tipo de informação útil.

É importante citar as heurísticas que foram utilizadas para fazer a detecção e a extração dos questionários na web: 1 - A descrição de uma pergunta normalmente começa com um número ou uma palavra qualquer iniciando com letra maiúscula, possuindo pelo menos quatro caracteres, um espaço e terminando com o caracter ‘:’, ‘?’ ou ‘.’. 2 - Perguntas normalmente possuem suas descrições e componentes de formulário próximos uns dos outros. 3 - É possível eliminar certos formulários/campos que não pertencem a um questionário, verificando a distância entre os componentes e certas palavras/frases que são normalmente encontradas nos mesmos. 4 - Páginas com questionários normalmente possuem certas palavras comuns em seu título e/ou em seu corpo. 5 - É possível determinar a presença de um questionário verificando a quantidade de clusters (agrupamento de nodos) que possuam pelo menos um componente de formulário ou que possuam uma certa quantidade de componentes de formulário. 6 - Todas as formas de perguntas seguem um ou mais padrões diferentes em suas estruturações nas páginas HTML e é possível utilizar tais padrões para fazer a extração das mesmas.

Como podemos ver o qFex foi dividido em scraping e extrator, a ideia de funcionamento é que ambos recebem um arquivo de configuração com as configurações de banco, biblioteca de scraping e as seeds para busca/extração, então o scraping deve varrer a

web com base nas seeds fornecidas, em busca de questionários com base nas heurísticas pré-definidas e salvar os links dos questionários no banco. O extrator deve pegar esses links que o scraping colocou no banco e extrair os dados dos questionários para um outro banco de dados onde ficam o respositório de questionários com base no que o usuário solicitou.

3.2 Absorção das tarefas de processamento de Linguagem Natural (NLP) pela Ciência da Informação (CI): uma revisão da literatura para tangibilização do uso de NLP pela CI

O presente artigo de Falcão, Lopes e Souza (2022), versa acerca da relevância das tecnologias digitais na Educação, destacando sua capacidade de proporcionar a democratização do acesso ao conhecimento e fomentar a aprendizagem autônoma. Ademais, são apresentados exemplos de aplicabilidade dessas tecnologias em distintos contextos educacionais, como no ensino a distância, na formação de docentes e na construção de ambientes de aprendizagem mais interativos e colaborativos. Por fim, são pontuados desafios e limitações no que tange ao uso dessas tecnologias, evidenciando a importância de um planejamento meticuloso e de uma reflexão crítica acerca de sua implementação na prática educacional.

O texto ressalta a capacidade do BERT em compreender o contexto em que as palavras são utilizadas em uma sentença, permitindo uma melhor compreensão de textos e um maior poder de processamento de informações. No entanto, o artigo também destaca que a aplicação do BERT na Educação ainda é uma área em desenvolvimento, sendo necessário um maior aprofundamento em pesquisas e estudos empíricos para avaliar sua eficácia e identificar possíveis limitações e desafios.

3.3 Técnicas Inteligentes para Análise de Agrupamento de Dados

Neste trabalho do Bordignon (2010), é utilizado um algoritmo (U*C) de IA para fazer o validação de técnicas de agrupamento de dados, é possível ter uma uma boa noção da acuracidade, desta forma dado dois conjuntos de dados, após aplicar o treinamento no dataset, conseguimos saber o quão bem o modelo está treinado e com isso se consegue extrair informações relacionadas entre os dados.

Utilizando aprendizado competitivo para guiar os exemplos de entradas, para isso foi utilizado uma Rede Neural Artificial de duas camadas conectadas sendo a saída chamada de camada competitiva e as conexões laterais entre os neurônios são utilizadas para inibição.

Para provar a acuracidade, foi utilizado o método K-médias que consiste em dividir

o conjunto de dados em K participações e de forma aleatória K centroides dentro do escopo do conjunto de dados. Também foi utilizado o método Agrupamento Nebuloso (Fuzzy) que é uma prática bastante utilizada para ajudar a definir os limites naturais vagos entre os grupos de dados. Outra técnica utilizada foi o Agrupamento Hierárquico para construir uma estrutura em forma de árvore denominada dendograma.

Após utilizar mais algumas técnicas como, Quantificação Vetorial por Aprendizagem, Agrupamento com Mapas Auto-Organizáveis, Matriz-U e Matriz P foi realizado o treinamento do conjunto em três passos, competitivo, cooperativo e adaptativo, assim tivemos o resultado que utilizando tantas técnicas não conseguiram obter o resultado desejado que era superar o autor do método.

3.4 Ferramenta para Coleta e Comparação de Dados de Publicações Acadêmicas dos Professores com o Currículo Lattes

É possível verificar conforme Branco et al. (2018), que neste trabalho relacionado, foi criado um scraping que é uma ferramenta para coletar dados da internet, e essa coleta de dados foi feita a partir três fontes específicas, que são sites do mundo acadêmico: DBLP, GoogleScholar e ResearchGate. A ideia é a comparação dos seus dados com os dados de publicações do currículo lattes a fim de criar uma visão unificada que tem como objetivo armazenar a maior quantidade possível de produções dos pesquisadores, visando ter essas publicações atualizadas pois nem sempre o currículo lattes está atualizado.

O scraping criado possui uma arquitetura simples que pode ser dividida em quatro partes: Conexão com um servidor de proxy que irá intermediar as requisições enviadas aos repositórios do GoogleScholar e ResearchGate; Acesso aos repositórios e extração dos dados das publicações; Análise sintática do currículo lattes para comparação com os dados extraídos dos repositórios; Persistência no banco de dados do currículo lattes usado para comparação das publicações que o scraping julgar que não estão contidas no lattes. Esse scraping foi desenvolvido utilizando JAVA e para o banco de dados PostgreSQL e a ferramenta pgAdmin 4 que facilita a visualização dos dados do banco através de uma interface acessível por um browser.

3.5 Análise dos trabalhos relacionados

Tanto no qFex quanto no trabalho do item 3.4 a ideia principal é a criação de um crawler para coletar e extrair os dados para construir um banco de dados, que é muito próximo do que fazemos no início do nosso trabalho para extrair dados do twitter e wikipedia. No artigo do item 3.2 aborda a utilização de NLP para democratizar o acesso ao conhecimento, utilizamos NLP e NER através da aplicação do BERT para facilitar

e de certa forma democratizar informações sobre política. No trabalho sobre análise de agrupamento de dados podemos relacionar com as fórmulas para cálculo de acuracidade e a validação de conferência de quão bem o modelo está treinado.

Utilizamos referências de trabalhos anteriores como guias para orientar a normalização dos dados e a validação do nosso estudo, como ilustrado na tabela abaixo.

	Scraping	Bert	NLP	Normalização	Validação/Acurácia
Ferramenta para Coleta e Comparação de Dados de Publicações Acadêmicas dos Professores com o Currículo Lattes	X			X	
Técnicas Inteligentes para Análise de Agrupamento de Dados			X	X	X
Absorção das tarefas de processamento de Linguagem Natural (NLP) pela Ciência da Informação (CI): uma revisão da literatura para tangibilização do uso de NLP pela CI		X	X		
qFex: um Crawler para Busca e Extração de Questionários de Pesquisa em Documentos HTML	X				
BERT PolREN: Análise da precisão sobre a inferência textual de reconhecimento de entidades nomeadas do BERTimbau no cenário político	X	X	X	X	X

Figura 1 – Tabela de correlação com os trabalhos relacionados

4 BERT PolREN

Neste capítulo é proposto o modelo BERT PolREN (**BERT** para **Pol**ítica com **R**econhecimento de **E**ntidade **N**omeada) que tem como base o BERTimbau[1](produzido pelo Fábio Souza, Rodrigo Nogueira e Roberto Lotufo) e faz uso dos seus fontes para implementação. Inicialmente é descrita uma visão geral do trabalho e depois os conceitos e os *scripts* utilizados no mesmo. Por fim efetuamos as análises e criamos gráficos para demonstrar de maneira visual os resultados encontrados.

4.1 Visão Geral

De forma geral, o processamento realizado passa pelas seguintes etapas, conforme mostra a Figura 1: (i) implementação e configuração dos Scrapers; (ii) Normalização e extração dos dados; (iii) Treinamento do modelo e aplicação do *conlleval script*; (iv) Aplicação das amostras do Wikipedia e dados do Twitter no algoritmo BERTimbau; (v) Validação manual e automatizada dos arquivos de retorno; (vi) Análise dos dados de precisão; (vii) Criação de gráficos e tabelas com os resultados.

[1]https://github.com/neuralmind-ai/portuguese-bert/blob/master/README.md

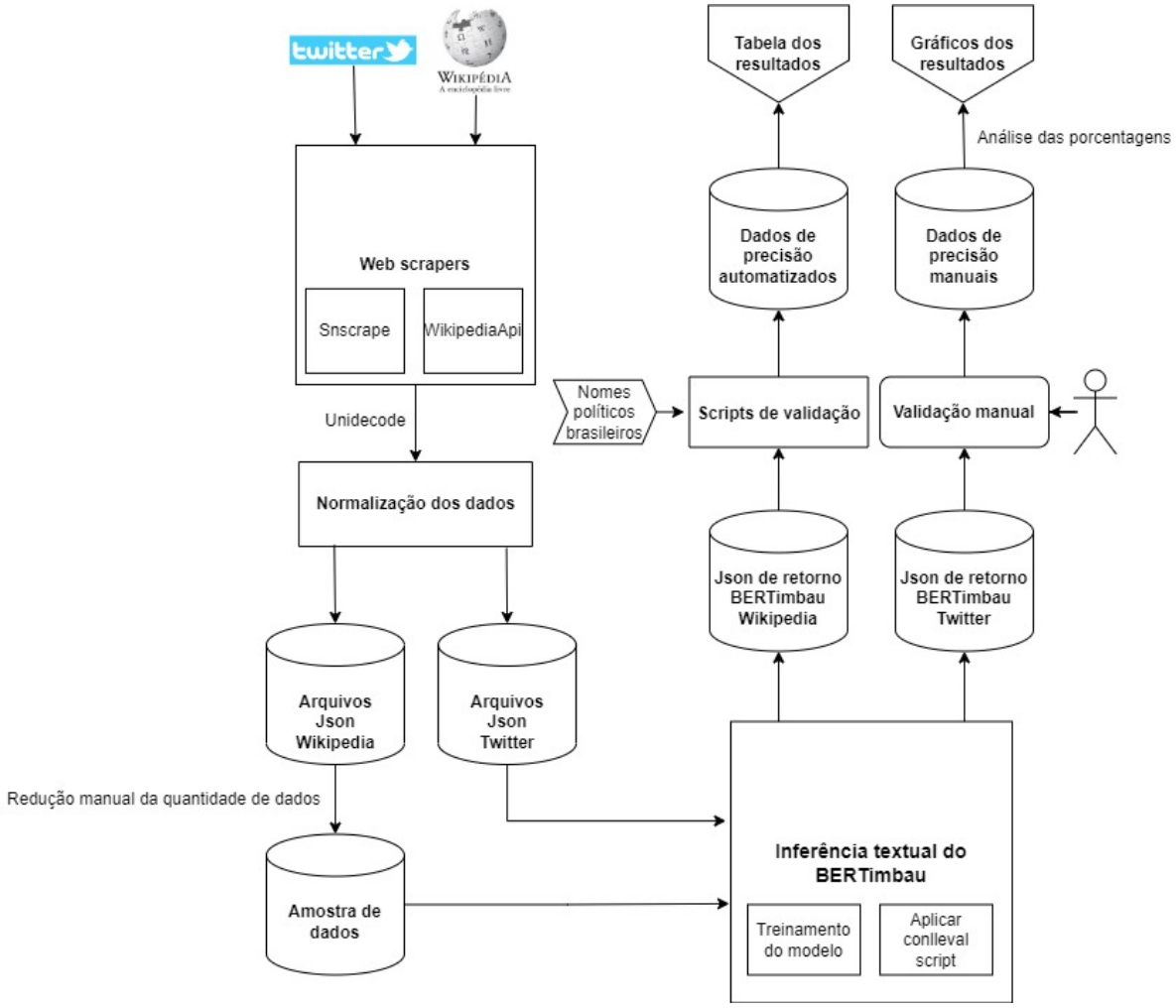


Figura 2 – Visão geral do BERT PolREN

Inicialmente é implementado e configurado dois web scrapings, um utilizando a WikipediaAPI[2] para extração dos dados da Wikipedia e outro utilizando o Tweepy[3] para extração de dados do Twitter, que mais tarde foi atualizado para o Snscrape[4] devido as mudanças de política do Twitter. Foram configurados os dois scrapings para que armazenassem os arquivos extraídos, então foi feito a normalização desses dados para que fiquem limpos e possam ser estruturados em um arquivo *JSON*.

Finalizada a primeira etapa, é feita a configuração do BERTimbau e treinado o modelo BERT-CRF com cenário seletivo de cinco classes: PESSOA/CARGO, ABSTRACCAO/IDEIA, ACONTECIMENTO, LOCAL/HUMANO e FISICO/REGIAO, utilizando o First HAREM dataset para treinamento. Com base no retorno desse treinamento é aplicado o Conlleval[5] Script para verificar a acurácia e precisão geral do modelo treinado.

Na sequência é executado o *script* para realizar a inferência textual NER do

[2]https://pypi.org/project/Wikipedia-API/
[3]https://docs.tweepy.org/en/stable/
[4]https://github.com/JustAnotherArchivist/snscrape
[5]https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt

BERTimbau, sobre os arquivos de retorno dos *Scrapings*, e os resultados dessas inferências são armazenados em *JSONs* de retorno para Wikipedia e Twitter.

Por fim, na abordagem automatizada, é executada uma sequência de *scripts* de validação, e com base no retorno desses *scripts* é possível verificar a precisão e acurácia sobre a fonte de dado do Wikipedia. Na abordagem manual, essa validação foi feita por uma pessoa analisando cada caso, sobre as fontes de dados do Twitter. Lembrando que foi assumido para este trabalho a definição de precisão e acurácia conforme descrito por Baeza-Yates, Ribeiro-Neto et al. (1999).

4.2 Scrapings

4.2.1 Scraping para Wikipédia

Nessa etapa foi utilizada uma *API* da Wikipédia para fazer o mapeamento dos dados que nos interessa, a *api* utilizada foi wikipediaAPI, ela permite passar um contexto e retorna tudo que a Wikipédia tem nesse contexto, utilizando uma função para buscar links relacionados, é feita uma busca em forma de árvore para trazer todo dado possível nesse contexto.

```python
import wikipediaapi
from datetime import datetime
import json

now = datetime.now()
date_time = now.strftime("%d-%m-%Y")
wiki_wiki = wikipediaapi.Wikipedia('pt')

file_name = "politica-wikipedia-0.2.json"
f = open("politica-wikipedia-2.json")
data = json.load(f)


page = wiki_wiki.page('Politicos Brasileiros')
all_data = [
    {
        "data_source": "Wikipedia",
        "date":date_time,
        "title":page.title,
        "body":page.text
    }
]
all_data = all_data + data
def get_references(page):
        links = page.links
        for title, body in links.items():
            all_data.append(
                        {
                    "data_source": "Wikipedia",
                    "date":date_time,
                    "title":title,
                    "body":body.text
                }
            )


new_data = ""
count = 0
for item in all_data:
    if count == 950:
        break
    if count > 700:
        new_data = item['body'] + "  " + new_data
    count = 1 + count

with open(file_name, 'w', encoding="utf-8") as output_file:
    output_file.write('')
    json.dump(new_data, output_file,ensure_ascii=False)
```

Figura 3 – Código para extração de dados da Wikipédia

É possível ver o código utilizado para efetuar a extração dos dados na Figura 3, é criado um algoritmo em python que utiliza a data atual para fazer uma busca na api, o título utilizado para a busca foi “Politicos Brasileiros”, com o retorno desses dados da API, é feito um loop para buscar toda informação relacionada a esse contexto. Por fim é criado um padrão de documento e salvo em formato *JSON*.

4.2.2 Scraping para Twitter

Para a extração dos dados do Twitter é necessário ter acesso à *API* do *Twitter*, para conseguir esse acesso basta seguir as intruções que se encontram no *about-twitter-api Link*[6]. Depois de criar uma conta de desenvolvedor e gerar a *Consumer Key*, *Consumer*

[6]https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api

Secret, *Access Token* e *Access Token Secret* será necessário instalar a biblioteca Tweepy e executar o código em Python da Figura 4.

```python
import tweepy
import json

consumer_key = "Sua consumer key aqui"
consumer_secret = "Sua consumer secret aqui"
access_token = "Seu access token aqui"
access_token_secret = "Seu access token secret aqui"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

tweets = api.search(q="política Brasil", lang="pt", count=100)

tweets_json = []
for tweet in tweets:
    tweets_json.append(tweet._json)

with open("tweets.json", "w") as outfile:
    json.dump(tweets_json, outfile)
```

Figura 4 – Código para extração de dados do Twitter com Tweepy

Segue abaixo os passos explicando o funcionamento do código:

Linhas 1 e 2 - As bibliotecas Tweepy e *JSON* são importadas;

Linhas 4 a 7 - As credenciais de acesso à *API* do *Twitter* são definidas como variáveis (necessário inserir as credenciais geradas na conta de desenvolvedor do Twitter);

Linhas 9 e 10 - A autenticação com a *API* do Twitter é realizada usando as credenciais;

Linha 12 - Um objeto *API* é criado para interagir com a *API* do *Twitter*;

Linha 14 - Uma busca por cem *tweets* sobre política Brasil na linguagem portugês, é realizada usando a função search da biblioteca Tweepy;

Linhas 16 a 18 - Os *tweets* são armazenados em uma lista chamada *tweets json*, um loop é usado para converter cada *tweet* em um formato *JSON*;

Linhas 20 e 21 - O *JSON* é escrito em um arquivo chamado *"tweets.json"*.

Infelizmente a biblioteca Tweepy parou de funcionar em janeiro devido a uma alteração nas políticas do twitter, a base de dados de cem *tweets* gerados antes dessa alteração continuou sendo utilizado, porém para quinhentos e mil *tweets* foi gerado a partir de outra biblioteca, pois ao tentar executar o código da Figura 4 retornou o seguinte erro: *"You currently have access to a subset of Twitter API v2 endpoints and limited v1.1 endpoints (e.g. media post, oauth) only. If you need access to this endpoint, you may need a different access level."* O que significa que a empresa alterou o nível de acesso para utilizar

o Tweepy.

Foi verificado a possibilidade de adiquirir o nível de acesso necessário para rodar o endpoint, porém o valor que o Twitter pedia pelo acesso era inviável, foi decidido então, utilizar a biblioteca Snscrape, que lançou uma atualização na versão de desenvolvimento em janeiro, com ela foi possível fazer a extração de dados sem nem precisar da *API* do Twitter.

A biblioteca precisa da versão do Python 3.8 ou maior, e a versão normal instalada através do *pip3 install snscrape* também não funciona devido a alteração do Twitter, é necessário instalar a versão de desenvolvimento *pip3 install git+https://github.com/JustAnotherArchivist/snscrape.git*, instalamos também a biblioteca pandas para criar o dataFrame e a biblioteca unidecode para limpar os *tweets* removendo os acentos dos dados extraídos.

O Snscrape possui dois métodos para obter *tweets* do Twitter: a interface de linha de comando (CLI) e um Python Wrapper, foi optado pelo segundo método, pois foi compreendido que é mais intuitivo, o código da Figura 5 foi utilizado para fazer a extração dos *tweets*.

```
1  import snscrape.modules.twitter as sntwitter
2  import pandas as pd
3  from unidecode import unidecode
4
5  attributes_container = []
6
7  for i,tweet in enumerate(sntwitter.TwitterSearchScraper('política Brasil lang:pt').get_items()):
8      if i>1000:
9          break
10     clean_content = unidecode(tweet.rawContent)
11     attributes_container.append([clean_content])
12
13 tweets_df = pd.DataFrame(attributes_container, columns=["Tweets"])
14
15 tweets_json = tweets_df.to_json(orient='records')
16
17 with open('tweets1000.json', 'w') as file:
18     file.write(tweets_json)
```

Figura 5 – Código para extração de dados do Twitter com Snscrape

Segue abaixo os passos explicando o funcionamento do código.

Linha 1 - O *snscrape.modules.twitter* é importado como sntwitter e é usado para acessar e coletar *tweets* do Twitter.

Linha 2 - O *pandas* é importado como pd e é usado para criar e manipular *DataFrames*.

Linha 3 - O *unidecode* é importado para remover os acentos dos textos dos *tweets*.

Linha 5 - Uma lista vazia chamada attributes_container é criada. Ela será usada para armazenar os atributos dos *tweets* coletados.

Linhas 7 a 11 - É feito um loop para coletar *tweets* relacionados à política no Brasil e no idioma português. A função *sntwitter.TwitterSearchScraper* é usada para realizar a busca no *Twitter*. O loop é executado até que mil *tweets* sejam coletados ou até que não haja mais *tweets* disponíveis. Para cada *tweet* coletado, o conteúdo é limpo (removendo os acentos) usando a função *unidecode* e, em seguida, adicionado à lista *attributes_container* como uma lista aninhada.

Linha 13 - Após a coleta dos *tweets*, um *DataFrame* chamado *tweets_df* é criado a partir dos dados armazenados em *attributes_container*. O *DataFrame* possui apenas uma coluna chamada *"Tweets"* e contém os *tweets* limpos.

Linha 15 - O *DataFrame tweets_df* é convertido em formato *JSON* usando o método *to_json()*.

Linhas 17 e 18 - O conteúdo *JSON (tweets_json)* é escrito no arquivo. Isso resulta na criação de um arquivo *JSON* de nome *tweets1000.json* contendo os *tweets* coletados no formato desejado.

4.3 Classes para aplicação NER

Para efetuar a inferência textual NER do BERTimbau, foi necessário definir classes que funcionam como rótulos, ou entidades nomeadas, como o próprio nome sugere, reconhecimento de entidades nomeadas. Foram definidas cinco classes com base no Exemplário[7] do Segundo HAREM:

i PESSOA/CARGO: Identifica todo nome de pessoa e todo cargo.

ii ABSTRACCAO/IDEIA: Identifica algumas ideias, sistemas políticos, direitos, entre outros. Como por exemplo "Liberdade", Direito de Expressão", "República", "Democracia".

iii ACONTECIMENTO: Identifica acontecimentos tanto histórico, quanto eventos. Como por exemplo "11 de setembro", "Proclamação da República", "Evento de posse do presidente da república"

iv LOCAL/HUMANO: Identifica um país, local, cidade, entre outros. Como por exemplo "Brasil", "Lisboa", "União Europeia", "Rocinha".

v FISICO/REGIAO: Identifica uma região física. Como por exemplo "Floresta Amazônica", "Deserto do Sahara", "Península Ibérica".

Essas classes foram definidas pensando em criar relações entre elas e analisar contextos, porém essa ideia é para um trabalho futuro. Para o BERT PolREN o foco é

[7] https://www.linguateca.pt/aval_conjunta/HAREM/ExemplarioSegundoHAREM.pdf

na classe PESSOA/CARGO, e analisar o resultado da inferência NER para um cenário Político, verificando dentro de PESSOA os políticos, e dentro de CARGOS os cargos políticos.

4.4 Implementação e configuração do BERTimbau

A implementação e configuração do BERTimbau foi criada com a linguagem Python, na versão 3.6, em um ambiente Conda. Para a criação da inferência textual com NER, utilização do modelo pré-treinado e importação do JSON, precisamos utilizar as bibliotecas *PyTorch 1.1.0*, *seqeval 0.0.12*, *scikit-learn 0.21.2* e *jsonlines 1.2.0*.

A seguir é listado e explicado cada módulo e *script* que foi utilizado para o funcionamento do BERTimbau, modelo do BERT adapatado para português.

i *__init__.py*

É utilizado apenas para pegar o caminho de referência do Python.

ii *dataset.py*

É responsável por receber os dados do *dataset*, para isso utiliza principalmente métodos da própria biblioteca do BERT e além disso utiliza a biblioteca *torch*, muito conhecida para se trabalhar com dados em Python.

iii *eval_tools.py*

Temos aqui a maioria das ferramentas para geração de métricas e a leitura/escrita de arquivos em json. Temos a camada CRF que gera uma lista de listas de IDs de rótulos de tamanho variável. Cada sequência tem um comprimento variável, definido pela máscara de saída do recurso, além da máscara de previsão. Aqui temos uma outra ocorrência de uma biblioteca muito utilizada em Python, o *numpy*.

iv *model.py*

Algumas classes do BERTimbau são instanciadas aqui, como o NER e CRF, todas elas são responsáveis pelos modelos que podemos aplicar caso necessário.

v *postprocessing.py*

São feitas algumas abstrações para serem utilizadas no *model.py*, algumas máscaras para o BERTimbau, podendo cada uma delas ser utilizada para qualquer modelo escolhido.

vi *preprocessing.py*

Utilizando as classes citadas acima, faz o processamento dos dados, cria alguns padrões de objetos para serem lidos pelos métodos de leitura de json, tendo até alguns exemplos de objetos que podemos utilizar.

vii results_*Writer.py*

Após ser feito todo o processamento dos dados, o treinamento em cima das métricas que utilizamos, o *resultsWrite* é responsável por escrever a saída do algoritmo.

viii *run*_Bert_*Harem.py*

Pré-processa um arquivo *JSON* de entrada com treinamento/avaliação bruto usando os exemplos do *preprocessing.py* e formatando para o BERTimbau de acordo com os argumentos fornecidos e com base no *First HAREM Dataset*.

ix run_*Inference.py*

Converte um arquivo txt com conteúdo de inferência textual/parâmetros em um arquivo *JSON* com padrão que será utilizado pelo readExamples. Tem o retorno de um arquivo *JSON* temporário.

x *tag_Encoder.py*

Manipula a criação de *tags*, método do BERTimbau(NER), basicamente faz um *"ENUM"* para alguns padrões que serão utilizados dentro do projeto, além disso tem algumas ferramenta de mapeamento de *tags*.

xi *tokenization.py*

Responsável pela criação de *token*, que é utilizado para dar unicidade a um determinado conjunto de dados que pode ser inserido no projeto.

xii *trainer.py*

Faz o treinamento do conjunto de dados, ou seja, dado um conjunto complexo, ele abstrai uma parte para ser utilizada no treinamento, assim o algoritmo de machine learning consegue saber a precisão das respostas que serão mostradas para o usuário, desta forma temos as métricas do resultado.

xiii *utils.py*

Por último mas não menos importante, os utilitários, tendo alguns métodos como saveModel e loadModel, responsáveis por salvar e carregar os dados que foram utilizados no documento *trainer.py*

O script de entrada para treinamento e avaliação é o *run_Bert_Harem.py*. O run_*Inference.py* pode ser usado para executar inferência textual em novos dados. Todos os outros explicados são módulos. Percebemos que com esses módulos era possível treinar e avaliar modelos BERTimbau com base no conjunto de dados HAREM de quatro maneiras distintas: Cenários totais e seletivos, abordagens baseadas em recursos e de ajuste fino, com e sem CRF.

No final optamos por treinar o modelo com CRF e cenário seletivo de cinco classes. Cada classe é uma entidade nomeada que através de similaridade textual a ferramenta identifica as strings que fazem parte dela.

4.5 Treinamento do modelo

O modelo utilizado foi treinado no cenário seletivo com cinco classes, utilizando o dataset first HAREM para treinamento e o mini HAREM para teste. Como o treinamento foi feito local em um notebook com duas *GPUs*, foi necessário limitar com o comando *CUDAVISIBLEDEVICES=1* antes do *run_Bert_Harem.py* para utilizar apenas a GPU com identificador 1, senão acabava dando erro.

Efetuamos o treinamento rodando o comando *CUDA_VISIBLE_DEVICES=1 python run_bert_harem.py - -bert_model bertimbau_base_bert-crf_selective - -labels_file data/classes.txt - -do_train - -train_file data/FirstHAREM-selective-train.json - -valid_file data/FirstHAREM-selective-dev.json - -num_train_epochs 2 - -per_gpu_train_ batch_size 1 - -gradient_accumulation_steps 16 - -do_eval - -eval_file data/MiniHAREM-selective.json –output_dir output_bert-crf_selective2*. A seguir explicamos cada parte desse comando:

- *-bert_model bertimbau-base_bert-crf_selective*: Especifica o modelo do BERTimbau a ser usado durante a execução, que é o modelo seletivo com crf.

- *-labels_file data/classes.txt*: Especifica o caminho do arquivo de texto que contém as classe a serem usadas durante o treinamento e avaliação.

- *-do_train*: Indica que o treinamento deve ser realizado. Isso instrui o script a realizar o treinamento do modelo.

- *-train_file data/FirstHAREM-selective-train.json*: Especifica o caminho do arquivo de treinamento a ser usado.

- *-valid_file data/FirstHAREM-selective-dev.json*: Especifica o caminho do arquivo de validação a ser usado durante o treinamento.

- *-num_train_epochs* 2: Especifica o número de épocas de treinamento, ou seja, o número de vezes que o conjunto de treinamento completo será percorrido. Neste caso, o treinamento será realizado por 2 épocas.

- *-per_gpu_train_batch_size* 1: Especifica o tamanho do lote de treinamento por GPU. Isso indica quantos exemplos de treinamento serão processados cada vez durante o treinamento em uma única GPU. Neste caso, apenas um exemplo de treinamento será processado cada vez por GPU.

- *-gradient_accumulation_steps 16*: Especifica o número de etapas de acumulação de gradiente. Durante o treinamento, os gradientes calculados em cada etapa são acumulados

e atualizados somente após um determinado número de etapas. Neste caso, os gradientes serão acumulados em grupos de 16 etapas antes de serem atualizados.

- *-do_eval*: Indica que a avaliação deve ser realizada. Isso instrui o script a avaliar o desempenho do modelo em um conjunto de dados separado.

- *-eval_file data/MiniHAREM-selective.json*: Especifica o caminho do arquivo de avaliação a ser usado durante a avaliação.

- *-output_dir output_bert-crf_selective2*: Especifica o diretório de saída onde os resultados e checkpoints do modelo serão armazenados.

Em resumo, o comando executa um *script* Python que treina um modelo BERTimbau cenário seletivo com CRF, com base em um conjunto de dados específico chamado *FirstHAREM-selective-train.json*, e em seguida avalia o modelo em outro conjunto de dados chamado *MiniHAREM-selective.json* e os resultados e checkpoints são salvos no diretório de saída *output_bert-crf_selective2*.

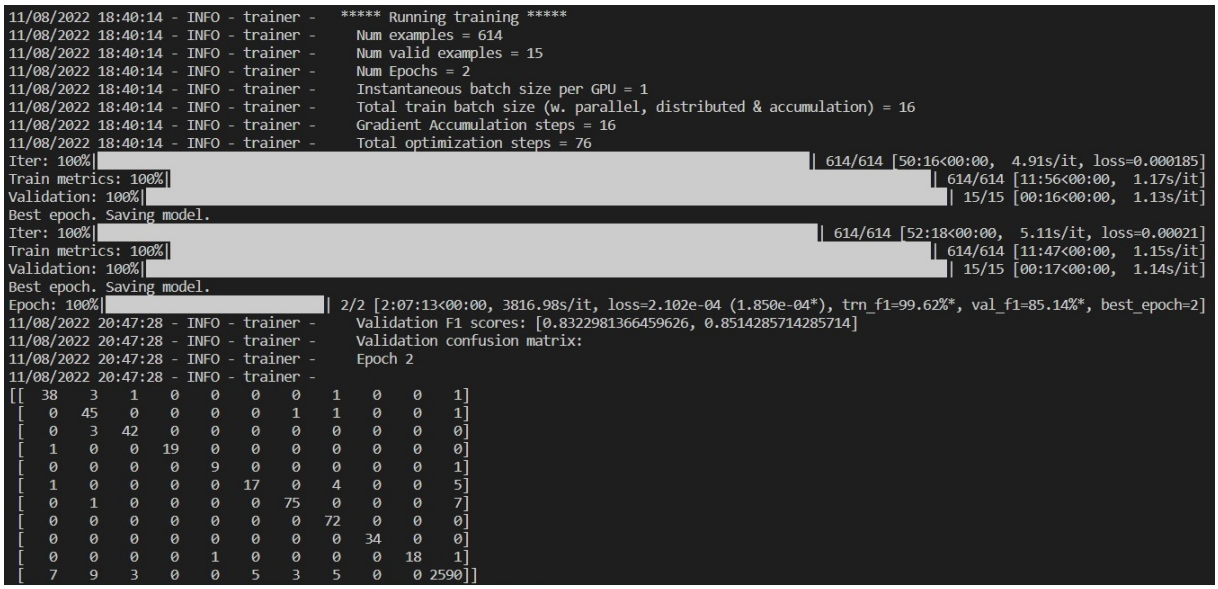


Figura 6 – Treinamento do modelo

Inicialmente houve uma tentativa de treinamento do modelo com trinta e duas épocas e mais de um treinamento por GPU em busca de uma melhor eficiência, mas depois de dois dias rodando com o notebook no limite, a tentativa foi abortada e efetuada novamente com apenas duas épocas e menos iterações, o que levou apenas algumas horas para treinar o modelo, tornando os testes mais viáveis e ainda assim mantendo uma boa acurácia e precisão.

Como é possível ver na Figura 6, após escolher a melhor época o script faz a validação através de uma matriz de confusão que é uma tabela que permite visualizar o desempenho de um algoritmo de classificação, ela é usada para avaliar a precisão de um modelo, onde o objetivo é prever a classe correta para cada exemplo.

A matriz de confusão organiza as previsões feitas pelo modelo em relação às classes reais dos exemplos. Ela é construída com base em quatro cenários:

i Verdadeiro Positivo: representa os exemplos classificados corretamente como positivos. Ou seja, são os casos em que o modelo previu corretamente uma classe positiva.

ii Verdadeiro Negativo: representa os exemplos classificados corretamente como negativos. São os casos em que o modelo previu corretamente uma classe negativa.

iii Falso Positivo: representa os exemplos classificados erroneamente como positivos. Nesses casos, o modelo previu uma classe positiva quando, na verdade, era negativa.

iv Falso Negativo: representa os exemplos classificados erroneamente como negativos. Ocorre quando o modelo previu uma classe negativa quando, na verdade, era positiva.

Na matriz de confusão montada no retorno do script, é compreendido que ela montou o cenário para as classes seletivas que se encontram no data/classes.txt, e fez a matriz colocando as classes reais como linha e as preditas como coluna.

Ao finalizar a validação e avaliação o *script* gera um arquivo chamado *predictions_conll* dentro do diretório criado *output_bert-crf_selective2* e a partir desse arquivo conseguimos aplicar o *conlleval script*.

4.6 Métricas de avaliação

Com base no retorno do treinamento *predictions_conll.txt*, foi possível aplicar o *conlleval*[8] script para computar as métricas de avaliação. O script usa a biblioteca *seqeval*[9] e *conlleval* para calcular métricas de avaliação para recuperação de informações, equivalentes as métricas escolhidas na CoNLL(*Conference on Natural Language Learning*), que são impressas no console. As métricas principais que o *script* calcula e imprime são:

i Precisão: A proporção de predições corretas (verdadeiros positivos) em relação ao total de predições positivas (verdadeiros positivos + falsos positivos). Essa métrica mede a precisão das predições feitas. Utilizaremos a porcentagem dela na classe PESSOA/CARGO para comparação com a análise manual.

ii Cobertura: A proporção de predições corretas (verdadeiros positivos) em relação ao total de exemplos reais positivos (verdadeiros positivos + falsos negativos). Essa métrica mede a capacidade de identificar corretamente os exemplos positivos.

[8]https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt
[9]https://github.com/chakki-works/seqeval

iii Valor F: A média harmônica da precisão e da cobertura. É uma métrica que combina os dois em uma única medida, fornecendo uma visão geral do desempenho geral.

iv Acurácia: A proporção total dos exemplos corretamente classificados em relação ao total de exemplos. Essa métrica mede a taxa geral de acertos.

Abaixo na Figura 7 temos o retorno da aplicação do *conlleval script* no modelo treinado que utilizamos para as inferências textuais.

```
(bert_crf) root@Allison-Note:/home/allison/ner_evaluation# ./conlleval.txt < output_bert-crf_selective2/predictions_conll.txt
processed 64853 tokens with 3018 phrases; found: 3022 phrases; correct: 2441.
accuracy:  97.89%; precision:  80.77%; recall:  80.88%; FB1:  80.83
    PESSOA/CARGO: precision:  84.50%; recall:  81.98%; FB1:  83.22  813
ABSTRACCAO/IDEIA: precision:  61.49%; recall:  75.08%; FB1:  67.61  740
   ACONTECIMENTO: precision:  82.53%; recall:  84.05%; FB1:  83.28  332
    LOCAL/HUMANO: precision:  88.61%; recall:  79.28%; FB1:  83.68  790
   FISICO/REGIAO: precision:  93.66%; recall:  89.04%; FB1:  91.29  347
```

Figura 7 – Resultado da aplicação do *Conlleval Script*

A acurácia geral segundo o script é de 97,89%. Outra informação importante, inclusive para comparação com as análises manuais e automatizada que são feitas nas inferências textuais, é a precisão da classe PESSOA/CARGO que é de 84,50%.

4.7 Inferência textual NER do BERTimbau

Após extrair os dados do Twitter e do Wikipedia com os *scrapings*, é efetuado três inferências textuais do BERTimbau sobre os dados extraídos de cem *tweets*, quinhentos *tweets* e mil *tweets*, e também uma inferência sobre um arquivo maior do Wikipedia que possui 4,4 *megabytes*. Para isso foi necessário rodar quatro vezes o seguinte comando, alterando apenas o *input_file e o output_file: python run_inference.py - -bert_model bertimbau-base_bert-crf_selective/ - -labels_file bertimbau-base_bert-crf_selective/classes.txt - -input_file tweets1000.txt - -output_format json - -output_file 1000tweets - -max_seq_length 512* onde cada parte desse comando é explicado a seguir:

- *-bert_model bertimbau-base_bert-crf_selective/*: Especifica o caminho para o modelo pré-treinado do BERTimbau que será usado para realizar as inferências textuais. Nesse caso, o modelo está localizado no diretório bertimbau-base_bert-crf_selective/.

- *-labels_file bertimbau-base_bert-crf_selective/classes.txt*: Indica o caminho para o arquivo que contém a lista de classes usados pelo modelo. Essas classes são usadas para classificar e rotular os textos de entrada durante as inferências textuais. O arquivo classes.txt contém essas classes.

- *-input_file tweets1000.txt*: Especifica o caminho para o arquivo de entrada que contém os textos nos quais você deseja realizar as inferências textuais. Nesse caso, o arquivo de entrada é chamado *tweets1000.txt*.

- *--output_format json*: Define o formato de saída das inferências textuais como JSON. Isso significa que os resultados serão salvos em um arquivo *JSON*.

- *--output_file 1000tweets*: Especifica o nome do arquivo de saída onde os resultados das inferências textuais serão armazenados. Nesse caso, o arquivo de saída será chamado *1000tweets*.

- *--max_seq_length* 512: Define o comprimento máximo da sequência de *tokens* de entrada. O BERTimbau possui um limite para o tamanho máximo de sequência que pode ser processado. Nesse caso, o limite é definido como 512 *tokens*.

O comando executa o *script run_inference.py* usando um modelo pré-treinado do BERTimbau, um arquivo de entrada contendo os textos a serem classificados, e gera um arquivo de saída no formato *JSON* com os resultados das inferências textuais e o parâmetro max_seq_length define o limite máximo de tokens para os textos de entrada.

Foram necessários cerca de trinta minutos para executar a inferência textual do BERTimbau sobre o arquivo com mil *tweets* e algumas horas pro arquivo do Wikipedia no ambiente computacional disposto, e obtivemos como retorno um JSON com o texto analisado e os resultados divididos em classes como na Figura 8.


```
"entities": [{"class": "PESSOA/CARGO", "start_char": 101,
"end_char": 106, "text": "Lula "}, {"class": "PESSOA/CARGO",
"start_char": 257, "end_char": 264, "text": "Hitler "},
{"class": "PESSOA/CARGO", "start_char": 328, "end_char": 333,
"text": "Lira "}, {"class": "PESSOA/CARGO", "start_char": 342,
"end_char": 347, "text": "Lula "}, {"class": "ACONTECIMENTO",
"start_char": 370, "end_char": 384, "text": "America Latina"},
{"class": "PESSOA/CARGO", "start_char": 561, "end_char": 566,
"text": "Zanin"}, {"class": "ABSTRACCAO/IDEIA", "start_char":
701, "end_char": 703, "text": "CF"}, {"class":
"ABSTRACCAO/IDEIA", "start_char": 802, "end_char": 808,
"text": "Brasil"}, {"class": "PESSOA/CARGO", "start_char":
1039, "end_char": 1045, "text": "Zanin "}, {"class":
"PESSOA/CARGO", "start_char": 1209, "end_char": 1222, "text":
"@LulaOficial "}, {"class": "FISICO/REGIAO", "start_char":
1252, "end_char": 1259, "text": "5 meses"}, {"class":
"ABSTRACCAO/IDEIA", "start_char": 1486, "end_char": 1496,
"text": "GloboNews "}, {"class": "ACONTECIMENTO",
"start_char": 1512, "end_char": 1518, "text": "BRASIL"},
{"class": "PESSOA/CARGO", "start_char": 1675, "end_char":
1685, "text": "Jefferson "}, {"class": "PESSOA/CARGO",
"start_char": 1833, "end_char": 1839, "text": "Zanin "},
{"class": "PESSOA/CARGO", "start_char": 1866, "end_char":
1876, "text": "Presidente"}, {"class": "ACONTECIMENTO",
"start_char": 2040, "end_char": 2046, "text": "Brasil"},
{"class": "ACONTECIMENTO", "start_char": 2168, "end_char":
2174, "text": "BRASIL"}, {"class": "ABSTRACCAO/IDEIA",
"start_char": 2393, "end_char": 2401, "text": "Brasil  "},
{"class": "ABSTRACCAO/IDEIA", "start_char": 2440, "end_char":
2447, "text": "Brasil "}, {"class": "ACONTECIMENTO",
"start_char": 2506, "end_char": 2521, "text": "Brasil Colonia
"}, {"class": "ABSTRACCAO/IDEIA", "start_char": 2541,
"end_char": 2555, "text": "Poder Central "}, {"class":
```


Figura 8 – Parte do resultado da inferência textual do BERTimbau sobre 1000 *tweets*

A inferência textual retorna uma das cinco classes definidas (class), o número da posição no texto do caracter inicial da palavra que encontrou e rotulou pertencente a classe (start_char), o número da posição do caracter final dessa mesma palavra(end_char) e a própria palavra (text).

Foram feitas três análises manuais nos resultados, para obter a precisão do BERTimbau aplicado sobre três tamanhos de dados do Twitter, resultantes do scraping de cem *tweets*, quinhentos *tweets* e mil *tweets*. Foi feita também uma análise automatizada sobre uma fonte de dado maior ainda resultante do scraping do Wikipedia.

4.8 Metodologia

4.8.1 Avaliação Automatizada

Nesta abordagem, foi buscando um método mais eficiente para lidar com grandes volumes de dados. Para esse fim, foi desenvolvida uma série de *scripts* que abordam casos específicos que possam surgir. Essa abordagem é necessária devido à inviabilidade de processar manualmente uma grande quantidade de dados. Além disso, se houver a necessidade de aplicar esse trabalho de forma regular, a execução manual seria impraticável.

Vale ressaltar que, para calcular as porcentagens de precisão, é assumido apenas erros de falso positivo ao inferir palavras para a classe PESSOA/CARGO que não correspondem a pessoas ou cargos. Erros em que a inferência textual não identifica uma pessoa ou cargo essenciais para o cálculo da acurácia geral não são contabilizados, devido à inviabilidade de realizar análises manuais em arquivos extensos. No entanto, ao aplicar o script conlleval no treinamento dos modelos, obtém-se a acurácia geral de PESSOA/CARGO considerando todos os erros, exceto nos cenários políticos.

No contexto desta metodologia de validação automatizada, a validação de políticos americanos ou de qualquer outra parte do mundo não é realizada devido à dificuldade em encontrar um conjunto de dados confiável que contenha todas as informações necessárias para obter uma prova concreta.

Essa abordagem automatizada não apresenta uma precisão de validação tão alta quanto aquela realizada manualmente. Isso ocorre devido à inviabilidade de abranger todos os possíveis cenários que podem ocorrer, tornando essa abordagem essencialmente manual.

Inicialmente nessa avaliação automatizada, para uma melhor compreensão dos dados, foi desenvolvido um script para filtrar os dados de retorno da inferênca com todas as classes, que encontrou 38.749 referências e reter apenas aquelas relacionadas a classe PESSOA/CARGO, conforme exemplificado na Figura 9.

```python
import csv
import json

arquivo_resultado_bert = "/home/vini/ufsc/resultado2.json"
arquivo_candidatos_2022 = "/home/vini/ufsc/consulta_cand_2022_BRASIL.csv"
arquivo = open(arquivo_resultado_bert)
dados = json.load(arquivo)

resultado_bert = []
todos_politicos = []



for item in dados[0]['entities']:
    if "PESSOA/CARGO" in item['class'] :
        resultado_bert.append({
            "nome": item['text'],
            "cargo":  item['class']
        })

with open("acerto_bert_resultado_acerto_cargo.json", 'w', encoding="utf-8") as output_file:
    output_file.write('')
    json.dump(resultado_bert, output_file,ensure_ascii=False)
```

Figura 9 – Script para filtrar apenas PESSOA/CARGO

Linhas 1 e 2 - Importação das bibliotecas *csv e json*;

Linhas 4 e 5 - Definição dos caminhos dos arquivos de entrada (arquivo de resultado do BERTimbau e arquivo *CSV* de candidatos de 2022);

Linhas 6 e 7 - Abertura do arquivo de resultado do BERTimbau e leitura dos dados;

Linhas 9 e 10 - Criação de listas vazias para armazenar os resultados do BERTimbau e todos os políticos encontrados;

Linhas 14 a 19 - Iteração pelos itens na lista de entidades no arquivo de resultado do BERTimbau e extração das informações relevantes (nome e cargo) apenas para as entidades da classe "PESSOA/CARGO";

Linhas 22 a 24 - Abertura de um novo arquivo *JSON* de saída para armazenar os resultados extraídos do BERTimbau;

Esse script da Figura 9, diminuiu de 38.749 referências para 5.506 que são apenas da classe PESSOA/CARGO. Posteriormente, foi desenvolvido outro script para extrair desses 5.506 casos retornados, apenas as pessoas que acertou e foram identificados 4.037 nomes de pessoas com base na utilização da biblioteca Spacy e do modelo pré-treinado "pt-core-news-sm", conforme demonstrado pela Figura 10.

```python
import spacy
import json

nlp = spacy.load("pt_core_news_sm")

arquivo_resultado_bert = "/home/vini/ufsc/tcc-allison-vinicios/acerto_bert_resultado_pessoa-cargo.json"
arquivo = open(arquivo_resultado_bert)
dados = json.load(arquivo)
pessoa_cargo = ""
resultado_bert = []

for item in dados:
    pessoa_cargo += item['nome'] + ","

doc = nlp(pessoa_cargo)

nomes = []
for entidade in doc.ents:
    if entidade.label_ == "PER":
        nomes.append(entidade.text)


with open("/home/vini/ufsc/tcc-allison-vinicios/nome-de-pessoas-encontrados.json", 'w', encoding="utf-8") as output_file:
    output_file.write('')
    json.dump(nomes, output_file, ensure_ascii=False)
```

Figura 10 – Script para validar nome de pessoas

Linha 1 - Importa a biblioteca *Spacy* para realizar o processamento de linguagem natural;

Linha 2 - Importa a biblioteca *JSON* para trabalhar com dados *JSON*;

Linha 5 - Carrega o modelo da biblioteca *Spacy* para processar texto em português;

Linha 7 - Define o caminho do arquivo *JSON* contendo os dados de pessoas e cargos;

Linha 8 - Abre o arquivo *JSON*;

Linha 9 - Carrega os dados do arquivo *JSON* em uma estrutura de dados Python;

Linha 10 - Inicializa uma variável de string vazia para armazenar os nomes de pessoas encontrados;

Linha 11 - Inicializa uma lista vazia para armazenar o resultado processado;

Linhas 13 e 14 - Percorre cada item nos dados do arquivo *JSON* e concatena os nomes de pessoas encontrados na variável pessoa-cargo;

Linha 16 - Processa o texto contido na variável pessoa-cargo utilizando o modelo do *Spacy*;

Linha 18 - Inicializa uma lista vazia para armazenar os nomes de pessoas extraídos;

Linhas 19 a 21 - Percorre todas as entidades (como nomes de pessoas) extraídas do documento processado e adiciona os nomes de pessoas na lista nomes;

Linhas 24 a 26 - Abre um novo arquivo *JSON* para escrever os nomes de pessoas encontrados. Limpa o conteúdo do arquivo e escreve os nomes de pessoas extraídos no arquivo *JSON*;

Para realizar a validação no contexto político brasileiro, dados provenientes do governo brasileiro foram utilizados, sendo obtidos por meio do website Dados Gov [10]. Esses dados possuem um tamanho de 21,33 MB e contêm informações como "DT-GERACAO", "HH-GERACAO", "ANO-ELEICAO", "CD-TIPO-ELEICAO", "NM-TIPO-ELEICAO", entre outros. Para essa validação, apenas os seguintes campos foram considerados: "DS-CARGO", "NM-CANDIDATO", "NM-URNA-CANDIDATO"e "SG-UF". Com base nessas informações de todos os candidatos, uma comparação foi feita com os resultados do modelo BERTimbau, como ilustrado na Figura 11.

[10]https://dados.gov.br

```python
import csv
import json

arquivo_resultado_bert = "/home/vini/ufsc/resultado.json"
arquivo_candidatos_2022 = "/home/vini/ufsc/consulta_cand_2022_BRASIL.csv"
arquivo = open(arquivo_resultado_bert)
dados = json.load(arquivo)

resultado_bert = {}
todos_candidatos = []


for item in dados[0]['entities']:
    resultado_bert.update({
        item['text']: item['class']
    })


with open(arquivo_candidatos_2022, 'r',  encoding='utf-8') as csvfile:
    columns = csv.DictReader(csvfile)
    for item in columns:
        todos_candidatos.append({
            "cargo": item['DS_CARGO'],
            "nome": item['NM_CANDIDATO'],
            "nome_na_urna": item['NM_URNA_CANDIDATO'],
            "estado": item['SG_UF']
        })

acerto_bert = {}
erro_bert = []
for item in todos_candidatos:
    nome = item['nome'].title()
    nome_na_urna = item['nome_na_urna'].title()
    if nome in resultado_bert:
        acerto_bert.update({nome: resultado_bert[nome]})
        continue
    if nome_na_urna in resultado_bert:
        acerto_bert.update({nome: resultado_bert[nome_na_urna]})

teste = set(resultado_bert) - set(acerto_bert)


with open("acerto_bert_resultado.json", 'w', encoding="utf-8") as output_file:
    output_file.write('')
    json.dump(acerto_bert, output_file,ensure_ascii=False)


with open("erro_bert_resultado1.json", 'w', encoding="utf-8") as output_file:
    output_file.write('')
    json.dump(erro_bert, output_file,ensure_ascii=False)
```

Figura 11 – Script de validação

Linhas 1 e 2 - Importa as bibliotecas csv e json, que são usadas para manipular arquivos *CSV e JSON*, respectivamente. Além disso, definem os caminhos para os arquivos de entrada e saída;

Linhas 4 a 7 - O arquivo arquivo-resultado-bert é aberto usando a função *open()* e, em seguida, carregado como um objeto *JSON* usando a função *json.load()*. Os dados são armazenados na variável dados;

Linhas 9 e 10 - É criado um dicionário vazio chamado resultado-bert e uma lista vazia chamada todos-candidatos;

Linhas 13 a 16 - O *script* itera sobre as entidades presentes em dados[0]['entities']. Para cada entidade, o valor do campo 'text' é usado como chave e o valor do campo *'class'* é usado como valor no dicionário resultado-bert;

Linhas 19 a 27 - Em seguida, o *script* abre o arquivo *CSV* arquivo-candidatos-2022 usando *csv.DictReader()*. Isso permite ler o arquivo *CSV* e tratar as colunas como dicionários.O *script* itera sobre as linhas do arquivo *CSV* e para cada linha, extrai os valores das colunas 'DS-CARGO', 'NM-CANDIDATO', 'NM-URNA-CANDIDATO' e 'SG-UF' e os adiciona como um dicionário na lista todos-candidatos;

Linhas 29 e 30 - É criado um dicionário vazio chamado acerto-bert e uma lista vazia chamada erro-bert;

Linhas 31 a 38 - Itera sobre os itens da lista todos-candidatos. Para cada item, os campos 'nome' e 'nome-na-urna' são convertidos para título (iniciando cada palavra com maiúscula). Em seguida, é verificado se o nome ou o nome na urna estão presentes no dicionário resultado-bert. Se estiverem, a chave e o valor correspondentes são adicionados ao dicionário acerto-bert. Caso contrário, o nome é adicionado à lista erro-bert;

Linha 40 - A variável teste é criada como a diferença entre o conjunto de chaves em resultado-bert e o conjunto de chaves em acerto-bert;

Linhas 43 a 45 - Cria um arquivo de saída chamado "acerto-bert-resultado.json"e escreve um objeto *JSON* vazio nele, usando *json.dump()*, seguido pelo dicionário acerto-bert. O argumento *ensure-ascii=False* garante que caracteres não-*ASCII* sejam salvos corretamente no arquivo;

Linhas 48 a 50 - Cria um arquivo de saída chamado "erro-bert-resultado1.json"e escreve um objeto *JSON* vazio nele, usando *json.dump()*, seguido pela lista erro-bert. O argumento *ensure-ascii=False* garante que caracteres não-*ASCII* sejam salvos corretamente no arquivo;

Ao analisar os cargos identificados, verificou-se que o modelo conseguiu identificar todos os níveis da política brasileira, incluindo os cargos de "Presidente", "Deputado Estadual", "Deputado Federal", "Governador", "Senador", "Vice-Governador"e "Vice-Presidente".

Isso demonstra que o modelo possui uma boa compreensão da hierarquia política brasileira. É importante ressaltar que os cargos mencionados acima são únicos, porém, dentro desses cargos, foi utilizado um script conforme ilustrado na Figura 12.

```
import csv
import json

arquivo_resultado_bert = "/home/vini/ufsc/resultado2.json"
arquivo_candidatos_2022 = "/home/vini/ufsc/consulta_cand_2022_BRASIL.csv"
arquivo = open(arquivo_resultado_bert)
dados = json.load(arquivo)

resultado_bert = []
todos_politicos = []

cargos = ["Presidente", "DEPUTADO ESTADUAL", "DEPUTADO FEDERAL", "GOVERNADOR", "SENADOR", "VICE-GOVERNADOR", "VICE-PRESIDENTE", "VICEPRESIDENTE"]

for item in dados[0]['entities']:
    if "PESSOA/CARGO" in item['class'] and item['text'] in cargos:
        resultado_bert.append({
            "nome": item['text'],
            "cargo":  item['class']
        })
```

Figura 12 – Script para validar cargos

Linhas 1 e 2 - Importa as bibliotecas *csv e json*, que são usadas para manipular arquivos *CSV e JSON*, respectivamente. Além disso, definem os caminhos para os arquivos de entrada e saída;

Linhas 4 a 7 - O arquivo arquivo-resultado-bert é aberto usando a função *open()* e, em seguida, carregado como um objeto *JSON* usando a função *json.load()*. Os dados são armazenados na variável dados;

Linhas 9 e 10 - É criado um dicionário vazio chamado resultado-bert e uma lista vazia chamada todos-candidatos;

Linhas 12 e 13 - É criado um um dicionário com todos os cargos encontrados;

Linhas 15 a 21 - Itera sobre os itens da lista de entidades para ser encontrado apenas os candidatos mencionados na variavel cargos;

Todas as validações mencionadas foram realizadas utilizando um conjunto de dados fornecido pelo próprio governo, como mencionado anteriormente, contendo informações sobre todos os candidatos a cargos políticos em 2022. O arquivo correspondente está disponível nos anexos. É importante ressaltar que os dados obtidos do governo não estão normalizados, o que significa que pode haver uma margem de erro nos valores mencionados anteriormente.

4.8.2 Avaliação Manual

A análise da classe PESSOA/CARGO foi feita considerando cinco cenários: PESSOA, POLÍTICO BRASIL, POLÍTICO MUNDO, CARGO POLÍTICO e CARGO.

i Para PESSOA são contabilizados todo nome de pessoa e nome de usuário do twitter como acerto e somados aos acertos de POLÍTICO MUNDO e POLÍTICO BRASIL,

por também serem pessoas.

ii Para POLÍTICO BRASIL são contabilizados e considerados acertos políticos, ex-políticos, ministros, ex-ministros, cientistas políticos, pessoas famosas de partidos e movimentos políticos, nomes de usuários do twitter como LulaOficial, jairbolsonaro, Haddad_Fernando, e algumas tags com nomes de políticos, como exceção teve quarenta e duas aparições do nome Zanin em mil *tweets* e doze em quinhetos *tweets* que também foram consideradas acertos em POLÍTICO BRASIL pelo cenário de possível indicação ao STF.

iii Para POLÍTICO MUNDO são contabilizados todos os políticos de fora do Brasil como acerto e somados aos acertos de POLÍTICO BRASIL.

iv Para CARGO POLÍTICO são contabilizados todos os cargos políticos existentes no Brasil como acerto.

v Para CARGO são contabilizados todos os cargos como acerto e somados aos acertos de CARGO POLÍTICO.

São utilizadas as seguintes fórmulas para encontrar a precisão de cada um dos cenários:

PP = P/(P+EP)

PPB = PB/(P+EP)

PPM = PM/(P+EP)

PC = C/(C+EC)

PCP = CP/(C+EC)

Onde PP é a precisão de PESSOA, P é o número de acertos de PESSOA, EP é o número de erros de PESSOA, PPB é a precisão de POLÍTICO BRASIL, PB é o número de acertos de POLÍTICO BRASIL, PPM é a precisão de POLÍTICO MUNDO, PM é o número de acertos de POLÍTICO MUNDO, PC é a precisão de CARGO, C é o número de acertos de CARGO, EC é o número de erros de CARGO, PCP é a precisão de CARGO POLÍTICO e CP é o número de acertos de CARGO POLÍTICO.

4.9 Resultados

4.9.1 Twitter

Foi efetuada a análise sobre o arquivo de retorno da inferência textual do BERTimbau em cem *tweets* e foram encontrados 81 acertos para POLÍTICO BRASIL, 84 acertos para POLÍTICO MUNDO, 102 acertos para PESSOA, 10 acertos para CARGO

POLÍTICO e 12 acertos para CARGO. Foram encontrados também, 13 erros discrepantes, que foram divididos proporcionalmente entre PESSOA (12 erros) e CARGO (1 erro) para que fosse possível efetuar o cálculo da precisão em cada cenário. É entendido como erro discrepante quando a palavra rotulada pela inferência textual na classe PESSOA/CARGO, não é considerada nem uma pessoa e nem um cargo. A seguir, na Figura 13, estão listados esses 13 erros e quantas vezes apareceram, para que seja possível em um trabalho futuro efetuar ajustes nos treinamentos em busca de uma melhor precisão.

DEUS 4 vezes	FROUXAS ARMADAS 1 vez	DA ACERVO 1 vez	FI 1 vez
@ 4 vezes	CONSTA TUDO 1 vez	SR 1 vez	

Figura 13 – Lista dos 13 erros da inferência textual em 100 *tweets*

Aplicando os valores encontrados na análise em cem *tweets*, nas fórmulas, temos PP = 102/(102+12) = 89,47%, PPM = 84/(102+12) = 73,68% e PPB = 81/(102+12) = 71,05% para a classe PESSOA, como é possível ver na Figura 14. Já para classe CARGO temos PC = 12/(12+1) = 92,30% e PCP = 10/(12+1) = 76,92%, como mostra a Figura 15.

Figura 14 – Precisão PESSOA 100 *tweets*

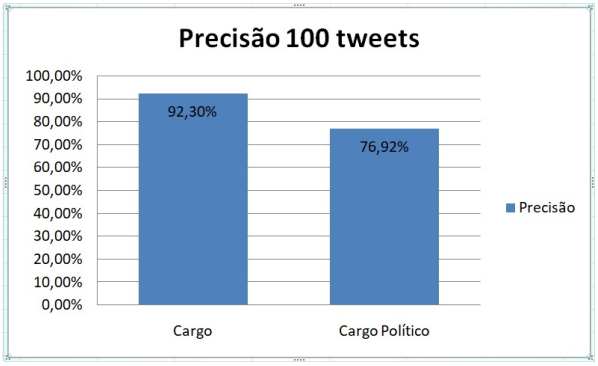


Figura 15 – Precisão CARGO 100 *tweets*

Foi efetuada a análise sobre o arquivo de retorno da inferência textual do BERTimbau em quinhentos *tweets* e foram encontrados 292 acertos para POLÍTICO BRASIL, 303 acertos para POLÍTICO MUNDO, 373 acertos para PESSOA, 20 acertos para CARGO POLÍTICO e 24 acertos para CARGO. Foram encontrados também, 27 erros discrepantes que dividimos proporcionalmente entre PESSOA (25 erros) e CARGO (2 erros), na Figura 16 estão listados esses 27 erros e quantas vezes apareceram.

@ 9 vezes	CUIABA 1 vez	CAPETA 1 vez	FAROFINH0 1 vez
DEUS 3 vezes	GOIAS 1 vez	BARCELONA 1 vez	CHOLDRAIGNOBIL 1 vez
PL 2 vezes	K 1 vez	CELTA DE VIGO 1 vez	CRUSOE 1 vez
JOVEMPANNEWS 1 vez	BOZO 1 vez	VALLADOLID 1 vez	BRASILEIRAO 1 vez

Figura 16 – Lista dos 27 erros da inferência textual em 500 *tweets*

Aplicando os valores encontrados na análise em quinhentos *tweets*, nas fórmulas, temos PP = 373/(373+25) = 93,72%, PPM = 303/(373+25) = 76,13% e PPB = 292/(373+25) = 73,36% para a classe PESSOA, como é possível ver na Figura 17. Para classe CARGO foram aplicadas as fórmulas, e coincidentemente foi obtido os mesmos valores de precisão que em cem *tweets*, temos PC = 24/(24+2) = 92,30% e PCP = 20/(24+2) = 76,92% como mostra a Figura 18.

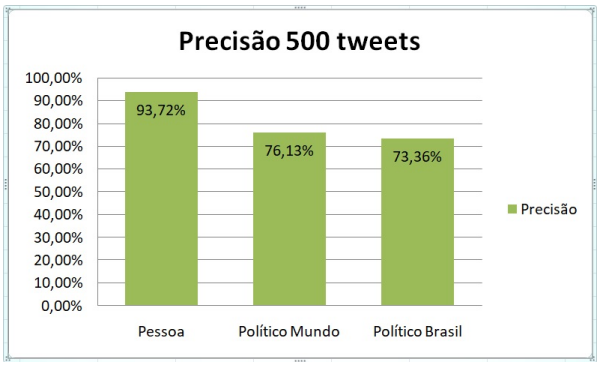


Figura 17 – Precisão PESSOA 500 *tweets*

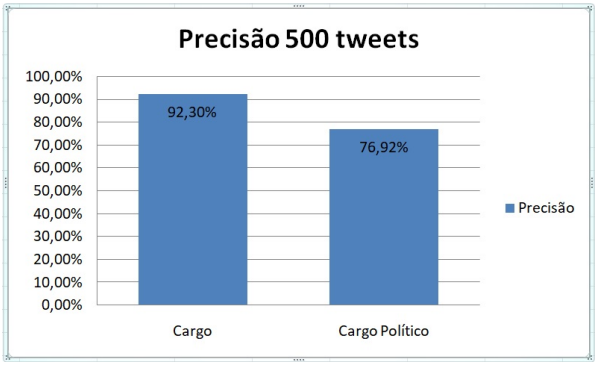


Figura 18 – Precisão CARGO 500 *tweets*

Foi efetuada a análise sobre o arquivo de retorno da inferência textual do BERTimbau em mil *tweets* e foram encontrados 636 acertos para POLÍTICO BRASIL, 666 acertos para POLÍTICO MUNDO, 805 acertos para PESSOA, 48 acertos para CARGO POLÍTICO e 59 acertos para CARGO. Foram encontrados também 66 erros discrepantes que foram divididos proporcionalmente entre PESSOA (62 erros) e CARGO (4 erros), a seguir na Figura 19 estão listados esses 66 erros e quantas vezes apareceram.

DEUS 12 vezes	JOVEM PAN NEWS 1 vez	GREG NEWS 1 vez	AM 1 vez
@ 9 vezes	CARAVANAMCCOY 1 vez	CRAMULHAO 1 vez	EX 1 vez
SENHOR 3 vezes	QUESTIONADOR190 1 vez	BRASILIA 1 vez	B.O. 1 vez
FLAMENGO 3 vezes	OCORNETEIROFLA 1 vez	JP NEWS 1 vez	FOLLA 1 vez
ATLETICO 3 vezes	VAN LIBERDADE 1 vez	GEMEOS 1 vez	PAPAI 1 vez
DIABO 2 vezes	DINOSSAURO 1 vez	FLUMINENSE 1 vez	BLAZE 1 vez
PALMEIRAS 2 vezes	DEMONIO 1 vez	ITUANO 1 vez	VASCO 1 vez
CORITIBA 2 vezes	INTERNACIONAL 1 vez	CRICIUMA 1 vez	MENINO 1 vez
BOGOTA 1 vez	BRAZUELA 1 vez	FILHO ROJAS 1 vez	IRMAO 1 vez
GENOCIDA 1 vez	BOTAFOGO 1 vez		

Figura 19 – Lista dos 66 erros da inferência textual em 1000 *tweets*

Aplicando os valores encontrados na análise em mil *tweets*, nas fórmulas, temos PP = 805/(805+62) = 92,85%, PPM = 666/(805+62) = 76,80% e PPB = 636/(805+62) = 73,35% para a classe PESSOA, como é visto na Figura 20. Já para classe CARGO temos PC = 59/(59+4) = 93,65% e PCP = 48/(59+4) = 76,20%, como mostra a Figura 21.

Figura 20 – Precisão PESSOA 1000 *tweets*

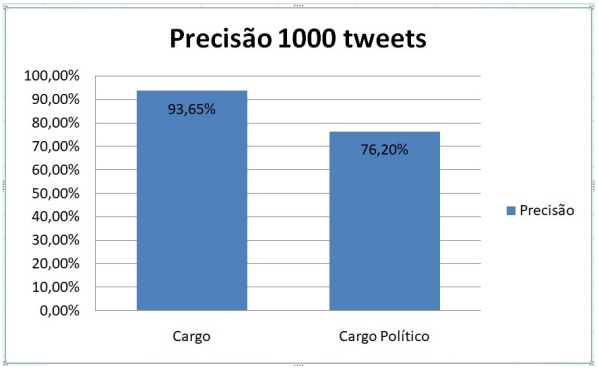
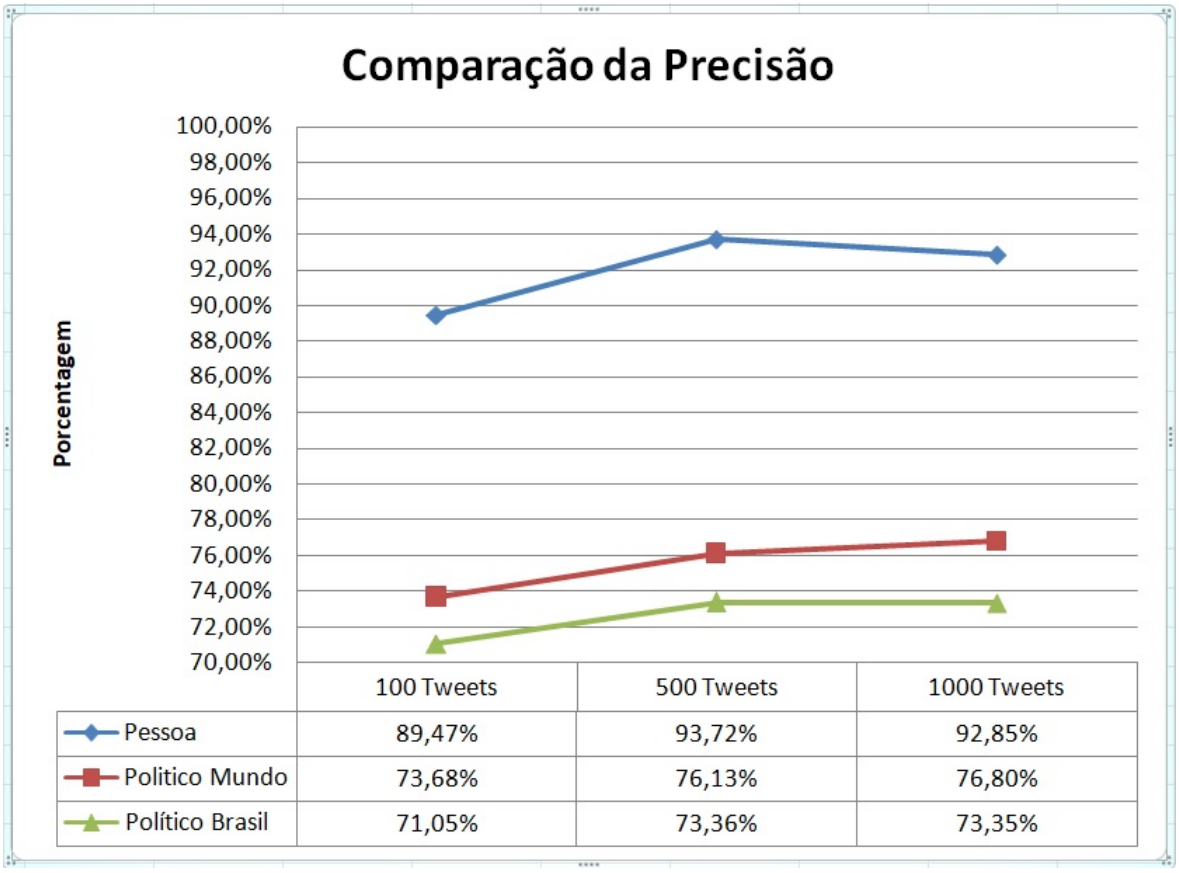


Figura 21 – Precisão CARGO 1000 *tweets*

Com base no resultados encontrados, foram criados gráficos de comparação entre os três tamanhos de dados, para tentar encontrar padrões e descobrir os melhores cenários.



	100 Tweets	500 Tweets	1000 Tweets
Pessoa	89,47%	93,72%	92,85%
Politico Mundo	73,68%	76,13%	76,80%
Político Brasil	71,05%	73,36%	73,35%

Figura 22 – Comparação do valor de precisão dos três tamanhos de dados para PESSOA

É possível ver na Figura 22, que os valores de precisão para a classe PESSOA mudaram muito pouco em diferentes tamanhos de dados e seguiram alguns padrões. O primeiro padrão era esperado, que é a melhor precisão ser de pessoa, depois político mundo e depois político Brasil. Isso ocorre pois é considerado que um político no Brasil é um político no mundo e que ambos são pessoas. Porém algo interessante é que a diferença de precisão entre eles nos três tamanhos de dados ficou muito próxima, oscilando entre pessoa e político mundo de 15,79% a 17,59% e entre político mundo e político Brasil de 2,63% a 3,45%.

Outro padrão identificado, é que a diferença maior entre os valores de precisão foi no salto de cem para quinhentos *tweets*, de quinhetos para mil *tweets* os valores de precisão praticamente se mantiveram, mudando menos de 1%. É possível reparar também, que o maior aumento no valor da precisão que ocorreu nos três tamanhos de dados, foi de pollítico mundo para pessoa, e de político Brasil para político mundo esse valor de precisão mudou pouco. Com isso podemos afirmar que a inferência textual NER encontrou mais pessoas que não eram políticas do que políticos de fora do Brasil.

Durante a análise manual foi percebido que a data que o dado foi exportado tem influência nos resultados, porém diferente do que era imaginado não teve grande impacto nas porcentagens de precisão. A extração dos dados de cem *tweets* foi feita em dezembro

de 2022, já a de quinhentos e mil *tweets*, foram feitas em junho de 2023, e nessas, mais da metade das referências eram Lula, Lira, Bolsonaro e Zanin, devido às notícias da semana que envolviam esses políticos e estavam sendo debatidas no Twitter. Foi compreendido que devido a esse influência da data de exportação dos dados de cem *tweets*, os três cenários para classe PESSOA têm pior precisão em relação aos dados de quinhentos e mil *tweets*.

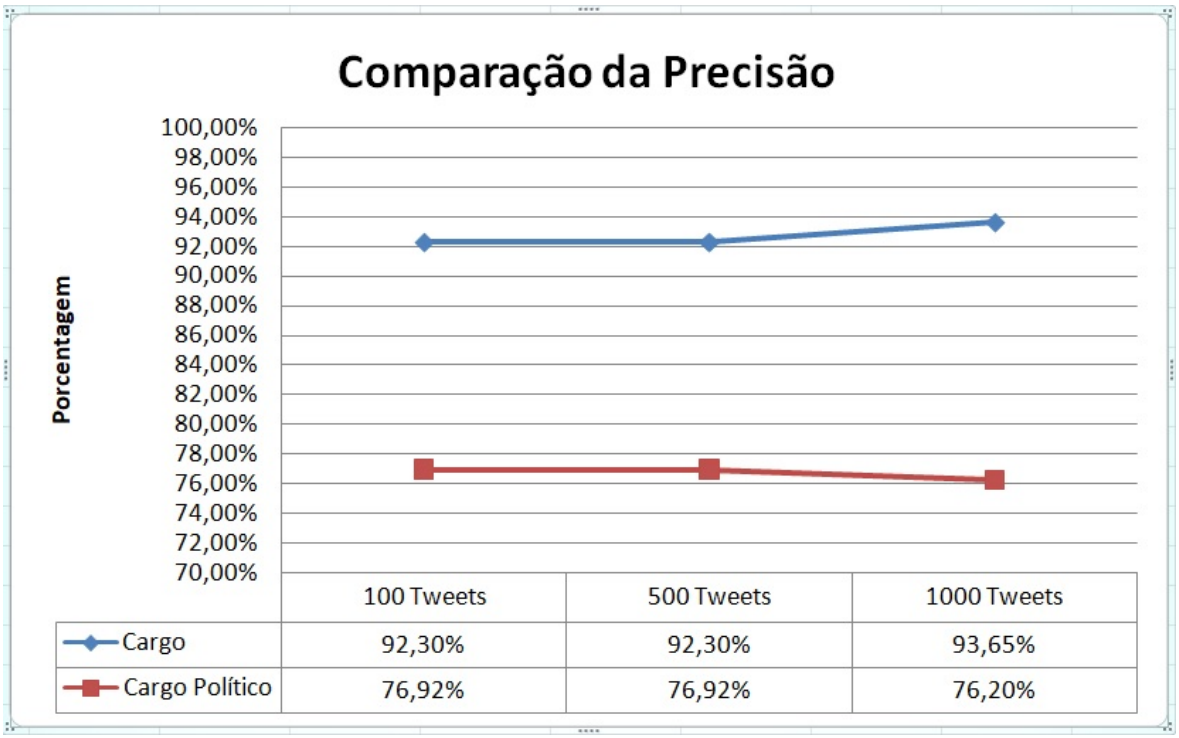


Figura 23 – Comparação do valor de precisão dos três tamanhos de dados para CARGO

É posível ver na Figura 23 que para classe CARGO praticamente não houve mudança de precisão, o que inviabiliza encontrar algum padrão de comportamento nas mudanças de tamanho de dados, apenas que Cargo sempre vai ter uma precisão melhor que Cargo Político, pois todo cargo político é considerado um cargo também.

4.9.2 Wikipedia

Após a aplicação do modelo BERTimbau nos dados do Wikipedia, um arquivo resultante com tamanho de 4,4 MB é gerado, contendo todas as classes utilizadas, que são: PESSOA/CARGO, ABSTRAÇÃO/IDEIA, ACONTECIMENTO, LOCAL/HUMANO e FÍSICO/REGIÃO. Nesse conjunto, um total de 38.749 classes foram encontradas.

Para validar a precisão desse modelo, um *script* em Python foi desenvolvido para facilitar o trabalho de validação, separando cada categoria utilizada no BERTimbau, selecionando o retorno apenas da categoria PESSOA/CARGO, o número de classes encontradas foi reduzido de 38.749 para 5.506. Com esse conjunto de dados mais refinado, a análise dos resultados pôde ser iniciada .

A saída desse script foi então utilizada para uma validação, a fim de determinar a precisão real do BERTimbau. No entanto, um problema foi identificado: a interpretação do BERTimbau considera a categoria (PESSOA/CARGO) independentemente da origem ser brasileira ou não, enquanto os dados utilizados na validação se referem apenas a políticos brasileiros, uma vez que não temos informações sobre toda a política mundial. Portanto, foram obtidos dois resultados de precisão, um específico para o Brasil e outro para o cenário político geral.

Inicialmente, foi realizada a análise da correspondência entre um politico e o seu cargo, identificando 41 classes em que tanto o nome quanto o cargo da pessoa foram identificados corretamente, como ilustrado na Figura 24:

Nome	Cargo
CARLOS ALBERTO OLIVEIRA DO COUTO JUNIOR	1-SUPLENTE
DULCINEIA COELHO DA SILVA	DEPUTADO ESTADUAL
PAULO BATISTA DOS REIS	DEPUTADO ESTADUAL
JOSE GOMES DA SILVA FILHO	DEPUTADO ESTADUAL
AIRTON RODRIGUES DE OLIVEIRA	DEPUTADO FEDERAL
MARIA VALDINEZ DE SOUZA	DEPUTADO ESTADUAL
SEVERINO RAMOS DE SANTANA	DEPUTADO ESTADUAL
IRANILDO DANTAS	DEPUTADO ESTADUAL
SEVERINO RAMOS DE VASCONCELOS	DEPUTADO FEDERAL
LEANDRO FERREIRA FELICIANO	DEPUTADO ESTADUAL
DHEURY MEDEIROS CARVALHO	DEPUTADO ESTADUAL
VANILSON ALVES DA SILVA	DEPUTADO ESTADUAL
SEVERINO DOS RAMOS SANTOS SILVA	DEPUTADO FEDERAL

Figura 24 – Amostra de nomes e cargos encontrados

Após a validação do *script* anterior, foi feita nova validação alterando os dados de políticos brasileiros, para nomes em geral. Foi identificado que dos 5.506 registros encontrados, 4.037 correspondiam a nomes de indivíduos, e dentro dessa categoria, constatou-se que 41 deles eram políticos brasileiros exclusivamente, como mostra a Figura 25.

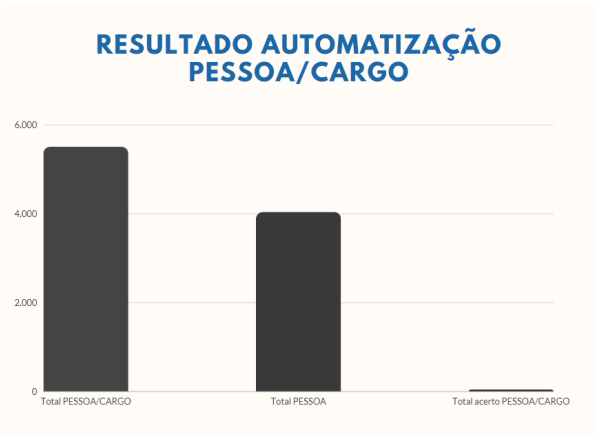


Figura 25 – Acerto de PESSOA/CARGO

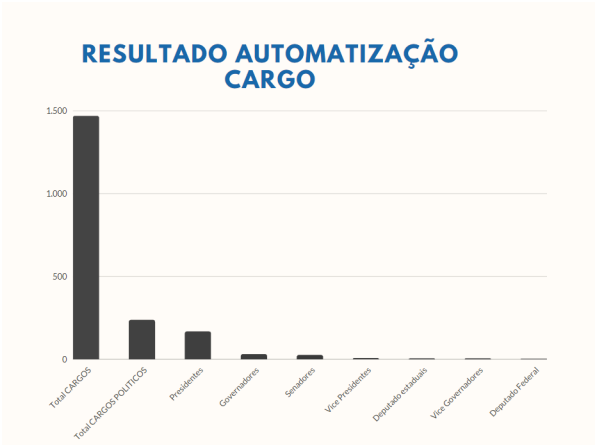


Figura 26 – Acerto de CARGO

Com base nas análises realizadas, pode-se concluir que o modelo BERTimbau apresentou uma precisão satisfatória. Considerando que os parâmetros não especificaram explicitamente a identificação exclusiva de políticos, a precisão alcançada foi de 73,32%. Com um total de 4.037 nomes encontrados e apenas 41 casos únicos de políticos, a precisão em relação políticos brasileiros é de 1,015%, esse valor é baixo pois o *script* funciona com casos únicos para cada político, portanto mesmo que nomes como Lula e Bolsonaro sejam citados diversas vezes, ele entende apenas como um acerto para cada, reduzindo drasticamente a porcentagem.

A Figura 26 ilustra visualmente os valores encontrado no retorno do *script* para validar cargos. Ao analisarmos especificamente os cargos, observamos uma precisão ainda maior. Dos 1.469 cargos identificados, constatou-se que 237 deles eram cargos políticos, sendo eles 167 presidentes, 4 deputados estaduais, 1 deputado federal, 30 governadores, 25 senadores, 4 vice-governadores e 6 vice-presidentes. Portanto, a precisão nesse contexto foi de 16,13%.

5 Considerações Finais

5.1 Conclusão

Após uma análise abrangente, conclui-se que é possível empregar o BERT PolREN para atingir uma precisão relativamente alta em um cenário político, quando se utilizou a validação manual, foram obtidos números muito próximos a precisão geral encontrada no *script conlleval* para classe PESSOA/CARGO da inferência NER do BERTimbau. Porém quando o arquivo é muito grande e precisamos aplicar a validação automatizada essa precisão cai drasticamente para o cenário político. É viável realizar adaptações no processo automatizado visando aprimorar a precisão e eficácia geral, utilizando conjuntos de dados mais abrangentes que englobem políticos de todo o mundo, tornando possível a aplicação do BERT PolREN para qualquer tamanho de arquivo sem a redução da precisão. Durante as análises, foram identificados alguns casos de falsos positivos, que interferem tanto na abordagem manual quanto automatizada, é recomendável considerar um dataset diferente do First HAREM para treinamento do modelo, buscando eliminar tais falsos positivos.

Com base no que foi disposto no parágrafo anterior, é possível afirmar que o objetivo geral do trabalho foi alcançado, conseguimos verificar a precisão e acurácia do modelo e identificar qual metodologia usar em cada cenário. Foi identificado que o BERT PolREN tem uma precisão satisfatória quando aplicado para um cenário político, tornando possível sua utilização para verificar fontes de dados e identificar relações com algum político ou cargo político específico.

5.2 Trabalhos Futuros

O BERTimbau utiliza a bilioteca Conda que é muito robusta, dificultando treinamento de modelos e aplicação de melhores inferências NER, no nosso ambiente computacional, sendo assim, encontramos alguns problemas que ficaram inviáveis de serem solucionados apenas nesse trabalho, que são:

i O poder computacional foi considerado o maior problema do trabalho, por isso seria interessante a aplicação do mesmo modelo em cloud, assim tendo um poder computacional maior que possa suportar um arquivo complexo.

ii Efetuar o treinamento de mais modelos com base no *Second HAREM Dataset*, ou qualquer outro dataset de processamento de linguagem natural, aplicando NER;

iii Ajustar no *dataset* de treinamento os cenário encontrados nas listas de erro da Inferência NER do BERTimbau;

iv Fazer a correlação EM do BERTimbau entre cargos e pessoa, para que não haja um trabalho excessivo na validação dos dados, tendo que, sem essa correlação é necessária duas validações para cada sentença analisada pelo BERTimbau, uma para cargo e outra para pessoa e além disso a precisão para política iria melhorar.

v Fazer correlações entre as cinco classes, para ter um contexto melhor e melhorar as análises.

vi Propõe-se aprimorar os roteiros de validação automatizada, incorporando um conjunto de dados global de políticos.

Referências

ACADEMY, D. S. *Deep Learning Book, Cap. 76.* 2022. Recuperado em 03 de Julho de 2023. Disponível em: <https://www.deeplearningbook.com.br/o-que-e-bert-bidirectional-encoder-representations-from-transformers/>.

BAEZA-YATES, R.; RIBEIRO-NETO, B. et al. *Modern information retrieval.* [S.l.]: ACM press New York, 1999. v. 463.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit.* [S.l.]: "O'Reilly Media, Inc.", 2009.

BORDIGNON, F. L. Técnicas inteligentes para análise de agrupamento de dados. 2010.

BRANCO, A. M. et al. Ferramenta para coleta e comparação de dados de publicações acadêmicas dos professores com o currículo lattes. Florianópolis, SC., 2018.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

ELINT. *Uma Visão Geral sobre Named Entity Recognition (NER).* 2021. Recuperado em 03 de Julho de 2023. Disponível em: <https://medium.com/elinttech/uma-vis%C3%A3o-geral-sobre-named-entity-recognition-ner-4dc4e3b5e37a>.

FALCÃO, L. C. de J.; LOPES, B.; SOUZA, R. R. Absorção das tarefas de processamento de linguagem natural (nlp) pela ciência da informação (ci): uma revisão da literatura para tangibilização do uso de nlp pela ci. *Em Questão*, p. 13–34, 2022.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* 2019.

KENNY, C. *What is web scraping?* 2020. Recuperado em 03 de Julho de 2023. Disponível em: <https://www.zyte.com/learn/what-is-web-scraping/>.

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

MARTINS, G. D. *Named Entity Recognition com BERT nos datasets HAREM.* 2021. Recuperado em 03 de Julho de 2023. Disponível em: <https://medium.com/@gdutramartins/named-entity-reconigtion-com-bert-nos-datasets-harem-f81bc38e6971>.

MATHIAS, G. N. et al. qfex: um crawler para busca e extração de questionários de pesquisa em documentos html. Florianópolis, SC., 2017.

MOTA, C.; SANTOS, D. *Avaliação Conjunta de Reconhecimento de Entidades Mencionadas em Português (AVAL-CONJUNTA-HAREM).* 2008. Recuperado em 03 de Julho de 2023. Disponível em: <https://www.linguateca.pt/aval_conjunta/HAREM/harem_ing.html>.

SOUZA, F. C. *Harem-preprocessing. v1.0.* 2023. Recuperado em 03 de Julho de 2023. Disponível em: <https://github.com/fabiocapsouza/harem_preprocessing>.

SUTTON, C.; MCCALLUM, A. et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, Now Publishers, Inc., v. 4, n. 4, p. 267–373, 2012.

Apêndice A - *Scripts* e arquivos de retorno

Os *scripts* utilizados para validação manual e automatizada, o *conlleval script* e os retornos desses *scripts* e das inferências do BERTimbau se encontram no endereço https://github.com/vinicamel/tcc-allison-vinicios.

Apêndice B - Artigo

Verificação de precisão e acurácia do BERTimbau na inferência textual NER para o cenário político

Allison de Souza[1], Vinicios Camello[1], Carina Friedrich Dorneles[1]

[1]Departamento de Informática e Estatística – INE – Universidade Federal de Santa Catarina, s/nº, 88040-900, Trindade – Florianópolis – SC– Brasil

{allisondesouza27@gmail.com, viniciosCamello3@gmail.com, carina.dorneles@ufsc.br}

Resumo. *Sobre o cenário político, para a identificação de qual pessoa, onde e quais cargos políticos ocupam, são necessários filtros que auxiliem a absorver informações reais. Este artigo propõe análises do BERTimbau aplicado sobre Scrapings e o parâmetro "Política Brasil". Objetiva-se verificar a precisão da inferência textual NER num cenário seletivo dividido em cinco classes e comparando essa precisão entre fontes de dados do Twitter e Wikipedia. Para validar essa abordagem, foram adotados os métodos de validação manual e automatizada. Com embasamento nas Análises e nas precisões encontradas, o BERT PolREN se mostrou eficiente para resolução do problema encontrado e pode ser utilizado como ferramenta confiável.*

Abstract. *On the political scene, in order to identify which person, where and what political positions they hold, filters are needed to help absorb real information. This paper proposes analyses of BERTimbau applied on Scrapings and the parameter "Politics Brazil". It aims to verify the accuracy of NER textual inference in a selective scenario divided into five classes and comparing this accuracy between Twitter and Wikipedia data sources. To validate this approach, manual and automated validation methods were adopted. Based on the Analyses and the accuracies found, BERT PolREN proved efficient for solving the problem encountered and can be used as a reliable tool.*

1. Introdução

A técnica de processamento de linguagem natural, denominada Reconhecimento de Entidades Nomeadas provém do termo Named Entity Recognition (NER), é muito utilizada para identificar e classificar entidades específicas. Estas entidades podem se apresentar na forma de texto, tais como nomes de pessoas, organizações, locais, datas, valores monetários, entre outros. Esta abordagem é amplamente utilizada em diversas áreas, como extração de informações, análise de sentimentos e opiniões, busca e recuperação de informações, análise de dados e mineração de texto, chatbots e assistentes virtuais, e classificação de documentos.

Para seu uso e pleno funcionamento e eficácia, é necessário que seja realizado o treinamento de um modelo de NER, e este pode apresentar algumas dificuldades. Existe a possibilidade da criação manual destes dados, no entanto, este pode ser um processo trabalhoso, considerando a variação de tipos de entidades, a presença de nomes

ambíguos ou incomuns, além da limitação de generalização para diferentes domínios. Por exemplo, mesmo um modelo treinado em notícias pode ter dificuldades para reconhecer entidades em textos políticos. Ou seja, para que haja eficácia, é necessário dispor de um conjunto de dados rotulados, nos quais as entidades nomeadas estejam devidamente anotadas.

Como um dos modelos que podem auxiliar nesse processo de criação, o modelo do Google BERT tem apresentado desempenho excepcional em diversas tarefas de processamento de linguagem natural, incluindo classificação de texto, reconhecimento de entidades nomeadas, análise de sentimento, entre outras. Porém tem-se o contratempo de que a versão original do BERT não é habilitada para a língua portuguesa, assim, torna-se necessário a utilização do BERTimbau, uma versão do BERT em português. A partir do uso do BERTimbau, é possível a análise dos dados coletados das fontes Wikipedia e Twitter, no contexto político.

Considerando estas informações, o presente artigo tem como objetivo a aplicação do algoritmo do BERTimbau para extração de informações, definidas como valiosas, de forma resumida. Para avaliar a eficácia do BERT, foram adotadas duas abordagens: uma análise manual dos dados de forma individualizada, a fim de medir a precisão com maior eficácia, e uma abordagem automatizada, utilizando scripts que utilizam dados públicos do governo contendo nomes de políticos para avaliar sua precisão. Por fim, objetiva-se que a metodologia proporcione a avaliação completa da eficácia do BERTimbau, utilizado para a análise de dados no contexto político, proporcionando insights valiosos para compreender e extrair informações relevantes dessas fontes de dados.

2. Revisão bibliográfica

Para o entendimento dos conceitos que norteiam o modelo do Google BERTimbau, são necessárias introduções a definições teóricas sobre Scraping, Named Entity Recognition (NER), *Natural Language Processing* (NLP), *Conditional Random Fields* (CRF), o funcionamento e aplicações do BERT e a descrição do HAREM Dataset, serão descritas.

2.1. Scrapings

O processo denominado Scrapings é, de forma geral, baseado no processo de coletar dados de um *site* de forma automatizada. Esse processo, pode envolver a extração de informações estruturadas ou não, provenientes e disponíveis em páginas da web. Como forma de conceituação, o *Web scraping* é o processo de coleta de dados da web de maneira automatizada. Também pode ser denominado como extração de dados da web.

Entre os principais uso da *web scraping*, estão inclusos os processos de monitoramento de preços e inteligência, monitoramento de notícias, geração de *leads* e pesquisa de mercado, entre muitos outros. Essa extração de dados da web é utilizada por pessoas e empresas que desejam ter acesso a vasta quantidade de dados da web disponíveis publicamente para decisões mais assertivas (KENNY, 2020). Para a realização deste artigo, o método utilizado para a coletar dados e execução da inferência textual NER e suas análises. A terminologia utilizada também será referenciada como, NER.

2.2. Named Entity Recognition (NER)

O *Named Entity Recognition (NER),* também é conhecido como Reconhecimento de Entidade Nomeada (REN) em português. Essa é a nomenclatura utilizada neste artigo como forma de integralizar as informações.

A técnica de NLP é denominada por identificar e categorizar informações-chave (entidades) em textos. Uma entidade pode ser descrita como sendo qualquer palavra ou série de palavras que se referem ao mesmo tema ou assunto. Desta forma, cada entidade detectada é classificada em uma categoria pré-determinada. Como exemplo, desta categorização, as entidades podem ser nomes de pessoas, organizações, locais, horários, quantidades, valores, entre outros (ELINT, 2021).

Ainda segundo Elint (2021), para o treinamento em algoritmos de NER, se faz necessária uma grande quantidade de dados classificados manualmente. Consequentemente, uma abordagem semi-controlada é recomendada para reduzir o esse esforço manual. Para o caso deste esforço, é utilizado um dataset conhecido como Fisrt HAREM Dataset.

2.3. Natural Language Processing (NLP)

O Natural Language Processing (NLP), conhecido em português como Processamento de Linguagem Natural (PLN), é um subcampo da inteligência artificial (IA) criado para auxiliar robôsna compreensão do processo orgânico da comunicação humana. Esse processamento envolve desde a compreensão de palavras únicas até a avaliação de frases e textos completos.

De acordo com Jurafsky e Martin (2019), o NLP é um tópico em constante mudança e que tem ganhado notoriedade devido à necessidade de automatizar operações relacionadas à linguagem e à quantidade de dados disponíveis de forma *online*. Por exemplo, algumas das aplicações do NLP são a tradução de máquina, análise de sentimento em redes sociais e resumo de textos. Para a realização dessas tarefas, são utilizados métodos como análise morfológica, sintética e semântica, além de algoritmos de aprendizado de máquina. Esses algoritmos são treinados com grandes quantidades de dados para encontrar conexões e padrões entre palavras e frases, permitindo que a máquina compreenda a linguagem natural de forma eficaz. Lidar com a ambiguidade e variedade da linguagem humana é um dos principais desafios enfrentados pelo NLP, como afirmado por Bird, Klein e Loper (2009).

Importante ressaltar que o significado das palavras pode mudar dependendo do contexto em que são usadas, e os falantes nem sempre seguem as regras gramaticais. Apesar dessas dificuldades, o processo teve avanços consideráveis, principalmente devido a novas estratégias e algoritmos, bem como à maior quantidade e qualidade de dados disponíveis. Para mecanismos de busca, o NLP não é algo novo, no entanto, o BERT, que utiliza treinamento bidirecional e é o foco da pesquisa neste trabalho, representa um grande avanço no processamento de linguagem natural.

2.4. Conditional Random Fields (CRF)

A *Conditional Random Fields* (CRF) é um *framework* utilizado para construir modelos probabilísticos utilizado em tarefas de processamento de linguagem natural e visão

computacional para segmentar e rotular dados sequenciais. A CRF oferece diversas vantagens em relação a modelos ocultos de Markov e gramáticas estocásticas para tarefas, incluindo a capacidade de relaxar as fortes suposições de independência.

O CRF é amplamente aplicado em diversas tarefas, como reconhecimento de entidades nomeadas, marcação de partes do discurso, segmentação de sequências de fala, entre outros. É possível modelar as relações contextuais entre as observações de entrada e os rótulos de saída por meio de probabilidades condicionais (SUTTON; MCCALLUM et al., 2012). No processo de treinamento do CRF, o algoritmo de otimização comumente utilizado é o algoritmo de *forward-backward*, responsável por ajustar os parâmetros com base nos dados e anotações das sequências rotuladas. Sua capacidade de modelar dependências de longo alcance em sequências o torna uma escolha poderosa para analisar dados de sequência.

2.5. Funcionamento e aplicações do BERT

Em termos de buscador na internet, a Google é certamente o principal utilizado. Atualmente, ele passou por atualizações em seu algoritmo, denominado BERT, que é um algoritmo de *Deep Learning* para NLP, que ajuda computadores a entender a linguagem como humanos. Desta forma, o BERT é capaz de auxiliar o Google a entender melhor o significado das palavras nas consultas no mecanismo de busca. Essa atualização também melhorou a capacidade de traduzir a linguagem humana para os computadores.

Segundo Devlin et al. (2018), ao contrário de modelos recentes de representação de linguagem, o BERT foi projetado para pré-treinar as representações profundas bidirecionais a partir de um texto não rotulado, condicionando conjuntamente o contexto à esquerda e à direita em todas as camadas analisadas. Como resultado, esse tipo de modelo pré-treinado BERT pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração. Estes podem atuar para uma ampla gama de tarefas, como resposta a perguntas e inferência de linguagem, sem modificações substanciais na arquitetura específica da tarefa BERT

Um dos principais benefícios do algoritmo se dá pela capacidade de reconhecer o contexto bidirecional de palavras em uma frase, o que lhe permite entender a relação entre uma palavra e seus predecessores e sucessores. O BERT tem sido a base para o desenvolvimento de outros modelos de PLN mais avançados, como o RoBERTa, o ALBERT e especificamente no BERTimbau, adaptado para língua portuguesa. Neste artigo, o BERTimbau, foi utilizado e seu funcionamento será detalhado.

2.6. Descrição do HAREM Dataset

Uma coleção de anotações linguísticas criada para fins de pesquisa e desenvolvimento no campo de processamento de linguagem natural (PLN) e aprendizado de máquina é denominado de conjunto de dados Harem. Essa denominaçao "Harem" é um acrônimo para "Harmonia, Anotação e Recuperação de Entidades Mencionadas".

Em específico, o *Harem dataset* foi desenvolvido objetivando o fornecimento de um conjunto de dados, anotados manualmente, que represente certa variedade de tipos de entidades e relações presentes na língua portuguesa. Essas anotações são particularmente relevantes para tarefas de extração de informações, análise de sentimentos, sumarização de texto, tradução automática e outras aplicações relacionadas à compreensão de linguagem natural.

O conjunto de dados Harem contém textos em português provenientes de diversas fontes, como notícias, artigos acadêmicos, blogs e páginas da web. Os documentos

As anotações no conjunto de dados Harem são realizadas por anotadores humanos especializados, que seguem diretrizes criteriosas e claras para garantir a consistência e qualidade das anotações. São anotadas informações sobre entidades, nomes de pessoas, organizações, locais e datas etc. Esse conjunto de dados passa por revisões e verificações para garantir a precisão e utilidade. O conjunto tem sido amplamente utilizado como referência em tarefas de PLN e aprendizado de máquina para a língua portuguesa.

3. BERT PolREN

O modelo BERT PolREN, ou seja, BERT para Política com Reconhecimento de Entidade Nomeada, utiliza como base o BERTimbau (produzido por Fábio Souza, Rodrigo Nogueira e Roberto Lotufo) e faz uso das suas fontes para implementação. Na metodologia desse artigo, será realizada a aplicação do algoritmo do BERTimbau para extração de informações, definidas como valiosas, de forma resumida. Para avaliar a eficácia do BERT, foram adotadas duas abordagens: uma análise manual dos dados de forma individualizada e uma abordagem automatizada. Por fim são apresentadas as análises utilizando gráficos para demonstrar de maneira visual os resultados encontrados.

3.1. Visão geral do modelo

Para que ocorra o processamento BERT PolREN devem ser realizadas as seguintes etapas, conforme Figura 1: (i) implementação e configuração dos Scrapers; (ii) Normalização e extração dos dados; (iii) Treinamento do modelo e aplicação do conlleval script; (iv) Aplicação das amostras do Wikipedia e dados do Twitter no algoritmo BERTimbau; (v) Validação manual e automatizada dos arquivos de retorno; (vi) Análise dos dados de precisão; (vii) Criação de gráficos e tabelas com os resultados. Nestas etapas são realizados os seguintes processos:

- Inicialmente são implementados e configurados dois *web scrapings*, um utilizando a WikipediaAPI (para dados da Wikipedia) e outro utilizando o Tweepy (para dados do Twitter).
- Realização da normalização destes dados, para que fiquem limpos e possam ser estruturados em um arquivo tipo JSON.
- Finalizada a primeira etapa, é feita a configuração do BERTimbau e treinado o modelo BERT-CRF utilizando o First HAREM dataset, com cenário seletivo de cinco classes:

- PESSOA/CARGO;
- ABSTRACCAO/IDEIA;
- ACONTECIMENTO;
- LOCAL/HUMANO;
- FISICO/REGIAO.

- Com base no retorno do treinamento realizado é aplicado o Conlleval Script para verificar a acurácia e precisão geral do modelo treinado.
- Na sequência o script é executado para realizar a inferência textual NER do BERTimbau, sobre os arquivos de retorno dos *Scrapings*, e os resultados dessas inferências são armazenados em JSONs de retorno para Wikipedia e Twitter.
- Finalizando o processo, na abordagem automatizada, é executada uma sequência de *scripts* de validação, e com base no retorno é possível verificar a precisão e acurácia sobre a fonte de dado do Wikipedia. Na abordagem manual, essa validação é feita por uma pessoa analisando cada caso, sobre as fontes de dados do Twitter.

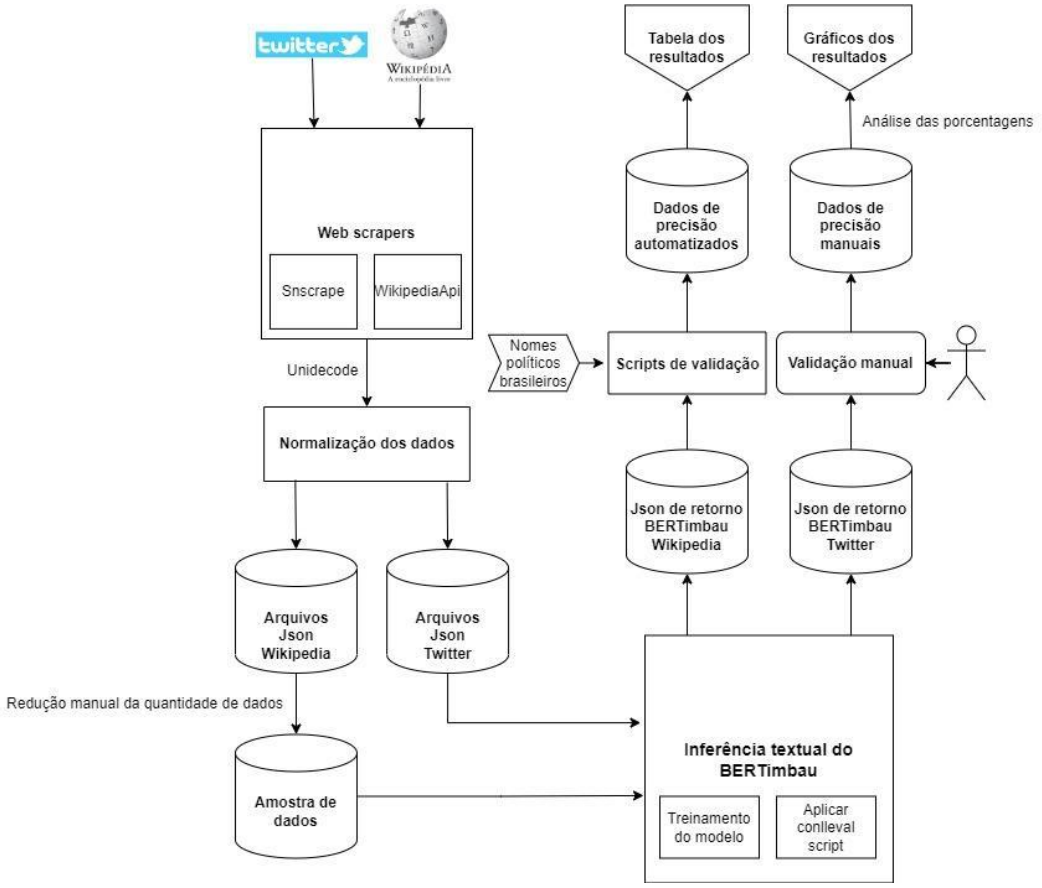


Figura 1. Visão geral do fluxograma BERT PoIREN

3.2. Scrapings

Os Scrapings foram divididos conforme a sua utilização, pelo melhor entendimento.

3.2.1. Scrapings para Wikipédia

Nessa a realização desta etapa foi utilizada a WikipediaAPI para realizar o mapeamento dos dados de interesse do estudo. Essa API permite que um contexto seja inserido e tudo que a Wikipédia tiver relacionado ao contexto, retornará ao usuário por meio de uma função para buscar *links* relacionados, dado todo possível nesse contexto.

Para este artigo, o título utilizado para a busca foi “Politicos Brasileiros”, e a partir do retorno desses dados da API, foi feito um *loop* para buscar toda a informação relacionada a esse contexto. O algoritmo foi criado em python e utilizando a data atual para fazer a busca na API, após a obtenção das informações, foi criado um padrão de documento e salvo em formato JSON.

3.2.2. Scrapings para Twitter

Com similaridade com a utilização para a Wikipédia, para a extração dos dados do Twitter foi necessário ter acesso à API do Twitter. As instruções para este acesso estão disponíveis na internet, por meio do link: “*https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-apiabout-twitter-api*”. Depois de criar uma conta de desenvolvedor e gerar a *Consumer Key*, *Consumer Secret, Access Token* e *Access Token Secret* é necessário instalar também a biblioteca *Tweepy* e então, executar o código em Python.

Infelizmente a biblioteca *Tweepy* parou de funcionar em janeiro de 2023, devido a alterações nas políticas do twitter. A base de dados da qual já se provinha, cerca de cem tweets, gerados antes da alteração continuaram sendo utilizado. Entretanto, para cerca de quinhentos mil tweets foram gerados a partir de outra biblioteca, pois mensagens de erro eram geradas. A mensagem exibida indicou que a empresa alterou o nível de acesso para utilizar o *Tweepy*.

Em função da inviabilidade financeira de aquisição da biblioteca do Twitter a biblioteca *Snscrape* foi utilizada pois com ela foi possível fazer a extração de dados sem a API do Twitter. A biblioteca precisa da versão do Python 3.8 ou maior, e a versão de desenvolvimento do pip3 *install snscrape*. O Snscrape possui dois métodos para obter tweets do Twitter: a interface de linha de comando (CLI) e um Python Wrapper.

3.3. Classes para aplicação NER

Para realizar a inferência textual NER do BERTimbau, foram definidas classes que funcionam como rótulos, ou entidades nomeadas. Com base no Exemplário segundo HAREM, foram definidas cinco classes, listadas a seguir:

- PESSOA/CARGO: Identifica todo nome de pessoa e todo cargo.
- ABSTRACCAO/IDEIA: Identifica algumas ideias, sistemas políticos, direitos, entre outros. Como por exemplo "Liberdade", Direito de Expressão".
- ACONTECIMENTO: Identifica acontecimentos tanto histórico, quanto eventos. Como por exemplo "11 de setembro", "Proclamação da República".
- LOCAL/HUMANO: Identifica um país, local, cidade, entre outros. Como por exemplo "Brasil", "Lisboa", "União Europeia", "Rocinha".

- FISICO/REGIAO: Identifica uma região física. Como por exemplo: "Deserto do Sahara", "Floresta Amazônica", “Península Ibérica".

Para este artigo e utilizando o BERT PolREN o foco se dá sobre a classe PESSOA/CARGO. Para analisar o resultado da inferência NER para um cenário Político, é verificado o filtro “PESSOA” para os políticos, e dentro de “CARGOS” os cargos políticos.

3.4. Implementação e configuração do BERTimbau

A implementação e a configuração do BERTimbau foram criadas com a linguagem Python, na versão 3.6, em ambiente Conda. Para a criação da inferência textual com NER, utilização do modelo pré-treinado e importação do JSON, foi necessário a utilização das bibliotecas PyTorch 1.1.0, seqeval 0.0.12, scikit-learn 0.21.2 e jsonlines 1.2.0. para visualização e explicação de cada um ds módulos e *scrip* sugere-se a leitura de Souza e Camello (2023).

Para entrada para treinamento e avaliação, o *script* utilizado foi o *runBertHarem.py.* O *runInference.py* pode ser usado para executar inferência textual em novos dados. Com o uso desses módulos foi possível treinar e avaliar modelos BERTimbau com base no conjunto de dados HAREM de quatro maneiras distintas: Cenários totais e seletivos, abordagens baseadas em recursos e de ajuste fino, com e sem CRF.

Ao final desta análise, e para este artigo, optou-se pelo treinamento do modelo com CRF e cenário seletivo de cinco classes. Cada classe corresponde a uma entidade nomeada que através de similaridade textual a ferramenta identifica as *strings* que fazem parte dela.

3.4. Treinamento do modelo desenvolvido

O modelo utilizado neste artigo no cenário seletivo com cinco classes, utilizou o *dataset first* HAREM para treinamento e o mini HAREM para teste. Como a realização do treinamento foi realizada em um notebook com duas GPUs, foi necessário limitar com o comando *CUDAVISIBLEDEVICES=1* antes do *runBertHarem.py*, caso fosse utilizado apenas a GPU com identificador 1, ocorriam erros.

Após decidir sobre a melhor época, o *script* realizou a validação através de uma matriz de confusão. Esta é uma tabela que permite visualizar o desempenho de um algoritmo de classificação, usada para avaliar a precisão de um modelo, onde o objetivo seja o de prever a classe correta para cada exemplo. A matriz de confusão organiza as previsões feitas pelo modelo em relação às classes reais dos exemplos e é construída com base em quatro cenários:

1. Verdadeiro Positivo: exemplos classificados corretamente como positivos, ou seja, onde o modelo previu corretamente uma classe positiva.
2. Verdadeiro Negativo: exemplos classificados corretamente como negativos, ou seja, onde o modelo previu corretamente uma classe negativa.
3. Positivo: nesses casos, o modelo previu uma classe positiva quando, na verdade, era negativa.

4. Falso Negativo: quando o modelo previu uma classe negativa quando, na verdade, era positiva.

Ao finalizar a validação e avaliação o *script* gera um arquivo chamado *predictionsconll* dentro do diretório criado *outputbert-crfselective2* e a partir desse arquivo conseguimos aplicar o *conlleval script.*

3.4. Métricas de avaliação

Com base no retorno do treinamento *predictionsconll.txt,* o *conlleval script* foi utilizado para computar as métricas de avaliação. Para calcular métricas de avaliação para recuperação de informações equivalentes as métricas escolhidas na CoNLL(Conference on Natural Language Learning), o *script* usa a biblioteca *seqeval* e *conlleval*. As métricas principais que o script calcula e imprime são:

1. Precisão: A proporção de predições corretas (verdadeiros positivos) em relação ao total de predições positivas (verdadeiros positivos + falsos positivos). A porcentagem sobre aa classe PESSOA/CARGO foi utilizada para comparação com a análise manual.
2. Cobertura: A proporção de predições corretas (verdadeiros positivos) em relação ao total de exemplos reais positivos (verdadeiros positivos + falsos negativos).
3. Valor F: A média harmônica da Precisão e da Cobertura. É uma métrica que combina os dois em uma única medida, fornecendo uma visão geral do desempenho geral.
4. Acurácia: A proporção de exemplos corretamente classificados em relação ao total de exemplos. Essa métrica mede a taxa geral de acertos.

A acurácia geral segundo o script é de 97,89%. A título de conhecimento, as análises manuais e automatizada que são feitas nas inferências textuais, apresentam precisão geral da classe PESSOA/CARGO de 84,50%.

3.7. Inferência textual NER do BERTimbau

A última etapa da verificação foi realizada após a extração dos dados do Twitter e do Wikipedia utilizando os *scrapings.* São efetuadas três inferências textuais do BERTimbau sobre os dados extraídos de 100 (cem) tweets, (500) quinhentos tweets e (1.000) mil tweets, e uma inferência sobre um arquivo maior do Wikipedia que possui 4,4 megabytes. Para essa análise, o comando foi realizado quatro vezes, com alteração do inputfile e do outputfile. Para mais detalhes sobre a rotina utilizada, recomenda-se a leitura de Souza e Camello (2023).

Para a realização deste artigo foram realizadas três análises manuais nos resultados, para obter a precisão do BERTimbau, com as quantidades definidas de *scraping*. Também foi realizada uma análise automatizada sobre uma fonte de dado maior, resultante do *scraping* do Wikipedia. A partir destes resultados, a validação foi realizada.

4. Metodologia desenvolvida

A metodologia utilizada para desenvolvimento deste artigo, foi dividida entre manual e automatizada. Cada uma destas será descrita brevemente neste capítulo.

4.1. Avaliação manual

Para a análise manual da classe PESSOA/CARGO foram considerados cinco cenários: PESSOA, POLÍTICO BRASIL, POLÍTICO MUNDO, CARGO POLÍTICO e CARGO.

1. PESSOA: contabilização de todo nome de pessoa e nome de usuário do twitter como acerto e somados aos acertos de POLÍTICO MUNDO e POLÍTICO BRASIL.
2. POLÍTICO BRASIL: contabilização e consideração acertos políticos de tofas as hierarquias, ex-políticos, pessoas famosas de partidos e movimentos políticos, nomes de usuários do twitter como LulaOficial, jairbolsonaro, HaddadFernando, e algumas tags com nomes de políticos. Também foram consideradas acertos em POLÍTICO BRASIL pelo cenário de possível indicação ao STF.
3. POLÍTICO MUNDO: contabilizados todos os políticos de fora do Brasil como acerto e somados aos acertos de POLÍTICO BRASIL.
4. CARGO POLÍTICO: contabilizados todos os cargos políticos existentes no Brasil como acerto.
5. CARGO: contabilizados todos os cargos como acerto e somados aos acertos de CARGO POLÍTICO.

Para a precisão de cada um dos cenários citados, foram utilizadas as seguintes fórmulas:

$$PP = P/(P+EP)$$

$$PPB = PB/(P+EP)$$

$$PPM = PM/(P+EP)$$

$$PC = C/(C+EC)$$

$$PCP = CP/(C+EC)$$

Onde, PP é a precisão de PESSOA, P é o número de acertos de PESSOA, EP é o número de erros de PESSOA, PPB é a precisão de POLÍTICO BRASIL, PB é o número de acertos de POLÍTICO BRASIL, PPM é a precisão de POLÍTICO MUNDO, PM é o número de acertos de POLÍTICO MUNDO, PC é a precisão de CARGO, C é o número de acertos de CARGO, EC é o número de erros de CARGO, PCP é a precisão de CARGO POLÍTICO e CP é o número de acertos de CARGO POLÍTICO.

4.2. Avaliação automatizada

Para o desenvolvimento desta avaliação, buscou-se um método eficiente para trabalhar com grandes volumes de dados. Assim, desenvolveu-se uma série de *scripts* que abordam casos específicos, necessários devido à inviabilidade de processamento manual de grande quantidade de dados. Para o cálculo das porcentagens de precisão, considerou-se apenas os erros de falso positivo ao inferir palavras para a classe

PESSOA/CARGO, que não correspondem a pessoas ou cargos. Erros em que a inferência textual não identificou uma pessoa ou cargo essenciais para o cálculo da acurácia geral, não foram contabilizados. No entanto, a aplicação do *script conlleval* no treinamento dos modelos, resultou na acurácia geral de PESSOA/CARGO e desta vez, considerando todos os erros, exceto nos cenários políticos.

Essa abordagem automatizada não apresenta uma precisão de validação tão alta quanto aquela realizada manualmente. Isso ocorre devido à inviabilidade de abranger todos os possíveis cenários que podem ocorrer, tornando essa abordagem essencialmente manual. Assim, a validação de políticos americanos ou de qualquer outra parte do mundo, não foi realizada.

Para a validação no contexto político brasileiro, os dados foram obtidos por meio do Site Oficial. Os dados possuem um tamanho de 21,33 MB e para essa validação, apenas os seguintes campos foram considerados: "DS-CARGO", "NM-CANDIDATO", "NM-URNA-CANDIDATO" e "SG-UF". A partir dos dados obtidos, uma comparação foi realizada com os resultados do modelo BERTimbau. Verificou-se que o modelo identificou todos os níveis hierárquicos da política brasileira, ou seja, os cargos de Presidente, Vice-Presidente, Senador, Deputado Federal, Deputado Estadual, Governador e Vice-Governador. Entretanto, ressalta-se que os dados obtidos do Governo não estão normalizados, o que pode significar uma margem de erro nos valores obtidos.

5. Resultados obtidos

Os resultados obtidos neste artigo são apresentados conforme a sua origem, sendo está a Wikipédia ou o Twitter.

5.1. Wikipédia

A partir da aplicação do modelo BERTimbau nos dados do Wikipedia, o resultado se deu em um arquivo com tamanho de 4,4 MB. Nesse conjunto, 38.749 classes foram as classes: PESSOA/CARGO, ABSTRAÇÃO/IDEIA, ACONTECIMENTO, LOCAL/HUMANO e FÍSICO/REGIÃO.

Para validar a precisão desse modelo, um *script* em Python foi desenvolvido para facilitar a validação, assim, cada categoria utilizada no BERTimbau foi separada, com retorno apenas da categoria PESSOA/CARGO. O número de classes encontradas foi reduzido de 38.749 para 5.506 e a partir deste refino, os resultados foram analisados. A saída do *script* foi utilizada para validação, a fim de determinar a precisão real do BERTimbau.

Durante a aplicação, um contratempo foi identificado, pois a interpretação do BERTimbau considerou a categoria (PESSOA/CARGO) independentemente da origem brasileira. Os dados utilizados na validação se referem apenas à políticos brasileiros, uma vez que não temos informações sobre a política mundial. Assim, foram obtidos dois resultados de precisão, um específico para o Brasil e outro para o cenário político geral. Inicialmente, foi realizada a análise entre o político e o seu cargo, com identificação correta de 41 classes entre o nome e o cargo da pessoa.

Após a validação do *script* anterior, uma nova validação foi realizada, com alteração dos dados de políticos brasileiros, para nomes em geral. Dos 5.506 registros encontrados, 4.037 correspondiam a nomes de indivíduos, e dentro dessa categoria, constatou-se que 41 deles eram políticos brasileiros exclusivamente, como mostra a Figura 2.

Com base nas análises realizadas, conclui-se que o modelo BERTimbau apresentou uma precisão satisfatória. Considerando que os parâmetros não especificaram explicitamente a identificação exclusiva de políticos, a precisão foi de 73,32%. Totalizando 4.037 nomes encontrados e apenas 41 casos únicos de políticos, a precisão em relação políticos brasileiros foi de 1,015%. O valor é tido como baixo, já que o *script* funciona com casos únicos, ou seja, mesmo que nomes como Lula e Bolsonaro sejam citados diversas vezes, apenas um acerto para cada foi analisado, reduzindo drasticamente a porcentagem.

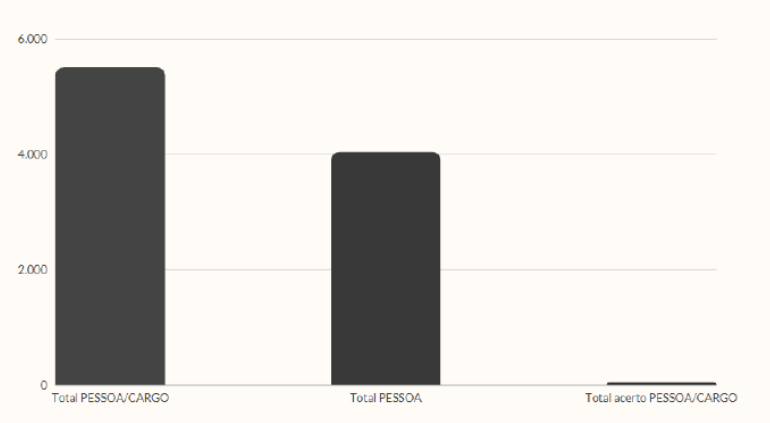


Figura 2. Resultado de automatização e acerto PESSOA/CARGO

Os valores encontrados no retorno do *script* para validação de cargos são apresentados na Figura 3. Neste caso, especificamente para os cargos, a precisão foi ainda maior, pois dos 1.469 cargos identificados, constatou-se que 237 deles eram cargos políticos. Divididos em 167 presidentes, 4 deputados estaduais, 1 deputado federal, 30 governadores, 25 senadores, 4 vice-governadores e 6 vice-presidentes, resultando em de 16,13%.

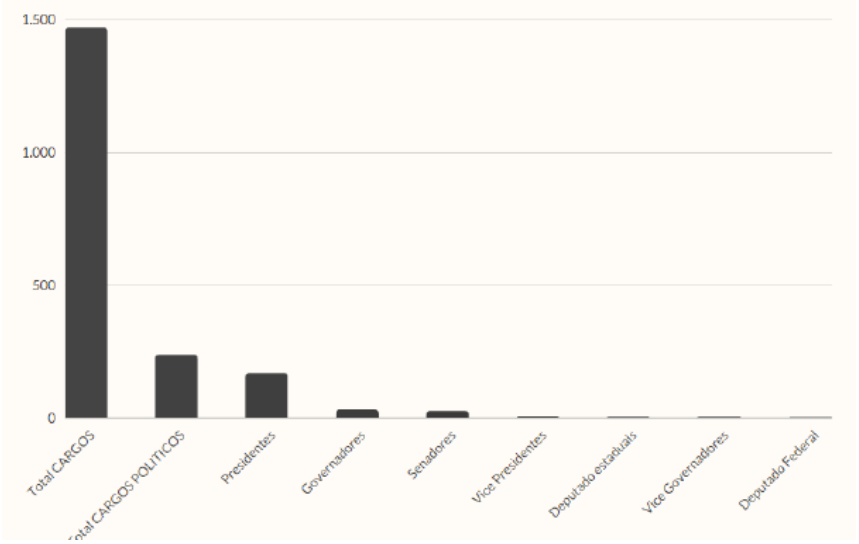


Figura 3. Resultado de automatização e acerto de CARGO

5.2. Twitter

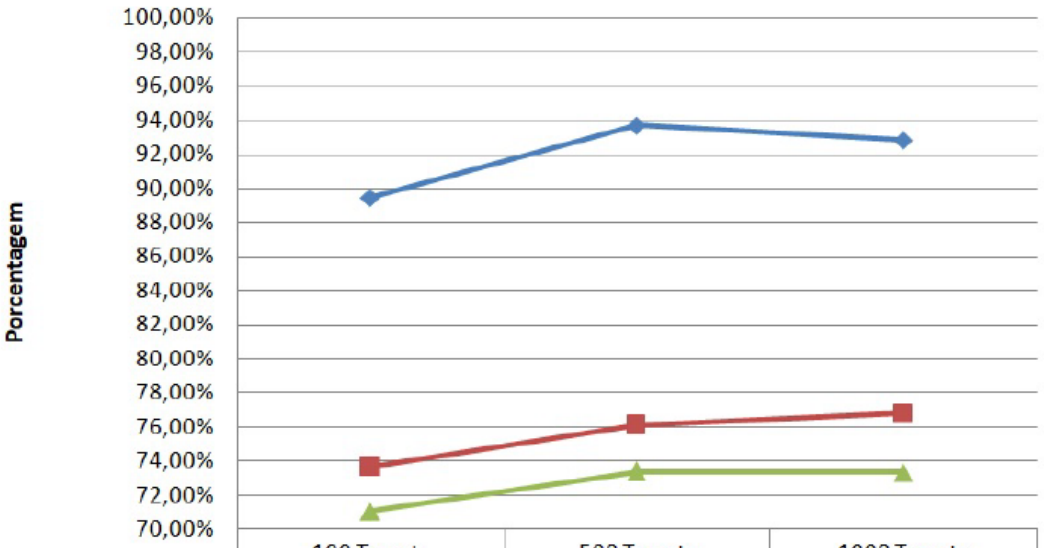
Para a análise sobre o arquivo de retorno da inferência textual do BERTimbau em 100 tweets, foram encontrados 81 acertos para POLÍTICO BRASIL, 84 acertos para POLÍTICO MUNDO, 102 acertos para PESSOA, 10 acertos para CARGO POLÍTICO e 12 acertos para CARGO. Também foram encontrados 13 erros, divididos proporcionalmente entre PESSOA (12 erros) e CARGO (1 erro) para cálculo da precisão em cada cenário. A partir da aplicando das fórmulas apresentadas no item 4.1., os resultados na análise em cem tweets, tem-se:

A análise sobre o arquivo de retorno da inferência textual do BERTimbau foi realizada em 100, 500 e em 1.000 tweets. Foram encontrados e validados os acertos para POLÍTICO BRASIL e POLÍTICO MUNDO, na classe PESSOA, para CARGO POLÍTICO na classe CARGO. Os resultados de precisão dos dois modelos apresentaram semelhante similaridade, conforme a classe, sendo apresentados na Tabela 1.

Tabela 1. Variação da precisão conforme quantidade de tweets analisada

Classe	100 tweets (%)	500 tweets (%)	1.000 tweets (%)
PESSOA	PP = 102/(102+12) = 89,47	PP = 373/(373+25) = 93,72	PP = 805/(805+62) = 92,85
	PPM = 84/(102+12) = 73,68	PPM = 303/(373+25) = 76,13	PPM = 666/(805+62) = 76,80
	PPB = 81/(102+12) = 71,05	PPB = 292/(373+25) = 73,36	PPB = 636/(805+62) = 73,35
CARGO	PC = 12/(12+1) = 92,30	PC = 24/(24+2) = 92,30	PC = 59/(59+4) = 93,65
	PCP = 10/(12+1) = 76,92	PCP = 20/(24+2) = 76,92	PCP = 48/(59+4) = 76,20

Como forma de análise conjunta de retorno de todas as quantidades de tweets (100, 500 e 1.000), percebe-se que os valores de precisão para a classe PESSOA mudaram pouco em diferentes tamanhos de dados e seguiram alguns padrões, conforme a Figura 5.



	100 Tweets	500 Tweets	1000 Tweets
Pessoa	89,47%	93,72%	92,85%
Politico Mundo	73,68%	76,13%	76,80%
Político Brasil	71,05%	73,36%	73,35%

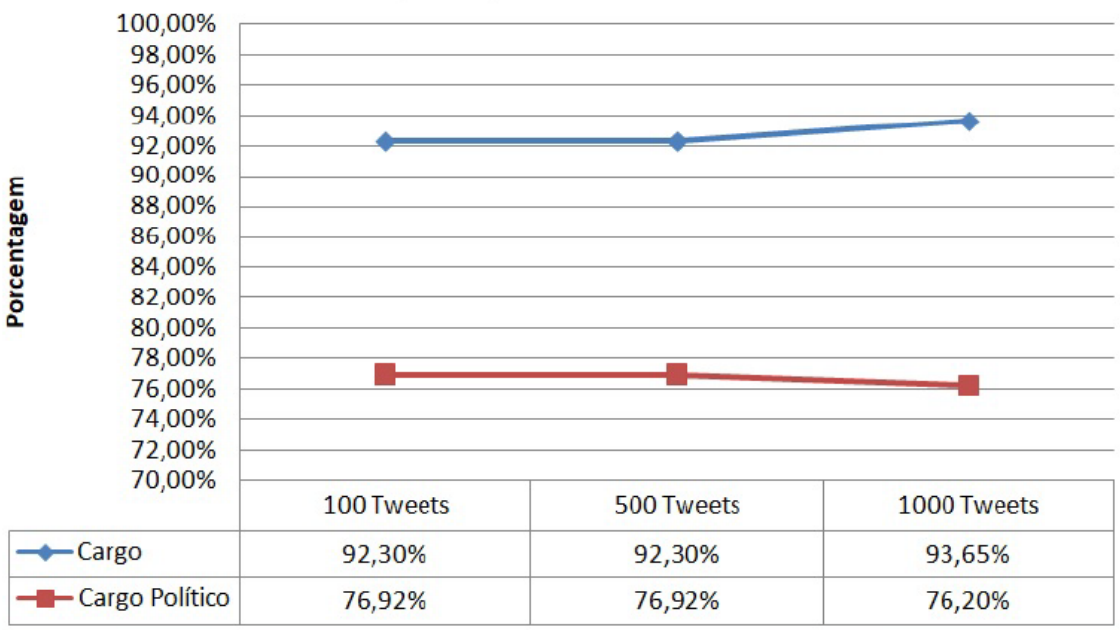
Figura 5. Valor de precisão dos três tamanhos de dados para PESSOA

A melhor precisão ser sobre a classe pessoa, era esperado. Isso ocorre pois é considerado que um político no Brasil é um político no mundo e que ambos são pessoas. Porém um ponto interessante foi de que a diferença de precisão entre eles nos três tamanhos de dados ficou muito próxima, oscilando entre pessoa e político mundo de 15,79% a 17,59% e entre político mundo e político Brasil de 2,63% a 3,45%.

Outro padrão identificado nos resultados, foi pela diferença entre os valores de precisão de cem para quinhentos tweets, com elevada oscilação. Para os dados de quinhetos para mil tweets os valores de precisão praticamente se mantiveram, mudando menos de 1%.

Outra constatação se deu sobre a data que o dado foi exportado e sua influência nos resultados, mesmo sem grande impacto nas porcentagens de precisão. A extração dos dados de cem tweets foi realizada em dezembro de 2022 já a de quinhentos e mil tweets, foram em junho de 2023. Mais da metade das referências citava Lula, Lira, Bolsonaro e Zanin, devido às notícias da semana que estavam sendo debatidas no Twitter.

Para os resultados referentes à classe CARGO praticamente não houve alteração (Figura 6) na precisão, o que inviabilizou encontrar o padrão de comportamento nas mudanças de tamanho de dados. Foi identificado que a classe CARGO sempre vai ter uma precisão melhor do que CARGO POLÍTICO, devido ao fato de que cargo político é considerado um cargo também.



	100 Tweets	500 Tweets	1000 Tweets
Cargo	92,30%	92,30%	93,65%
Cargo Político	76,92%	76,92%	76,20%

Figura 6. Valor de precisão dos três tamanhos de dados para CARGO

6. Conclusões

Após o desenvolvimento do trablaho que deu origem a este artigo, conclui-se que é possível empregar o BERT PolREN para atingir elevada precisão em análises quando ao cenário político. Para isso, utilizou-se a validação manual a partir da obtenção de números próximos a precisão geral encontrada no *script conlleval* para classe PESSOA/CARGO da inferência NER do BERTimbau.

Verificou-se também que quando o arquivo utilizado é muito grande se faz necessária a aplicação da validação automatizada, pois a precisão cai drasticamente para o cenário político. Assim, se torna viável realizar adaptações no processo automatizado visando aprimorar a precisão e eficácia geral, utilizando conjuntos de dados mais abrangentes que englobem políticos de todo o mundo. Tornando possível também a aplicação do BERT PolREN para qualquer tamanho de arquivo sem a redução da precisão.

Durante as análises, foram identificados alguns casos de falsos positivos, que interferiram tanto na abordagem manual quanto automatizada. Recomenda-se então que nestes casos seja considerado um *dataset* diferente do First HAREM para o treinamento do modelo, buscando eliminar tais falsos positivos.

Por fim, é possível afirmar que o objetivo geral do trabalho foi alcançado, uma vez em que foi verificada a precisão e acurácia do modelo, bem como a identificação sobre qual a melhor metodologia para cada cenário. Sobre a precisão, identificou-se que o BERT PolREN se apresenta de forma satisfatória quando aplicado para um cenário político, possibilitando sua utilização para verificação de fontes de dados e identificação de relações com políticos ou cargos políticos específicos.

Referencias

Bird, S.; Klein, E.; loper, E. (2009) “Natural language processing with Python: analyzing text with the natural language toolkit. [S.l.]. O’Reilly Media, Inc..

Devlin, J. et al. (2018) “Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Elint. (2021) “Uma Visão Geral sobre Named Entity Recognition (NER)”, https://medium.com/elinttech/uma-vis%C3%A3o-geral-sobre-named-entity-recognition-ner-4dc4e3b5e37a, Julho de 2023.

Jurafsky, D.; martin, J. H. (2019) “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition”.

Kenny, C. (2020) “What is web scraping?”, https://www.zyte.com/learn/what-is-web-scraping, Junho de 2023.

Souza, A. e Camello, V. (2023) “BERT PolREN: Análise da precisão sobre a inferência textual de reconhecimento de entidades nomeadas do BERTimbau no cenário político”, Trabalho de Conclusão de Curso. Universidade Federal de Santa Catarina. Florianópolis. Brasil.

Sutton, C.; Mccallum, A. et al. (2012) “An introduction to conditional random fields”. Foundations and Trends® in Machine Learning, Now Publishers, Inc., v. 4, n. 4, p. 267–373.