Trabalho de Conclusão de Curso

Estudo de Aplicabilidade de Técnicas de Processamento de Linguagem Natural em Petições Iniciais

Ana Geórgia de Souza Silva Gama Pereira
agssgp@ic.ufal.br

Orientador:

Prof. Dr. André Lage Freitas

Maceió, 14 de junho de 2023

Ana Geórgia de Souza Silva Gama Pereira

Estudo de Aplicabilidade de Técnicas de Processamento de Linguagem Natural em Petições Iniciais

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. André Lage Freitas

Maceió, 14 de junho de 2023

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecária Responsável: Maria Helena Mendes Lessa– CRB4 - 1616

P436e Pereira, Ana Geórgia de Souza Silva Gama.
Estudo de Aplicabilidade de Técnicas de Processamento de Linguagem Natural em Petições Iniciais / Ana Geórgia de Souza Silva Gama Pereira. – 2023.
23 f. : il. color.

Orientador: André Lage Freitas.
Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2023.

Bibliografia: f. 22-23.

1. Computação. 2. Ensino e aprendizagem. 3. Processamento de linguagem natural. 4. Tecnologia e educação. I. Título.

CDU: 004.434

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Prof. Dr. André Lage Freitas - Orientador
Universidade Federal de Alagoas

Prof. Dr. Leonardo Viana Pereira - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Prof. Dr. Orivaldo Vieira Santana Jr. - Examinador
Escola de Ciências e Tecnologia
Universidade Federal do Rio Grande do Norte

Maceió, 14 de junho de 2023

Agradecimentos

Gostaria de declarar meus sinceros agradecimentos a todas as pessoas que contribuíram para a realização deste trabalho.

Expresso aqui minha profunda gratidão ao meu orientador, por todo apoio, paciência e sabedoria fornecida ao longo desses anos de orientação. Sua experiência e incentivo foram imprescindíveis ao longo de minha jornada acadêmica.

Além disso, sou grata a mim mesma. Este trabalho representa uma trajetória repleta de desafios e obstáculos. Diante de todas as dificuldades encontradas, sou grata por não ter desistido.

À todos os que colaboraram, direta ou indiretamente, meu mais intenso agradecimento. Nada disso teria sido possível sem o apoio e encorajamento de cada um.

Ana Geórgia Gama

"Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes."

– Luther King, *Marthin*

Resumo

Este trabalho teve como objetivo estudar a aplicação de algoritmos de processamento de linguagem natural (NLP) na classificação de petições iniciais do sistema jurídico brasileiro. Foram explorados os algoritmos Doc2Vec e BERT, com a finalidade de analisar sua eficácia e comparar seus desempenhos. A metodologia envolveu etapas de pré-processamento dos dados, treinamento e avaliação dos modelos. Os resultados foram avaliados utilizando métricas como precisão, acurácia e matriz de confusão. O modelo BERT demonstrou uma precisão de 85% na classificação de documentos, enquanto o modelo Doc2Vec apresentou resultados moderados. A importância do pré-processamento dos dados e a influência da dimensionalidade também foram discutidas. Além disso, foi destacada a necessidade de recursos computacionais adequados para a análise de modelos de linguagem grandes. Esses resultados contribuem para a compreensão do desempenho dos algoritmos de NLP na classificação de petições iniciais e podem auxiliar na seleção e implementação de modelos adequados em diferentes contextos jurídicos.

Palavras-chave: processamento de linguagem natural, classificação de documentos, petições iniciais, Doc2Vec, BERT.

Abstract

This study aimed to investigate the application of natural language processing (NLP) algorithms in the classification of initial petitions in the Brazilian legal system. The Doc2Vec and BERT algorithms were explored to analyze their effectiveness and compare their performances. The methodology involved data preprocessing, model training, and evaluation stages. The results were assessed using metrics such as precision, accuracy, and confusion matrix. The BERT model demonstrated an 85% precision in document classification, while the Doc2Vec model showed moderate results. The importance of data preprocessing and the influence of dimensionality were also discussed. Additionally, the need for adequate computational resources for analyzing large language models was emphasized. These findings contribute to understanding the performance of NLP algorithms in classifying initial petitions and can assist in selecting and implementing suitable models in various legal contexts.

Key-words: natural language processing, document classification, initial petitions, Doc2Vec algorithms, BERT.

Lista de Figuras

Conteúdo

1

Introdução

O Relatório Justiça em Números 2022 (1) fornece informações detalhadas sobre o fluxo processual no sistema de justiça brasileiro. Nesta edição, destacou-se que o Poder Judiciário teve 77,3 milhões de processos em tramitação no ano de 2021. A depender da esfera jurídica considerada, esses processos levam mais de dois anos para ser concluídos. Em 2021, os processos eletrônicos representaram 80,8% das ações em tramitação e 89,1% dos casos baixados. Dos 90 órgãos do Judiciário, 44 aderiram integralmente ao programa Justiça 4.0, o que abrange 67,7% das serventias judiciais.

Isso nos mostra que ainda há certa morosidade nas tramitações de processos judiciais. Por outro lado, é crescente a adesão do sistema jurídico brasileiro à transformação digital e uso de inteligência artificial em seu âmbito. O judiciário já usa robôs, alguns com funções específicas e outros que podem acelerar e até mesmo concluir processos. Como é o caso da Elis, ferramenta capaz de examinar e selecionar os processos de execução fiscal. De acordo com o Tribunal de Justiça de Pernambuco (2), esses processos representam mais de 50% de todas as ações em andamento no estado nordestino. Enquanto a triagem manual de setenta mil processos leva em média um ano e meio, Elis analisa pouco mais de oitenta mil processos em quinze dias. Temos também o exemplo do Victor (3), uma inteligência artificial voltada para apoiar a atividade de análise de admissibilidade recursal, mediante sinalização de que um ou mais temas de repercussão geral, se aplica ao caso dos autos.

Este trabalho tem como **objetivo realizar um estudo da aplicabilidade de técnicas de Processamento de Linguagem Natural em petições iniciais**. Especificamente, neste estudo aplicamos algoritmos de processamento de linguagem implementados pelos modelos Doc2Vec (4) e BERT (5). **Este objetivo evidencia uma especificidade pouco encontrada na área, que é a aplicação desses modelos a este tipo de documento jurídico**.

A metodologia adotada consistiu em analisar o desempenho de cada abordagem em etapas, envolvendo pré-processamento dos dados, treinamento e avaliação dos modelos.

Inicialmente, fizemos a escolha do dataset que seguiríamos trabalhando. Tínhamos um dataset multi classes com uma quantidade menor de amostras e um dataset com duas classes e uma quantidade maior de amostras. Baseado em critérios de análise dos passos seguintes, seguimos com o modelo de duas classes, contendo 6000 amostras de cada, devido ao fato de ter maior representatividade e menor complexidade ao lidar com o modelo de classificação binária.

Partimos então para a etapa de pré-processamento, onde acontece a limpeza dos documentos. Removemos *stopwords* e reduzimos o ruído textual. Em seguida, utilizamos os modelos Doc2Vec e BERT, que são amplamente conhecidos por sua capacidade de gerar representações vetoriais e analisar relações entre as palavras.

Os modelos foram treinados com um conjunto de dados rotulados, onde cada documento foi associado a uma classe específica, podendo ser `dir_consumidor` (direito do consumidor) ou `dir_tributario` (direito tributário). Para o Doc2Vec, utilizamos regressão logística para classificação, enquanto no BERT utilizamos uma camada adicional de classificação no topo do modelo para realizar a tarefa de classificação das petições. Durante o treinamento de ambos, utilizamos os parâmetros padrões indicados para cada modelo e avaliamos sua performance em um conjunto de teste.

Os resultados obtidos foram analisados com base em métricas de avaliação como: precisão, acurácia e matriz de confusão. Além disso, para o modelo Doc2Vec, temos ainda uma visualização do agrupamento utilizando o algoritmo k-means com t-SNE, para analisar as classificação sem rotulação prévia. Já para o BERT, realizamos uma análise de perda do treinamento e o cálculo do coeficiente de correlação de Matthews (MCC) para obter uma melhor compreensão do comportamento e do desempenho dos modelos.

Este manuscrito está organizado da seguinte maneira. Este capítulo contendo uma **Introdução geral** do trabalho. A seguir, o **capítulo Contribuições**, onde descrevemos todos os percursos trilhados durante o estudo. Este, foi ainda **divido em quatro seções: Análise dos dados, Pré-processamento, Transformação dos dados categóricos em dados numéricos e Aplicação dos modelos Doc2Vec e BERT**. Nesta última seção do capítulo Contribuições, **descrevemos a metodologia, resultados e discussões para cada um dos modelos**, incluindo o processo de treinamento, classificação e o uso das métricas de avaliação para cada um deles. No capítulo seguinte temos as **Conclusões gerais** extraídas a partir do que foi demonstrado no capítulo anterior.

2

Contribuição

A contribuição deste trabalho é um estudo de viabilidade de técnicas de NLP para o documento jurídico Petição Inicial. Foram percorridos diversos caminhos durante a pesquisa. Alguns mais relevantes que outros, porém juntos, nos levaram às conclusões aqui expostas. Considerando o contexto de um TCC, julgamos válido registrar aqui o roteiro trilhado para que os demais pesquisadores possam se beneficiar do aprendizado durante esse trabalho investigativo, seja continuando a percorrer os caminhos que foram exitosos ou seja evitando investigar frentes cujos resultados não foram promissores ou satisfatórios.

Baseando-se no objetivo do trabalho - realizar um estudo da aplicabilidade de técnicas de Processamento de Linguagem Natural em petições iniciais, aplicamos as seguintes etapas metodológicas para alcançar este propósito:

1. Análise dos dados
2. Pré-processamento
3. Transformação de dados categóricos em dados numéricos
4. Aplicação dos modelos Doc2Vec e BERT
    - Doc2Vec - Metodologia
    - Doc2Vec - Resultados e discussões
    - BERT - Metodologia
    - BERT - Resultados e discussões

Na **metodologia** descrevemos o que e como foi feito nas **etapas de Treinamento, Classificação e Métricas de avaliação de cada modelo**. Da mesma forma, nos **resultados**, mostramos o que foi gerado a partir da metodologia, em cada uma das etapas, bem como uma breve uma breve **discussão** sobre eles.

2.1 Análise dos dados

À princípio, tínhamos um dataset com 1.000 petições iniciais, contendo 200 amostras de cada um dos ramos do direito: Direito Civil, Direito do Consumidor, Direito Previdenciário, Direito Administrativo e Direito Tributário. No entanto, ao mergulhar mais a fundo na forma como os modelos funcionavam, vimos que trabalhar com dataset multi classes iria aumentar ainda mais a complexidade do nosso estudo, demandando um tempo que não tínhamos disponível. Desta forma, focamos na representatividade dos dados, aumentando o número de amostras e na escolha de um dataset com duas classes, uma vez que tínhamos uma gama maior de exemplos com aplicações de classificações binárias.

Em um primeiro momento, realizamos uma análise simples dos dados com o objetivo de ter uma visão geral do conjunto de dados utilizados. De antemão, sabemos que o conjunto escolhido é formado por **12.000 petições iniciais** do Tribunal de Justiça de São Paulo, **divididas igualmente em dois ramos do direito: Direito do Consumidor e Direito Tributário**. Os dados foram coletados utilizando *scripts* do repositório `Pacote tjsp` (6).

A princípio, nosso dataset tem três colunas: `processo_id`, `peticao_inicial_texto` e `assunto_peticao` [Figura 2.1].

```
~> df_12K.describe:

<bound method NDFrame.describe of                processo_id  ...                                    assunto_peticao
0       xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO DO CONSUMIDOR - Contratos de Consumo -...
1       xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO DO CONSUMIDOR - Contratos de Consumo -...
2       xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO DO CONSUMIDOR - Contratos de Consumo -...
3       xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO DO CONSUMIDOR - Contratos de Consumo -...
4       xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO DO CONSUMIDOR - Contratos de Consumo -...
...                          ...  ...                                               ...
11996   xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO TRIBUTÁRIO - Impostos - ITCD - Imposto...
11997   xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO TRIBUTÁRIO - Contribuições - Contribui...
11998   xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO TRIBUTÁRIO - Impostos - IPTU/ Imposto ...
11999   xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO TRIBUTÁRIO - Impostos - ICMS/ Imposto ...
12000   xxxxxxxxxxxxxxxxxxxxxxxx  ...  DIREITO TRIBUTÁRIO - Impostos - IPTU/ Imposto ...

[12001 rows x 3 columns]>


~> df_12K.columns:

Index(['processo_id', 'peticao_inicial_texto', 'assunto_peticao'], dtype='object')
```

Figura 2.1: Dataset original - `head` e `columns`

Observamos que a coluna `assunto_peticao` contém o ramo do direito ao qual aquela petição pertence, seguido de seus sub ramos [Figura 2.2].

```
~> df_12K['assunto_peticao'].value_counts():

_____________________________________________________________________________________________________
assunto_peticao
DIREITO DO CONSUMIDOR - Contratos de Consumo - Bancários
   2738
DIREITO TRIBUTÁRIO - Impostos - ICMS/ Imposto sobre Circulação de Mercadorias
   1595
DIREITO TRIBUTÁRIO - Impostos - IPTU/ Imposto Predial e Territorial Urbano
   1519
DIREITO DO CONSUMIDOR - Contratos de Consumo - Fornecimento de Energia Elétrica
    767
DIREITO TRIBUTÁRIO - Impostos - ISS/ Imposto sobre Serviços
    740

   ...
DIREITO TRIBUTÁRIO - Limitações ao Poder de Tributar - Imunidade
      1
DIREITO DO CONSUMIDOR - Contratos de Consumo - Transporte Aéreo - Problemas com bagagem (exceto extravio)
      1
DIREITO DO CONSUMIDOR - Contratos de Consumo - Planos de Saúde-Fornecimento de medicamentos
      1
DIREITO TRIBUTÁRIO - Contribuições - Contribuições Especiais - FGTS/Fundo de Garantia Por Tempo de Serviço
      1
DIREITO TRIBUTÁRIO-Crédito Tributário-Base de Cálculo-Exclusão - ICMS
      1
Name: count, Length: 87, dtype: int64
_____________________________________________________________________________________________________
```

Figura 2.2: Dataset original - `assunto_peticao`

Com foco na coluna `peticao_inicial_texto`, podemos visualizar através da prévia do texto demonstrada abaixo [Figura 2.3], que há muitos espaços vazios, números e palavras que não têm muita utilidade para nossa análise.

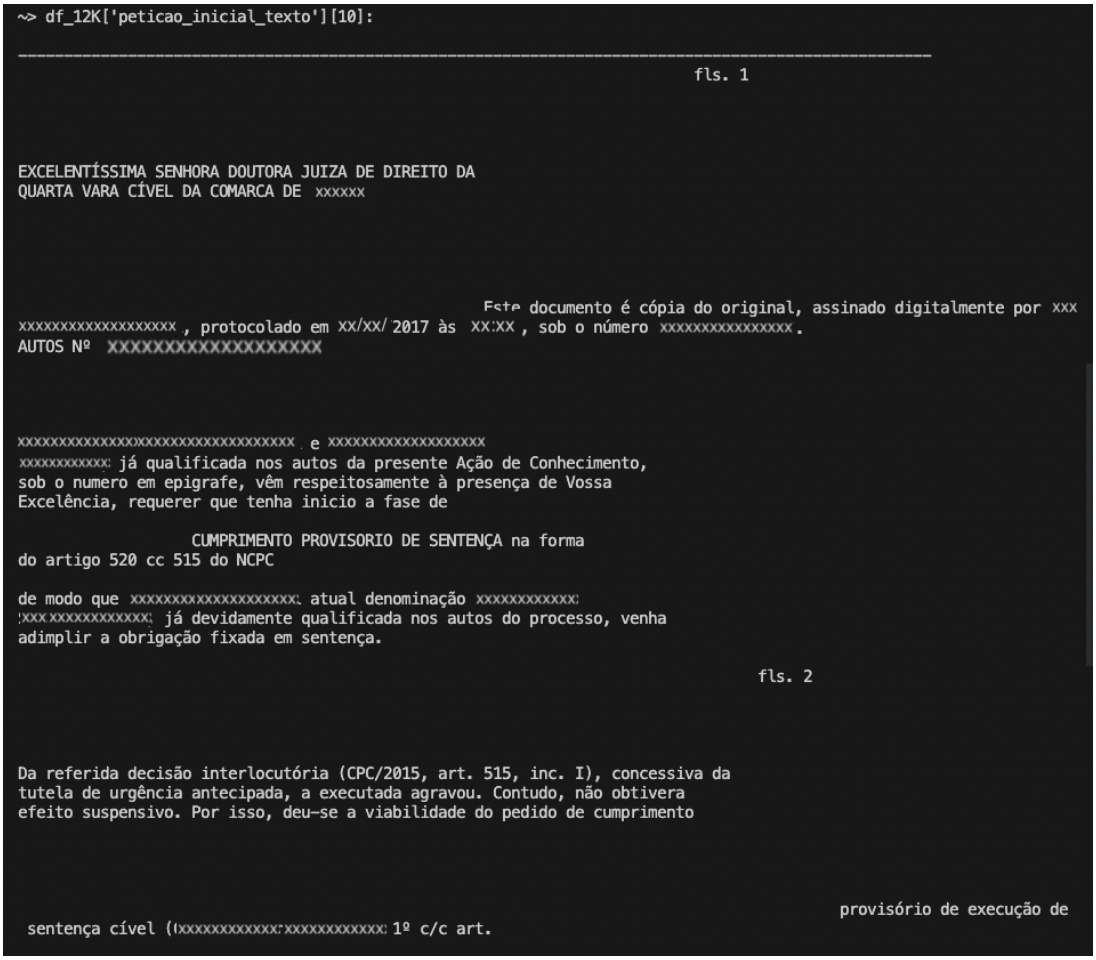

Figura 2.3: Dataset original - `peticao_inicial_texto`

Diante desse cenário, começamos renomeando a coluna `peticao_inicial_texto` para `texto_peticao` e a partir da coluna `assunto_peticao`, criamos a coluna `ramo`, que armazena apenas o ramo principal do direito ao qual o documento pertence. Renomeamos também estas classes para `dir_consumidor` e `dir_tributario`. Estas mudanças foram aplicadas ao conjunto de dados e a partir deste ponto vamos nos referenciar ao conjunto de dados original considerando que ele inclui essas mudanças [1].

Uma visão geral do que o nosso dataset original se tornou [Figura 2.4]. Agora temos uma coluna a mais contendo apenas o ramo principal.

[1] O código utilizado para os passos descritos acima é o mesmo da etapa de pré-processamento, indicado mais adiante.

```
~> df_12K.head:

------------------------------------------------------------------------------------------------------------
<bound method NDFrame.head of               id_processo  ...           ramo
0      XXXXXXXXXXXXXXXXX  ...  dir_consumidor
1      XXXXXXXXXXXXXXXXX  ...  dir_consumidor
2      XXXXXXXXXXXXXXXXX  ...  dir_consumidor
3      XXXXXXXXXXXXXXXXX  ...  dir_consumidor
4      XXXXXXXXXXXXXXXXX  ...  dir_consumidor
...                  ...  ...             ...
11996  XXXXXXXXXXXXXXXXX  ...  dir_tributario
11997  XXXXXXXXXXXXXXXXX  ...  dir_tributario
11998  XXXXXXXXXXXXXXXXX  ...  dir_tributario
11999  XXXXXXXXXXXXXXXXX  ...  dir_tributario
12000  XXXXXXXXXXXXXXXXX  ...  dir_tributario

[12000 rows x 4 columns]>
------------------------------------------------------------------------------------------------------------


~> df_12K.columns:

------------------------------------------------------------------------------------------------------------
Index(['id_processo', 'texto_peticao', 'assunto_peticao', 'ramo'], dtype='object')
------------------------------------------------------------------------------------------------------------
```

Figura 2.4: Dataset limpo - `head` e `columns`

2.2 Pré-processamento

Como visto na seção anterior, fica evidente a necessidade da aplicação de técnicas de limpeza textual nos documentos. Esse passo é essencial para a aprimorar a qualidade e consistência dos textos para análises e modelagem subsequente.

Temos elementos que podem ser considerados "ruídos"ou informações indesejadas no texto, tais como: caracteres especiais, excesso de espaços em branco, informações de formatação, textos repetidos, dentre outros. Então, realizamos a aplicação de técnicas de limpeza textual como remoção de caracteres especiais, dígitos, *stopwords*, palavras frequentes e espaços excessivos.

Iniciamos a etapa de pré-processamento. Executamos o pré-processamento [2] uma primeira vez, removendo os ruídos percebidos na análise anterior. A partir desse texto já limpo, contamos[3] as seis [4] palavras que mais apareciam nos documentos, gerando um gráfico [Figura 2.5] que ilustra a quantidade que cada uma aparece.

[2] Acesso ao código de pré-processamento: pre-processamento.py
[3] Acesso ao código de análise das palavras: analise-das-palavras.py
[4] Quantidade escolhida de maneira aleatória.

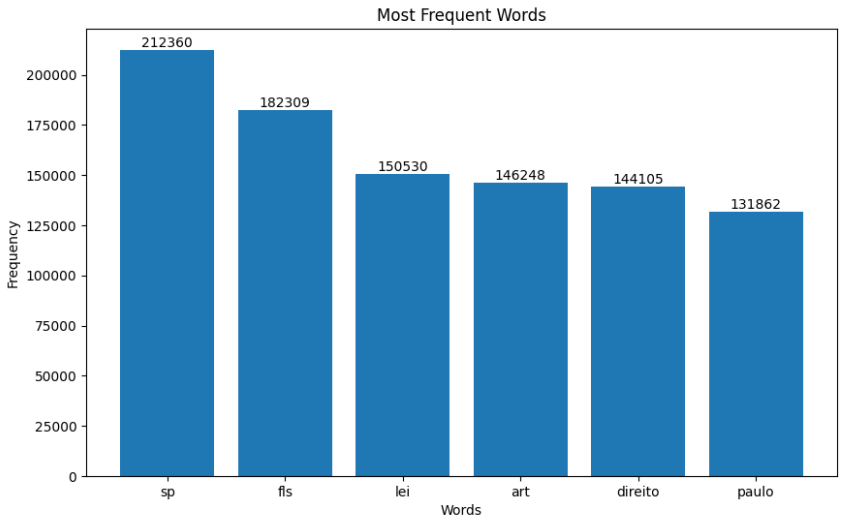

Figura 2.5: Palavras mais frequentes - Dataset limpo

Executamos uma segunda vez o pré-processamento, desta vez, adicionando as palavras frequentes na lista de *stopwords*. Desse modo, as palavras frequentes também seriam removidas dos textos. Isso possibilita que os algoritmos de processamento de linguagem natural tenham um desempenho mais eficiente e preciso na extração de informações pertinentes dos textos.

Na sequência, geramos gráficos [Figuras 2.6 e 2.7] que comparam a quantidade de palavras que cada documento tinha antes e depois do pré-processamento do texto, bem como, a média de palavras nos documentos de cada ramo.

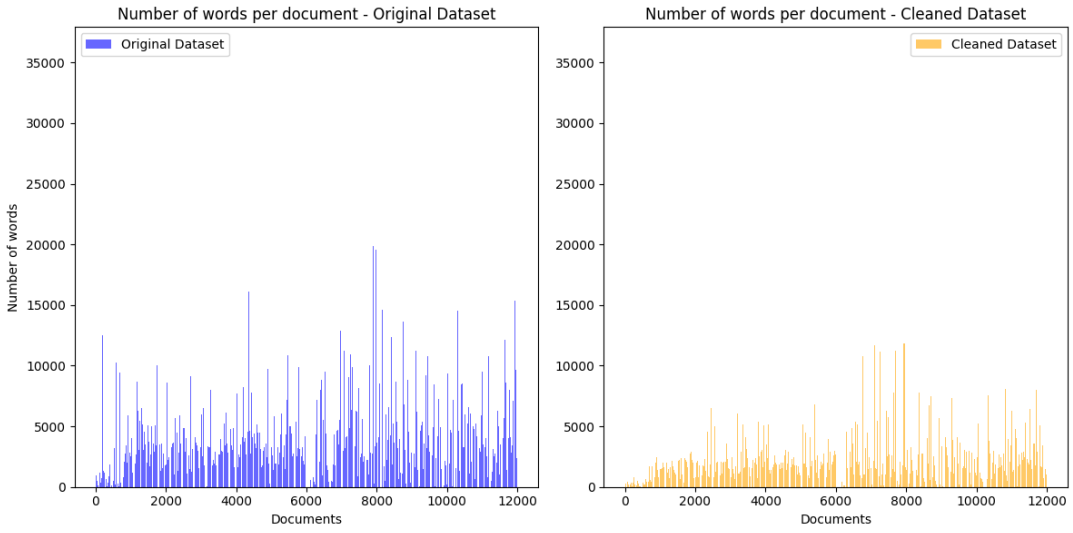

Figura 2.6: Número de palavras por documento

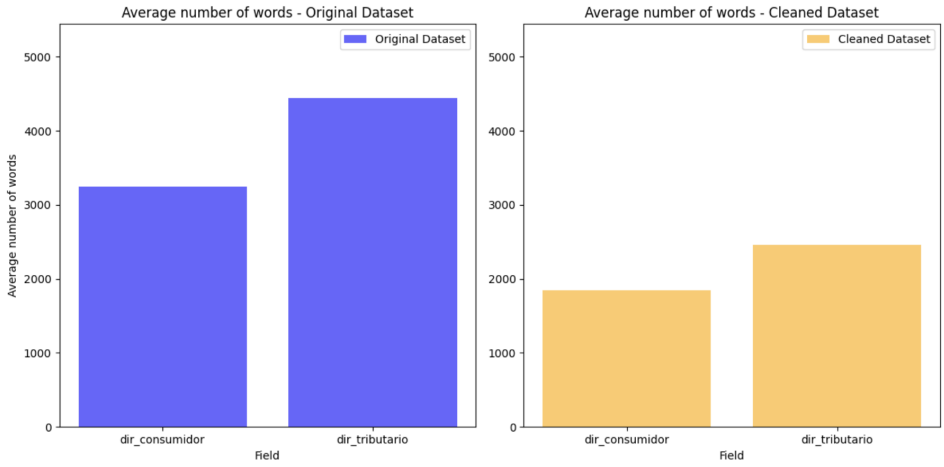

Figura 2.7: Média de palavras contidas nos documentos de cada ramo

Temos agora um número significativamente menor de palavras no dataset limpo. Isso trás mais consistência para os dados, uma vez que a maioria dessas palavras eram comuns a todos os documentos e isso poderia dificultar no processo de agrupamento, pois possivelmente apresentariam muitas similaridades.

Por fim, a coluna `texto_peticao`, apresenta uma versão [Figura 2.8] mais condensada e concisa, onde podemos considerar um texto quase sem ruídos.

```
~> df_12K['texto_peticao'][10]:

excelentíssima senhora doutora juiza quarta vara cível comarca xxxxxxx documento cópia original assinado digitalmente xxxxxxx xxx xxxxxxx protocolado
 sob número wbre autos xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx xx xxxxxxx qualificada autos presente ação conhecimento sob numero epigrafe vê
m respeitosamente presença vossa excelência requerer inicio fase cumprimento provisorio sentença forma artigo cc ncpc modo xxxxx xxxxxxx real atual d
enominação xxxxxx xxxxxxx devidamente qualificada autos processo venha adimplir obrigação fixada sentença referida decisão interlocutória cpc inc con
cessiva tutela urgência antecipada executada agravou contudo obtivera efeito suspensivo deu viabilidade pedido cumprimento provisório execução sentenç
a cível cpc documento cópia original assinado digitalmente xxxxx xxxxxxx xxxxxx protocolado sob número wbre inc processo conhecimento tramitou perante
 juízo deu parcial provimento pedidos formulados ação condenando requerida pagamento planos econômicos expurgos relacionados sentença exequenda referi
da sentença executada apelou segunda instancia julgada improvido recurso ingressando recurso superior tribubal justiça contudo obtivera efeito suspens
ivo deu viabilidade pedido cumprimento provisório execução sentença cível cpc inc tendo vista requerida cumpriu sentença homologada vossa excelência f
az necessário inicio fase cumprimento sentença diante exposto requer vossa excelência inicio fase cumprimento sentença intimação requerida quinze dias
 pague valor dois milhoes sessenta dois mil novecentos sessenta reais trinta seis centavos ainda ocorrer pagamento voluntário prazo dias deverá acresc
ida multa honorários advogado dez cento termos ncpc devendo vossa excelência procer penhora on line valor devido atualizado momento documento cópia or
iginal assinado digitalmente xxxxxxx xxx xxxxxxx protocolado sob número wbre bloqueio termos artigo ambos ncpc requer ainda arbitrado honorários sucu
mbência fase cumprimento sentença valor pago caso pagamento espontâneo nestes termos pede deferimento xxxxxxx xxx xxxxxxx oab
```

Figura 2.8: Dataset limpo - `texto_peticao`

Esses passos foram relevantes para entender a natureza dos documentos, preparando-os da melhor maneira para auxiliar na construção de modelos de Processamento de Linguagem Natural (NLP) aplicados a esses dados.

2.3 Transformação de dados categóricos em dados numéricos

Tendo em vista que os modelos de classificação tendem a lidar melhor com dados numéricos, uma vez que são baseados em algoritmos que processam textos modelados em números, transformamos a coluna de Ramo do Direito em uma representação numérica utilizando a função **LabelEnconder()** da biblioteca `sklearn`. Com isso, adicionamos ao dataset a coluna `labels`,

contendo a representação numérica da coluna `ramo`, onde 0 representa `dir_consumidor` e 1 representa `dir_tributario`.

2.4 Aplicação dos modelos Doc2Vec e BERT

O objetivo desta fase é realizar o treinamento dos modelos Doc2Vec e BERT, seguido de um processo de classificação e uma análise da qualidade da classificação em cada modelo por meio do uso de métricas de avaliação. Para cada modelo, seguimos a sequência:

1. Treinamento do modelo
2. Submissão do dataset de petições ao modelo treinado, gerando uma representação vetorial dos documentos
3. Classificação
4. Métricas de avaliação

Para ambos os modelos, estruturamos a narrativa contendo cada um suas respectivas metodologias, resultados e discussões.

2.4.1 Doc2Vec - Metodologia

Treinamento e teste

Para o modelo Doc2Vec[5], começamos carregando o conjunto de dados e posteriormente o dividimos em conjuntos de teste e treinamento, de maneira aleatória na proporção de 20% para teste e 80% para treino. Em seguida, treinamos o modelo Doc2Vec, implementado pela biblioteca Gensim (7), com os dados de treinamento.

Classificação - Regressão Logística

Com o modelo treinado, partimos para a validação dele, ou seja, para avaliar o desempenho do modelo em relação à tarefa de classificar Petições Iniciais de acordo com o Ramo do Direito. Utilizamos o conjunto de treino para treinar o modelo de regressão logística. Após treinado, utilizamos o modelo de regressão logísica para classificar o conjunto de teste, ou seja, para classificar se as Petições Iniciais do conjunto de teste são do Ramo do Direito do Consumidor ou Direito Tributário.

[5] Acesso ao código Doc2Vec: Doc2Vec.py

Métricas de avaliação

Após a classificação do modelo, aplicamos as métricas para visualizar seu desempenho. Começamos pelo relatório de classificação, seguido da geração de matriz confusão. Resolvemos ainda, avaliar a aplicabilidade do modelo Doc2Vec treinado para agrupar Petições Iniciais, utilizando os vetores gerados pelo modelo. Utilizamos o algoritmo de agrupamento (*clustering*) K-means para separar automaticamente as Petições Iniciais em dois grupos. O objetivo era entender se os grupos formados estariam próximos de acordo com o Ramo do Direito.

Por fim, para compreender se o algoritmo k-means foi efetivo, avaliamos a qualidade dos dois grupos que foram formados utilizando a Análise de *Silhouette*.

2.4.2 Doc2Vec - Resultados e conclusões

Relatório de classificação

De acordo com o relatório de classificação (8) [Figura 2.9], o desempenho da regressão logística para classificar as Petições Iniciais pode ser considerado de moderado a ruim, dado que a precisão com a qual ele acertou os dois rótulos foi de 51%. Se considerarmos os rótulos isolados, vemos que para o rótulo `dir_tributario` ele se saiu melhor, mostrando `recall` de 61% contra 42% do outro rótulo. E ainda, teve F1-score de 0.55 classificando Petições Iniciais do Ramo `dir_tributario`. Isso quer dizer que o modelo teve um desempenho moderado na predição dessa classe. Quanto mais próximo de 0 for o F1-score, significa que o modelo não conseguiu prever corretamente essa classe em nenhuma instância.

[Doc2Vec | 2 Classes Dataset] - Classification Report:

	precision	recall	f1-score	support
0	0.52	0.39	0.44	1236
1	0.49	0.62	0.55	1164
accuracy			0.50	2400
macro avg	0.50	0.50	0.49	2400
weighted avg	0.50	0.50	0.49	2400

Figura 2.9: Doc2Vec - Relatório de classificação

Construímos também a matriz confusão [Figura 2.10]. Podemos ver que o modelo classificou corretamente 528 documentos como pertencentes à classe 0 (`dir_consumidor`) enquanto que ele classificou 493 erroneamente como classe 0, quando na verdade eles pertenciam à classe 1 (`dir_tributario`), por exemplo.

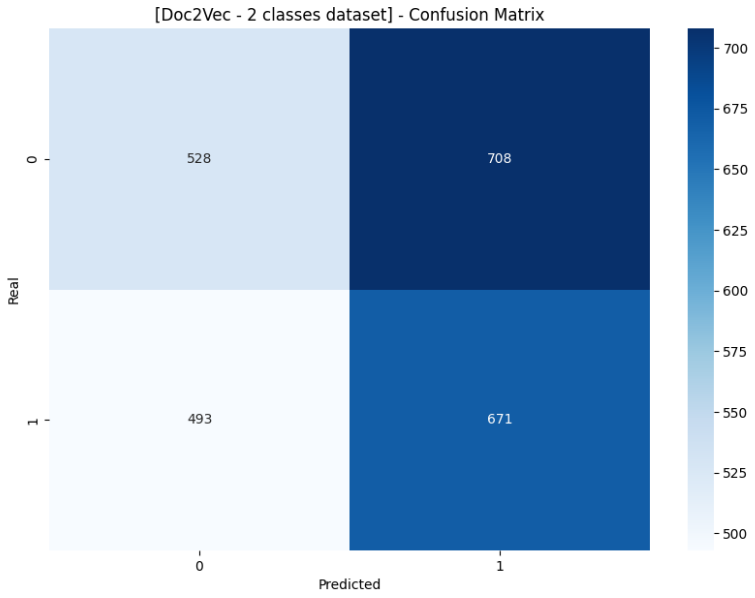

Figura 2.10: Doc2Vec - Matriz Confusão

Agrupamento

Visualizando os grupos formados, na Figura 2.11, e a Análise Silhouette, na Figura 2.12.

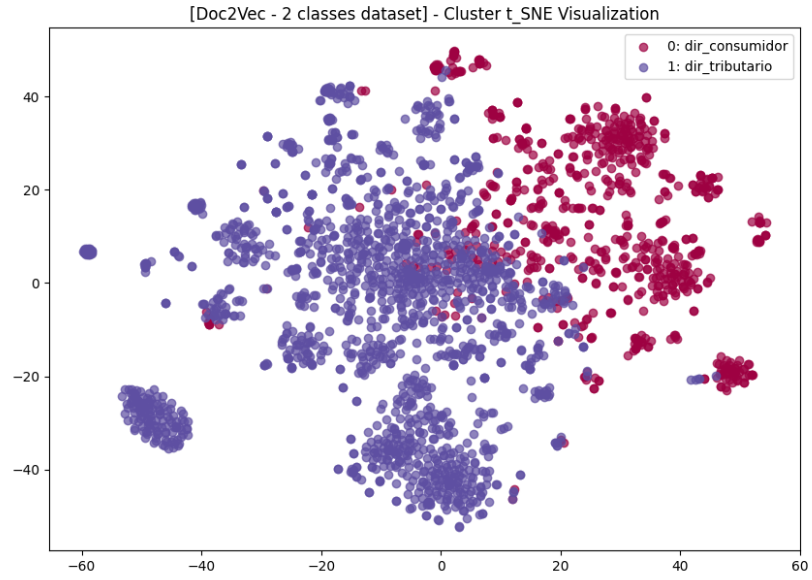

Figura 2.11: Doc2Vec - t-SNE e K-means

No agrupamento, os pontos roxos representam os documentos que ele classificou como `dir_consumidor` e em vermelho os `dir_tributario`. A localização de cada ponto representa

o vetor bidimensional gerado pelo algoritmo t-SNE, reduzindo o vetor de cem dimensões do modelo Doc2Vec.

Com base nisso, podemos ver que o modelo Doc2Vec proporcionou um agrupamento dos documentos relativamente parecido com a forma que os humanos classificaram, pois se fosse da mesma forma teríamos os pontos roxos agrupados entre si e os vermelhos da mesma forma, sem se misturar. Isso nos leva a crer que o agrupamento feito pelos modelos computacionais de aprendizagem de máquina não correspondem inteiramente ao agrupamento feito por seres humanos. Em outras palavras, houve algumas similaridades entre os conteúdos dos textos das Petições Iniciais para os dois Ramos do Direito.

O valor de 0.1083705 indica um resultado moderado na qualidade de formação dos grupos [Figura 2.12]. Um valor muito próximo de zero pode indicar que há sobreposições entre os grupos, o que significa que alguns pontos podem ter sido atribuídos a grupos incorretos ou estão localizados na fronteira entre os grupos.

Assim, os grupos formados demonstram satisfatoriamente o conteúdo das Petições Iniciais.

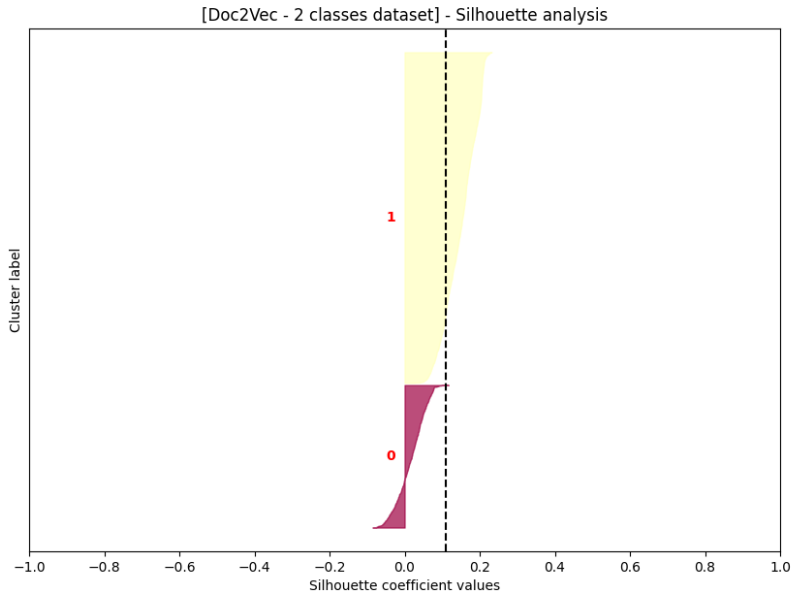

Figura 2.12: Doc2Vec - Silhouette

2.4.3 BERT - Metodologia

Para esta seção, seguimos o tutorial *Bert - Word Embeddings*(9) [6].

[6] Acesso ao código BERT: BERT - fine-tuning

Configuração

Iniciamos o treinamento e avaliação de um modelo de classificação de sequências utilizando o modelo pré-treinado BERT (*Bidirectional Encoder Representations from Transformers*). Foi rodado a primeira vez com o dataset original [7] e na sequência com o dataset limpo [8].

O código carrega um conjunto de dados com os textos das petições e os `labels` (rótulos de classe) correspondentes. Os documentos judiciais são pré-processados por meio da inclusão de tokens especiais ([CLS] e [SEP]) no início e no final de cada sentença. Considerando que uma sentença é o texto completo do documento.

Em seguida, o tokenizador do BERT é usado para dividir as sentenças em sub-palavras, ou tokens. Para garantir que todas as sequências tenham o mesmo tamanho, foi criado um tamanho máximo de sequência (`MAX_LEN`). A sequência mais longa é truncada, enquanto a sequência mais curta contém *tokens* de *padding* (são adicionados zeros até o vetor atingir o tamanho máximo).

Treinamento

Executando a função **train_test_split()**, dividimos o conjunto de dados em um conjunto de teste e outro de treino. Essa seção é usada para avaliar o desempenho do conjunto de validação e treinar o modelo no conjunto de treinamento.

O modelo é colocado no modo de treinamento (**model.train()**). O treinamento é realizado em várias épocas (definido pelo parâmetro "*epochs*"). Para cada *epoch*, o código itera sobre os lotes de treinamento (**train_dataloader**) e realiza as seguintes etapas:

- Move o lote para a GPU, se disponível.
- Limpa os gradientes acumulados anteriormente.
- Realiza a passagem para frente (*forward pass*) do modelo e calcula a perda (*loss*) em relação às *labels* verdadeiras.
- Realiza a passagem para trás (*backward pass*) para calcular os gradientes e atualiza os parâmetros do modelo com o otimizador.

A perda média por lote, bem como o número de exemplos de treinamento e etapas de treinamento, são monitorados durante o treinamento.

Avaliação

O modelo é colocado no modo de avaliação (**model.eval()**). O código itera sobre os lotes de validação (**validation_dataloader**) e realiza as seguintes etapas:

[7] Aquele que passou somente pela etapa de renomeação das colunas
[8] Aquele que passou pelo processo completo de limpeza textual

- Move o lote para a GPU, se disponível.
- Desativa o cálculo e armazenamento dos gradientes para economizar memória.
- Realiza a passagem para frente do modelo para obter as previsões (`logits`).
- Move as previsões e as *labels* para a CPU.
- Calcula a acurácia para cada lote e acumula o total de acurácia e o número de etapas de avaliação.

A acurácia média é calculada dividindo a acurácia total pelo número de etapas de avaliação.

Classificação

Carregamos o modelo pré-treinado do BERT para classificação de sequências (através da função **BertForSequenceClassification**) com base na versão multilíngue[9] (10) do BERT e definimos aqui que o número de rótulos/classes que o modelo deve prever é 2.

É válido ressaltar que o BERT, por si só, não é um modelo de classificação. Portanto, para utilizar o BERT para classificação de texto, foi necessário fazer o ajuste fino (*fine-tuning*), que consiste em adicionar uma camada de classificação no topo do BERT e treiná-la com dados rotulados para a tarefa de classificação desejada.

O conjunto de dados é carregado novamente para ser testado pelo código. O pré-processamento é executado de forma semelhante à dos dados de treinamento e validação.

O modelo faz previsões nas sequências de teste e calcula o coeficiente de correlação de *Matthew* para o conjunto de dados completo e para cada lote de teste. O coeficiente varia entre -1 e 1, onde 1 indica uma predição perfeita, 0 indica um desempenho aleatório e -1 indica uma discordância total entre as previsões e os rótulos verdadeiros.

2.4.4 BERT - Resultados e conclusões

Relatório de classificação e matriz confusão

O relatório de classificação mostrou que o BERT teve um melhor desempenho quando submetemos a ele o dataset limpo [2.14], obtendo 85% de precisão versus 62% de precisão quando executado no original [2.13].

[9] `bert-base-multilingual-cased`

[BERT | 2 Classes Dataset] - Classification Report:

	precision	recall	f1-score	support
0	0.51	0.98	0.67	6000
1	0.72	0.06	0.12	6000
accuracy			0.52	12000
macro avg	0.62	0.52	0.39	12000
weighted avg	0.62	0.52	0.39	12000

Figura 2.13: BERT - Dataset original - Relatório de classificação

[BERT | 2 Classes Dataset] - Classification Report:

	precision	recall	f1-score	support
0	0.77	0.95	0.85	6000
1	0.94	0.72	0.81	6000
accuracy			0.83	12000
macro avg	0.85	0.83	0.83	12000
weighted avg	0.85	0.83	0.83	12000

Figura 2.14: BERT - Dataset limpo - Relatório de classificação

No geral, o BERT mostrou melhor desempenho na classificação das petições iniciais, mesmo quando não houve tratamento dos dados. Isso já era esperado, visto que, uma das principais diferenças entre os modelos é que o BERT considera o contexto no qual a palavra está inserida, fazendo com que ele assemelhe os documentos de acordo com o contexto formado e não apenas na comparação de palavras.

As matrizes da Figura 2.15 e da Figura 2.16 vêm para corroborar os resultados mostrados nos relatórios.

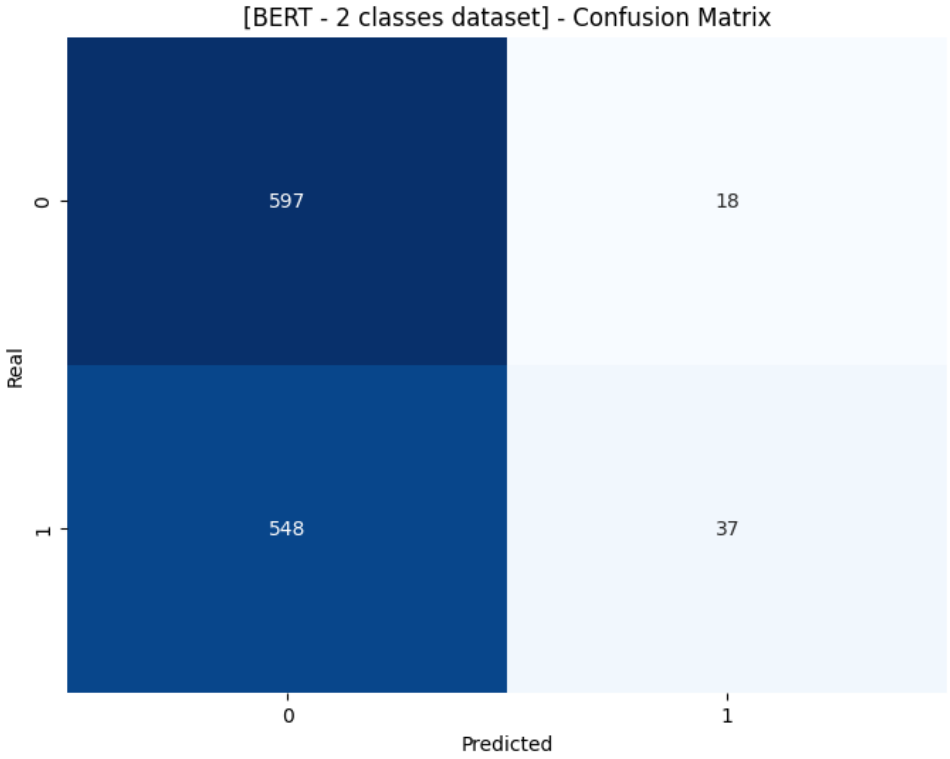

Figura 2.15: BERT - Dataset original - Matriz Confusão

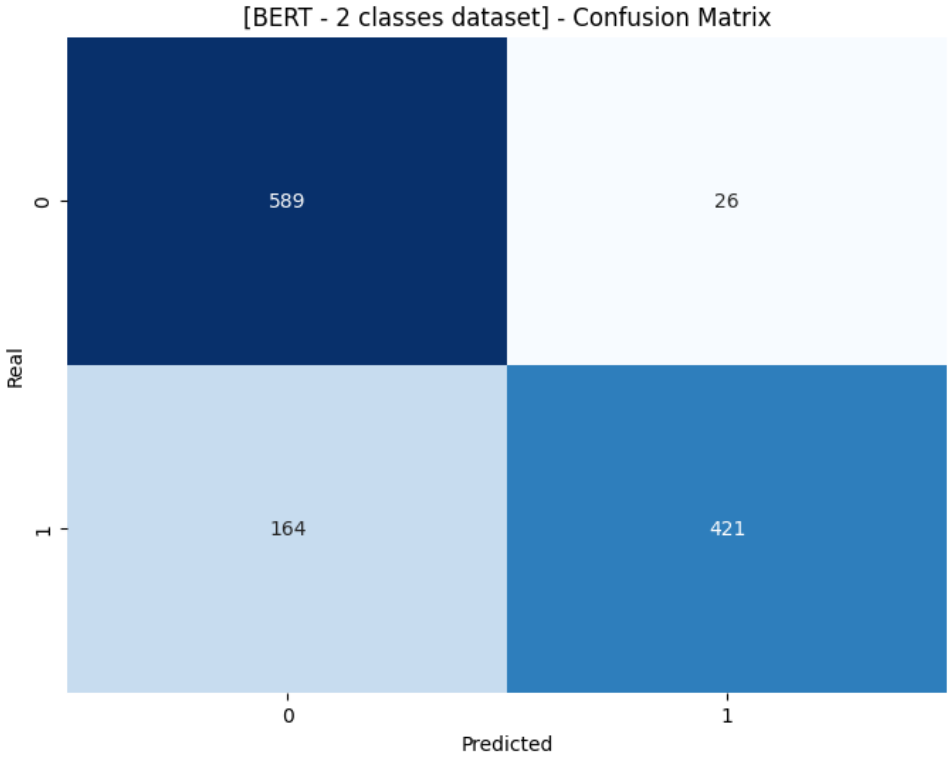

Figura 2.16: BERT - Dataset limpo - Matriz Confusão

No entanto, é interessante observar que quando executado sobre os dados brutos há uma

espécie de enviesamento na classificação. O modelo tendeu a classificar mais instâncias da classe 0 (`dir_consumidor`) que da classe 1 (`dir_tributario`). Isso provavelmente se deve ao fato de, por não haver remoção das *stopwords*, muito provavelmente esses documentos terão um aumento no número de atributos que fez o modelo considerá-los semelhantes.

Agrupamento e análise *Silhouette*

É válido mencionar que não conseguimos efetuar o agrupamento e o cálculo do score *Silhouette*. Infelizmente devido a alta complexidade de diminuir a dimensionalidade do vetor gerado, bem como a limitação de recursos computacionais, as funções de gerar o gráfico K-means e *Silhouette* não chegaram a ser concluídas. Os recursos oferecidos pelo Google Colab PRO foram insuficientes. Apesar de usar apenas 11GB da GPU RAM, esse processo estava levando mais de 6h horas, fazendo com que as unidades computacionais disponíveis se esgotassem antes da conclusão da execução.

Perda no treinamento e MCC

No entanto, fizemos uma análise de perda durante o treinamento. Considerando que a perda é uma medida da discrepância entre as saídas previstas pelo modelo e os rótulos verdadeiros dos dados de treinamento.

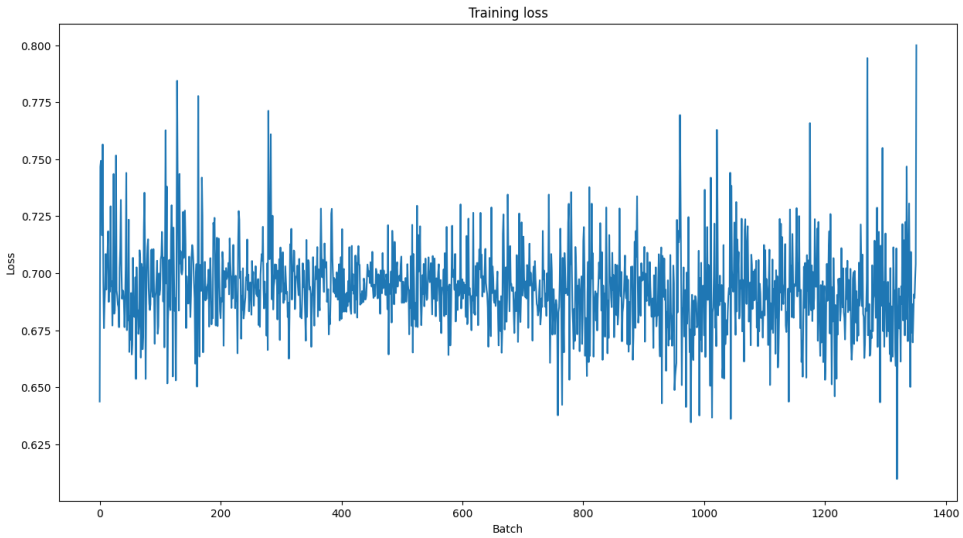

Figura 2.17: BERT - Dataset Limpo - Training Loss

O fato do nosso gráfico de perda apresentar oscilações ao longo das *epochs* pode significar que nosso modelo pode estar com problemas de convergência (o modelo não está aprendendo da maneira mais eficiente) ou que os dados de treinamento não são representativos o suficiente para proporcionar um aprendizado adequado.

Já com relação ao coeficiente de correlação de Matthews (MCC), obtivemos um total de 0.09512930716469688 para o dataset limpo. Isso é um indicativo de baixo desempenho e que o modelo está tendo dificuldade em fazer previsões precisas.

3

Conclusão

Este trabalho teve como objetivo realizar um estudo da aplicabilidade de técnicas de Processamento de Linguagem Natural em petições iniciais. Foram realizadas várias análises, aplicadas diferentes técnicas de NLP e utilizados modelos de NLP clássicos (Doc2Vec) e do estado-da-arte (BERT). A principal conclusão deste trabalho é que as técnicas do modelo BERT foram mais eficazes na análise de Petições Iniciais do que as técnicas do modelo Doc2Vec.

O modelo BERT demonstrou uma precisão de 85% na classificação do conjunto de dados limpos. Por outro lado, os resultados moderados do modelo Doc2Vec mostram que o agrupamento dos documentos não atendeu totalmente à classificação humana.

Neste estudo falamos sobre o pré-processamento dos dados e como a dimensionalidade afeta a performance dos modelos. Verificamos que, na análise de modelos de linguagem grandes (BERT) com técnicas como agrupamento e análise *Silhoutte*, há necessidade de recursos computacionais adequados para sua execução, ressaltando aqui a complexidade do estudo feito.

Os resultados mostram que o modelo BERT, com o tratamento de dados adequado, superou o Doc2Vec na classificação de petições iniciais. No entanto, ainda há desafios a serem resolvidos, como a melhoria da convergência do modelo, ajuste dos hiperparâmetros, o tratamento adequado dos dados e a consideração das limitações computacionais para realizar análises mais abrangentes, como agrupamento e análise Silhouette.

Esses resultados podem melhorar a compreensão do desempenho dos modelos na classificação de petições iniciais e podem ajudar na escolha e implementação de modelos adequados para tarefas semelhantes em vários campos. Considerando ainda a crescente adesão do meio jurídico aos recursos computacionais em seu dia-a-dia, podemos também enfatizar que, aplicações como esta podem ser inseridas no meio jurídico, podendo ajudar a dar maior celeridade aos processos.

Para trabalhos futuros, poderíamos investir mais em validar os modelos em dataset maiores e com mais classe, observando se os resultados obtidos se mantêm. Para o caso do Doc2Vec,

executar o algoritmo utilizando algum modelo pré-treinado e, para o BERT, realizar os testes com o modelo pré-treinado BERTimbau (11) e Albertina PT-* (12).

Outra frente futura de trabalho é avaliar o desempenho de outros modelos de linguagem grande (LLM, `Large Language Models`), tais como RoBERTa (13), GPT-4 (14) e DistilBERT (15), na classificação de petições iniciais.

Bibliografia

Conselho Nacional de Justiça (CNJ). Justiça em números 2022. PDF, 2022.

Tribunal de Justiça de Pernambuco. TJPE disponibiliza ferramenta de inteligência artificial para execução fiscal em programa de formação do CNJ. https://www.tjpe.jus.br/-/tjpe-disponibiliza-ferramenta-de-inteligencia-artificial-para-execucao-fiscal-e 2023. Acessado em: 04 de junho de 2023.

Mamede Said Maia Filho and Tainá Aguiar Junquilho. Projeto victor: perspectivas de aplicação da inteligência artificial ao direito. *Revista de Direitos e Garantias Fundamentais*, 19(3):218–237, 2018.

Michael D Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the annual meeting of the cognitive science society*, volume 27, 2005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

J. Jesus. Tjsp repository. Disponível em: https://github.com/jjesusfilho/tjsp. Acessado em: 04 de junho de 2023.

Gensim - Doc2Vec Tutorial. https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html#sphx-glr-auto-examples-tutorials-run-doc2vec-lee-py. Acessado em: 09 de maio de 2023.

Scikit-learn - Machine Learning in Python. https://scikit-learn.org/stable/. Acessado em 09 de maio de 2023.

Chris McCormick and Nick Ryan. Bert word embeddings tutorial. http://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/, 2019.

Thomas Wolf et al. Bert base, multilingual cased. https://huggingface.co/bert-base-multilingual-cased, 2021.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

João Rodrigues, Luís Gomes, João Silva, António Branco, Rodrigo Santos, Henrique Lopes Cardoso, and Tomás Osório. Advancing neural encoding of portuguese with transformer albertina pt. *arXiv preprint arXiv:2305.06721*, 2023.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

OpenAI. Gpt-4 technical report, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.