UNIVERSIDADE FEDERAL DE SANTA CATARINA

Zero-shot learning e Few-shot learning no desenvolvimento de modelos inteligentes para classificação e análise de documentos jurídicos

Fernando Cesar da Costa Junior

Florianópolis - SC

2024

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Zero-shot learning e Few-shot learning no desenvolvimento de modelos inteligentes para classificação e análise de documentos jurídicos

Fernando Cesar da Costa Junior

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal de Santa Catarina.

Florianópolis - SC
2024

Fernando Cesar da Costa Junior

Zero-shot learning e Few-shot learning no desenvolvimento de modelos inteligentes para classificação e análise de Documentos Jurídicos

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal de Santa Catarina

Orientação:

_____

Prof. Elder Rizzon Santos, Dr.
UFSC/CTC/INE

Banca Examinadora:

_____

Prof. Alexandre Gonçalves Silva, Dr.
UFSC/CTC/INE

_____

Thiago Ângelo Gelaim, Dr.

AGRADECIMENTOS

Agradeço, antes de tudo, a Jesus, que me deu a capacidade de ter chegado até aqui, "Pois todas as coisas vêm dele, existem por meio dele e são para ele. A ele seja toda a glória para sempre. Amém!" (Romanos 16:36). Agradeço a minha esposa pela paciência e incentivos constantes que me deram ânimo para vencer essa etapa. Agradeço também a meus pais pelo esforço e pelo amor que dedicaram a mim. Obrigado por todo apoio e ensinamentos recebidos ao longo de minha vida! Agradeço a minha irmã pelo carinho e pela torcida para meu sucesso. Agradeço ao meu orientador, professor Elder Rizzon Santos, pelo conhecimento e pelos conselhos compartilhados. Por fim, agradeço a todos os professores do INE/UFSC e meus companheiros de graduação, com quem pude aprender bastante.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ML	*Machine Learning*
DP	*Deep Learning*
IA	Inteligência Artificial
PLN	Processamento de Linguagem Natural
FSL	*Few-shot learning*
ZSL	*Zero-shot learning*
GFSL	*Generalized Few-shot learning*
GZSL	*Generalized Zero-shot learning*
ICL	*In-context learning*
GPT	*Generative Pretrained Transformer*
LLM	*Large Language Model*
PLM	*Pre Trained Language Model*
BERT	*Bidirectional Encoder Representations from Transformers*
ALBERT	*A Lite* BERT
RoBERTa	*Robustly Optimized* BERT *Pretraining Approach*
RNR	Rede Neural Recorrente
MAML	*Model-Agnostic Meta-Learning*
GPT	Generative pre-trained transformer

RESUMO

No campo jurídico, é comum enfrentar o desafio de lidar com uma enorme quantidade de documentos e processos que demandam análise e interpretação minuciosas. Esse trabalho exige um esforço considerável por parte dos profissionais do ramo. No entanto, ao empregar técnicas avançadas de Processamento de Linguagem Natural (PLN), é possível obter benefícios significativos, aprimorando a eficiência, a precisão e a acessibilidade desse processo. Este Trabalho de Conclusão de Curso de graduação busca explorar o desenvolvimento do PLN e o papel que essa área pode exercer no campo jurídico, investigando o potencial do *Zero-shot Learning* (ZSL) e *Few-shot Learning* (FSL) na classificação de documentos do âmbito legal. Foram implementados modelos de *Deep Learning* (DL), baseados em *Transformers*, aplicando-os ao Direito, a partir do texto contido em peças judiciais, em cenários com pouca ou nenhuma disponibilidade prévia de exemplos de treinamento para cada classe ou categoria de interesse. Foram comparados o desempenho dos modelos em métodos de ZSL e FSL convencionais, em que as classes são todas não vistas ou pouco vistas no treinamento, e em ZSL e FSL generalizados, em que há classes não vistas ou pouco vistas juntamente com classes muito vistas no treinamento. Os resultados foram promissores quando aplicados o ZSL e o FSL generalizados, em especial com o uso dos modelos JurisBERT e BERTimbau, alcançando-se valores em torno de 90% para acurácia e F1-score em um dos *datasets* utilizados. Mas também foram encontradas dificuldades quando aplicados os métodos de ZSL e FSL tradicionais, obtendo-se desempenhos insatisfatórios mesmo em modelos com pré treinamento especializado em dados jurídicos em português.

Palavras-chave: Processamento de Linguagem Natural; Zero-Shot Learning; Few-Shot Learning.

ABSTRACT

In the legal field, it is common to face the challenge of dealing with a huge amount of documents and processes that require detailed analysis and interpretation. This work requires a specific effort on the part of professionals in the field. However, by employing advanced Natural Language Processing (NLP) techniques, it is possible to obtain significant benefits, improving the efficiency, accuracy, and accessibility of this process. This undergraduate course completion work seeks to explore the development of NLP and the role that this area can play in the legal field, investigating the potential of Zero-shot Learning (ZSL) and Few-shot Learning (FSL) in the classification of documents in the legal field. Deep Learning (DL) models, based on Transformers, were implemented and applied to Law, based on the text contained in legal documents, in scenarios with little or no prior availability of training examples for each class or category of interest. The performance of the models was compared using conventional ZSL and FSL methods, in which all classes are rarely seen or rarely seen in training, and generalized ZSL and FSL, in which there are unseen or seen classes together with classes that are often seen in training. The results were promising when applying generalized ZSL and FSL, especially when using the JurisBERT and BERTimbau models, reaching values around 90% for accuracy and F1-score in one of the datasets used. However, difficulties were also encountered when applying traditional ZSL and FSL methods, with unsatisfactory performances being obtained even in models with specialized pre-training on legal data in Portuguese.

Keywords: Natural Language Processing; Zero-Shot Learning; Few-Shot Learning.

1 INTRODUÇÃO

O Processamento de Linguagem Natural (PLN) tem desempenhado um papel cada vez mais importante na área da Inteligência Artificial, permitindo que as máquinas compreendam e processem a linguagem humana (Nadkarni; Ohno-Machado; Chapman, 2011). Com o avanço das técnicas de *Machine Learning* (ML), surgiram abordagens inovadoras, como o *Few-Shot Learning* (FSL) e o *Zero-Shot Learning* (ZSL), que têm sido exploradas em diversas esferas e desempenham um papel importante no desenvolvimento do campo do PLN.

Conforme Brown et al. (2020), em "*Language Models are Few-Shot Learners*", o *Few-Shot Learning* refere-se à capacidade de um modelo aprender com um número limitado de exemplos de treinamento. O mesmo autor também explica que o *Zero-Shot Learning* possui o enfoque no aprendizado com classes ou conceitos para os quais o modelo não foi explicitamente treinado. Essas abordagens são significativas, pois, os modelos usados em tarefas de aprendizado tradicionais, como, por exemplo, classificadores baseados em Redes Neurais Artificiais, costumam requerer grandes volumes de dados para generalizar bem para novos exemplos (Kavzoglu, 2009). No entanto, em muitos cenários do mundo real, a coleta de dados pode ser cara, demorada ou simplesmente inviável.

Os desafios para a obtenção de uma grande quantidade de dados qualificados envolvem extensas anotações manuais para rotulá-los em categorias e a dependência de expertise de domínio para uma classificação precisa. Nessa conjuntura, o domínio jurídico apresenta um terreno fértil para a aplicação de estratégias de ZSL e FSL, pois aquela área tem como uma de suas características o dispendioso trabalho de análise e interpretação de documentos textuais e processos formais, demandando um esforço expressivo por parte dos profissionais do Direito. Além disso, há situações em que o número de documentos jurídicos pode ser limitado, levando a uma pouca ou nenhuma disponibilidade de exemplos para treinamento para as classes ou categorias de interesse. Então, a aplicação de técnicas avançadas de *Machine Learning*, especificamente o ZSL e o FSL, no âmbito do PLN, surge como uma solução capaz de proporcionar benefícios substanciais a partir da contribuição para a otimização dos processos legais e da abertura de caminho para a automação de tarefas complexas e o acesso mais rápido e efetivo ao conhecimento jurídico, mesmo com poucos dados.

Segundo Polo et al. (2021), as aplicações dos métodos de PLN no Direito, em muitos países, estão cada vez mais presentes e representam um futuro cada vez mais promissor. Como exemplo, Aguiar (2024) traz em seu trabalho o caso do Conselho Nacional de Justiça

(CNJ), que, em 2021, identificou a necessidade urgente de automatizar processos judiciais no Brasil, envolvendo o uso de robôs e de inteligência artificial, especialmente de aprendizado de máquina, visando aumentar a produtividade do judiciário e liberar os profissionais para tarefas mais complexas.

A utilização de modelos de ML em cenários ZSL e FSL com o intuito de aprimorar a eficiência, a precisão e a acessibilidade na análise de textos jurídicos requer a execução de tarefas de PLN como a Classificação de texto, que representa uma tarefa chave neste trabalho. A classificação de texto envolve atribuir rótulos a documentos ou partes de documentos, com base em seu conteúdo, tornando-se útil para organizar os dados em categorias específicas.

Logo, o propósito deste trabalho é estudar o uso das técnicas FSL e ZSL na execução de tarefas de classificação de documentos, dentro do contexto jurídico, nos casos em que a disponibilidade de dados é restrita, e como tais abordagens podem ampliar a aplicabilidade de modelos de PLN em ocorrências do mundo real.

De maneira mais específica, vislumbra-se neste trabalho a oportunidade de pesquisar, analisar e comparar diferentes modelos e estratégias de ML que, com poucos dados, sejam capazes de classificar peças judiciais em português automaticamente em categorias específicas, superando os desafios relacionados à escassez de dados rotulados e à necessidade de generalização para casos não vistos anteriormente. Com isso, possibilita-se uma forma de lidar com novos tipos de documentos e situações no terreno do Direito, mesmo sem exemplos rotulados disponíveis, facilitando a organização e a recuperação de informações relevantes.

2 OBJETIVOS

2.1 OBJETIVO GERAL:

Explorar e avaliar o desempenho de diferentes algoritmos e modelos de Processamento de Linguagem Natural na classificação de documentos e informações jurídicas em português a partir das abordagens de *Zero-shot learning* e *Few-shot learning*.

2.2 OBJETIVOS ESPECÍFICOS:

- Analisar o estado da arte em Processamento de Linguagem Natural (PLN) e nas técnicas de aprendizagem de máquina *Zero-shot learning* e *Few-shot learning*.

- Aplicar método de pré-processamento de texto para documentos jurídicos, considerando sua eficácia para a tarefa de classificação.
- Aplicar as abordagens *Zero-shot learning* e *Few-shot learning* em dados textuais jurídicos em português.
- Realizar experimentos comparativos para avaliar o desempenho das abordagens de PLN escolhidas aplicadas aos dados jurídicos, identificando suas vantagens e limitações.
- Examinar os resultados obtidos e contrastar as diferentes abordagens de PLN.
- Identificar lacunas na pesquisa existente em PLN aplicado ao Direito e Gestão de processos jurídicos e propor possíveis direções para futuros trabalhos.

3 MÉTODO DE PESQUISA

A pesquisa proposta neste trabalho é de natureza exploratória, com o objetivo de investigar e compreender o uso de técnicas de PLN e, em especial, da abordagem FSL e do ZSL na área do Direito.

Foram realizados experimentos sobre uma importante tarefa em PLN: A classificação de texto. No estudo da abordagem FSL, vislumbra-se a utilização de aprendizado de máquina supervisionado na classificação de texto, pois requer um conjunto de dados rotulados, ou seja, pares de entrada e saída, para que os modelos possam aprender a mapear os dados de entrada para as categorias desejadas e que serão pré-definidas. Para a abordagem ZSL, conforme Schopf, Braun e Matthes (2022), considera-se que essa se trata de uma estratégia não supervisionada, pois não requer treinamento ou ajuste fino nos dados rotulados das classes alvo. Além do ZSL e FSL convencionais, foram testados variações desses paradigmas: ZSL e FSL generalizados.

O desenho metodológico adotado envolve a coleta, preparação e análise de dados jurídicos, bem como a implementação de algoritmos de *deep learning*, escolhidos a partir de pesquisa realizada tanto na literatura como em artigos acadêmicos relevantes, relacionados às abordagens supracitadas.

O método de pesquisa ainda passa por realizar uma avaliação do desempenho geral (considerando todas as classes) dos vários modelos propostos nas abordagens estudadas, comparando-os com os resultados das execuções dos mesmos modelos sobre os dados completos dos *datasets* utilizados, ou seja, sem restrições de amostras, de forma a garantir uma *baseline* contextualizada com o problema de pesquisa definido. Para isso, métricas

adequadas, como a acurácia, f1-*score, recall* e precisão, são usadas para, então, interpretar os resultados obtidos, discutindo as implicações no contexto jurídico.

Ao final, vantagens ou possíveis limitações no uso da abordagem de ZSL e FSL no campo legal são identificadas, sugerindo-se melhorias ou direções para pesquisas futuras.

4 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz ao leitor os conceitos necessários para entender como funcionam as tecnologias deste trabalho e os motivos de serem utilizadas.

4.1 PROCESSAMENTO DE LINGUAGEM NATURAL

A linguagem natural diz respeito à forma como nos comunicamos uns com os outros, seja de maneira falada ou de maneira escrita (Brownlee, 2017). Kochmar (2022) afirma que a capacidade de domínio da linguagem natural representa um elemento central da inteligência humana, pois quase todas as atividades desempenhadas pelo ser humano envolvem algum uso da linguagem. Em vista disso, há um notório interesse de pesquisadores no âmbito da Inteligência Artificial no desenvolvimento da área de Processamento de Linguagem Natural (PLN), que abrange a análise e a representação da linguagem natural, tanto fonética quanto escrita (Polo et al., 2021), exigindo a aplicação de algoritmos de aprendizado de máquina, processamento estatístico e técnicas avançadas de processamento de texto (Bird, Klein, & Loper, 2009).

O início dos estudos nessa seara, segundo Nadkarni, Ohno-Machado e Chapman (2011), se deu na década de 1950, como a interseção da Inteligência Artificial (IA) e da Linguística, que é a ciência que tem por objeto a linguagem humana em seus aspectos fonético, morfológico, sintático, semântico, social e psicológico. O objetivo do PLN é permitir que as máquinas compreendam, interpretem e gerem linguagem humana de forma a facilitar a comunicação entre humanos e computadores (Jurafsky e Martin, 2009).

Esse campo de estudo desafia questões linguísticas intrincadas, como a ambiguidade, o contexto, erros ortográficos e a variação linguística. Isso trouxe muitos desafios no desenvolvimento inicial do PLN, pois, na época, seguia-se uma abordagem baseada em regras, que eram codificadas por especialistas, conferindo um conhecimento aos sistemas (Kochmar, 2022). Havia confiabilidade, mas, por outro lado, maior rigidez. Com isso, houve uma reorientação na década de 1980, resultando no nascimento do PLN estatístico, que dão

bons resultados ao se treinar modelos estatísticos com dados abundantes e representativos (Nadkarni; Ohno-Machado; Chapman, 2011). As abordagens estatísticas não fazem suposições e tentam aprender com os dados (Kochmar, 2022). Nesse mesmo período, técnicas de *machine learning* começaram a ser usadas em complemento às abordagens estatísticas, com a aplicação de algoritmos que aprendiam as distribuições estatísticas dos padrões linguísticos contidos nos dados (Zang e Teng, 2021). Com a explosão da quantidade de dados produzidos a partir do surgimento da *World Wide Web* na década de 1990 e os avanços das tecnologias de *hardware*, a partir dos anos 2010, a abordagem com *deep learning* passou a ser proeminente na área de PLN (Kochmar, 2022), mostrando resultados empíricos mais fortes em comparação aos métodos estatísticos tradicionais para uma ampla gama de tarefas de PLN (Zang e Teng, 2021).

Entre os recentes avanços nessa área, LeCun et al. (2015) destacam a importância do desenvolvimento de modelos de representação distribuída, como *word embeddings*, que capturam relações semânticas entre palavras. Mas o advento das Redes Neurais Recorrentes (RNRs) e, mais recentemente, das arquiteturas baseadas em *Transformers* revolucionou o campo nos últimos anos. A arquitetura *Transformer*, introduzida por Vaswani et al. (2017), tornou-se fundamental em tarefas de PLN, permitindo a atenção contextualizada, em que o modelo considera não apenas a palavra em si, mas também as palavras ao seu redor e suas relações contextuais.

O PLN desempenha um papel crucial na análise de texto, fornecendo *insights* em áreas como pesquisa acadêmica, análise de mídia social, extração de informações em documentos jurídicos, entre muitas outras (Manning & Schütze, 1999). Também nos deparamos com PLN em tarefas como recuperação de informações em motores de busca, predição de texto e geração de linguagem, assistentes virtuais inteligentes, filtro de *spam* e tradução automática (Kochmar, 2022, p. 4).

À medida que a quantidade de dados de texto disponíveis na internet e em repositórios digitais cresce exponencialmente, a capacidade de extrair informações significativas desses dados torna-se cada dia mais fundamental.

4.1.1 Pré-processamento

Os dados de texto são desestruturados. Há muitos elementos inúteis para processamento por máquina presente no texto, exigindo um pré-processamento sobre os dados brutos (Kulkarni, Shivananda e Kulkarni, 2022). Essa etapa é vital, porque o texto indesejado

pode alterar a direção dos resultados que ele produz (Kulkarni, Shivananda e Kulkarni, 2022). As técnicas utilizadas no pré-processamento de texto incluem remoção de palavras irrelevantes (*stopwords*), padronização de texto, normalização de texto (conversão de palavras para uma forma comum), *stemming* e *lemmatization* (redução das palavras às suas raízes ou formas de dicionário para padronizar variações da mesma palavra), *part-of-speech tagging* (identificação da classe gramatical de cada palavra), entre outras.

a) *Tokenization*

Tokenization ou Tokenização é a identificação de cada unidade "atômica" de um texto, representando a primeira operação a ser realizada no processamento de documentos (Habert, 1998). Isso permite dividir um fluxo de caracteres em unidades linguísticas.

Segundo Minaee et al. (2024), embora a ferramenta de tokenização mais simples corte o texto em *tokens* com base em espaços em branco, a maioria das ferramentas de *tokenização* depende de um dicionário de palavras, e para aumentar a cobertura dos dicionários, os *tokenizers* populares usados para modelos de linguagens grandes são baseados em subpalavras, que podem ser combinadas para formar um grande número de palavras, incluindo palavras não vistas nos dados de treinamento ou palavras em diferentes idiomas.

Como exemplo, usando a tokenização feita a partir da divisão do texto em palavras, a frase originalmente escrita "O juiz proferiu uma sentença de improcedência do pedido" seria convertida para a seguinte lista de *tokens* (palavras, neste caso): ["O", "juiz", "proferiu", "uma", "sentença", "de", "improcedência", "do", "pedido"].

Os três tokenizadores mais populares o *Byte Pair Encoding*, *Word Piece Encoding* (usado principalmente para modelos muito conhecidos como BERT e Electra) e o *Sentence Piece Encoding*.

b) Remoção de *Stopwords*

Stopwords são palavras consideradas sem importância em uma frase que geralmente não adicionam muito significado semântico a um texto, como artigos, preposições, conjunções. Por exemplo, no texto "O produto é muito bom, a qualidade é excelente e o preço é justo, logo eu recomendo para todos", o resultado da remoção de *stopwords* seria: Produto muito bom qualidade excelente preço justo recomendo todos.

O uso dessa técnica deve ser considerada com cuidado, pois pode prejudicar modelos que usam contexto de palavras para detectar significado semântico (Campesato, 2021).

c) *Stemming*

O *stemming* é um processo que consiste em remover os afixos (prefixos e sufixos) de uma palavra para obter sua raiz, ou *stem*, cuja ideia é reduzir as palavras flexionadas a uma forma comum, mesmo que essa forma não seja necessariamente uma palavra válida no dicionário.

d) *Lemmatization*

Lemmatization é um processo mais sofisticado que leva em consideração o contexto da palavra e busca a sua forma canônica, ou lema, que é a forma dicionarizada da palavra, podendo facilitar a análise e a comparação de textos ao permitir uma extração de informações mais eficiente. Por exemplo, para o conjunto de palavras "correndo", "corrida" e "corredor", o lema associado é "correr".

e) *Part-of-speech (POS) tagging*

Part-of-speech tagging é uma técnica em PLN que consiste em atribuir uma etiqueta a cada palavra em um texto, indicando a função gramatical dessa palavra na frase, como substantivo, verbo, adjetivo, advérbio, preposição, conjunção. Essa tarefa acontece após aplicação da tokenização e permite uma análise da estrutura gramatical de um texto, facilitando a compreensão do seu significado.

Conforme Rathod e Govilkar (2015), as técnicas de *POS tagging* são tradicionalmente baseadas em regras, em estatística ou podem ser híbridas. Um *tagger* baseado em regras usa léxico ou dicionário onde as palavras são armazenadas junto com as informações sintáticas e, caso haja mais de uma categoria sintática possível para uma palavra, regras baseadas em gramática são aplicadas. Por outro lado, um *tagger* estatístico seleciona a sequência mais provável de *tags* de um corpus de texto, com base em evidências estatísticas anteriores. Já a abordagem híbrida é uma combinação de abordagem baseada em regras e abordagem estatística, que atribui a *tag* mais provável à palavra usando estatística e, depois disso, se uma ambiguidade for encontrada, aplicam-se regras gramaticais para definição da etiqueta.

4.1.2 Transformação de dados

4.1.2.1 *Word embedding*

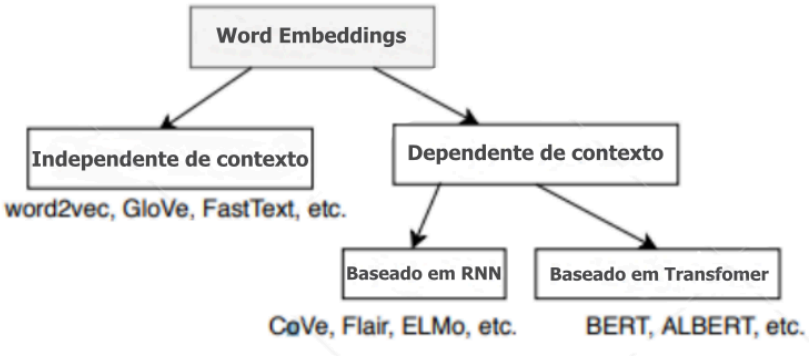
Word embedding é uma técnica utilizada em PLN para representar palavras como vetores densos (de valores reais), distribuídos (em várias dimensões) e de comprimento fixo, cuja construção usa estatísticas de coocorrência de palavras de acordo com a hipótese distributiva (Almeida e Xexéo, 2019). Ou seja, essa representação numérica permite mapear cada palavra de um vocabulário para um ponto em um espaço vetorial (Khattak et al., 2019), de forma que palavras que ocorrem no mesmo contexto, cujas representações vetoriais são mais próximas, têm semântica semelhante (Wang, Zhou e Jiang, 2020). A ideia por trás de *word embeddings* é a de que palavras com significados semelhantes provavelmente serão usadas juntas em frases com mais frequência (Singh, 2023). Ao capturar as relações semânticas entre as palavras, desde as mais simples, como as baseadas no mero significado, até as mais complexas, como os relacionamentos entre palavras de diferentes significados, mas de um mesmo contexto, os modelos de PLN são habilitados por essa a técnica a entender e processar melhor o texto.

Incitti, Urli e Snidaro (2023) afirmam que os *embeddings* de palavras evoluíram a ponto de conseguiram obter representações relevantes do significado das palavras com grandes níveis de precisão. Por outro lado, esses pesquisadores também dizem que, em representações para trechos de texto mais longos, os desafios e problemas a superar são maiores.

Segundo Wang, Nulty e Lillis (2020), em termos gerais, existem dois tipos principais de word embeddings: *Word embeddings context-independent* (independentes de contexto) e *word embeddings context-dependent* (dependentes de contexto ou que capturam contexto). Na taxonomia por eles proposta, o último tipo ainda pode ser subdividido em RNN-*based* (baseada em RNR) e *Transformer-based* (baseada em Transformadores).

A Figura 1 mostra exemplos de métodos para *word embeddings* conforme a classificação apresentada. Podemos complementar a lista de métodos do tipo *context-independent* com alguns mais tradicionais como o *Bag of Words* - BoW - e o *Term Frequency – Inverse Document Frequency* - TF-IDF - (Incitti, Urli e Snidaro, 2023). Os métodos Glove - *Global Vectors for Word Representation* - (Pennington et. al., 2014) e Word2Vec (Mikolov et. al., 2013) destacam-se nesse rol.

Figura 1 - Métodos para *word embeddings*



Fonte: Adaptado de Wang, Nulty e Lillis (2020)

Os métodos *context-dependent*, como ELMo e BERT, mais recentemente, passaram a sobressair por produzir melhorias significativas em tarefas de PLN (Ethayarajh, 2019). Segundo Albrecht, Ramachandran e Winkler (2021), o principal benefício dos métodos *context-dependent* é a aprendizagem por transferência, com a capacidade de usar um modelo de linguagem pré-treinado e ajustá-lo para tarefas posteriores específicas. De acordo com Wang, Nulty e Lillis (2020), normalmente, existem duas maneiras pelas quais as *embeddings* contextualizadas podem ser aproveitadas para tarefas posteriores. Em primeiro lugar, eles podem ser usados como extratores de *features* fixas, em que a representação obtida é usada como recurso estático no modelo *downstream*, ou seja, no modelo usado posteriormente e que é treinado para uma tarefa específica. Em segundo lugar, os *embeddings* pré-treinados podem ser ajustados em um conjunto de dados *downstream*, mais específico em uma determinado domínio, para fornecer recursos de entrada de uma sequência de dados, em um processo denominado “ajuste fino”.

Wang, Nulty e Lilis (2020) complementam que, por meio do ajuste fino, os modelos baseados em *Transformer*, usados para representar uma sequência, alcançaram um forte desempenho de última geração para muitas tarefas de PLN, embora os extratores fixos ofereçam uma vantagem de não requererem treinamento de parâmetros e, consequentemente, consomem menos tempo e memória.

4.1.3 Tarefa de PLN

4.1.3.1 Classificação de Documentos

Conforme Kulkarni, Shivananda e Kulkarni (2022), a tarefa de ML de classificação diz respeito a uma técnica de aprendizado de máquina supervisionada em que a variável dependente é categórica, podendo ser binária ou multiclasse. As classes possíveis são conhecidas antecipadamente e imutáveis, sendo elas frequentemente, mas nem sempre, mutuamente exclusivas. No caso em que as instâncias de dados recebem potencialmente rótulos sobrepostos, a classificação é definida como multirrótulo (Fields; Chovanec; Madiraju, 2024).

Quando se lida com dados textuais, a classificação, de acordo com Patwardhan, Marrone e Sansone (2023), representa a categorização de um texto em uma ou mais categorias predefinidas com base em seu conteúdo, atribuindo um rótulo a esse texto para que ele seja organizado de modo a facilitar sua análise e seu gerenciamento.

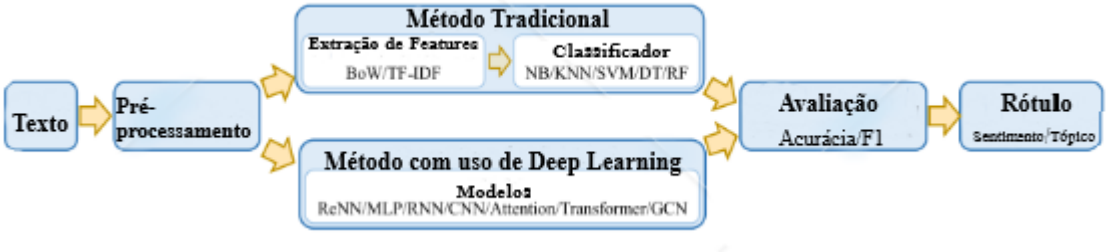
Campesato (2021) argumenta que, do ponto de vista comercial, os algoritmos de aprendizado de máquina para classificação de texto são valiosos quando estruturam e analisam texto de maneira econômica, agilizando os processos de negócios e a tomada de decisão. Nessa linha, Qian Li et al. (2022) declaram que é desejável utilizar métodos de ML para automatizar o procedimento de classificação de texto para produzir resultados mais confiáveis e menos subjetivos, uma vez que a classificação manual de grandes quantidades de dados textuais pode ser facilmente influenciada negativamente por fatores humanos, como fadiga.

Os métodos de classificação de texto podem ser categorizados, de forma geral, em duas abordagens principais: Algoritmos tradicionais de aprendizado de máquina e métodos de aprendizado profundo. Até a década de 2010, algoritmos tradicionais, como SVM, NB e RF, dominaram a classificação de texto. Eles contam com recursos artesanais e modelos estatísticos para classificar documentos de texto. Dessa forma, Qian et al. (2022) alegam que a eficácia desses métodos tradicionais é amplamente restrita à etapa de extração de recursos (*feature extraction*) tradicional, que é o processo feito por um especialista humano de identificar e quantificar características relevantes em dados textuais e que se baseiam em regras ou em técnicas estatísticas, como TF-IDF (*Term Frequency - Inverse Document Frequency*). Além disso, esses últimos pesquisadores citados lembram que a extração tradicional de recursos geralmente desconsidera a estrutura sequencial ou as informações contextuais, tornando desafiador a obtenção de informações semânticas dos textos.

Por outro lado, os métodos de aprendizagem profunda, proeminentes a partir da década de 2010, aproveitam arquiteturas de redes neurais para aprender automaticamente representações de dados de textos brutos, permitindo uma classificação mais robusta e precisa

(Palaniappan; Vedhamani; Sundharakumar, 2023). Isso significa que esses modelos de DL integram a engenharia de recursos, aprendendo um conjunto de transformações não lineares em suas camadas mais internas, que mapeiam características importantes dos dados (*features*) diretamente para as saídas do modelo, garantindo mais precisão e estabilidade e sendo mais úteis em abordagens orientadas a dados com alta complexidade computacional (Qian et al. 2022). A Figura 2 demonstra um fluxograma das etapas envolvidas na classificação de texto com os diferentes métodos.

Figura 2 - Fluxograma da classificação de texto usando métodos tradicionais de ML e PLN ou usando métodos de DL



Fonte: Adaptado de Qian et al., 2022

Wang, Pan e Lin (2023) afirmam que a classificação de texto é considerada a tarefa mais fundamental e crucial no campo do processamento de linguagem natural, sendo abordada em inúmeras aplicações práticas como análise de sentimento e rotulagem de tópicos. Eles prosseguem dizendo que, devido à explosão de informações, processar e classificar manualmente grandes quantidades de dados de texto é demorado e desafiador, tornando indispensável o uso de métodos de aprendizado de máquina.

Patwardhan, Marrone e Sansone (2023) complementam dizendo que, em geral, há duas subcategorias de classificação de texto proeminentes:

• Classificação de Documentos: Esta tarefa envolve atribuir um rótulo ou categoria a um documento completo, como um artigo de notícias, postagem de *blog* ou artigo científico. O rótulo pode ser definido com base no assunto do texto do documento, ou na identificação do autor do documento, ou em que período ele foi produzido, por exemplo. A classificação de documentos normalmente é realizada primeiro representando o documento como um vetor numérico e depois usando um modelo de aprendizado de máquina para fazer uma previsão com base na representação do documento.

• Classificação de causa e efeito: Esta tarefa envolve a identificação da relação de causa e efeito entre dois eventos descritos em uma frase ou parágrafo. Por exemplo, na frase

"A chuva causou o cancelamento do jogo", a classificação identificaria a “chuva” como causa e o “cancelamento do jogo” como o efeito.

4.2 MACHINE LEARNING

Machine Learning (ML) ou Aprendizado de Máquina é uma parte do paradigma da inteligência artificial e pode ser considerado um subcampo da IA focada em fazer com que as máquinas aprendam com os dados e em permitir que as máquinas tomem decisões sem a intervenção humana (Kulkarni, Shivananda e Kulkarni, 2022). Conforme afirmado por Janiesch, Zschech, e Heinrich (2021, apud Bishop, 2006), ML visa automatizar a tarefa de construção de modelos analíticos para realizar tarefas cognitivas como detecção de objetos ou tradução de linguagem natural. Isto é alcançado através da aplicação de algoritmos que aprendem iterativamente a partir de dados de treinamento específicos do problema, o que permite que os computadores encontrem *insights* ocultos e padrões complexos sem serem explicitamente programados.

Segundo Géron (2019), ML pode ser útil em problemas para os quais as soluções existentes exigem muitos ajustes manuais ou longas listas de regras, como problemas complexos para os quais não existe uma boa solução utilizando uma abordagem tradicional ou problemas que precisam se adaptar a uma grande quantidade de novos dados.

Os principais estilos de aprendizado de máquina, de acordo com Campesato (2021), são os de aprendizagem supervisionada, de aprendizagem não supervisionada, de aprendizagem semi supervisionada, de aprendizagem ativa (subconjunto de aprendizado não supervisionado), de aprendizagem auto-supervisionada, de aprendizagem fracamente supervisionada e de aprendizagem por reforço. Dentre esses, os mais usados estão apresentados nos subtópicos seguintes.

4.2.1 Aprendizado Supervisionado

O aprendizado supervisionado é uma parte do ML em que dados de treinamento rotulados são aproveitados para derivar o padrão ou função e criar modelos (Kulkarni; Shivananda; Kulkarni, 2022). Os dados consistem em uma variável dependente (rótulo de destino) e nas variáveis independentes ou preditores. A máquina tenta aprender a função a partir de dados rotulados e prever a saída em dados não vistos. (Kulkarni; Shivananda; Kulkarni, 2022).

Entre os algoritmos de aprendizagem supervisionada mais importantes estão KNN (K-*nearest neighbors* ou K-vizinhos mais próximos), regressão linear, regressão logística, máquinas de vetores de suporte, árvores de decisão, florestas aleatórias e algumas redes neurais.

4.2.2 Aprendizado Semi Supervisionado

Géron (2019) explica que alguns algoritmos podem lidar com dados de treinamento parcialmente rotulados, geralmente muitos dados não rotulados e alguns dados rotulados, definindo o tipo de aprendizagem semi supervisionada, em que a maioria dos algoritmos são combinações de algoritmos não supervisionados e supervisionados, como as *deep belief network* (DBN) ou redes de crenças profundas.

4.2.3 Aprendizado Não Supervisionado

Aqui, a máquina aprende o padrão oculto sem aproveitar os dados rotulados, de modo que o treinamento não acontece. Em vez disso, esses algoritmos aprendem com base em semelhanças ou distâncias entre recursos para capturar os padrões (Kulkarni; Shivananda; Kulkarni, 2022).

As tarefas não supervisionadas mais comuns são agrupamento, detecção de anomalias e detecção de novidades, visualização e redução de dimensionalidades e regras de associação (Géron, 2019).

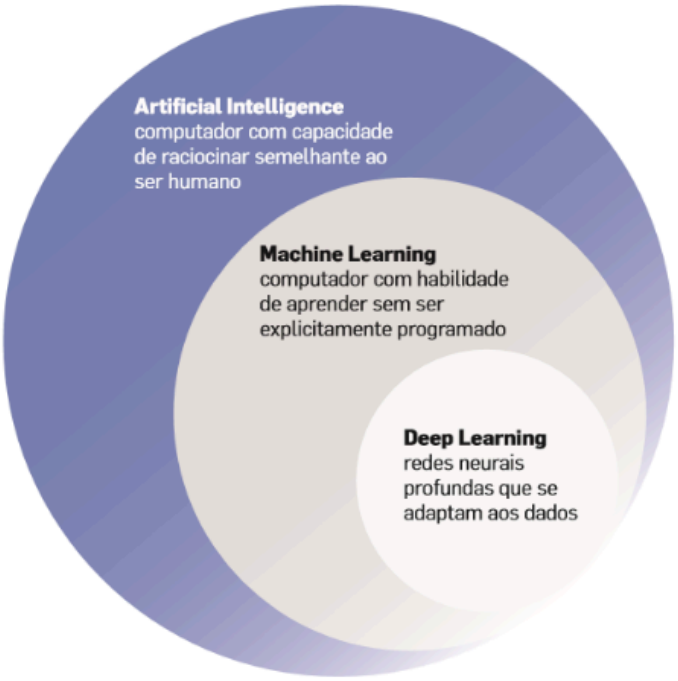
4.2.4 Aprendizado por reforço

Segundo Russell e Norvig (2021), na aprendizagem por reforço, um agente tomador de decisão aprende com uma sequência de sinais de recompensa, que indicam a qualidade do seu próprio comportamento, otimizando a soma das recompensas futuras. Dessa forma, o objetivo da aprendizagem por reforço, de acordo com Haykin (2009), é minimizar uma função de custo restante, definida como a expectativa do custo cumulativo das ações realizadas ao longo de uma sequência de etapas, em vez de simplesmente minimizar o custo imediato de uma ação.

4.3 DEEP LEARNING

Deep Learning (DL) ou Aprendizado Profundo é o subcampo da inteligência artificial e do aprendizado de máquina (Figura 3) que se concentra na criação de grandes modelos de redes neurais que são capazes de tomar decisões precisas baseadas em dados, sendo particularmente adequada para contextos onde os dados são complexos e onde existem grandes conjuntos de dados disponíveis (Kelleher, 2019).

Figura 3 - Hierarquia de domínios de DL, ML e IA



Fonte: Alessandro Cauduro, 2018[1]

Segundo Raff (2022), DL é obtido a partir da combinação de algoritmos que atuam como blocos de construção, os quais devem ser unidos para criar um modelo maior para o problema. Cada bloco de construção é projetado para resolver determinados problemas, fornecendo informações importantes ao modelo. Alguns dos blocos são mais genéricos e podem ser aplicados a uma variedade maior de situações, enquanto outros são mais específicos, afetando quando e como são usados.

---

[1] Disponível em: <https://www.meioemensagem.com.br/opiniao/deep-learning-o-motor-dos-negocios-na-era-da-inteligencia-artificial>. Acesso em 5 de maio de 2024.

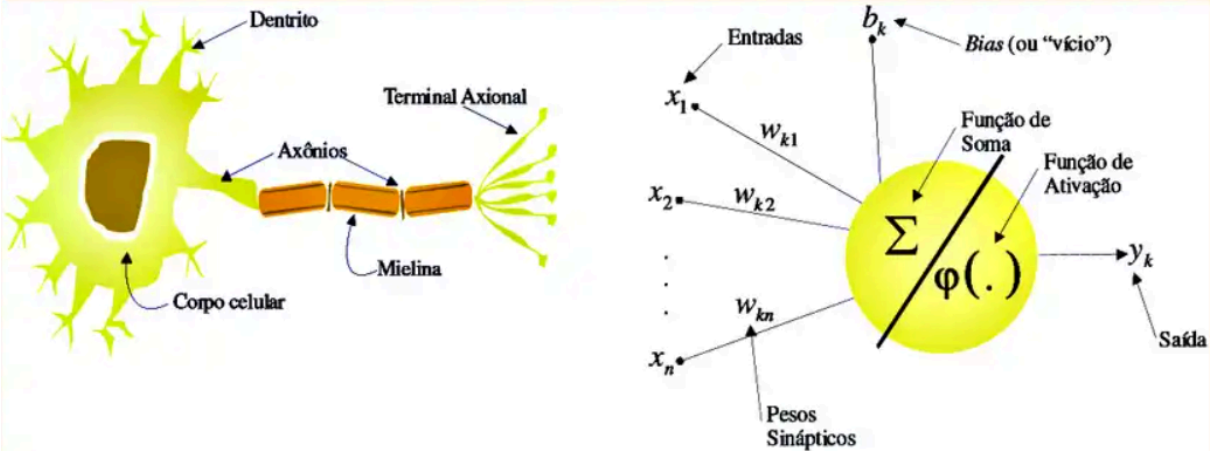
O aprendizado profundo é atualmente a abordagem mais amplamente utilizada em aplicações de Visão Computacional e de PLN (Russell e Norvig, 2021), devido a sua capacidade de modelar relacionamentos intrincados e não lineares em dados (Wang; Pan; Lin, 2023). Segundo Wang, Pan e Lin (2023), os métodos de ML supervisionado tradicionais, como Regressão Logística, K-vizinhos mais próximos e *Naive Bayes*, enfrentam limitações significativas em tarefas de classificação, principalmente porque requerem grandes quantidades de dados rotulados e específicos de tarefas para treinamento antes de poder prever com eficácia os resultados em novos dados, além de serem incapazes de categorizar os dados em classes que não são vistas ou não são rotuladas nos dados de treinamento.

4.3.1 Redes Neurais Artificiais

Os blocos de construção definidos no subtópico anterior são, abstratamente, camadas de uma Rede Neural Artificial (RNA). Essas camadas podem ser empilhadas (Raff, 2022), em que a saída gerada em uma camada alimenta a entrada da próxima camada. O empilhamento sucessivo gera um rede profunda.

Gershenson (2003) afirma que uma RNA é um modelo computacional inspirado nos neurônios naturais do cérebro. O cérebro consiste em um grande número de elementos altamente conectados chamados neurônios, que possuem três componentes principais: o corpo celular; os dendritos, que transportam sinais elétricos para o corpo celular; e o axônio, que transporta o sinal do corpo celular para outros neurônios (Hagan et al., 2014). Gershenson (2003) explica que, quando os sinais recebidos são fortes o suficiente e ultrapassam um certo limite, o neurônio é ativado e emite um sinal através do axônio. Este sinal pode ser enviado para outra sinapse e ativar outros neurônios. De forma similar, continuando em seu raciocínio, uma RNA consiste basicamente em entradas (como sinapses), que são multiplicadas por pesos (força dos respectivos sinais), e depois computadas por uma função matemática que determina a ativação do neurônio, havendo outra função para o cálculo da saída do neurônio artificial. Além disso, as RNAs combinam neurônios artificiais para processar informações. A Figura 4 representa as semelhanças entre os dois conceitos de neurônios.

Figura 4 - Neurônio biológico (à esquerda) e Neurônio em RNA (à direita)



Fonte: Pedro Siqueira, 2024[2]

O surgimento das Redes Neurais Artificiais (RNAs) foi motivado pelo reconhecimento de que o cérebro humano processa dados, muitas vezes, mais rápido do que o computador digital mais rápido que existe hoje, organizando seus constituintes estruturais, os neurônios, em redes neurais para realizar tarefas como, por exemplo, reconhecimento de padrões, percepção e controle motor (Haykin, 2009). Na década de 1960, foi demonstrado que as redes dos modelos neurais têm propriedades semelhantes às do cérebro: elas podem realizar reconhecimento sofisticado de padrões e podem funcionar mesmo se alguns dos neurônios forem destruídos (Krogh, 2008).

Hagan et al. (2014) explanam que a base para o campo das RNAs surgiu no final do século XIX e início do século XX, como resultado, principalmente, de trabalhos interdisciplinares em física, psicologia e neurofisiologia, que apresentaram teorias gerais de aprendizagem, visão, condicionamento, sem modelos matemáticos específicos de operação de neurônios. Hagan et al. (2014) estabelecem alguns marcos no desenvolvimento histórico de RNAs: o desenvolvimento do *perceptron* (RNA de única camada) e a introdução da regra de aprendizagem associativa (que descreve como as conexões entre os neurônios são modificadas com base na entrada e na saída da rede) na década de 50 por Frank Rosenblatt, a teoria de redes recorrentes de Hopfield e o algoritmo de retropropagação, que permitiu o treinamento eficiente de redes perceptron multicamadas, na década de 80.

Conforme Qiu et al. (2020), com o avanço do *deep learning*, várias redes neurais têm sido amplamente utilizadas para resolver tarefas de PLN, como redes neurais convolucionais

[2] Disponível em: <https://www.dio.me/articles/redes-neurais-classficacao>. Acesso em 5 de maio de 2024.

(RNCs), redes neurais recorrentes (RNRs), redes neurais baseadas em gráficos (RNGs) e mecanismos de atenção. Essas redes neurais profundas aliviam o problema de engenharia de recursos (*feature engineering*), criando representações densas e de baixa dimensão que capturam as características sintáticas e semânticas do texto.

Segundo Raff (2022), as RNCs têm foco na localidade espacial, em que os itens próximos uns dos outros estão relacionados, mas os itens distantes não têm relacionamento, permitindo a construção de redes neurais que aprendem mais rápido e fornecem soluções mais precisas para classificação de imagens. Já as RNRs, de acordo com Kelleher (2019), são adaptadas para o processamento de dados sequenciais, processando cada elemento da sequência, um por vez. Isso se dá por meio de um fluxo recorrente em que a rede processa cada entrada dentro do contexto gerado pelo processamento da entrada anterior, cuja saída gerada realimenta a camada oculta que está realizando esse processamento. Isso, seguindo o raciocínio de Kelleher (2019), permite que a rede mantenha uma memória acerca das informações da sequência que permitirão ao modelo tomar decisões sobre a próxima entrada.

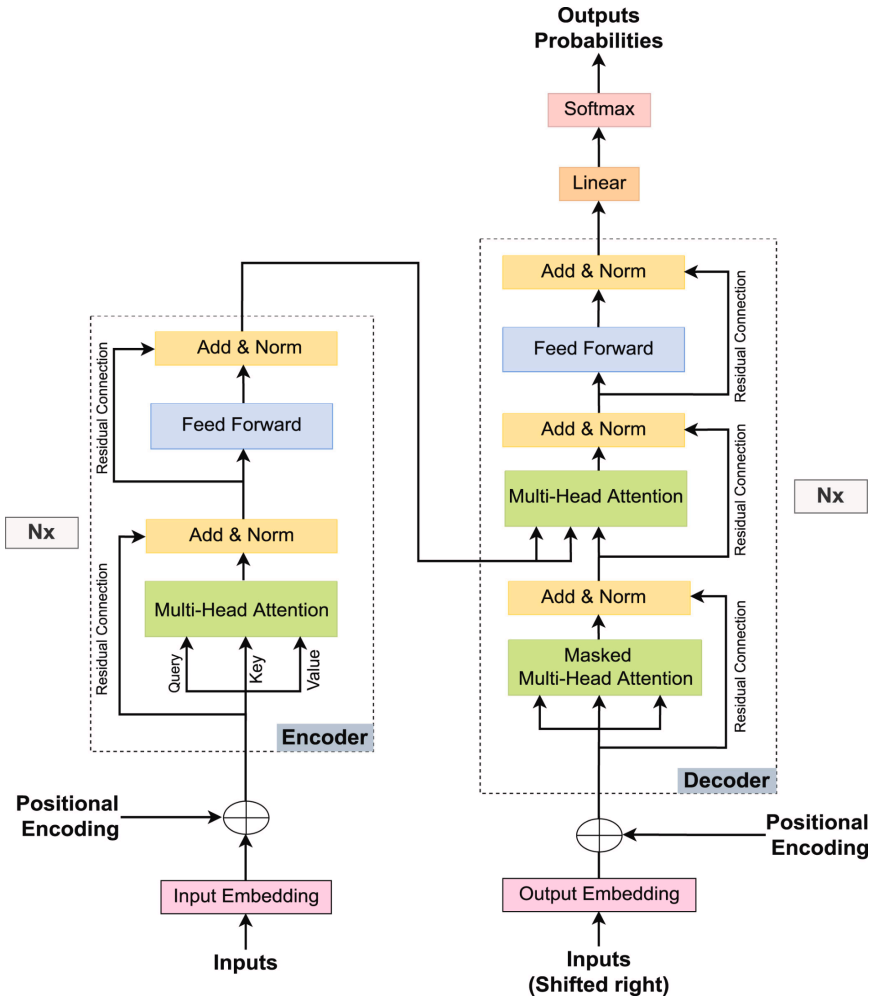
Segundo Wang, Pan e Lin (2023), nas tarefas de classificação, os métodos de DL baseados em RNRs possuem limitações por precisarem de muitos dados de treinamento, além do fato de que um conjunto de dados rotulados com um conjunto específico de classes não pode ser reaproveitado para treinar um método que foi projetado para prever um conjunto totalmente diferente de classes. Mas outro modelo despontou e tem sido motivo de grande interesse no contexto de DL (Kelleher, 2019), especialmente em PLN: O *Transformer* (Vaswani et al. 2017).

4.4 TRANSFORMERS

Segundo Ghojogh e Ghodsi (2020), antes de 2017, as arquiteturas tradicionais de modelagem de dados sequenciais, como RNRs, em especial LSTMs (Memória de longo e curto prazo), eram as mais usadas para aplicações de PLN. Mas, no ano de 2017, pesquisadores do Google, publicaram um artigo propondo uma nova arquitetura de rede neural, apelidada de *Transformer* ou Transformador (Figura 5), para modelar dados processados sequenciais (Vaswani et. al, 2017). Conforme Bishop e Bishop (2024), esses modelos são conhecidos como transformadores porque transformam um conjunto de vetores de entrada (embeddings obtidos do texto, no caso de tarefas de PLN), que estão em algum espaço de representação, em um conjunto correspondente de vetores, tendo a mesma dimensionalidade, em algum novo espaço. Conforme Ghojogh e Ghodsi (2020), essa

transformação passa por uma etapa intermediária de criação de um vetor de contexto, que simula um conceito que a mente humana guarda sobre alguma coisa (Perlovsky, 2006). Desse modo, a saída é gerada a partir do vetor de contexto que o modelo obtém processando a entrada. Logo, objetiva-se que a entrada e a saída permaneçam relacionadas entre si, mas com o novo espaço gerado passando a ter uma representação interna mais rica em informação e mais adequada para resolver tarefas posteriores (Bishop e Bishop, 2024).

Figura 5 - Arquitetura original *Transformer* (Vaswani et al, 2017)



Fonte: Islam et al, 2023

Segundo os criadores do Transformer, Vaswani et. al (2017), essa arquitetura foi construída com o objetivo de lidar com a limitação de modelos baseados em CNN, nos quais há dificuldades para aprender relações entre elementos distantes na sequência, como, por exemplo, palavras longínquas em um texto, devido ao aumento do custo computacional com a distância. O *Transformer* resolve isso usando um mecanismo de atenção que permite a análise

de todas as posições simultaneamente, atribuindo pesos ou coeficientes de ponderação diferentes a diferentes entradas (Bishop e Bishop, 2024). Isso faz com que algumas palavras sejam consideradas mais importantes para entender o significado do texto do que outras, porém introduz uma pequena perda de precisão. Essa perda é mitigada pela técnica de *Multi-Head Attention*.

Além disso, conforme Singh (2023), a nova arquitetura também minimiza as limitações de modelos baseados em RNR, nos quais o processamento serial pode ser lento e ineficiente, especialmente para sequências longas, habilitando o uso do paralelismo nos treinamentos com o processamento em GPUs.

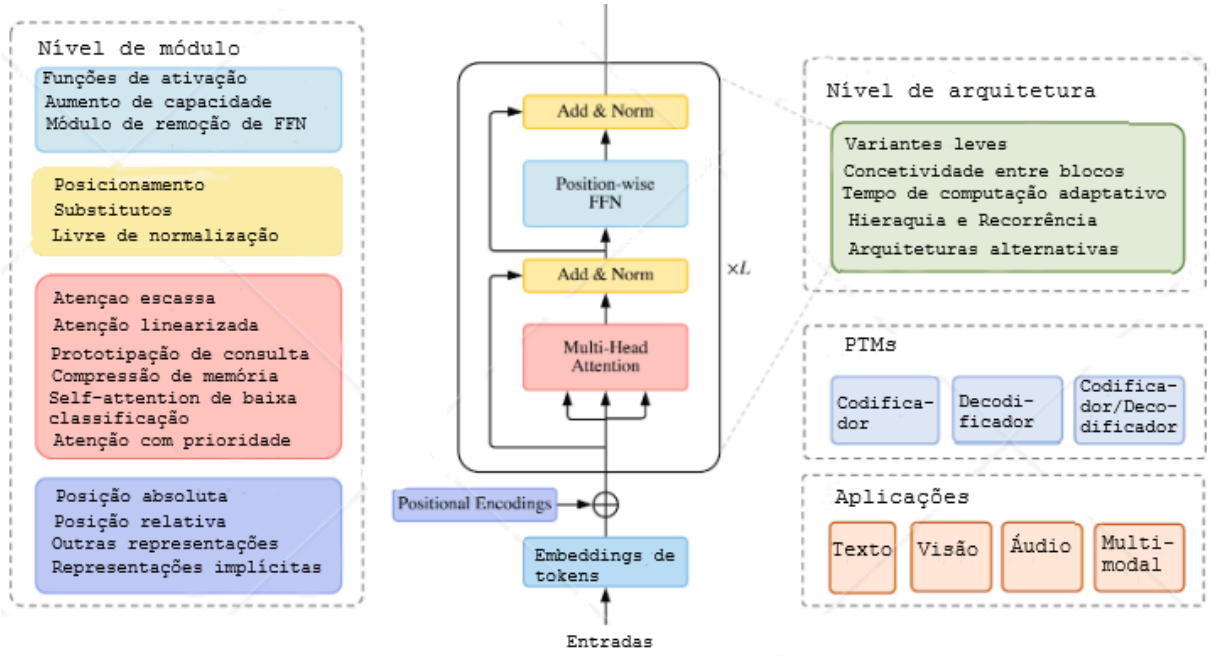
Paralelamente ao desenvolvimento do *Transformer*, um método de aprendizagem por transferência chamado ULMFiT comprovou ser possível treinar modelos de linguagem em um *corpus* extenso e diversificado e gerar classificadores de texto de alto desempenho, mesmo com poucos dados rotulados (Tunstall; Von Werra; Wolf, 2022).

Implementando essa abordagem de aprendizagem por transferência de forma eficaz, os transformers podem ser treinados em um grande corpo de dados para, em seguida, serem aplicados a muitas tarefas posteriores usando alguma forma de ajuste fino (Bishop e Bishop, 2024). Isso impulsionou ainda mais o desenvolvimento de *transformers*, pois eliminou a necessidade de treinar arquiteturas para tarefas específicas do zero, quebrando quase todos os *benchmarks* em PLN por uma margem significativa (Tunstall; Von Werra; Wolf, 2022).

Nesse contexto, variações do modelo *Transformer* original foram propostas ao longo do tempo a partir de três perspectivas: Tipos de modificação de arquitetura, métodos de pré-treinamento e aplicações (Lin et al., 2022). A Figura 6 ilustra essa categorização de variantes do *Transformer*.

Por tudo isso, os modelos baseados em *Transformers* alcançaram status de estado da arte para muitas tarefas importantes de PNL nos últimos anos (Singh, 2023), facilitando o desenvolvimento de vários modelos de linguagem grande (*Large Language Models* - LLMs) bem conhecidos, como BERT, T5, e GPT-3 (Campesato, 2024).

Figura 6 - Categorização de variantes do *Transformer* proposta por Lin et al. (2022)



Fonte: Adaptado de Lin et al., 2022

4.4.1 Arquitetura original do Transformer

Singh (2023) explica que os *Transformers* são uma estrutura de aprendizagem semi supervisionada. Em uma etapa inicial, eles são treinados em grandes volumes de dados não rotulados. Isso permite que o modelo aprenda representações de linguagem valiosas, como a relação entre palavras e conceitos. Posteriormente, esses modelos pré-treinados (com conhecimento adquirido no aprendizado não supervisionado) são finamente ajustados (*fine-tuned*) em tarefas específicas usando técnicas de aprendizado supervisionado com dados rotulados.

Os *Transformers* foram inspirados na arquitetura *encoder-decoder* encontrada em RNRs (Campesato, 2024). Arquiteturas de RNR eram uma das abordagens preferidas para lidar com dados sequenciais antes do advento dos *Transformers* (Ghojogh e Ghodsi, 2020). Porém, em vez de usar recorrência, como essas redes neurais, o modelo *Transformer* é totalmente baseado no mecanismo de atenção (Lee, 2024), que permite que ele dê mais importância a certas palavras ou *tokens* em relação a outros durante o processamento (Ghojogh e Ghodsi, 2020), por mais distantes que estejam. Isso ocorre, porque o *Transformer* calcula uma pontuação quanto à atenção entre cada *embedding* e todos os outros *embeddings* de palavras do *corpus* (Patwardhan, Marrone e Sansone, 2023), para que, com isso, consiga

atribuir às palavras uma relevância relativa e, então, concentre-se nas partes mais importantes do texto para determinar sua classificação. Para Galassi, Lippi e Torroni (2020), soluções eficazes para diversos problemas de PLN devem levar em consideração uma noção de relevância, de modo a concentrar os recursos computacionais num conjunto restrito de elementos importantes.

A arquitetura original do *Transformer* deu origem a outras arquiteturas, que trazem algumas melhorias, como por exemplo:

• *Reformer*: Tem desempenho equivalente aos modelos Transformer, sendo muito mais eficiente em termos de memória e muito mais rápido em sequências longas (Kitaev; Kaiser; Levskaya, 2020).

• *Longformer*: Projetado para processar documentos longos (Beltagy; Peters; Cohan, 2020).

• *Switch*: É uma técnica que usa apenas um subconjunto dos parâmetros de um modelo, mas aproveita muito bem as vantagens de GPUs e TPUs para operações intensas de multiplicação de matrizes (Campesato, 2024). Ela utiliza vários "especialistas" dentro do modelo, cada um treinado para lidar com diferentes tipos de dados ou tarefas, ativa apenas alguns desses especialistas para cada exemplo de entrada, tornando o treinamento mais eficiente (Fedus; Zoph; Shazeer, 2022).

• ELECTRA (*Efficiently Learning an Encoder that Classifies Token Replacements Accurately*): Diferentemente do BERT, que mascara alguns *tokens* e treina o modelo para prever esses *tokens* mascarados, o ELECTRA corrompe a entrada substituindo alguns *tokens* por alternativas plausíveis geradas por uma pequena rede geradora. Em seguida, um modelo discriminativo é treinado para prever se cada *token* foi substituído por um gerador ou se permaneceu inalterado. A eficiência do ELECTRA vem do fato de que ele considera todos os *tokens* na entrada, não apenas um subconjunto mascarado. Isso permite que o modelo aprenda representações contextuais mais ricas e seja treinado de forma mais eficiente (Clark; Loung; Manning, 2020).

a) Encoder-Decoder

Segundo Islam et al. (2023), a arquitetura *Transformer* (Figura 5) proposta por Vaswani et. al (2017) o bloco codificador possui seis camadas codificadores concatenados, mas é possível especificar um número maior de blocos. Cada elemento do codificador possui duas subcamadas. Lee (2024) afirma que a primeira subcamada é a *Self-Attention* ou

Autoatenção, e a segunda, é uma *Feedforward* simples e totalmente conectada. A saída da primeira subcamada torna-se a entrada da segunda subcamada, e a saída final da sexta (na arquitetura original) camada codificadora é então passada para cada camada decodificadora no bloco decodificador.

Semelhantemente, de acordo com Campesato (2024), o bloco decodificador também possui seis (às vezes mais) camadas decodificadoras concatenadas, em que a saída de uma camada se torna a entrada para a próxima camada decodificadora. Lee (2024) acrescenta que, além de duas subcamadas idênticas a cada camada de codificação, o bloco decodificador possui uma terceira subcamada para realizar a *Multi-head Attention* ou Atenção de múltiplas cabeças. Conexões residuais e normalização de camada são usadas sequencialmente para todas as subcamadas, tanto no bloco codificador como no decodificador, ajudando a evitar erros com distorções ou perda de informações.

O componente codificador pega uma sequência de comprimento variável e a converte em um estado de saída de comprimento fixo. O componente decodificador assume um estado de comprimento fixo e o converte novamente em uma saída de comprimento variável (Kamath; Graham; Emara, 2022). O decodificador extrai vetores de palavras de entrada, transforma palavras de saída em vetores de palavras e gera a probabilidade de cada palavra de saída ser a próxima palavra na sequência de saída (Lee, 2024).

b) Mecanismo de Atenção

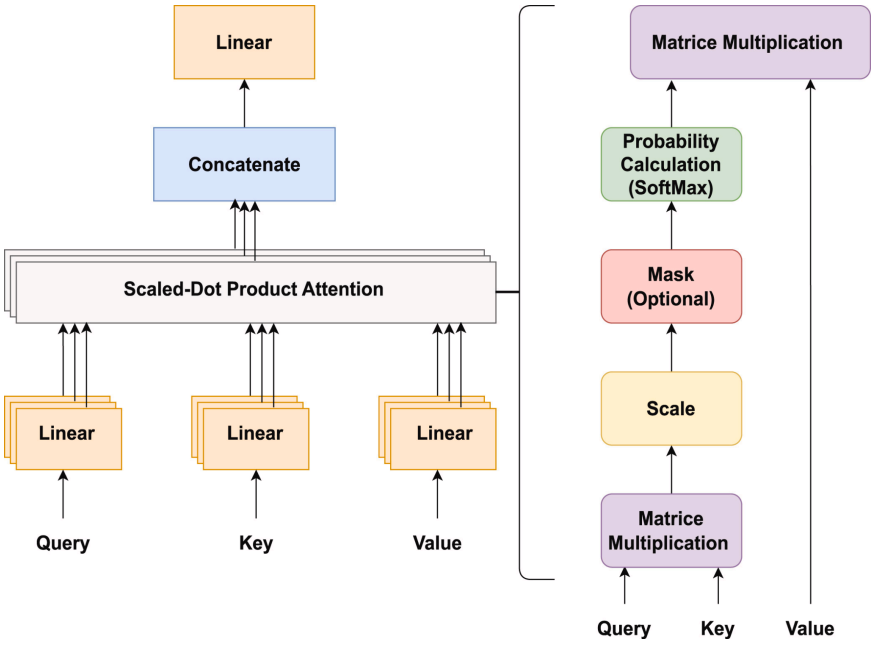
Galassi, Lippi e Torroni (2020) afirmam que a atenção foi introduzida em PLN por Bahdanau, Cho e Bengio (2014) a partir de tarefas de tradução automática. Aqueles pesquisadores explicam que o mecanismo de atenção faz parte de uma arquitetura neural que permite destacar dinamicamente características relevantes dos dados de entrada, que, em tarefas de PLN, são tipicamente uma sequência de elementos textuais.

Ghojogh e Ghodsi (2020) dizem que o conceito de atenção é uma simples ponderação de dados modelada por aprendizado de máquina, de modo que, as partes mais informativas ou mais importantes dos dados recebem pesos maiores para maior atenção. Segundo Islam et al. (2023), na arquitetura original do *Transformer*, o mecanismo de atenção é implementado usando a técnica "*Scaled Dot Product Attention*" (Figura 7), que se baseia em três matrizes de parâmetros fundamentais (Consulta, Chave e Valor) que encapsulam, cada uma, uma representação codificada de cada elemento de entrada da sequência. O modelo calcula a similaridade entre a Consulta e cada uma das Chaves, usando uma operação de produto

escalar (*dot product*). Essas pontuações de similaridade são então passadas por uma função SoftMax, que as transformam em probabilidades, as quais representam a importância relativa de cada parte dos dados de entrada para a tarefa atual.

Esse mecanismo é parte central na arquitetura *Transformer* e é por meio dele que os embeddings contextuais são determinados para as palavras em um *corpus*. Ao contrário do word2vec ou gloVe, o mecanismo de atenção leva em consideração todas as palavras em uma frase durante o processo de criação de *embeddings* de palavras para uma determinada palavra em uma determinada frase (Campesato, 2024).

Figura 7 - Mecanismo de atenção



Fonte: Islam et al. (2023)

c) Feed Forward Networks

De acordo com Campesato (2024), a rede neural *feedforward* (FFN) no codificador (*encoder*) de um modelo *Transformer* desempenha um papel fundamental no processamento das informações obtidas nas camadas anteriores e no mapeamento delas para um espaço potencialmente mais complexo para capturar padrões intrincados nos dados.

O autor detalha que a FFN envolve as funções de ativação não linear ReLU ou GELU, normalmente com uma ou mais camadas lineares antecedendo essas funções, de modo a processar cada *token* independentemente. Isso permite que a FFN se concentre na extração de

recursos de *tokens* individuais sem ser influenciado por outros *tokens* na sequência, adicionando profundidade ao modelo e aprimorando o aprendizado de representações hierárquicas e complexas dos dados. Dessa forma, seu funcionamento complementa a camada de *Multi-head Attention* para processar e analisar sequências eficazmente em uma ampla gama de tarefas de PLN.

d) Positional Encoding

Ghojogh e Ghodsi (2020) explicam que, como não há recorrência nem convolução, o modelo *transformer* não tem senso de ordem na sequência, processando todas as palavras de uma vez. Como a ordem das palavras é importante para o significado das frases, é necessário uma forma de explicar a ordem dos *tokens* ou palavras na sequência. Para isso, adiciona-se um vetor que contabiliza a posição de cada incorporação de palavra de entrada. Ou seja, cada posição na sequência tem um vetor *positional encoding* associado a ela, e esse vetor é somado à representação da palavra (*embedding*). Assim, mesmo palavras iguais terão representações diferentes dependendo de onde aparecem na frase.

Além disso, as codificações posicionais permitem que um modelo trabalhe com sequências que consistem em comprimentos variáveis, e também permite o processamento paralelo de sequências, o que é uma vantagem significativa dos modelos de transformadores sobre as redes neurais recorrentes (Campesato, 2024).

No entanto, conforme Campesato (2024), há pelo menos três técnicas diferentes que produzir informações relacionadas à posição, conforme mostrado aqui:

• Incorporação de posição (*position embedding*);

• Codificação de posição (*position encoding*);

• Posições relativas (*relative positions*).

Apesar do *Transformer* original usar *positional encoding*, outros modelos de linguagem baseados em *Transformer* podem usar diferentes técnicas de codificação. Campesato (2024) cita como exemplo o BERT, que usa incorporações de posição em vez de codificação de posição.

e) *Multi-Head Attention*

Islam et al. (2023) explicam que os módulos *encoder* e *decoder* da arquitetura *Transformer*, e suas respectivas camadas, são executados diversas vezes, conforme exigido

pela tarefa em questão, de modo que o mecanismo de atenção é executado em paralelo múltiplas vezes, denotando a presença de múltiplas "Cabeças de Atenção" ou *Multi-Head Attention*.

Islam et al. (2023) complementam que a utilização do *Multi-Head Attention* facilita a rede neural no aprendizado e captura diversas características dos dados sequenciais de entrada, melhorando a representação dos contextos de entrada, uma vez que cada "cabeça" de atenção tem seus próprios pesos e pode se concentrar em diferentes partes dos dados de entrada.

Campesato (2024) explica que, no *Transformer* original, o decodificador contém uma camada adicional chamada *Masked Multi-Head Attentio*n (Atenção multicabeça mascarada), que é projetada para prevenir que o decodificador "olhe para o futuro", ou seja, que veja *tokens* que estão à frente na sequência de entrada. Isso é importante porque, durante o treinamento, o decodificador deve prever o próximo *token* com base apenas nos *tokens* anteriores.

No entanto, o BERT, que é uma variante do *Transformer*, é treinado de maneira diferente. O BERT é um modelo bidirecional, o que significa que ele leva em consideração tanto os *tokens* anteriores quanto os próximos ao fazer previsões (Campesato, 2024), logo o BERT não usa uma camada de atenção multicabeça mascarada da mesma maneira que o *Transformer* original.

f) Add & Norm

A etapa de Adição e Normalização *(Add & Norm*), segundo Campesato (2024), combina a entrada de um componente com a saída de outro componente, de forma que, após essa combinação, tem-se uma normalização. Esta etapa é executada na camada de *Multi-head attention*, bem como na camada de *Feed forward*.

A adição essencialmente "aumenta" um sinal enfraquecido, como se fosse um repetidor, e a etapa de normalização garante que os valores resultantes estejam no mesmo intervalo, sem valores discrepantes, ajudando a na obtenção de cálculos mais rápidos (Campesato, 2024).

4.4.2 Variantes da Arquitetura Transformer quanto ao método de Pré-treinamento

Conforme explica Campesato (2024), existem modelos que implementam diferentes componentes da arquitetura do *transformer*, embora sempre forneçam um mecanismo de autoatenção. Essas variações refletem no método de pré-treinamento, e ocorrem, segundo o pesquisador, das seguintes maneiras:

• Arquitetura com decodificador e sem codificador: São conhecidos como modelos *Transformers* Autorregressivos e são úteis em tarefas de geração de texto, já que utilizam previsões anteriores para gerar uma nova previsão, aprendendo as interdependências entre palavras e frases, incluindo a semântica, quando usado em PLN;

• Arquitetura com codificador e sem decodificador: São modelos *Transformers* de Autocodificação (*Autoencoding*), que, no contexto de PLN, são adequados para tarefas de compreensão da linguagem natural, pois, em termos simplificados, são pré-treinados corrompendo parte dos *tokens* de entrada de alguma forma. Em seguida, tentam reconstruir esses *tokens* corrompidos, seguindo o princípio dos *Autoencoders* tradicionais, embora não reconstruam toda a entrada de dados como esses. Ao fazer isso, esse tipo de *transformer* examina as dependências bilaterais (antes e depois) de *tokens* em uma frase para obter contexto bidirecional.

• Arquitetura com codificador e com decodificador (arquitetura original), em que o codificador processa a sequência de origem e o decodificador gera a sequência de destino, sendo um modelo adequado para tarefas de PLN como a tradução automática.

Campesato (2024) exemplifica afirmando que modelos GPT usam apenas o componente decodificador da arquitetura *transformer*, enquanto o BERT usa apenas o componente codificador. Já o BART e o T5 são dois modelos que utilizam tanto o codificador quanto o decodificador da arquitetura *transformer*.

4.4.3 Biblioteca Hugging Face

Embora existam vários modelos excelentes de *deep learning* produzidos por vários grupos de pesquisa, e neste momento eles estão fragmentados e não são compatíveis entre diferentes estruturas. O modelo BERT, por exemplo, foi desenvolvido pela equipe de pesquisa do Google especialmente no TensorFlow e não funciona automaticamente no PyTorch, para o qual seria necessário portar/reescrever todo esse código. A biblioteca *Transformers* da *Hugging Face* traz as vantagens de permitir escolher facilmente entre diferentes modelos pré-treinados apenas alterando o valor de um parâmetro; e de funcionar em PyTorch e TensorFlow. (Albrecht; Ramachandran; Winkler, 2021)

4.5 TRANSFER LEARNING

Weiss, Khoshgoftaar e Wang (2016) explicam que, em metodologias tradicionais de aprendizado de máquina, os dados de treinamento e de teste são extraídos do mesmo domínio, de modo que as características de distribuição desses dados sejam as mesmas. Mas há casos em que os dados de treinamento são caros ou difíceis de coletar. Com isso, surgiu uma necessidade de criar modelos de aprendizado de alto desempenho treinados com dados de diferentes domínios, obtidos mais facilmente, e que poderiam ser aplicados a um domínio alvo relacionado. Essa metodologia ficou conhecida como *Transfer Learning* ou aprendizagem por transferência.

Transfer Learning é inspirado na forma como os humanos aprendem, porque normalmente não aprendemos coisas do zero, ao contrário, construímos uma ideia ou solução com base em conhecimentos relacionados anteriores (Azunre, 2021). Azunre (2021) complementa que essa abordagem trouxe o benefício de permitir que um modelo treinado com recursos massivos, incluindo dados, poder de computação, tempo e custo, pudesse ser ajustado e reutilizado em novos ambientes pela comunidade por uma fração muito menor dos recursos originais.

Para implementar o *Transfer Learning*, de acordo com Albrecht, Ramachandran e Winkler (2021), primeiramente, define-se uma rede neural profunda para ser treinada em um grande corpus de texto, geralmente derivado de fontes de dados públicos, como a Wikipedia. Em geral, a arquitetura do modelo escolhida é uma variante do LSTM ou Transformer (Albrecht, Ramachandran e Winkler, 2021).

Essa estratégia de ajustar um grande modelo de linguagem pré-treinado foi amplamente adotada e alcançou desempenho de última geração em uma variedade de tarefas de PLN (Souza; Nogueira; Lotufo, 2023). Caso um modelo pré-treinado não funcione em determinada tarefa, ele ainda poderá servir como uma boa entrada para um modelo em desenvolvimento, exigindo menos dados novos de treinamento (Singh, 2023).

4.6 PRE-TRAINED LANGUAGE MODEL (PLM)

Palaniappan, Vedhamani e Sundharakumar (2023) afirmam que um *pre-trained language model* é um modelo de linguagem já treinado em um grande corpo de texto, podendo compreender padrões estatísticos e relações entre palavras e frases com base no processo de treinamento. Geralmente, PLMs usam métodos não supervisionados para minerar

conhecimento semântico de forma automática e, em seguida, constroem alvos de pré-treinamento, ou seja, são treinados em tarefas específicas, como prever a próxima palavra em um frase, para aprender a representação semântica da linguagem (Qian Li, 2022). Como resultado, as informações aprendidas podem ser usadas para realizar diversas tarefas, como categorização de texto, inferência em linguagem natural e resposta a perguntas.

Os PLMs oferecem vários benefícios, como permitir usar um conjunto de dados de treinamento significativamente maior para o classificador, levando a uma maior precisão e a uma melhor consciência dos padrões de linguagem, o que aumenta a capacidade de categorizar e compreender novas ocorrências de texto. Além disso, a utilização de um modelo de linguagem pré-treinado libera recursos para serem utilizados no treinamento do classificador, permitindo a concentração apenas nessa tarefa, otimizando tempo e recursos computacionais (Palaniappan; Vedhamani; Sundharakumar, 2023).

Esse tipo de modelo provê o *transfer learning* para o classificador em um tarefa de classificação de texto, o que, conforme a analogia sugerida por Albrecht, Ramachandran e Winkle (2021), seria como um ser humano aprender primeiro a tocar violão usando os muitos recursos disponíveis na internet para, então, aplicar esse conhecimento a uma tarefa diferente, como tocar harpa ou violoncelo, cujos recursos de aprendizado são mais difíceis de encontrar.

Conforme Zhao et al. (2023), uma das primeiras tentativas de PLM proposta foi o ELMo, que busca capturar representações de palavras sensíveis ao contexto, primeiro pré-treinando uma rede LSTM bidirecional (biLSTM), em vez de aprender representações de palavras fixas, e depois ajustando a rede biLSTM de acordo com tarefas posteriores específicas. Depois, com o advento dos *Transformers*, o BERT (Devlin et al, 2019) foi proposto como modelo pré-treinado de linguagem bidirecional com tarefas de pré-treinamento especialmente projetadas em corpus não rotulados de grande escala, elevando amplamente o nível de desempenho das tarefas de PLN e inspirando um grande número de trabalhos que estabelecem o paradigma de aprendizagem de pré-treinamento e *fine tuning*. Outras arquiteturas baseadas em *Transformers* foram introduzidas, e, conforme Fields, Chovanec e Madiraju (2024), esses modelos se tornaram a abordagem padrão para muitas tarefas gerais de classificação de texto, como análise de sentimentos, classificação de documentos e reconhecimento de entidades nomeadas

Ainda segundo Zhao et al. (2023), a comunidade de pesquisa cunhou o termo *Large Language Model* (LLM) para os PLMs de grande porte que foram obtidos a partir da escala dos tamanhos dos modelos e dos conjuntos de dados de treinamento. Minaee et al. (2024) acrescentam que, comparados aos PLMs tradicionais, os LLMs não são apenas muito maiores

em tamanho, mas também exibem habilidades mais fortes de compreensão e geração de linguagem, além de habilidades emergentes como aprendizagem em contexto (*In-context Learning*), onde os LLMs aprendem uma nova tarefa a partir de um pequeno conjunto de exemplos apresentados no *prompt* no momento da inferência.

Fields, Chovanec e Madiraju (2024) explicam que o PLN passou por uma verdadeira revolução por conta dos LLMs, levando várias tarefas clássicas de PLN ao estado da arte. Eles acrescentam que uma dessas tarefas, de grande interesse na academia e na indústria, é a classificação de texto, relacionada à categorização e organização de texto.

4.6.1 Bidirectional Encoder Representation from Transformers (BERT)

O BERT é um modelo de linguagem baseado em *Transformer* que foi projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente o contexto à esquerda e à direita em todas as camadas. Ele foi treinado em duas etapas principais: Primeiro é pré-treinado em um grande *corpus* (BooksCorpus, com 800 milhões de palavras e Wikipedia em inglês, com 2,5 bilhões de palavras) de texto usando as tarefas de Modelagem de Linguagem Mascarada (MLM), em que algumas palavras do texto são mascaradas (ocultadas) para então ser executada a previsão das palavras faltantes, e de Previsão de Próxima Sentença (NSP), em que ele tenta prever essas palavras com base no contexto fornecido pelas palavras ao redor (Devlin et al., 2018).

As duas versões mais conhecidas do BERT são chamadas BERT-*Base*, com 12 camadas, 12 núcleos de atenção e 110 milhões de parâmetros, e BERT-*Large*, que consiste em 24 camadas, 16 cabeças de atenção e 340 milhões de parâmetros (Campesato, 2024). No entanto, existem também vários modelos menores baseados em BERT, tais como BERT-*medium*, BERT-*small*, BERT-*mini* e BERT-*tiny*.

Segundo Campesato (2024), existem muitas variantes de modelos BERT pré-treinados em diferentes tipos de conjuntos de dados (como jurídicos, biomédicos, científicos e assim por diante), e muitas dessas melhoraram características de desempenho, como é o caso do RoBERTa, do ALBERT e do DistilBERT.

Palaniappan, Vedhamani e Sundharakumar (2023) afirmam que o *tokenizer* integrado ao BERT desempenha um papel crucial no modelo, convertendo texto bruto em *tokens* e codificando-os adequadamente. Ele executa tokenização, truncamento e preenchimento para garantir que todas as sequências de entrada tenham comprimentos uniformes, permitindo que o modelo lide com palavras fora do vocabulário de maneira eficiente. Palaniappan,

Vedhamani e Sundharakumar (2023) concluem que a arquitetura do modelo BERT permite a captura eficaz de dependências de palavras e a codificação adequada dos dados de entrada, permitindo que o modelo execute tarefas ZSL para prever probabilidades de categoria.

Além disso, Martins e Silva (2022) alude ao estudo de Sun et al. (2019), que faz um estudo comparativo de como usar o BERT para a tarefa de classificação de texto, para citar que o modelo BERT pode ser utilizado em tarefas com pequenas quantidades de dados disponíveis para treinamento.

4.6.2 A Lite BERT (ALBERT)

A complexidade e o custo computacional do BERT limitaram sua aplicação em cenários com recursos computacionais restritos. Para superar essas limitações, pesquisadores propuseram o ALBERT (A Lite BERT) na busca de otimizar o BERT, tornando-o mais eficiente e eficaz. A principal motivação por trás do ALBERT é a redução do número de parâmetros, sem comprometer o desempenho.

Conforme Lan et al. (2020), as principais características do ALBERT são a parametrização fatorizada de *embeddings*, que permite reduzir significativamente o número de parâmetros, especialmente para vocabulários grandes, e o compartilhamento de parâmetros entre camadas, que reduz ainda mais o número de parâmetros a serem treinados e promove uma regularização implícita.

Ademais, ao invés de usar a previsão da próxima sentença (NSP) utilizada no BERT, o ALBERT utiliza a predição de ordem de sentença (SOP) no seu treinamento, concentrando-se em modelar a coerência entre as sentenças, em vez de simplesmente prever a ordem delas.

4.6.3 Robustly Optimized BERT Pretraining Approach (RoBERTa)

O modelo RoBERTa representa um refinamento do BERT. Diferente do BERT, que usa um padrão de mascaramento fixo durante o pré-treinamento, o RoBERTa aplica o mascaramento dinâmico, o que significa que as palavras a serem mascaradas são alteradas a cada vez que os dados são passados pelo modelo, ajudando o modelo a obter um entendimento mais robusto do contexto da linguagem. Além disso, o RoBERTa não usa a tarefa de Previsão de Próxima Frase (NSP) que está presente no BERT. Em vez disso, ele é treinado apenas com sentenças completas, o que os autores concluíram como mais eficaz (Liu et al., 2019).

O RoBERTa é treinado com tamanhos de lote maiores em comparação com o BERT, o que pode contribuir para um aprendizado mais estável e eficiente, e usa um vocabulário de *Byte-Pair Encoding* (técnica usada para construir um vocabulário de *tokens* a partir de um grande *corpus* de texto, combinando iterativamente os pares de *bytes* ou caracteres mais frequentes) maior, permitindo que o modelo lide com um conjunto mais amplo de *tokens*, podendo ajudar o modelo a entender melhor o texto e a ser mais eficaz em tarefas de PLN (Liu et al., 2019).

4.6.4 BERTimbau

O modelo BERTimbau (Souza; Nogueira; Lotufo, 2020) foi desenvolvido na esteira do crescente interesse em modelos monolíngues, devido ao seu desempenho superior e à eficiência de recursos em comparação com modelos multilíngues, como o mBERT (BERT multilíngue). O BERTimbau foi avaliado em três tarefas de PLN: “Similaridade textual de sentenças”; “reconhecimento de implicação textual"; e “reconhecimento de entidade nomeada”. Além disso, o estudo de Souza, Nogueira e Lotufo (2020) comparou os tokenizadores do BERTimbau e do mBERT, analisando o impacto dessa tokenização no desempenho das tarefas de PLN para ajudar a entender as vantagens de um modelo monolíngue em relação a um multilíngue em termos de compreensão de linguagem.

O BERTimbau usou os dados brWaC (Brazilian Web as a Corpus), um *corpus* grande e diversificado (2,68 bilhões de *tokens* de 3,53 milhões de documentos) de páginas web em português brasileiro, para treinamento, permitindo que o modelo aprenda a estrutura e os padrões do português brasileiro de forma mais eficaz. Os resultados do trabalho, conforme Souza, Nogueira e Lotufo (2020), mostraram que o BERTimbau melhorou o desempenho nas tarefas alvos do estudo em relação aos modelos multilíngues e abordagens monolíngues anteriores.

O modelo foi disponibilizado para a comunidade acadêmica em bibliotecas de código aberto[3].

[3] https://github.com/neuralmind-ai/portuguese-bert

4.7 PROMPT ENGINEERING

Segundo Giray (2023), a engenharia de *prompt* ou *prompt engineering* se refere à prática de desenvolver e otimizar *prompts* para utilizar efetivamente grandes modelos de linguagem, especialmente em tarefas de PLN. Campesato (2024) afirma que um *prompt*, no contexto de LLMs, é uma sequência de texto de tamanho variável que os usuários fornecem aos modelos, auxiliando esses a concluir uma solicitação ou tarefa. Ainda de acordo com o pesquisador, embora os *prompts* possam ser qualquer sequência de texto, incluindo uma sequência aleatória, a qualidade e a estrutura dos *prompts* afeta a qualidade das conclusões do modelo.

Conforme Zhao et al. (2023), prompts cuidadosamente projetados, com uma descrição da tarefa expressa de forma mais clara ou em um formato amigável ao modelo, podem aumentar o desempenho de *zero-shot* ou *few-shot* de LLMs. Para Luo (2023), esse método se adequa bem aos cenários de FSL e ZSL e é uma alternativa à abordagem de ajuste fino, de forma que, com prompts apropriados, o aprendizado de máquina por meio de *prompt* (*prompt learning*) pode orientar o modelo para gerar saídas desejadas, transformando-o em um classificador sem a necessidade de parâmetros adicionais ou de um ajuste fino extensivo, o que representa uma possibilidade de adaptação mais rápida a novas tarefas e maior flexibilidade no tratamento de diversos problemas de classificação.

Minaee et al. (2024) afirmam que a *prompt engineering* transcende a mera construção de *prompts*, requerendo uma combinação de conhecimento de domínio, compreensão do modelo de IA e uma abordagem metódica para personalizar *prompts* para diferentes contextos. Isso pode envolver a criação de modelos que podem ser modificados programaticamente com base em um determinado conjunto de dados ou contexto. Além disso, a *prompt engineering* é um processo iterativo e exploratório, semelhante às práticas tradicionais de aprendizado de máquina.

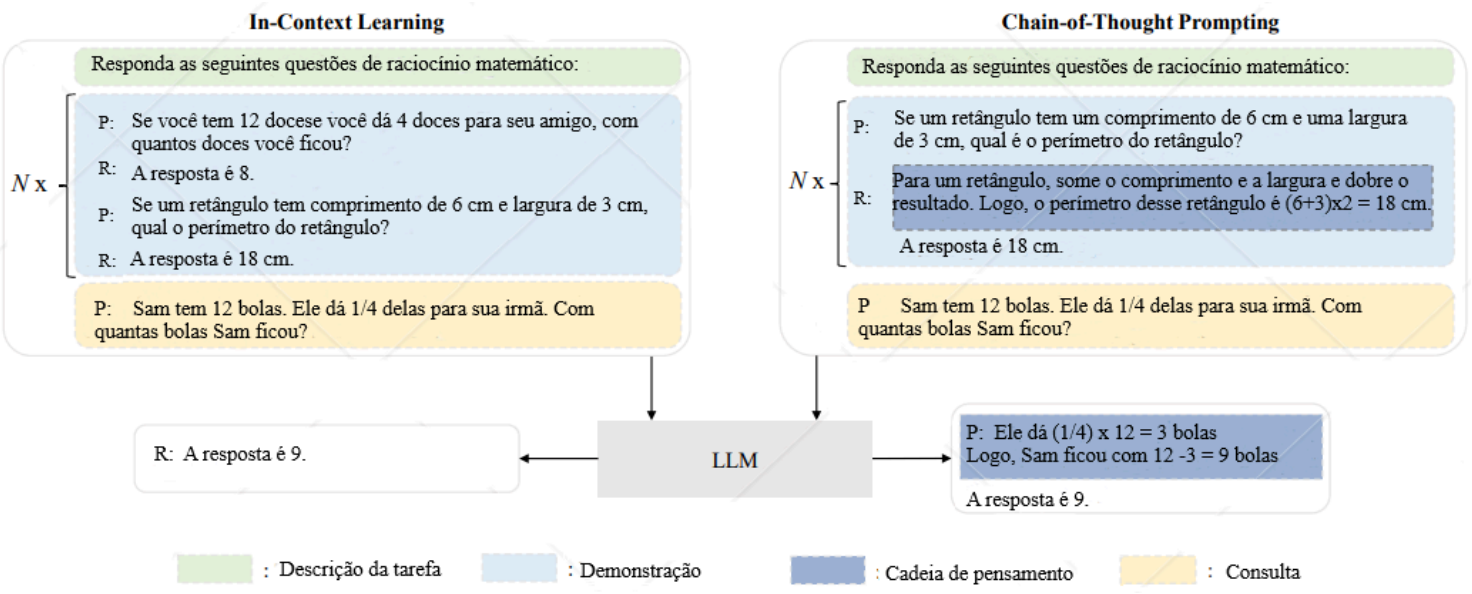
Zhao et al. (2023) apresentam duas abordagens para *prompt learning* (Figura 8):

- *Chain-of-Thought Prompting* (CoT): Proposto por Wei et al (2022), o método “Prompting por cadeia de pensamento” guia o modelo através de uma sequência de passos lógicos para chegar a uma resposta. Por exemplo, Kojima et al. (2022) mostram que, anexando a frase 'Vamos pensar passo a passo', LLMs poderiam realizar solicitações de CoT em uma configuração de *zero-shot*. Contudo, conforme Kim et al. (2023), embora o prompt CoT funcione de forma eficaz para LLMs, com mais de 100 bilhões de parâmetros,

ele não confere necessariamente os mesmos benefícios para modelos de linguagem menores;

- *In-Context Learning*: Nessa abordagem, o modelo aprende a realizar uma tarefa específica a partir dos exemplos fornecidos no contexto do *prompt*. Segundo Zhao et al. (2023), foi primeiramente proposto com o GPT-3.

Figura 8 - Ilustração comparativa entre ICL e CoT prompting



Fonte: Adaptado de Zhao et al., (2023)

4.7.1 Modelos LLaMA

A coleção de modelos LLaMA foi lançada pela Meta AI em 2023, composta por quatro tamanhos em parâmetros (7 bilhões, 13 bilhões, 30 bilhões e 65 bilhões), alcançando um desempenho excelente em vários *benchmarks* abertos (Zhao et al, 2023).

Ao contrário dos modelos GPT, os modelos LLaMA são de código aberto, ou seja, os pesos dos modelos são liberados para a comunidade de pesquisa sob uma licença não comercial. Assim, a família LLaMA cresce rapidamente à medida que esses modelos são amplamente utilizados por muitos grupos de pesquisa para desenvolver melhores LLMs de código aberto para competir com os de código fechado ou para desenvolver LLMs de tarefas específicas para aplicações de missão crítica (Minaee et al, 2024).

Segundo Zhao et al. (2023), um grande número de pesquisadores estenderam os modelos LLaMA por meio do ajuste de instruções ou do pré-treinamento contínuo, criando

modelos customizados. Para adaptar efetivamente os modelos LLaMA em idiomas diferentes do inglês, muitas vezes é necessário estender o vocabulário original, treinado principalmente no *corpus* inglês, ou ajustá-lo com instruções ou dados no idioma alvo.

4.7.2 Modelos GPT (Generative pre-trained transformer)

Os modelos GPT (Radford et al., 2018) empregam dois estágios para aprender representações gerais, que são transferidas com adaptação limitada para várias tarefas de PLN: O pré-treinamento não supervisionado em um conjunto massivo de dados de texto e o posterior ajuste fino em um tarefa supervisionada específica. Esses modelos usam uma arquitetura *Transformer* generativa somente decodificado, cujo princípio básico é comprimir o conhecimento mundial no modelo de modo que ele possa recuperar a semântica desse conhecimento e servir como um solucionador de tarefas de uso geral. Dois pontos-chave para o seu sucesso são: Treinar modelos de linguagem *Transformer* somente decodificadores que podem prever com precisão a próxima palavra e aumentar o tamanho dos modelos de linguagem (Zhao et al., 2023).

A utilização de modelos GPT para classificação de texto emprega um método de etapa única baseado em *prompt*. Esta abordagem simplificada aproveita os recursos geradores da GPT para produzir diretamente rótulos de classificação específicos, orientando o modelo GPT a gerar um rótulo de classificação em um formato predefinido (Wang; Pan; Lin, 2023).

4.7.2.2 GPT 3

O GPT 3 foi lançado em 2020, dimensionando os parâmetros do modelo para um tamanho de 175 Bilhões. No artigo do GPT-3, "*Language models are few-shot learners*" (Brown et al., 2020), ele introduziu formalmente o conceito de aprendizagem *in-context* (ICL), que utiliza LLMs em tarefas de *few-shot* ou *zero-shot*. A ICL pode instruir LLMs a compreenderem as tarefas na forma de texto em linguagem natural, e o desenvolvimento dessa habilidade teve um grande salto, devido ao dimensionamento significativo da arquitetura de rede neural do GPT 3 (Zhao et al, 2023).

4.7.2.3 GPT-4

Como outro progresso notável, o GPT-4 foi lançado em março de 2023, trazendo a novidade da extensão da entrada de texto para sinais multimodais (ver, ouvir e falar), além de

trazer capacidades mais fortes que o GPT 3.5 na resolução de tarefas complexas, mostrando uma grande melhoria de desempenho em muitas tarefas de avaliação (Zhao et al, 2023).

Com base no trabalho realizado para o GPT-4, a OpenAI lançou ainda o GPT-4V, que se concentrou na implantação segura das capacidades de visão do GPT-4, que exibiu fortes habilidades nesse aspecto em vários cenários de aplicação, mostrando o grande potencial como um poderoso sistema de aprendizagem multimodal (Zhao et al, 2023). Mais recentemente, a OpenAI lançou uma geração atualizada do modelo GPT-4, denominada GPT-4 Turbo, com uma série de melhorias técnicas.

4.8 META LEARNING

Conforme Işık e Paçal (2024), *Meta learning* ou Meta aprendizado é um subcampo de aprendizado de máquina que dota os modelos com a capacidade de se adaptarem rapidamente a novas tarefas com base na experiência anterior, ou seja, utilizando o conhecimento que adquiriu em tarefas anteriores, mesmo com dados limitados.

Işık e Paçal (2024) explicam que, geralmente, existem 3 abordagens diferentes em meta-aprendizagem: “Baseada em métricas”; “baseada em modelo”; e “baseada em otimização”. Na abordagem baseada em métricas, o foco está em aprender uma métrica de similaridade ou distância entre exemplos de dados, que facilita a aprendizagem de novas tarefas. A abordagem baseada em modelo envolve modificar a estrutura do modelo para que ele possa ser treinado rapidamente em novas tarefas. Já a abordagem baseada em otimização está interessada em aprender como otimizar melhor os parâmetros do modelo para novas tarefas, geralmente através de meta-gradientes.

ProtoNets e MAML estão entre os algoritmos de *meta learning* com maior eficácia em cenários de aprendizagem *few shot learning* (Işık e Paçal, 2024)

4.8.1 MAML (Model-Agnostic Meta-Learning)

O *Model-Agnostic Meta-Learning* foi proposto por Chelsea Finn et al. (2017) como um algoritmo de meta aprendizado agnóstico de modelo, porque foi projetado para funcionar com qualquer modelo treinado com descida de gradiente, além de ser aplicável a vários problemas de aprendizagem.

Conforme Griva et al. (2023), a ideia central é treinar os parâmetros iniciais do modelo para maximizar seu desempenho em uma nova tarefa após atualizar esses parâmetros,

mesmo com uma pequena quantidade de dados disponíveis. O algoritmo proposto não aumenta o número de parâmetros aprendidos nem impõe restrições à arquitetura do modelo, maximizando a sensibilidade das funções de perda de novas tarefas com relação aos parâmetros. Finn et al. (2023) avaliou o algoritmo em problemas de classificação FSL, regressão supervisionada e aprendizagem por reforço (RL). Como desafio, o MAML pode ser computacionalmente caro durante o treinamento e a inferência e requer que o modelo passe por um ajuste de hiperparâmetros caro.

4.8.2 Prototypical Networks

Prototypical Networks são um algoritmo de *meta learning* baseado em métricas e que um espaço de incorporação (vetores de características) é aprendido, de forma que as instâncias pertencentes à mesma classe estão próximas umas das outras de acordo com certas métricas (Işık e Paçal, 2024). Usando essas métricas, calcula-se a semelhança ou distância das instâncias no conjunto de consultas com as instâncias no conjunto de suporte. Consequentemente, a média dos vetores de características de todas as instâncias no conjunto de suporte é calculada. Dada uma nova instância de consulta é classificada de acordo com sua distância ou semelhança com esses meios

4.9 ZERO-SHOT LEARNING

O *Zero-Shot Learning* é uma estratégia de aprendizado em que um modelo é treinado para generalizar para classes não vistas durante o treinamento. ZSL usa a compreensão contextual e as habilidades de codificação de modelos de linguagem pré-treinados para permitir a classificação sem a necessidade de entrada rotulada das classes invisíveis (Palaniappan; Vedhamani; Sundharakumar, 2023).

Essa técnica teve sua origem, sob essa designação, na classificação de imagens na área de Visão Computacional, com o trabalho de Palatucci et al. (2009). Desde então, o ZSL tem passado por notáveis avanços, notadamente influenciado pela teoria da transferência de aprendizado e pela representação semântica. Em um marco importante, o trabalho de Lampert et al. (2009), intitulado "*Learning to Detect Unseen Object Classes by Between-Class Attribute Transfer*", introduziu a inovadora ideia de transferência de atributos entre classes previamente observadas e aquelas que não foram vistas. Esse conceito de transferência entre atributos de classes conhecidas e desconhecidas contribuiu significativamente para o

desenvolvimento do ZSL, ampliando suas aplicações e eficácia em diversas áreas, incluindo o PLN.

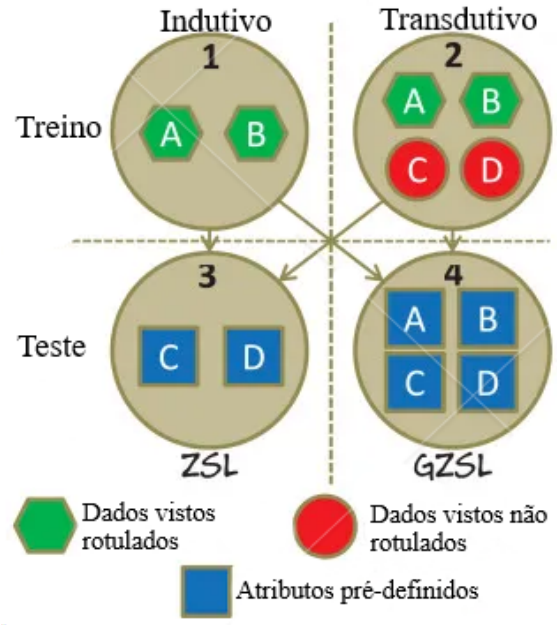
Conforme Wang et al. (2023), o ZSL pode ser implementado em duas configurações de treinamento: A transdutiva e a indutiva. No ZSL indutivo, os dados de origem que aparecem no treinamento (ou pré-treinamento) podem ter rótulos disponíveis nessa fase, mas esses dados não correspondem às classes invisíveis que são alvos na tarefa de predição. Dessa forma, a predição das classes de destino de um determinado atributo dos dados ocorre sem qualquer informação semântica prévia sobre as classes invisíveis. Essa abordagem tem sido a mais utilizada devido a sua flexibilidade e extensibilidade relativamente boas, segundo Sun, Gu e Sun (2021). Já no ZSL transdutivo, dados vistos rotulados e dados não vistos não rotulados estão disponíveis no treinamento. Ou seja, embora não tenham os rótulos das classes invisíveis durante o treinamento, os modelos utilizam informações descritivas ou representações vetoriais dessas classes para auxiliar na classificação. A Figura 9 exemplifica a diferença entre as duas configurações, em que as categorias C e D são as classes invisíveis e as classes A e B são categorias de dados usados no treinamento (ou pré-treinamento) e que não são classes invisíveis.

Outra metodologia para configuração de ZSL utilizada é a que separa a aprendizagem em classes invisíveis entre o ZSL convencional e o *Generalized* ZSL (GZSL). De acordo com Pourpanah et. al (2022), o GZSL visa treinar um modelo para classificar amostras de dados sob a condição de que algumas (mas não todas) classes de saída sejam desconhecidas durante o aprendizado supervisionado. Com isso, o GZSL aproveita as informações semânticas das classes vistas no treinamento para preencher a lacuna entre as classes vistas e não vistas. Por outro lado, no ZSL convencional, o treinamento (ou pré-treinamento) ocorre sem que nenhuma rotulagem para as classes de destino apareça no treinamento.

Na Figura 9, vemos as diferentes formas de se combinar as metodologias de ZSL, quais sejam, ZSL convencional indutivo, ZSL convencional transdutivo, GZSL indutivo e GZSL transdutivo.

Para Wang, Pan e Lin (2023), o ZSL pode economizar mais custos computacionais e consumo de tempo, ignorando totalmente as etapas de rotulagem, tokenização, pré-processamento de dados e extração de recursos sobre as classes invisíveis.

Figura 9 - ZSL/GZSL transdutivo e indutivo



Fonte: Adaptado de Nguyen (2023)[4]

4.10 FEW-SHOT LEARNING

Few-Shot Learning (FSL) é uma subárea do aprendizado de máquina que lida especificamente com a capacidade de um modelo de aprender com um número limitado de exemplos por classe. O desafio central do FSL é treinar modelos para realizar tarefas de classificação com base em uma quantidade restrita de dados de treinamento, muitas vezes composta por apenas algumas instâncias por categoria.

O FSL busca imitar a capacidade humana de aprender com poucas amostras, uma vez que as pessoas são capazes de compreender novos conceitos rapidamente e depois reconhecer variações desses conceitos em percepções futuras (Lake et al., 2011).

Conforme Gomes (2022), seu surgimento se deu no início do século, numa abordagem de aprendizagem não profunda, passando pela proposta do estudo de Fe-Fei et al. (2003), com o conceito de *One-shot learning*. Gomes (2022) também escreve que o período de aprendizagem profunda se dá a partir do trabalho de Koch et al. (2015), ganhando maior projeção a partir de então.

Semelhantemente ao ZSL, o FSL também pode ser configurado tanto com uma abordagem indutiva, em que há a inferência de regras gerais a partir de dados de rotulados de

[4] Disponível em: <https://medium.com/@minhanh.dongnguyen/zero-shot-learning-mls-ability-to-generalize-beyond-training-data-making-it-more-human-7b80259a1a41>. Acesso em 14 de dezembro de 2024.

treinamento (Colombo et. al, 2023), quanto numa abordagem transdutiva, modelando simultaneamente, durante treinamento, dados rotulados e dados não rotulados (Zhou et. al, 2024), que adicionam informações para a tarefa subsequente. Além disso, conforme Kukleva, Kuehne e Schiele (2021), FSL pode ser configurado em *Generalized* FSL (GFSL), em que a modelagem ocorre com classes base, para as quais existem muitas amostras de treinamento, e com novas classes, para as quais há apenas algumas amostras de treinamento disponíveis, contrastando com o cenário convencional de FSL, no qual somente as classes com poucas amostras aparecem no treinamento. Os dois tipos de configurações acima descritos podem ser combinados de forma parecida ao mostrado na Figura 9 para ZSL.

Wang, Pan e Lin (2023) afirmam que o FSL é vantajoso, mesmo com acesso a conjuntos de dados maiores, porque rotular dados é demorado e o treinamento em conjuntos de dados extensos pode ser computacionalmente caro.

5 TRABALHOS RELACIONADOS

O presente capítulo visa proporcionar uma visão abrangente do contexto de pesquisa no qual se insere o trabalho desenvolvido. O objetivo principal é examinar e contextualizar as contribuições prévias e os avanços significativos na área de estudo, estabelecendo um alicerce para a compreensão do estado-da-arte. Para isso, esta seção oferece uma síntese de alguns trabalhos relacionados relevantes encontrados. Cada resumo destaca as principais metodologias, técnicas, parâmetros e objetivos desses trabalhos, proporcionando uma visão clara das diversas abordagens que foram exploradas.

Foram realizados resumos expandidos de três trabalhos que foram considerados como referência muito próximas para o deste TCC. Além disso, foram feitos resumos menores de outros trabalhos que também mantêm relações com o tema desenvolvido neste projeto.

Ao final do capítulo, foi incluída uma análise crítica e comparativa dos trabalhos relacionados, para permitir destacar os pontos de convergência e as lacunas existentes neste campo de pesquisa, evidenciando, assim, a relevância do presente trabalho e seu potencial para realizar contribuições distintivas e para responder aos desafios específicos na área.

5.1 *FEW-SHOT AND ZERO-SHOT APPROACHES TO LEGAL TEXT CLASSIFICATION: A CASE STUDY IN THE FINANCIAL SECTOR*

Os autores do artigo *Few-shot and Zero-shot Approaches to Legal Text Classification: A Case Study in the Financial Sector* (Sarkar et al., 2021) propõem investigar as abordagens de *Few-shot* e *Zero-shot learning*. O contexto dessa tarefa diz respeito ao das organizações que são regidas por regulamentos em suas comunicações com o público, o que requer o cumprimento de padrões de conteúdo textual e a revisão de conteúdo por pessoas especializadas, para garantir a adequação do conteúdo às normas. Naturalmente, o trabalho para revisar esses textos é bem extenso, o que abre espaço para uso do PLN. No entanto, conforme apresenta o autor, há o desafio de se obter grandes conjuntos de treinamento para que modelos que usam PLN sejam eficazes, além de haver uma variedade de tarefas jurídicas e de legislação em constante evolução, tornando difícil construir dados de treinamento suficientes para cobrir todos os casos. Esses problemas, pela hipótese do autor, poderiam ser solucionados por meio do uso das técnicas anteriormente citadas, o que reduziria significativamente o custo da revisão jurídica de documentos.

Especificamente, o artigo traz um único exemplo de conformidade regulatória no domínio financeiro sob a regulamentação do FINRA (uma organização sem fins lucrativos autorizada pelo governo americano para supervisionar as corretoras dos EUA) examinando como seria possível treinar um sistema em um ambiente tradicional supervisionado com muitos dados e, em contraste, uma situação de treino *Zero-shot*, em que um perito jurídico fornece tão somente informações aproximadas sobre o que não está em conformidade com os requisitos normativos dentro de um texto analisado. O objetivo foi mostrar ser possível, com poucos dados de treinamento, alcançar um desempenho equivalente ao ambiente supervisionado com muitos dados e, assim, permitir que sistemas de classificação de texto para conformidade regulatória sejam construídos rapidamente e com pouco esforço, permitindo-lhes cobrir uma ampla gama de indústrias e quadros regulatórios.

Quanto à metodologia para tratar a problemática descrita no artigo, o autor usa uma arquitetura tripla (Wei et al., 2021), que consiste em três instâncias da mesma rede neural com parâmetros compartilhados, onde uma frase de entrada é comparada com um exemplo positivo e um exemplo negativo. Ou seja, em vez de olhar apenas para pequenos detalhes das frases, essa técnica considera informações importantes em toda a frase. Ela faz isso comparando três exemplos de cada vez: um exemplo principal, um exemplo semelhante e um exemplo diferente. Para fazer essa comparação, a metodologia usa um modelo chamado *Sentence*-BERT, que ajuda a entender o significado das frases, pois é treinado para entender as diferenças e semelhanças entre as frases, gerando dois valores: um que representa a diferença entre o exemplo principal e o exemplo positivo e outro que representa a diferença

entre o exemplo principal e o exemplo negativo. O modelo *Sentence*-BERT captura as informações contextuais em uma frase em uma representação de vetor de tamanho fixo. Essa representação contextual da frase é então alimentada em um *perceptron* de duas camadas. A camada oculta do *perceptron* possui ativação ReLU (Nair e Hinton, 2010) para introduzir não linearidade no *perceptron*.

Além disso, usa-se o conceito de Perda Tripla que permite que a rede distinga exemplos semelhantes e não semelhantes negativos de uma classe. Outro aspecto importante na metodologia trazida pelo artigo é a classificação de frases, onde exemplos da mesma classe estão próximos a partir do cálculo de proximidade proposto, com a medida da distância euclidiana e do uso de algoritmo Máquina de Vetores de Suporte (SVM) com *kernel* de Função de Base Radial (RBF).

Para viabilizar a tarefa do artigo, dados de fontes de dados internas e externas do setor de serviços financeiros foram recuperados para criar os conjuntos de dados iniciais para a abordagem. Após a configuração dos dados, limpou-se os dados para remover conteúdo duplicado e irrelevante para, então, garantir a qualidade dos dados antes da revisão. O conjunto de dados foi dividido em conjunto de treinamento, desenvolvimento e teste. Para o modelo de *Few-shot learning*, foram amostradas 40 sentenças de exemplos de promissórias e 190 de não promissórias do conjunto de treinamento.

A abordagem do artigo foi comparada com os métodos de aprendizado supervisionado de *Naive Bayes*, o *Multi-Layer Perceptron* (MLP), o *Support Vector Machine* (SVM), o *Sentence*-BERT e o *Laser embeddings*. Além das abordagens supervisionadas, a abordagem de *Few-shot learning* do artigo foi comparada com uma abordagem de *Zero-shot learning*. Foi usado o modelo BART (Lewis et al., 2020) como modelo de *Zero-shot learning* para avaliar, com base nas probabilidades calculadas, se uma frase se encaixa na categoria “promissória". Para probabilidade máxima maior que 0,7, classificou-se a frase como "promissória".

Os autores desse artigo obtiveram os seguintes resultados: Mesmo ao treinar com um número limitado de exemplos, o modelo de *Few-shot learning* alcança melhor desempenho de *recall* em comparação com diferentes modelos supervisionados. Descobriu-se que a precisão do modelo proposto é melhor do que o modelo de *Zero-shot learning*, mas inferior aos modelos supervisionados. No geral, o F-*Measure* mostra que resultados semelhantes podem ser obtidos com uma abordagem de *Few-shot* e isso permite o objetivo de um rápido treinamento de sistemas para diferentes tarefas jurídicas. Também concluiu-se que, na situação em que o classificador é aplicado como primeiro filtro, uma *recall* alto é preferível,

pois, segundo os autores, é melhor criar mais trabalho para uma segunda anotação manual do que perder alguns textos importantes.

Os autores também mostraram alguns exemplos de erros de classificação feitos pelo algoritmo e ressaltam o desafio e a necessidade de mudanças para melhorar os resultados. Eles também reconheceram que, quanto mais contexto, maior a probabilidade de melhorar a eficácia da abordagem.

Tabela 1 - Desempenho do modelo de aprendizagem de *Few-shot* do trabalho 5.1 em comparação com outros métodos de aprendizagem supervisionados e de *Zero-shot*

Model	Precision	Recall	F1	Accuracy
Naive Bayes	0.78	0.48	0.60	0.75
MLP	0.66	0.70	0.68	0.75
SVM	0.76	0.67	0.71	0.79
S-BERT	0.72	0.69	0.70	0.78
Laser	0.75	0.68	0.71	0.79
Zero-Shot	0.48	0.75	0.59	0.60
Few-Shot(ours)	0.61	0.74	0.67	0.72

Fonte: Sarkar et al., 2021

5.2 *UNLOCKING PRACTICAL APPLICATIONS IN LEGAL DOMAIN: EVALUATION OF GPT FOR ZERO-SHOT SEMANTIC ANNOTATION OF LEGAL TEXTS*

No artigo *Unlocking Practical Applications in Legal Domain: Evaluation of GPT for Zero-Shot Semantic Annotation of Legal Texts* (Savelka et al, 2023), os autores se propõem a analisar a capacidade, no estado da arte, de um modelo *Transformer Generative Pretrained* (GPT) para realizar anotação semântica de pequenos trechos de texto (de uma a poucas frases) provenientes de documentos legais de vários tipos, umas vez que, segundo o autor, até o momento da publicação do artigo, não havia uma análise rigorosa da capacidade de grandes modelos de linguagem (LLM) quanto à anotação semântica de textos legais em abordagens de *Few-shot learning*.

Como indicado no artigo, as aplicações práticas que podem se beneficiar desse trabalho envolvem, por exemplo, a revisão de contratos e a investigação em estudos jurídicos. Foi examinado se e com que sucesso o modelo pode anotar semanticamente pequenos lotes de trechos de texto curtos (10–50) com base exclusivamente em definições concisas dos tipos semânticos. Descobriu-se que o modelo GPT tem um desempenho surpreendentemente bom

em abordagens *Zero-shot* em diversos tipos de documentos (F1 = 0,73 em uma tarefa que envolve opiniões judiciais, 0,86 para contratos e 0,54 para estatutos e regulamentos).

De maneira mais específica, o autor avaliou a eficácia do GPT-3.5 (text-davinci-003) em tarefas focadas em revisão de contratos (usando o conjunto de dados Atticus de compreensão do contrato - CUAD - que é um corpo de 510 contratos jurídicos comerciais que foram rotulados manualmente sob a supervisão de advogados profissionais), investigação de disposições estatutárias/regulamentares federais, estaduais e locais americanos (conjunto de dados PHASYS, relacionados à preparação e resposta a emergências do sistema de saúde pública) e análise jurisprudencial (conjunto de dados do U.S. *Board of Veterans' Appeals* - BVA - que envolvem recursos de veteranos de guerra nos Estados Unidos em processos sobre benefícios, invalidez, etc). Cada um dos conjuntos de dados oferece suporte a diversas tarefas que envolvem diferentes tipos de documentos jurídicos, além de serem equipados com anotações especializadas anexadas, em geral, a pequenos trechos de texto.

Foi comparado o desempenho do modelo GPT-3.5 geral (não ajustado) na realização de anotações de pequenos lotes de trechos de textos curtos provenientes dos tipos de documentos legais mencionados acima em relação ao desempenho de um modelo tradicional de aprendizado de máquina estatístico (ML) (com o algoritmo *Random Forest*) e modelo BERT ajustado (RoBERTa). As anotações do modelo GPT são baseadas em textos compactos de uma frase com definições de tipo semântico fornecidas ao modelo na forma de um *prompt*. Foi analisada a seguinte questão de pesquisa no contexto das três tarefas de anotação legal: Dadas as breves definições de tipo de um único sistema não hierárquico que descreve pequenos trechos de texto, com que sucesso um modelo geral GPT-3.5 pode classificar automaticamente tais textos em termos de categorias?

O autor utilizou a medida de similaridade de Jaccard como *baseline* (*tokens* como conjuntos). Cada trecho de texto é comparado com as definições de tipo disponíveis na respectiva tarefa. Em seguida, é atribuído o rótulo cuja definição possui a maior pontuação de similaridade com o trecho. Além disso, o desempenho do GPT-3.5 foi comparado com um algoritmo tradicional de aprendizado estatístico supervisionado (*Random Forest*). Para comparar o desempenho do modelo GPT-3.5 de *zero-shot learning* com um LLM, foi usado o modelo RoBERTa com ajustes. Para avaliar quantos documentos rotulados foram necessários para que o sistema de ML supervisionado correspondesse e excedesse o desempenho do GPT-3.5, foi treinado a *Random Forest* e o RoBERTa em conjuntos de treinamento de tamanhos variados. Para testar o desempenho do text-davinci-003, o autor enviou um lote de trechos de texto usando a biblioteca OpenAI (Python), que é um *wrapper* para a API REST

do OpenAI. Os lotes foram tornados tão grandes quanto possível para alcançar a máxima rentabilidade. O autor também explicou no artigo a configuração de vários parâmetros do modelo com o fito de tentar maximizar a relação custo-benefício da abordagem proposta.

Ao apresentar os resultados dos experimentos que avaliaram o desempenho do text-davinci-003 em três tarefas (envolvendo pareceres adjudicatórios, cláusulas contratuais e disposições legais e regulamentares), o autor afirma que esse modelo superou a *baseline* na similaridade de Jaccard em todas as três tarefas, demonstrando um desempenho competitivo, especialmente considerando o tamanho limitado dos dados de treinamento. O modelo alcançou F1 = 0,73 para os papéis retóricos das sentenças de decisões judiciais, 0,86 para os tipos de cláusulas contratuais e 0,54 para fins de avaliação do sistema de saúde pública. Segundo o artigo, o modelo RoBERTa exigiu mais dados para igualar o desempenho do GPT-3.5. E o *Random Forest* mais dados ainda.

A Tabela 2 mostra os resultados experimentais em termos de pontuações micro-F1. A coluna Jaccard mostra o desempenho da linha de base de similaridade Jaccard. Os rótulos @N denotam quantos exemplos foram usados no treinamento dos dois sistemas de ML supervisionados (RandF – *Random Forest*, BERT – RoBERTa *base*), onde @Max significa que todos os exemplos disponíveis foram usados. A coluna GPT relata o desempenho do modelo text-davinci-003. As células azuis significam o ponto em que o sistema BERT correspondeu ao desempenho de qualquer um dos modelos GPT-3.5. As células sombreadas em verde fazem o mesmo para a *random forest*.

A análise detalhada feita no trabalho quanto ao desempenho do GPT-3.5 revelou que, embora tenha sido promissor, ainda há desafios, como a dificuldade em distinguir certas classes. O desempenho das cláusulas contratuais da CUAD pareceu, no geral, positivo. Já quanto aos documentos adjudicatórios, a classe "Raciocínio" foi particularmente problemática, com várias sentenças classificadas erroneamente. Finalmente, as disposições legais e regulamentares pareceram ser as mais desafiadoras, pois um grande número de disposições de “Resposta a Emergências” são rotuladas como “Preparação para Emergências”.

No geral, o desempenho é muito bom, mas não perfeito. Além disso, o desempenho do GPT-3.5 variou consideravelmente nos três conjuntos de dados analisados, dependendo, além do próprio conjunto de dados, da complexidade dos tipos semânticos. Isso, conforme o autor, sugeriu que, em alguns casos, o ajuste fino de um LLM pode ser necessário para obter resultados mais precisos, como, por exemplo, no conjunto de dados PHASYS, onde o desempenho do GPT-3.5 não é satisfatório, uma vez que foram identificados vários desafios

que este conjunto de dados apresentaram e que dificultam até mesmo os modelos de *machine learning* supervisionados.

O artigo concluiu que descobertas são importantes para profissionais jurídicos, educadores e acadêmicos que pretendem aproveitar os recursos de LLMs de última geração para reduzir o custo das cargas de trabalho existentes de alto volume, envolvendo anotação semântica de documentos legais, ou para desbloquear novos fluxos de trabalho que não seriam economicamente viáveis serem realizados manualmente ou usando *machine learning* supervisionado.

Tabela 2 - Resultados a sobre a aplicação do modelo text-davinci-003 para as três tarefas do trabalho 5.2

		@20		@50		@100		@250		@500		@1000		@Max		
	Jaccard	RandF	BERT	RandF	BERT	RandF	BERT	RandF	BERT	RandF	BERT	RandF	BERT	RandF	BERT	GPT
BVA	**.35**	**.30**	**.27**	**.31**	**.26**	**.51**	**.52**	**.65**	**.82**	**.65**	**.85**	**.73**	**.89**	**.83**	**.92**	**.73**
Citation	.71	.11	.00	.12	.00	.77	.49	.94	.97	.97	.99	.97	1.0	.99	1.0	.96
Evidence	.28	.58	.59	.61	.60	.69	.66	.77	.91	.76	.92	.81	.93	.87	.94	.77
Finding	.08	.22	.10	.14	.00	.35	.21	.50	.55	.48	.65	.63	.77	.74	.86	.53
Legal Rule	.15	.00	.00	.06	.00	.15	.15	.38	.77	.41	.84	.68	.91	.89	.95	.72
Reasoning	.24	.11	.10	.04	.00	.07	.00	.23	.52	.15	.56	.21	.64	.40	.71	.43
CUAD	**.38**	**.25**	**.12**	**.31**	**.14**	**.52**	**.43**	**.73**	**.68**	**.80**	**.87**	**.86**	**.93**	**.89**	**.95**	**.86**
Anti-assignment	.56	.12	.00	.16	.10	.78	.71	.87	.81	.90	.94	.93	.97	.94	.99	.96
Audit Rights	.50	.39	.12	.35	.06	.49	.71	.71	.82	.81	.94	.86	.97	.90	.96	.93
Cvnt. not to Sue	.31	.04	.00	.00	.00	.04	.00	.73	.26	.76	.87	.76	.93	.90	.96	.78
Governing Law	.52	.80	.17	.80	.22	.96	.92	.94	1.0	.98	1.0	.99	1.0	1.0	1.0	.99
IP Assignment	.05	.40	.00	.51	.00	.53	.08	.73	.13	.76	.84	.83	.88	.87	.94	.84
Insurance	.38	.11	.13	.23	.13	.64	.73	.93	.92	.90	.97	.93	.98	.95	.97	.96
Min. Commitment	.20	.00	.00	.23	.00	.52	.00	.64	.38	.72	.69	.77	.87	.81	.91	.67
Post-term. Services	.43	.21	.15	.29	.10	.44	.29	.60	.57	.72	.77	.77	.84	.78	.85	.69
Profit Sharing	.25	.08	.17	.04	.16	.03	.01	.56	.75	.76	.88	.86	.92	.87	.94	.78
Termination Cnv.	.43	.27	.00	.60	.00	.42	.00	.81	.81	.88	.91	.88	.95	.91	.97	.89
Volume Restriction	.07	.00	.00	.01	.00	.03	.00	.08	.30	.15	.41	.45	.78	.61	.90	.46
Warranty Duration	.21	.02	.00	.12	.00	.15	.00	.48	.21	.56	.76	.79	.91	.84	.93	.81
PHASYS	**.24**	**.52**	**.48**	**.53**	**.48**	**.53**	**.49**	**.56**	**.65**	**.59**	**.68**	**.61**	**.72**	**.63**	**.72**	**.54**
Preparedness	.24	.01	.00	.00	.00	.00	.00	.08	.31	.22	.46	.20	.56	.31	.61	.45
Response	.24	.77	.77	.77	.77	.77	.77	.78	.79	.78	.78	.79	.80	.80	.81	.63
Recovery	.27	.27	.00	.33	.00	.33	.03	.38	.55	.35	.59	.48	.60	.53	.67	.28

Fonte: Savelka et al, 2023

5.3 *PROCESSAMENTO DE LINGUAGEM NATURAL APLICADO AO RECONHECIMENTO DE ENTIDADES NOMEADAS EM TEXTOS LEGAIS EM PORTUGUÊS BRASILEIRO*

O trabalho *Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro* (Heck, 2022) propôs realizar um estudo sobre o Reconhecimento de Entidades Nomeadas (REN), que é uma técnica de Processamento de Linguagem Natural (PLN) que visa encontrar e identificar nomes específicos em documentos de texto a partir de entidades pré-definidas. No trabalho em

questão, a ideia utilizada foi a de aplicar a técnica para extrair informações importantes de documentos jurídicos em português, desenvolvendo um modelo de Inteligência Artificial que torna mais fácil analisá-los.

A monografia em epígrafe introduziu o tema abordando a importância da Extração de Informação (EI) e do Reconhecimento de Entidades Nomeadas (REN) no contexto da crescente quantidade de informações disponíveis em formato de texto. Segundo o trabalho desenvolvido, a EI visa localizar e estruturar informações relevantes em documentos, facilitando sua análise e manipulação. Ao usar o REN, buscou-se identificar nomes de entidades como pessoas, locais e organizações em um conjunto de dados de textos jurídicos conhecido como LeNER-Br, construído a partir de petições, mandados e decisões judiciais em geral.

Para construir o modelo, foram utilizadas arquiteturas de redes neurais, como a Bidirectional *Long Short-Term Memory* (BiLSTM) e a *Bidirectional Encoder Representation from Transformers* (BERT), que é uma tecnologia pré-treinada em língua portuguesa brasileira. Além disso, foram experimentadas versões com e sem o uso de camadas de *Conditional Random Field* (CRF).

Nessa conjuntura, a monografia estabeleceu como objetivo geral "produzir um modelo treinado de *deep learning* baseado em IA (especificamente, em técnicas de PLN) que encontre a citação de entidades pré-definidas em textos legais em língua portuguesa (isto é, que realize a tarefa de NER)" (Heck, 2022, pág. 16). Os objetivos específicos compreenderam: Testar diferentes tipos de modelos de *deep learning*; Disponibilizar um modelo adaptado ao contexto da empresa em que o projeto foi aplicado, no caso, para utilização em inteligência investigativa e segurança pública; Identificar dentre as métricas de avaliação de desempenho de modelos NER mais relevantes; Comparar desempenhos de diferentes modelos, em especial entre modelos de redes neurais, especificamente as Redes Neurais Recorrentes (RNR) e a arquitetura *Transformer*; E comparar o desempenho dos modelos obtidos com os de trabalhos relacionados.

Para contextualizar a pesquisa e a implementação dos modelos, a autora apresentou uma descrição da empresa onde o projeto foi realizado quanto à sua área de atuação (comunicação corporativa e inteligência investigativa) e como o trabalho desenvolvido contribuía para a organização.

Avançando no trabalho, a autora cita as ferramentas e métodos essenciais empregados na execução do projeto e fornece uma visão geral dos procedimentos seguidos para criar os modelos. Dentre esses elementos estão: O *dataset* de textos legais em português LeNER-Br,

formado por 70 textos jurídicos manualmente anotados e divididos entre os conjuntos de treino, de validação e de teste; As bibliotecas NumPy e PyTorch, escritas em Python, sendo a primeira otimizada para cálculos que envolvem *arrays* multidimensionais, e a segunda um *framework* de *Deep Learning*; JSON e CoNLL, usados como formatos de texto para algoritmos para treinamento das redes neurais; Modelos BERT pré-treinados oriundos do artigo de Souza, Nogueira e Lotufo (2019); Parâmetros para LSTM, que foram usados com base no trabalho de Luz de Araújo et al. (2018), além dos parâmetros baseados nos de Souza, Nogueira e Lotufo (2019) para as redes BERT; E os *embeddings* disponibilizados no trabalho de NILC (NILC, 2017), selecionados para serem usados no treinamento das redes LSTM.

Já para realizar a avaliação dos resultados, a autora escolheu o método CoNLL-2003 (TJONG KIM SANG; DE MEULDER, 2003). Para treinamento e teste das redes neurais, foi usada uma máquina com GPU, apesar de que os testes, em sua maioria, foram realizados com uso de CPU, o que, conforme descrito no trabalho, não afetou os resultados, levando-se em conta as métricas selecionadas.

Chegando próximo à parte final, os resultados são apresentados sob três aspectos. O primeiro aspecto se deu a partir do treinamento de redes BiLSTM, com e sem a camada CRF. Foi escolhido, para os dois cenários, um mesmo tipo de embedding Wang2Vec de dimensão 100, com o algoritmo de treinamento SkipGram. Foi demonstrado as métricas obtidas em forma de comparação, observando-se que o uso de uma camada CRF influencia nos modelos, gerando uma melhora substancial nas inferências. Foi realizado um segundo experimento, que consistiu em variar os tipos de *embeddings* usados e as dimensões deles. Com base em resultados obtidos por Hartmann et al. (2017) e usando uma rede de base com a camada CRF na saída, dois tipos de *embeddings* foram selecionados: Wang2Vec, com dois tipos de algoritmo de treinamento - SkipGram e CBoW; e GloVe. As duas dimensões selecionadas foram 100 e 300. Os melhores resultados alcançados através das variações de BiLSTM-CRF treinadas estão na casa dos 88% para a métrica F1-*score*, *Recall* e Precisão. O F1-*score*, por ser uma combinação de precisão e *recall*, foi usado para indicar o melhor modelo, no caso, o Wang2Vec - CBoW - dimensão 100.

O segundo aspecto se deu a partir de variações de redes BERT, no caso, dos tipos large e base, com e sem a camada CRF. Conclui-se que a rede BERT large tem melhor desempenho, e as redes com CRF também obtiveram valores de métricas mais altos em relação às sem essa camada.

O último aspecto sobre os resultados foi a realização de comparações entre os melhores resultados obtidos no projeto e os apresentados no modelo *baseline*, do trabalho de

Luz de Araújo et al. (2018). Observou-se que a rede BERT-CRF demonstrou os melhores resultados, superando o desempenho do baseline em aproximadamente 4,4 pontos percentuais quanto ao F1-*score* e do BiLSTM-CRF em 2,9 pontos percentuais. Esse resultado foi considerado pela autora como esperado, tendo em vista, segundo seu argumento, que os modelos estado-da-arte de PLN, no geral, são redes BERT. Também observou-se que o melhor modelo BiLSTM também superou o desempenho do modelo *baseline* em quase 1,5 ponto porcentual, resultado esse atribuído pela autora por conta do uso de diferentes *embeddings* e outras alterações na rede neural, como o tamanho de cada camada, em comparação com o trabalho de Luz de Araújo et al. (2018).

Segundo esse trabalho, "constatou-se que o desempenho em todas as classes de entidade do baseline é superado pelo modelo BERT-CRF por, em média, cerca de 3 pontos percentuais na precisão, 9,5 pontos percentuais no recall e 6,2 pontos percentuais no F1-*score*. "A melhora mais significativa foi na classe PESSOA, com 13,62 pontos percentuais de diferença no F1-*score*, seguida da classe LOCAL, com uma melhora de 12,18 pontos" (Heck, 2022, pág. 59). Além disso, a classe em que obteve-se o pior desempenho nos dois modelos foi LOCAL já que dados dessa classe foram mais raros no *dataset*. Por fim, a autora indica algumas inferências que o modelo BERT-CRF do trabalho fez corretamente, ao contrário do *baseline*, de Luz de Araújo et al. (2018).

Como conclusão, a autora resume as técnicas, modelos, métricas e resultados, mencionando que, no seu ponto de vista, o projeto alcançou seu objetivo geral e os objetivos específicos. Além disso, ela sugere, para trabalhos futuros, a expansão do conjunto de dados de treinamento com mais documentos jurídicos, o aumento da representação da classe "LOCAL", além da exploração da possibilidade de treinamento de *embeddings* de palavras específicas para o domínio jurídico.

Tabela 3 - Comparação de resultados dos modelos NER no trabalho 5.3

Modelo	Precisão	*Recall*	*F1-score*
Melhor BiLSTM-CRF	88,71%	87,50%	88,10%
Melhor BERT-CRF	**90,16%**	**91,86%**	**91,00%**
Luz de Araujo *et al.* (2018)	87,98%	85,29%	86,61%

Fonte: Heck, 2022

5.4 OUTROS TRABALHOS

5.4.1 *EXTRAÇÃO DE ENTIDADE DE PRODUTOS UTILIZANDO TÉCNICAS DE FEW-SHOT LEARNING*

O trabalho *Extração de Entidades de Produtos Utilizando Técnicas de Few-Shot Learning* (Chaves, A. , Barbosa, B. , Ferreira, D., 2022) tem como área de pesquisa a análise de dados não estruturados em formato de texto, especialmente para o comércio eletrônico, motivado pelo crescimento dos marketplaces, que apresenta o desafio da inclusão diária de novos produtos por diversos vendedores, podendo resultar em classes de produtos com poucas amostras nas bases de dados.

Diante desse cenário, esse estudo comparou o desempenho dos classificadores FSL *Matching Networks* e Redes Neurais Siamesas na tarefa de classificação de produtos em um *marketplace*, que envolve 34 classes e 394 amostras. Esses classificadores foram contrastados com o algoritmo K-Nearest Neighbors (KNN), e a Análise de Componentes Principais foi aplicada para redução da dimensão do banco de dados.

Os resultados indicaram que os algoritmos FSL superaram o KNN no teste de validação cruzada *leave-one-out*, alcançando uma acurácia de 96,85%, mesmo ao lidar com classes contendo um número reduzido de amostras.

Tabela 4 - Desempenho dos classificadores no trabalho 3.4.1

***Alg.**	***Acc.(1200)**	**Tempo (1200)**	**Acc.(307)**	**Tempo (307)**
SN	96,32%	3,2 horas	96,85%	2,3 horas
SNC	95,27%	2,1 horas	95,80%	1,0 horas
KNN	95,80%	NA*	95,80%	NA*
MN	96,32%	44,4 horas	96,85%	12,3 horas

*Abreviações: Acc.=acurácia em (%), Alg.=algoritmo, NA=não aplicável

Fonte: Chaves, A. , Barbosa, B. , Ferreira, D., 2022

5.4.2 *COMPANY CLASSIFICATION USING ZERO-SHOT LEARNING*

O trabalho *Company classification using zero-shot learning* (Rizinski et al., 2023) aborda a classificação de empresas, uma abordagem comum na pesquisa financeira, para

agrupar empresas semelhantes em categorias ou clusters. Tradicionalmente, essa classificação é feita por padrões estabelecidos, mas esses métodos têm limitações, como falta de interoperabilidade e lentidão nas atualizações.

O estudo propõe o uso de avanços recentes em aprendizado de máquina (ML) e processamento de linguagem natural (NLP) para superar essas limitações. Em particular, explora a classificação de texto usando métodos NLP, incluindo modelos de *transformer* pré-treinados. O foco é na abordagem de aprendizado de *zero-shot*, que permite que o modelo classifique inputs em várias classes sem treinamento específico para essas classes.

O experimento utiliza o modelo valhalla/distilbart-mnli-12-3 em um conjunto de dados da *Wharton Research Data Services* (WRDS) contendo informações de empresas classificadas pelo índice GICS. Os resultados mostram uma pontuação F1 de 0.56 usando as categorias originais, mas essa pontuação aumenta para 0.64 com modificações nos nomes dos setores. O método mostra potencial para automatizar a classificação de empresas, sendo útil em setores financeiros, de marketing e inteligência de negócios.

5.4.3 *JURISBERT: TRANSFORMER-BASED MODEL FOR EMBEDDING LEGAL TEXTS*

O trabalho *JurisBERT: Transformer-based model for embedding legal texts* (Viegas, 2022) propõe uma nova extensão do modelo BERT, especificamente voltada para o domínio jurídico, denominada JurisBERT. O foco principal desse modelo é aprimorar a tarefa de Similaridade Semântica Textual (SST) em textos legais, sendo treinado do zero com textos específicos da área legal, como leis, doutrinas e precedentes, obtendo maior precisão do que outros modelos BERT

Uma das vantagens do JurisBERT, segundo o autor, é que ele considera o conceito de sub linguagem, ou seja, ele foi refinado para entender melhor o jargão jurídico. Além disso, ele foi treinado com 24 mil pares de "ementas" (resumos de decisões judiciais) com graus de similaridade pré-definidos, tornando-o mais preciso para tarefas jurídicas. No contexto jurídico, isso pode ser útil para identificar casos semelhantes, analisar jurisprudências e auxiliar na busca por informações relevantes.

Para validar a abordagem, os autores criaram um conjunto de dados contendo 24 mil pares de ementas com graus de similaridade variando entre 0 e 3, extraídas de mecanismos de busca disponíveis nos sites dos tribunais brasileiros. A criação de um *dataset* especializado é um dos contributos importantes do trabalho.

O JurisBERT inclui 24.000 pares de ementas com graus de similaridade variando de 0 a 3. Os experimentos dos autores mostraram que o JurisBERT é melhor do que outros modelos em quatro cenários: BERT multilíngue e BERTimbau sem ajuste fino em cerca de 22% e 12% de precisão (F1), respectivamente; e com ajuste fino em cerca de 20% e 4%. Além da abordagem ter reduzido em 5 vezes as etapas de treinamento, usando hardware acessível.

5.5 TABELA COMPARATIVA

	Domínio	**Fonte dos Dados**	**Idioma dos dados**	**Baseline**	**Métrica de Avaliação**	**Resultados**
5.1	Conformidade regulatória na comunicação de organiza- ções financei- ras com o público	Fontes de dados internas e externas do setor de servi- ços financeiros americano (dados fechados)	Inglês	Naive Bayes; Perceptron Multicamadas; Support Vector Machine; Sentence-BERT; Laser.	Precisão, Recall, F1-score, Acurácia	Few-shot learning superou modelos supervisionados e mostrou eficácia na classificação de conformidade regulatória.
5.2	Legal (Contratos, Estatutos e Regulamentos)	https://www.phasys.pitt.edu/pdf/Code_Book_Numerical_Defintions.pdf (leis e regula- mentos) https://www.atticusprojectai.org/cuad (Contratos jurídi- cos comerciais) https://www.bva.va.gov/ (Decisões judiciais) (dados abertos)	Inglês	Similaridade de Jaccard entre os textos e as definições de tipo disponíveis; E Random Forest e modelo RoBERTa	F1-score, Similari- dade de Jaccard	GPT-3.5 supera a baseline em todas as tarefas pela simila- ridade de Jaccard, alcançando F1 = 0,73 para os papéis retóricos das sentenças de decisões judiciais, 0,86 para os tipos de cláusulas contratuais e 0,54 para fins de avaliação do sistema de saúde pública.

5.3	Jurídico (Reconhecimento de Entidades Nomeadas)	LeNER-Br https://paperswithcode.com/dataset/lener-br (petições, mandados, decisões judiciais) (dados abertos)	Português	Modelo do trabalho de Luz de Araújo et al. (2018)	Precisão, Recall, F1-score	BERT-CRF obteve melhor performance com precisão, recall e F1-score, respectivamente, 90,16%, 91,86% e 91,00%, superando a baseline.
5.4.1	E-commerce (Classificação de Produtos em Marketplaces)	Marketplace com 34 classes e 394 amostras, cedida pela empresa parceira do trabalho (dados fechados)	Português	K-Nearest Neighbors (KNN)	Acurácia e tempo	Matching Networks e Redes Siamesas superaram KNN em acurácia (96,85%) mesmo com classes com poucas amostras.
5.4.2	Análise de investimentos e Gestão de ativos	https://wrds-www.wharton.upenn.edu/ (descrições tex- tuais de empresas de capital aberto) (dados abertos)	Inglês	Modelos *facebook/bart-large-mnli* e *joeddav/xlm-roberta-large-xnli*	Precisão, Recall e F1-score	F1 score inicial de 0,56 usando os nomes originais das categorias do índice GICS. A aplicação da técnica TF-IDF vectorization aumentou a F1 score para 0,64
5.4.3	Jurídico (Similaridade Semântica Textual)	Súmulas, decisões, acordos e votos judiciais (Dados abertos do STF, STJ, TJRJ, TJMS)	Português	BERT multilíngue e BERTimbau sem ajuste fino e com ajuste fino	F1-score	JurisBERT comparado ao mBERT sem *fine-tuning* obteve F1 cerca de 22% maior. Comparado ao BERTimbau sem *fine-tuning*, obteve F1 cerca de 12% maior. Comparado ao mBERT com fine-tuning, obteve F1 cerca de 20% a maior. Comparado ao BERTimbau com *fine-tuning*, obteve F1 cerca de 4% maior.

5.6 DISCUSSÃO

Os trabalhos relacionados apresentados exploram abordagens diversas, desde *Few-shot* e *Zero-shot learning* até o uso de modelos *Transformer* e redes neurais para diferentes tarefas de PLN. Os resultados indicam aplicações práticas significativas, como a redução de custos na anotação semântica, automação na classificação de empresas e produtos, e melhorias na extração de informações de documentos. Além disso, todos os trabalhos reconhecem desafios, como a necessidade de dados de treinamento adequados, a variabilidade nas tarefas jurídicas e a importância de ajustes específicos para obter resultados mais precisos.

No presente trabalho, a tarefa pretendida para testar e analisar as abordagens que são objeto da pesquisa é a de classificação de texto, mais especificamente a classificação de documentos. Para essa tarefa, vislumbra-se a utilização de aprendizado de máquina supervisionado na classificação de texto, pois requer um conjunto de dados rotulados, ou seja, pares de entrada e saída, para que o modelo possa aprender a mapear os dados de entrada para as categorias desejadas e que serão pré-definidas. Nessa direção, usar a proposta do trabalho de Heck (2022) quanto à utilização do *Bidirectional Encoder Representations from Transformers* (BERT) ou similar parece ser útil, uma vez que esse é um modelo pré-treinado para Processamento de Linguagem Natural, de código aberto, que têm uma arquitetura flexível e já possui uma compreensão semântica rica para ser usada em uma variedade de tarefas de PLN (Cordeiro, 2019), o que o torna oportuno no estudo de FSL e ZSL em dados jurídicos. A literatura, conforme revisada por Vaswani et al. (2017) e Radford et al. (2018), sustenta a importância do BERT como um avanço significativo no campo de PLN. Além disso, conforme Heck (2022), o BERT é pré-treinado em português brasileiro.

Adicionalmente, a utilização do JurisBERT surge como uma opção promissora, pois, conforme demonstrado por Viegas (2022), é um modelo BERT treinado do zero em um extenso corpus de textos jurídicos brasileiros. Essa característica crucial o diferencia do BERT padrão, que abrange um vocabulário mais geral. Ao ser treinado especificamente com linguagem jurídica, o JurisBERT internaliza nuances, terminologias e a semântica peculiar do domínio legal de forma mais eficaz.

Nos trabalhos relacionados predominam as métricas Precisão, *Recall* e F1-*score*. Essas métricas também podem ser aplicadas neste TCC. Em um contexto jurídico, um alto *recall* é importante, pois é preferível identificar todas as instâncias relevantes (documentos jurídicos que requerem atenção) mesmo que isso resulte em alguns falsos positivos. A precisão também é importante para garantir que as instâncias identificadas como positivas pelo modelo sejam

verdadeiramente relevantes. No campo jurídico, a precisão pode ser crítica para evitar retrabalho associado a revisões manuais desnecessárias. Já o F1-*score* é uma métrica abrangente que equilibra *recall* e precisão, avaliando o desempenho global do modelo.

6. DESENVOLVIMENTO

Neste capítulo, será detalhado o desenvolvimento da pesquisa (Figura 12), por meio da qual será analisado o emprego de diferentes arquiteturas, metodologias e algoritmos de DL na classificação de documentos jurídicos em português brasileiro, de modo a avaliar o desempenho de modelos diversos na tarefa proposta, considerando as abordagens de ZSL e FSL no campo de PLN para os dados do trabalho. Em todos os métodos e configurações de treinamento abaixo descritos, foi avaliado o aprendizado indutivo.

Os 3 conjuntos de dados (VICTOR *small*, Votos TCU e Decisões STJ) utilizados nesta pesquisa foram obtidos a partir de trabalhos desenvolvidos por pesquisadores na área de PLN que disponibilizaram dados textuais jurídicos anotados no âmbito do Direito brasileiro. Todos os *datasets* se referem a textos de documentos legais disponíveis publicamente.

Foi necessário realizar uma etapa de pré-processamento, com a subamostragem de uma categoria específica do VICTOR, para tratamento de um desbalanceamento excessivo de classe. Outrossim, tanto esse *dataset* como os demais foram reduzidos a duas *features*, sendo uma para categorização do texto documental e a outra para o próprio texto.

Para uso do *dataset* VICTOR, foram escolhidas os métodos de ZSL generalizado (GZSL), em que, conforme Pourpanah et. al (2022), o conjunto de teste pode potencialmente conter tanto classes vistas como classes não vistas no treinamento, e FSL generalizado (GFSL), o qual, segundo Kim e Choi (2023), não visa apenas aprender sobre novas classes que foram pouco vistas no treinamento, mas também preservar o conhecimento existente para classes base. Nesse sentido, 1 classe de documento foi definida para ser a classe não vista (GZSL) ou pouco vista (GFSL), sendo então realizada a transformação adequada nos dados de treinamento e validação. Para as outras classes, vistas no treinamento, foram mantidos os números de exemplos, com exceção de um de uma delas, que sofreu a ação de *undersampling* para mitigação de desbalanceamento de classes.

Em relação ao *dataset* do STJ, também foram empregados os métodos de GFSL e GZSL, com 1 só classe, dentre 7 categorias de documentos, exercendo o papel de classe pouco vista ou não vista no treinamento. Mas, além disso, tanto esse último *dataset* quanto o *dataset* Votos TCU serviram para análise da tarefa de classificação de documentos jurídicos

com todas as suas categorias sendo pouco vistas ou não vistas, representando uma aplicação do FSL e ZSL puros. Por exemplo, em FSL (1-*shot*), todas as classes fornecem 1 exemplar para o treinamento, enquanto, em ZSL, nenhuma classe fornece exemplo de treinamento, de modo que a execução da tarefa é apoiada exclusivamente nos pré-treinamentos, que dão conhecimento prévio aos modelos.

Para os métodos GFSL e FSL aplicados aos 3 *datasets*, foram definidas as configurações para treinamento de cada um dos modelos com 1 exemplar (1-*shot*) e 5 exemplares (5-*shot*) da classe única pouco vista, no caso do GFSL, e 1 exemplar (1-*shot*) e 5 exemplares (5-*shot*) de cada classe pouco vista no FSL. As amostragens realizadas neste trabalho, tanto para *undersampling*, quanto para seleção de exemplos das classes pouco vistas, foram feitas usando funções em python que usam aleatoriedade.

Os modelos de DL utilizados são baseados no BERT e foram extensamente pesquisados, sendo considerados performáticos em tarefas de classificação de texto. São eles: ALBERT, RoBERTa, BERTimbau e JurisBERT. Todos esses modelos, obtidos a partir da plataforma *Hugging Face*, foram usados em suas versões menores em virtude das restrições de recursos computacionais desta pesquisa, sendo limitados a uma quantidade máxima de 512 *tokens* processados em uma única entrada. Em cada processamento de documentos que ultrapassaram esse limite, houve o truncamento automático, e todos os documentos truncados entraram nas fases treinamento, validação e teste.

O desenvolvimento dos códigos e treinamento dos modelos se deu no ambiente Jupyter *Notebook*, por meio da plataforma Google Colab, utilizando máquinas com GPU. A implementação completa da análise exploratória, do pré-processamento e da transformação dos dados, bem como os algoritmos de treinamento para todos os modelos em cada configuração selecionada para este trabalho está disponível no link https://github.com/fernando-cesar/TCC_SI_UFSC_24.2. Os conjuntos de dados derivados dos originais após as etapas de manipulação dos dados são encontrados no link https://drive.google.com/drive/folders/1sKMXvshAvP4G5KgCxm6-77nKZrUtBnVu?usp=sharing .

A Figura 10 demonstra como foram especificados nas implementações os hiperparâmetros ou argumentos da função de treinamento de cada modelo nos *datasets*, que seguiu valores fixos: 3 épocas (“*num_train_epochs*”), ou seja, o números de vezes que o modelo percorre todo o conjunto de dados de treinamento; taxa de aprendizado (“*learning_rate*”), representando tamanho dos passos que o algoritmo dá para ajustar os pesos da rede neural a cada iteração de treinamento, no valor de 0,00002; decaimento de peso

(“*weight_decay*”), que é uma técnica de regularização para minimizar o *overfitting*, de 0,01. A etapa de validação foi realizada ao final de cada época, usando o argumento “*eval_strategy*” do código (Figura 10) para defini-la, de forma que, após o *fine tuning* nos dados utilizados, o melhor modelo foi escolhido automaticamente para a etapa de teste. Essa última definição ocorre por meio do argumento “*load_best_model_at_end*”, sinalizando-o com “*True*” (Figura 10).

Figura 10 - Argumentos passados para os parâmetros internos dos treinamentos

```
# Argumentos de treinamento
training_args = TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    num_train_epochs=3,
    learning_rate=2e-5,
    weight_decay=0.01,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    push_to_hub=False,
    report_to="none",
)
```

Fonte: O autor

Não foi utilizada a etapa de o *fine tuning* manual com ajuste de hiperparâmetros, sendo os experimentos restritos aos ajustes automáticos dos modelos realizados sobre os dados durante a execução da função de treinamento (Figura 11), a partir dos hiperparâmetros fixos acima descritos. Ademais, os valores da taxa de aprendizado e decaimento de peso foram estabelecidos a partir de valores comuns de hiperparâmetros encontrados em pesquisas em ML. O número de épocas foi definido considerando os recursos computacionais disponíveis para este trabalho.

Figura 11 - Função de treinamento dos modelos

```
# Configuração do treinador
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=encoded_dataset["train"],
    eval_dataset=encoded_dataset["validation"],
    compute_metrics=compute_metrics
)
```

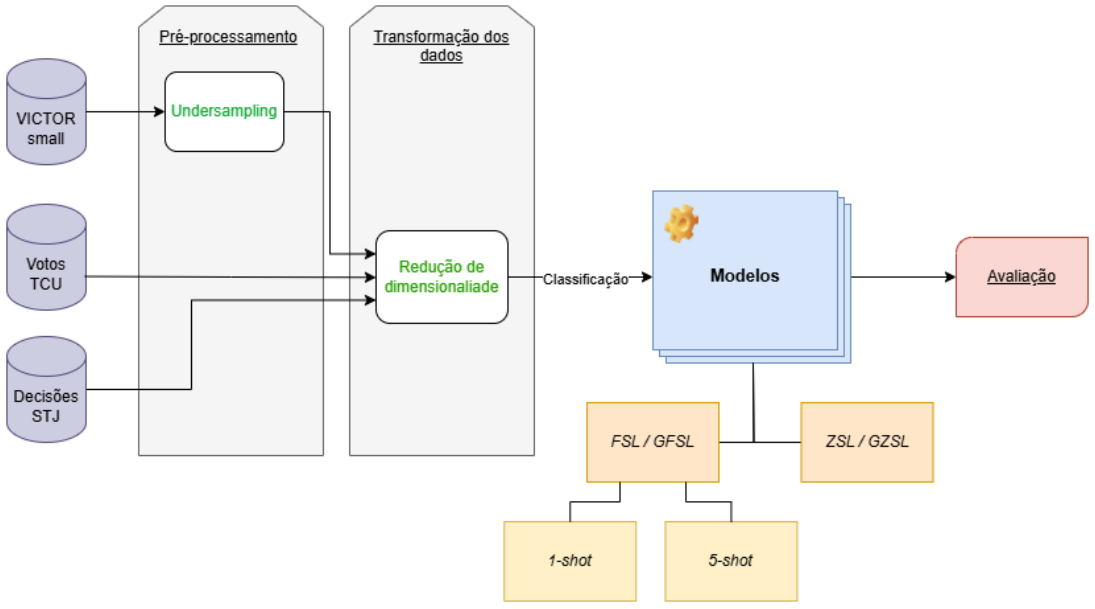
Fonte: O autor

É importante ressaltar que o *fine tuning* foi executado apenas nos métodos GFSL, GZSL e na abordagem FSL convencional, onde todas as categorias de documentos foram pouco vistas no treinamento. Na abordagem ZSL convencional, somente os dados de testes são utilizados, não havendo nenhuma classe vista disponível para treinamento.

Foi aproveitada a camada de classificação, geralmente linear, disponível em cada modelo, sendo apenas ajustado o número de neurônios dessa camada de saída para corresponder ao número de classes de cada *dataset*.

A avaliação do desempenho foi realizada por meio das métricas de acurácia, f1 *score*, precisão e *recall*, pois são métricas comuns e, geralmente, eficazes para trabalhos dessa natureza e facilitam a comparação com outros trabalhos da área.

Figura 12 - Etapas gerais do desenvolvimento do trabalho



Fonte: O autor

6.1. CONJUNTOS DE DADOS

6.1.1 VICTOR

O primeiro conjunto de dados que deu suporte a este trabalho foi o VICTOR (Luz de Araújo et al., 2020), construído a partir de documentos jurídicos digitalizados do Supremo Tribunal Federal do Brasil (STF), sendo parte de mais de 45 mil recursos feitos à corte constitucional brasileira, incluindo aproximadamente 692 mil documentos com cerca de 4,6 milhões de páginas. Destes processos, 44.855 são rotulados por especialistas, assim como seus 628.820 documentos. Neste trabalho de conclusão de curso, foi usada a versão pequena (*small*) do VICTOR disponibilizado pelos pesquisadores: VICTOR pequeno ou SVic, contendo 6.510 Recursos Extraordinários, 94.267 documentos e 339.478 páginas, sendo todas as amostras rotuladas. O conjunto de dados foi disponibilizado em arquivo com extensão .csv em três versões de diferentes tamanhos e conteúdos.

Segundo Luz de Araújo et al. (2020), os dados são resultados de alguns mecanismos de limpeza e pré-processamento como a remoção de caracteres especiais, o uso de minúsculas em todos os caracteres, a transformação de termos que são e-mails ou links em *tokens* "EMAIL" e "LINK", o stemming para reduzir palavras em sua raiz, reduzindo o número de palavras com semântica semelhante, dentre outros.

O *dataset* original contém dois tipos de anotação para as duas tarefas suportadas:

- Classificação de tipo de documento, cujas etiquetas são Acórdão, para decisões de primeira instância em revisão; Recurso Extraordinário (RE), para petições de apelação; Agravo de Recurso Extraordinário (ARE), para embargos à petição de apelação; Despacho, para ordens judiciais; Sentença para julgamentos; e Outros, para documentos que os autores não incluíram nas classes anteriores e que foram agrupados de maneira genérica, sem detalhamento do tipo de informação associada à categoria.
- Classificação de tema de processo, que atribuem um ou mais temas a cada Recurso Extraordinário. São 28 opções de temas mais frequentes, identificados por números inteiros, com uma classe possuindo o identificador 0 para os temas restantes.

As features do VICTOR estão ilustradas na Figura 13. Neste trabalho, o *dataset* foi segmentado para destacar a feature categoria do tipo de documento ("*document_type*"), além do próprio texto do documento. Isso tornou o desenvolvimento dos modelos mais eficiente ao

focar nos aspectos essenciais da classificação de documentos jurídicos (rótulo da classe e conteúdo textual). Gerou-se, então, um subconjunto do *dataset* original com dimensionalidade reduzida às colunas "*body*" e "*document_type*".

Figura 13 - *Features* do VICTOR

```
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149217 entries, 0 to 149216
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   themes         149217 non-null  object
 1   process_id     149217 non-null  object
 2   file_name      149217 non-null  object
 3   document_type  149217 non-null  object
 4   pages          149217 non-null  int64
 5   body           149217 non-null  object
dtypes: int64(1), object(5)
```

Fonte: O autor

Os conjuntos de dados originais são divididos em dados de treinamento, validação e teste, respeitando a proporção de 70%, 15% e 15%, respectivamente. Abaixo, a Tabela 5 mostra a distribuição das classes de documentos em cada subconjunto dos dados (treinamento, validação e teste) em cada versão do VICTOR.

Tabela 5 - Distribuição de classes por subconjunto dos dados do VICTOR

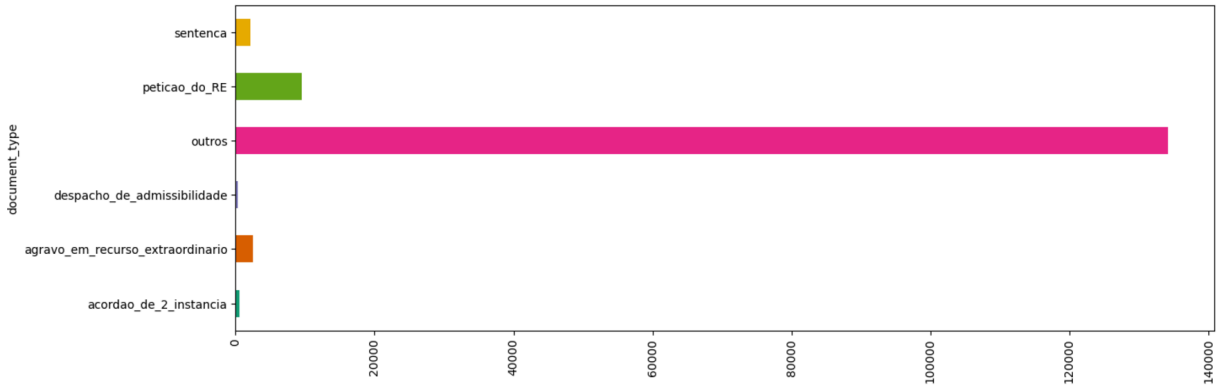
Dataset	Category	Training set		Validation set		Test set	
		Documents	Pages	Documents	Pages	Documents	Pages
MVic	Acórdão	1,966	4,740	354	656	358	659
	ARE	2,894	34,640	760	8,373	721	7,347
	Despacho	2,415	3,952	326	457	346	490
	Others	420,494	1,323,841	92,696	280,399	93,855	283,763
	RE	4,396	77,893	902	15,753	849	15,129
	Sentença	4,065	21,210	727	3,970	696	3,627
SVic	Acórdão	301	553	201	299	199	273
	ARE	270	2,546	237	2,149	213	1,841
	Despacho	265	346	147	183	147	198
	Others	38,585	134,134	25,898	84,104	25,744	85,408
	RE	453	9,509	326	6,364	312	6,331
	Sentença	420	2,129	284	1,636	265	1,475

Fonte: Luz de Araújo et al. (2020)

Todas as 6 classes de documentos do VICTOR *small* foram utilizadas na tarefa de classificação de documentos. É possível identificar um desbalanceamento extremo da classe

"*Others*" ou "Outros" (Figura 14), de forma a possuir um número de exemplares associados muito maior que aqueles identificados com as demais classes.

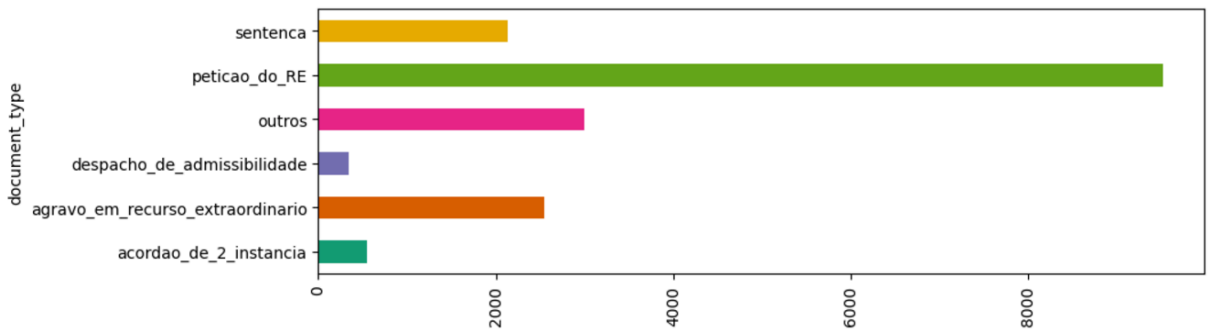
Figura 14 - Gráfico com a distribuição de exemplos por classes em subconjunto dos dados do VICTOR



Fonte: O autor

Como estratégia para lidar com esse problema, que pode afetar a acurácia dos resultados das aplicações dos modelos em uma tarefa de classificação, foi realizado um *undersampling* ou subamostragem, tanto no conjunto de dados de treinamento quanto nos conjuntos de dados de validação e teste, reduzindo o número de exemplos da classe "Outros" para um patamar próximo ao número de exemplos das outras classes. Nesses casos, a redução se deu para 3000 exemplares no conjunto de treinamento (Figura 15) e para 2000 exemplares nos conjuntos de validação e teste, de forma a reduzir o número de amostras da classe "outros" a patamares semelhantes ao dos números de amostras das demais classes, sendo os quantitativos em questão escolhidos de forma empírica.

Figura 15 - Gráfico com a distribuição de exemplos por classes em subconjunto dos dados de treinamento do VICTOR após *undersampling* da classe 'outros'



Fonte: O autor

Para uso do *dataset* VICTOR junto aos métodos de GZSL e GFSL, foi necessário escolher quais classes seriam consideradas classes “vistas”, cujos exemplos foram usados no treinamento dos modelos; quais seriam as classes “não vistas”, que foram usadas para avaliar os modelos em uma abordagem GZSL; e quais seriam as classes “pouco vistas”, utilizadas em abordagens GFSL. Para essa definição, foi utilizado o critério de frequência, em que as classes mais frequentes no *dataset* foram priorizadas como classes vistas no treinamento. Dessa forma, das 6 classes do *dataset* VICTOR, a classe que exerceu o papel de classe não vista na abordagem GZSL e de classe pouco vista na abordagem GFSL é a classe “Despacho”, pois contém o menor número de páginas no conjunto de dados. As outras classes do *dataset* foram atribuídas ao conjunto de classes vistas no treinamento.

No caso da abordagem GZSL, os conjuntos de dados de treinamento e validação não devem incluir exemplos da classe não vista, para que a avaliação seja adequada. Por isso, foi necessária uma transformação no *dataset* original, excluindo daquelas porções de dados as instâncias da classe “Despacho”. Já para a avaliação da abordagem FSL, foram constituídos *datasets* de treinamento subjacentes com 1 e 5 exemplos da classe pouco vista, ou seja, da classe “Despacho” (Tabelas 6 e 7), sendo selecionadas amostras aleatórias da referida classe. Para os conjuntos de validação usados em FSL, foram mantidos a quantidade de exemplos originais da classe “Despacho”.

Tabela 6 - Exemplos por classes para FSL com 1 exemplar da classe Despacho

Tipo de Documento	**Quantidade de Exemplos**
Acórdão de 2ª instância	553
Agravo em recurso extraordinário	2546
Despacho de Admissibilidade	1
Outros	3000
Petição do recurso extraordinário	9509
Sentença	2129

Fonte: O autor

Tabela 7 - Exemplos por classes para FSL com 5 exemplares da classe Despacho

Tipo de Documento	Quantidade de Exemplos
Acórdão de 2ª instância	553
Agravo em recurso extraordinário	2546
Despacho de Admissibilidade	5
Outros	3000
Petição do recurso extraordinário	9509
Sentença	2129

Fonte: O autor

Ao explorar os subconjunto de dados gerados, foi constatada uma quantidade considerável de documentos com quantitativo de *tokens* excedendo o limite de processamento dos modelos usados neste trabalho, a saber, de 512 *tokens*. A Tabela 8 traz informações quanto a esse aspecto.

Tabela 8 - Informações quanto ao tamanho dos documentos do *dataset* obtido a partir do VICTOR *small*

Tipo de Documento	Total de amostras do conjunto	Quantidade de amostras com mais de 512 *tokens*	Mínimo de *tokens* em um único documento	Máximo de *tokens* em um único documento	Média de *tokens* do conjunto
Conjunto de treinamento após *undersampling*	18083	1286	6	2324	309
Conjunto de validação após *undersampling*	12631	1286	5	1228	308
Conjunto de validação após *undersampling*	12118	861	6	1751	311

Fonte: O autor

6.1.2 Votos TCU e Decisões STJ

Outros dois subconjuntos foram utilizados: Um sendo dados anotados de 7403 decisões do Superior Tribunal de Justiça (STJ) e outro de 371 votos do Tribunal de Conta da União (TCU), ambos obtidos a partir da coleta de dados dos respectivos sites dos órgão colegiados. Esses *datasets* foram produzidos no âmbito do trabalho de Da Silva et. al (2024), *Datasets for Portuguese Legal Semantic Textual Similarity,* e foram disponibilizados em repositório público[5]. O conjunto de dados de votos do TCU contém os seguintes atributos (Figura 16): "Área", "tema", "subtema", "enunciado", "processo", "ano", "tipo_processo", "relator" e "voto". Por outro lado, o conjunto de dados STJ compreende os atributos "matéria", "natureza", "tema", "processo", "relator", "órgão", "data_julgamento", "data_publicação" e "ementa" (Figura 17). Para a tarefa de classificação deste TCC, foram definidas as classes "matéria" no conjunto de dados do STJ e "Área" no conjunto de dados do TCU para serem as classes alvos que serviram de sinal para o aprendizado supervisionado dos modelos da classificação, sendo os conjuntos de dados reduzidos às classes "matéria" e "ementa" no primeiro *dataset* e "área" e "voto" no segundo *dataset*.

Figura 16 - *Features* do *dataset* Votos TCU

```
df_tcu.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371 entries, 0 to 370
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Unnamed: 0     371 non-null    int64 
 1   AREA           371 non-null    object
 2   TEMA           371 non-null    object
 3   SUBTEMA        371 non-null    object
 4   ENUNCIADO      371 non-null    object
 5   PROCESSO       371 non-null    object
 6   ANO            371 non-null    object
 7   TIPO_PROCESSO  371 non-null    object
 8   RELATOR        371 non-null    object
 9   VOTO           371 non-null    object
dtypes: int64(1), object(9)
```

Fonte: O autor

[5] https://osf.io/k2qpx/.

Figura 17 - *Features* do *dataset* Decisões STJ

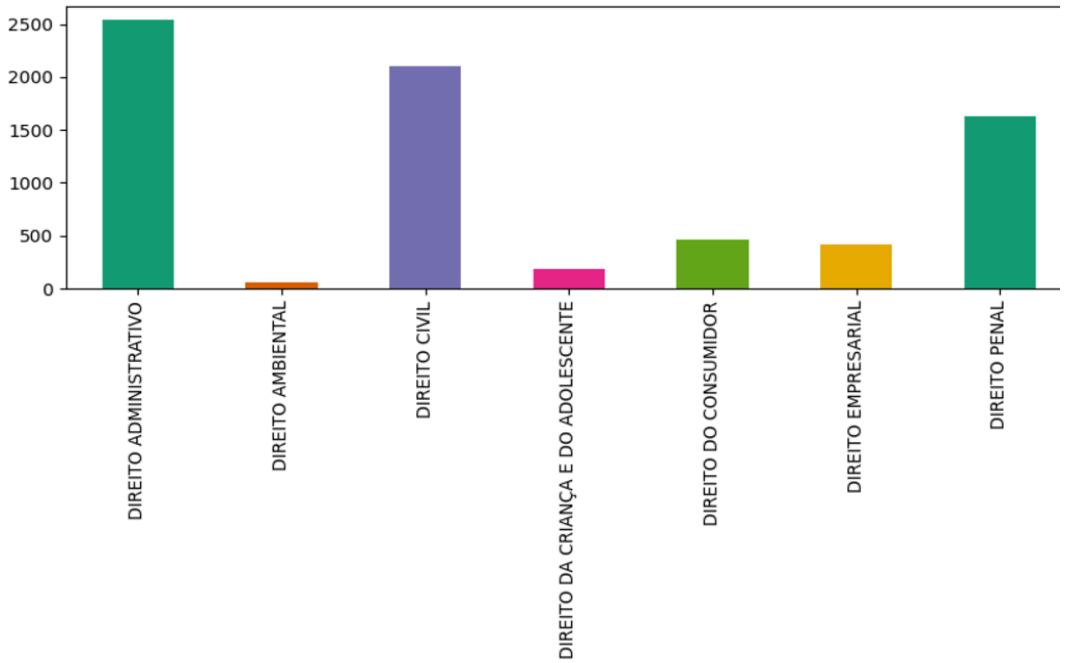
```
df_stj.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7407 entries, 0 to 7406
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Unnamed: 0       7407 non-null   int64 
 1   MATERIA          7407 non-null   object
 2   NATUREZA         7407 non-null   object
 3   TEMA             7407 non-null   object
 4   PROCESSO         7407 non-null   object
 5   RELATOR          7407 non-null   object
 6   ORGAO            7407 non-null   object
 7   DATA_JULGAMENTO  7407 non-null   object
 8   DATA_PUBLICACAO  7407 non-null   object
 9   EMENTA           7407 non-null   object
dtypes: int64(1), object(9)
```

Fonte: O autor

As matérias no conjunto de dados do STJ são das seguintes categorias: “Direito Administrativo”, “Direito Civil”, “Direito da Criança e do Adolescente”, “Direito do Consumidor”, “Direito Empresarial”, “Direito Ambiental”, “Direito do Consumidor” e “Direito Penal”. A distribuição de exemplos das categorias da classe “matéria” está representada no gráfico da Figura 18 e discriminada na Tabela 9.

Figura 18 - Distribuição de exemplos por categoria da feature “Matéria“



Fonte: O autor

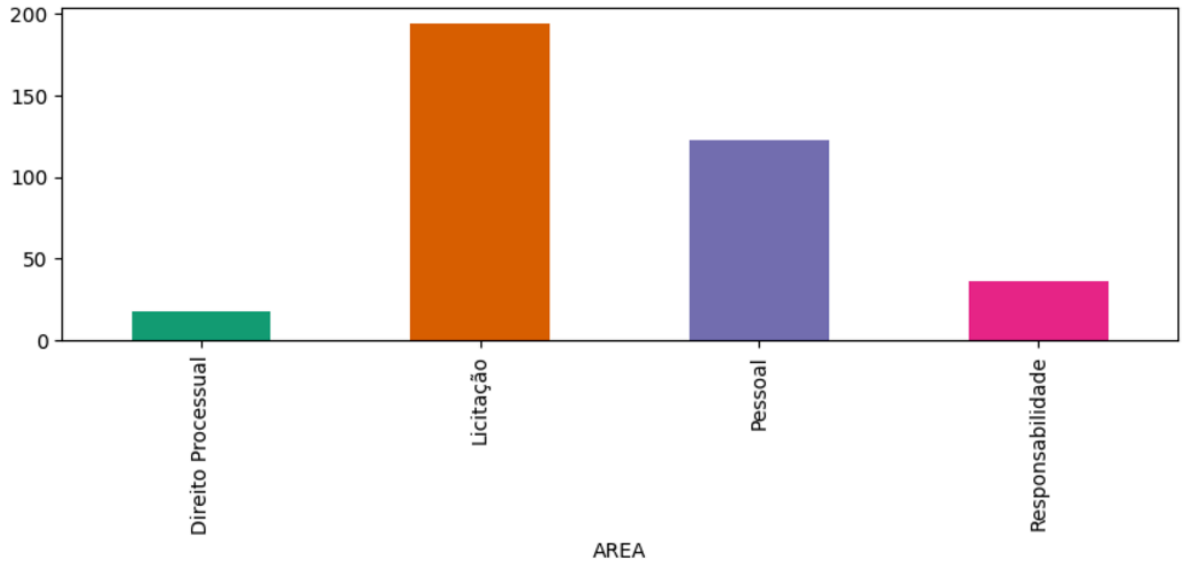
Tabela 9 - Distribuição de exemplos por categoria da feature “Matéria“

Tipo de Documento	Quantidade de Exemplos
Direito Administrativo	2542
Direito Civil	2108
Direito Penal	1631
Direito do Consumidor	459
Direito Empresarial	417
Direito da Criança e do Adolescente	188
Direito Ambiental	62

Fonte: O autor

Já as áreas atribuídas aos votos no conjunto de dados do TCU são das seguintes categorias: “Responsabilidade”, “Licitação”, “Direito Processual” e “Pessoal”. A distribuição de exemplos das categorias da classe “Área” está representada no gráfico da Figura 19 e discriminada na Tabela 10.

Figura 19 - Distribuição de exemplos por categoria da feature “Área”



Fonte: O autor

Tabela 10 - Distribuição de exemplos por categoria da feature “Área”

Tipo de Documento	Quantidade de Exemplos
Licitação	194
Pessoal	123
Responsabilidade	36
Direito Processual	18

Fonte: O autor

Para viabilizar a análise dos modelos aplicados, os dados do STJ e do TCU foram divididos em dados de treinamento, validação e teste usando-se a proporção de 70%, 15% e 15% (Tabelas 11, 12, 13, 14 e 15), respectivamente.

Tabela 11 - Distribuição de exemplos por classe da feature “Área” nos dados de treinamento do TCU

Tipo de Documento	Quantidade de Exemplos
Licitação	142
Pessoal	83
Responsabilidade	22
Direito Processual	12

Fonte: O autor

Tabela 12 - Distribuição de exemplos por classe da feature “Área” nos dados de validação e, igualmente, nos dados de teste do TCU

Tipo de Documento	Quantidade de Exemplos
Licitação	27
Pessoal	19
Responsabilidade	6
Direito Processual	4

Fonte: O autor

Tabela 13 - Distribuição de exemplos por classe da feature “Matéria“ nos dados de treinamento do STJ

Tipo de Documento	Quantidade de Exemplos
Direito Administrativo	1753
Direito Civil	1499
Direito Penal	1144
Direito do Consumidor	327
Direito Empresarial	278
Direito da Criança e do Adolescente	139
Direito Ambiental	44

Fonte: O autor

Tabela 14 - Distribuição de exemplos por classe da feature “Matéria“ nos dados de validação do STJ

Tipo de Documento	Quantidade de Exemplos
Direito Administrativo	386
Direito Civil	301
Direito Penal	252
Direito do Consumidor	73
Direito Empresarial	71
Direito da Criança e do Adolescente	25
Direito Ambiental	4

Fonte: O autor

Tabela 15 - Distribuição de exemplos por classe da feature “Matéria“ nos dados de teste do STJ

Tipo de Documento	Quantidade de Exemplos
Direito Administrativo	403
Direito Civil	308
Direito Penal	235
Direito do Consumidor	66
Direito Empresarial	61
Direito da Criança e do Adolescente	24
Direito Ambiental	14

Fonte: O autor

Além disso, do mesmo modo que o *dataset* VICTOR teve que ser adaptado para ser usado em abordagens GZSL e GFSL, também foi necessário escolher a classe não vista ou pouco vista nos treinamentos de modelos aplicados ao *dataset* Decisões STJ. A classe não vista/pouco vista do *dataset* do STJ foi “Direito Ambiental”. O critério foi o mesmo usado anteriormente, ou seja, a classe com menos exemplos.

Na sequência, foram criados *datasets* de treinamento subjacentes com 1 e 5 exemplos das classes pouco vistas a partir da seleção aleatória de amostras das respectivas classes, com o intuito de analisar a abordagem GFSL nos modelos usados nesses conjuntos de dados. De igual forma, foram derivados os conjuntos de dados de treinamento e validação sem as classes não vistas, para serem empregados na análise da abordagem GZSL.

A Tabela 16 traz informações acerca do quantitativo de *tokens* nos documentos dos 2 *datasets* descritos neste item, revelando limitações para processamento desses dados textuais nos modelos abordados, dado o limite de *tokens* explicado anteriormente.

Tabela 16 - Informações quanto ao tamanho dos documentos do *dataset* obtido a partir dos *datasets* Votos TCU e Decisões STJ

Tipo de Documento	**Total de amostras do conjunto**	**Quantidade de amostras com mais de 512 *tokens***	**Mínimo de *tokens* em um único documento**	**Máximo de *tokens* em um único documento**	**Média de *tokens* do conjunto**
Votos TCU	371	369	34	143566	13662
Decisões STJ	7407	2610	15	5018	507

Fonte: O autor

6.2. FERRAMENTAS

O desenvolvimento desta pesquisa se apoiou em um conjunto de ferramentas computacionais e bibliotecas de software, que possibilitaram a aplicação de técnicas de PLN e aprendizado de máquina para a classificação de documentos jurídicos. A seguir, são detalhadas as principais ferramentas utilizadas:

- Linguagem de Programação e Ambiente de Desenvolvimento Python
- Google *Collaboratory* (Colab) para a execução dos experimentos, com acesso a recursos computacionais de alto desempenho, como GPUs.
- Bibliotecas de Processamento de Linguagem Natural e Aprendizado de Máquina, como *Transformers* (*Hugging Face*), que fornece implementações de modelos de linguagem de última geração além de ferramentas para tokenização, treinamento e avaliação de modelos; Pytorch; Pandas; e Scikit-learn.

6.3. MODELOS

6.3.1 ALBERT

A versão do modelo ALBERT utilizada foi o ALBERT base, versão 2 (albert/albert-base-v2), com 12 camadas que compartilham pesos entre si, ou seja, em vez de cada camada ter seus próprios pesos independentes, todas as camadas utilizam o mesmo conjunto de pesos, reduzindo significativamente o número de parâmetros do modelo sem

alterar a profundidade. Além disso, são 768 dimensões ocultas, 12 cabeças de atenção, e apenas 11 milhões de parâmetros.

Essa abordagem economiza memória e acelera o treinamento, pois evita redundâncias entre camadas, mas mantém o modelo suficientemente profundo para capturar representações ricas do texto.

6.3.2 RoBERTa

A versão do modelo RoBERTa utilizada foi o RoBERTa base (FacebookAI/roberta-base), com 125 milhões de parâmetros, baseado em BERT, mas com vocabulário maior. Possui 12 camadas completas de *Transformer Encoder* e usa o *Byte Pair Encoding* (BPE) como método de tokenização, diferente do *Word Piece* usado no BERT.

A camada de classificação é uma camada linear simples, com função de ativação Softmax, que mapeia a dimensão oculta do modelo (768 neurônios) para o número de classes.

6.3.3 BERTimbau

A versão do modelo BERTimbau utilizada foi o BERTimbau base (neuralmind/bert-base-portuguese-cased), baseado na arquitetura do BERT base, com 110 milhões de parâmetros, distribuídos entre embeddings, pesos do mecanismo de atenção e *feed-forward*, além de possuir 12 camadas completas de *Transformer Encoder*.

A camada de classificação é uma camada linear simples, com função de ativação Softmax, que mapeia a dimensão oculta do modelo (768 neurônios) para o número de classes.

6.3.4 JurisBERT

O modelo JurisBERT (alfaneo/jurisbert-base-portuguese-uncased) utilizado neste trabalho tem a mesma arquitetura do BERT base, com 110 milhões de parâmetros, 12 camadas completas de *Transformer Encoder* e 768 neurônios na camada oculta.

7 RESULTADOS

A Tabela 17 exibe os resultados dos testes realizados nos modelos utilizando a abordagem GZSL aplicada aos *datasets* VICTOR *small* modificado e Decisões STJ. Pode-se

observar que o modelo JurisBERT com o último *dataset* atingiu os melhores resultados, com uma acurácia de 89,89% nos testes de classificação de documentos. Além disso, outras métricas como precisão, recall e F1-Score também apresentam os melhores resultados, com índices acima de 89%. Esse resultado pode ser atribuído ao treinamento do JurisBERT em textos jurídicos em Português, indicando como mais adequado à tarefa de classificação de documentos jurídicos em português do que os outros modelos multi-línguas ou treinados em textos genéricos.

Porém, quando esse mesmo resultado é comparado ao resultado da classificação convencional aplicada no *dataset* Decisões STJ com os mesmos modelos, mas sem fazer uso da abordagem de GZSL (Tabela 20), ou seja, usando-se todos os dados disponíveis para treinamento, incluindo a classe não vista em GZSL, repara-se que os resultados de classificação GZSL no geral é melhor, mesmo esperando que fosse pior, já que, nesse método, os modelos não dispõem de informações sobre 1 das classes durante o treinamento.

Esse fenômeno pode indicar que, para esse *dataset*, os modelos talvez estejam sendo tendenciosos para a as classes vistas, que são majoritárias em relação a classe não vista, uma vez que essa foi definida como a classe com menor frequência no *dataset*. Dessa forma, a abordagem GZSL, por não treinar diretamente para a classe "não vista", pode ter evitado esse viés em alguma medida. O mesmo não ocorre com o VICTOR modificado, provavelmente, por ter um proporção entre a classe minoritária (não vista/pouco vista) mais equitativa comparado à mesma proporção no *dataset* Decisões STJ.

Em relação ao *dataset* VICTOR *small* modificado, o modelo com melhor acurácia na abordagem GZSL foi o BERTimbau base cerca 80,73%, enquanto o JurisBERT obteve o f1-*score* mais alto, com 79,99%. Os valores obtidos por BERTimbau base e JurisBERT ficaram muito próximos, demonstrando que um modelo treinado em textos genéricos em português tem potencial de lidar bem com alguns documentos jurídicos, a depender da linguagem utilizada em documentos ou do tamanho do *dataset* usado para avaliação.

Os resultados comparados entre os GZSL aplicados aos dois *datasets* mostram uma relevante vantagem dos modelos quando executados sobre o *dataset* Decisões STJ. Analisando os dois conjuntos, verifica-se que o Decisões STJ possuem palavras chaves no início do documento, que podem facilitar a tarefa de classificação, especialmente considerando que os modelos usados só conseguem processar uma entrada de 512 *tokens* e que há muitos documentos com um número de *tokens* acima dessa quantitativo.

Tabela 17 - Resultados sobre o método GZSL com o *dataset* VICTOR *small* modificado e Decisões STJ

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**VICTOR modificado**	0.78358	0.78953	**0.80215**	0.78358
	Decisões STJ	0.85018	0.84728	0.84938	0.85018
RoBERTa base	**VICTOR modificado**	0.79482	0.79094	0.79403	0.79483
	Decisões STJ	0.83032	0.82641	0.82807	0.83032
BERTimbau base	**VICTOR modificado**	**0.80731**	0.79588	0.79095	**0.80731**
	Decisões STJ	0.89620	0.89466	0.89425	0.89620
JurisBERT	**VICTOR modificado**	0.80648	**0.79993**	0.79776	0.80649
	Decisões STJ	**0.89891**	**0.89638**	**0.89616**	**0.89892**

Fonte: O autor

Nos métodos GFSL, tanto com configuração 1-*shot* e quanto 5-*shot*, verificamos nas Tabelas 18 e 19 que o modelo JurisBERT ficou na frente no desempenho sobre o *dataset* Decisões STJ em todos os critérios estudados, com acurácia de 90% e F1-*score* de 89,7%, correspondendo à expectativa do modelo treinado em dados jurídicos obter o melhor resultado. Já quando se olha para os resultados do método sobre o *dataset* VITOR modificado, o modelo BERTimbau obtém desempenho ligeiramente superior, mas muito próximo do obtido com o JurisBERT, indicando com o BERTimbau tem capacidade para considerável de generalizar textos em português, mesmo em domínio específico, podendo superar um modelo treinado do zero em um único domínio.

Comparando os resultados obtidos em 1-*shot* e 5-*shot*, há um desempenho muito parecido dos modelos em ambas as configurações, com vantagens para a aplicação em 5-*shot*,

em especial no *dataset* VICTOR, sendo coerente ao número maior de dados da classe pouco vista que os modelos dispõem em abordagem 5-*shot*.

Tabela 18 - Resultados sobre o método GFSL (1-*shot*) com o *dataset* VICTOR *small* modificado e Decisões STJ

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**VICTOR modificado**	0.80540	0.79656	0.79010	0.80540
	Decisões STJ	0.85252	0.84551	0.84504	0.85252
RoBERTa base	**VICTOR modificado**	0.79709	0.78467	0.77948	0.79709
	Decisões STJ	0.85252	0.84814	0.84675	0.85252
BERTimbau base	**VICTOR modificado**	**0.80723**	**0.79820**	**0.79205**	**0.80723**
	Decisões STJ	0.89478	0.89253	0.89161	0.89478
JurisBERT	**VICTOR modificado**	0.80508	0.79593	0.78926	0.80508
	Decisões STJ	**0.90018**	**0.89793**	**0.89665**	**0.90018**

Fonte: O autor

Os resultados, no geral, aproximam o desempenho das abordagens de GZSL e GFSL (1-*shot* e 5-*shot*) do desempenho dos modelos quando a tarefa de classificação envolve dados de treinamento com rotulagem completa, sem classes não vistas ou pouco vistas.

Tabela 19 - Resultados sobre o método GFSL (5-*shot*) com o *dataset* VICTOR *small* modificado e Decisões STJ

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**VICTOR modificado**	0.77793	0.76660	0.76077	0.77793
	Decisões STJ	0.83273	0.81919	0.83418	0.83273
RoBERTa base	**VICTOR modificado**	0.78964	0.77615	0.77482	0.78964
	Decisões STJ	0.83543	0.81964	0.84047	0.83543
BERTimbau base	**VICTOR modificado**	**0.81870**	**0.81421**	**0.82079**	**0.81870**
	Decisões STJ	0.89748	0.89511	0.89527	0.89748
JurisBERT	**VICTOR modificado**	0.80558	0.79808	0.79149	0.80558
	Decisões STJ	**0.90108**	**0.89887**	**0.89794**	**0.90108**

Fonte: O autor

Tabela 20 - Resultados sobre a classificação de documentos do *dataset* VICTOR *small* modificado e Decisões STJ sem abordar GZSL/ZSL ou GFSL/FSL

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**VICTOR modificado**	0.80627	0.80256	0.80277	0.80627
	Decisões STJ	0.78147	0.73610	0.765327	0.78147
RoBERTa base	**VICTOR modificado**	0.80524	0.80167	0.802638	0.80524
	Decisões STJ	0.83543	0.82569	0.83979	0.83543
BERTimbau base	**VICTOR modificado**	**0.83208**	**0.82548**	**0.83514**	**0.83208**
	Decisões STJ	0.88759	0.88784	0.89162	0.88759
JurisBERT	**VICTOR modificado**	0.82741	0.82331	0.83296	0.82741
	Decisões STJ	**0.89478**	**0.89481**	**0.89585**	**0.89478**

Fonte: O autor

Quanto à aplicação dos modelos em abordagens ZSL e FSL ‘puras’ (Tabelas 22 e 23), o BERTimbau mostrou desempenho relativamente mais competitivo em alguns cenários, especialmente no *dataset* Votos TCU, em ZSL, enquanto o JurisBERT destacou-se no *dataset* Decisões STJ, tendo o melhor desempenho nos 3 cenários de ZSL e FSL avaliados. Isso provavelmente se deve à relevância dos dados legais brasileiros no treinamento do JurisBERT, permitindo uma compreensão mais profunda do vocabulário e das estruturas específicas do campo jurídico.

Esse comportamento se assemelha aos cenários anteriores em GZSL e GFSL, sendo possível obter conclusões parecidas no que diz respeito à influência das características dos

conjuntos de dados, uma vez que as palavras-chave jurídicas do início dos textos do *dataset* Decisões STJ, atreladas ao limite de 512 tokens de processamento de entradas nos modelos, podem favorecer o modelo treinado em textos jurídicos, o que não ocorre com o *dataset* Votos TCU.

Tabela 21 - Resultados sobre o método ZSL com os *datasets* Votos TCU e Decisões STJ

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**Votos TCU**	0.07143	0.00952	0.00510	0.07143
	Decisões STJ	0.03420	0.00511	0.00281	0.03420
RoBERTa base	**Votos TCU**	0.10714	0.02074	0.01148	0.10714
	Decisões STJ	0.01260	0.00031	0.00016	0.01260
BERTimbau base	**Votos TCU**	**0.19643**	**0.15580**	**0.14161**	**0.19643**
	Decisões STJ	0.01350	0.00210	0.13877	0.01350
JurisBERT	**Votos TCU**	0.10714	0.02074	0.01148	0.10714
	Decisões STJ	**0.33123**	**0.18943**	**0.13270**	**0.33123**

Fonte: O autor

Nota-se que, nessas abordagens convencionais de ZSL e FSL os modelos tiveram dificuldade em identificar corretamente as classes, com métricas significativamente mais baixas em relação aos métodos GFSL e GZSL. Isso ocorre porque o modelo depende exclusivamente de *embeddings* gerados a partir do seu pré-treinamento, sem qualquer ajuste específico aos dados, no caso do ZSL, ou com ajuste a pouquíssimos dados, no caso do FSL.

Tabela 22 - Resultados sobre o método FSL (1-shot) com os *datasets* Votos TCU e Decisões STJ

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**Votos TCU**	0.08929	0.06970	0.08162	0.08929
	Decisões STJ	0.27543	0.18621	0.16429	0.27543
RoBERTa base	**Votos TCU**	0.071429	0.00952	0.00510	0.071429
	Decisões STJ	0.05491	0.00572	0.00301	0.05491
BERTimbau base	**Votos TCU**	0.10714	0.08117	0.15271	0.10714
	Decisões STJ	0.21332	0.15337	0.37358	0.21332
JurisBERT	**Votos TCU**	**0.33929**	**0.20314**	**0.23798**	**0.33929**
	Decisões STJ	**0.36094**	**0.19993**	**0.20294**	**0.36094**

Fonte: O autor

Quando se compara os resultados da aplicação do método FSL na configuração de 1-*shot* com a configuração em 5-*shot*, nota-se um aumento relativo considerável, especialmente na métrica F1-*score*, em favor da configuração 5-*shot*, indicando que poucos amostras a mais nesses cenários de dados escassos podem ter impacto significativo nos resultados, embora os valores das métricas obtidos ainda estão muito aquém do observado no cenário sem restrição de dados (Tabela 20).

Tabela 23 - Resultados sobre o método FSL (5-shot) com os *datasets* Votos TCU e Decisões STJ

Modelos	Dataset	Acurácia	F1-score	Precisão	Recall
ALBERT base, versão 2	**Votos TCU**	0.3750	0.26489	0.40903	0.3750
	Decisões STJ	0.28803	0.18600	0.27236	0.28803
RoBERTa base	**Votos TCU**	0.10714	0.02143	0.01190	0.10714
	Decisões STJ	**0.35914**	0.19430	0.13733	0.35914
BERTimbau base	**Votos TCU**	0.35714	0.29781	0.25986	0.35714
	Decisões STJ	0.31773	**0.30401**	0.32064	0.31773
JurisBERT	**Votos TCU**	**0.41071**	**0.30382**	0.24107	0.41071
	Decisões STJ	0.09271	0.11650	0.34445	0.09271

Fonte: O autor

A Tabela 24 compara os resultados obtidos pelo modelo JurisBERT, que, no aspecto geral, foi o modelo de desempenho mais proeminente na tarefa de classificação dos dados jurídicos em português deste trabalho, em relação a aplicação dos paradigmas GZSL e GFSL (1-*shot* e 5-*shot*). É possível identificar que os valores obtidos para as métricas utilizadas foram muito próximos uns dos outros, mas ainda com vantagem para o 5-*shot*, seguido de 1-*shot* e, depois, *zero-shot*, considerando o *dataset* Decisões STJ. Isso denota que o treinamento com um exemplar único (1-*shot*) ou com 5 exemplares (5-*shot*) traz incrementos positivos nos resultados, embora a preponderância para contribuição para o resultado seja, em grande parte, relacionado ao conhecimento prévio do modelo pré-treinado.

Quanto aos resultados desses métodos sobre o *dataset* VICTOR *small* modificado, foi observado que a acurácia e o F1-*score* foram mais altos quando aplicado o ZSL, indicando que o(s) exemplo(s) da classe pouco vista nas configurações 1-*shot* e 5-*shot* não foram determinantes para a obtenção do melhor resultado. Quanto a isso, o formato dos documentos

e o truncamento dos textos, discutidos anteriormente, podem ser elencados como prováveis causas.

Tabela 24 - Resultados dos métodos GZSL e GFSL (1-*shot* e 5-*shot*) sobre os *datasets* VICTOR *small* modificado e Decisões STJ usando o modelo JurisBERT

Modelo JurisBERT	Dataset	Acurácia	F1-score	Precisão	Recall
Zero-shot generalizado	**VICTOR modificado**	0.80648	0.79993	0.79776	0.80649
	Decisões STJ	0.89891	0.89638	0.89616	0.89892
1-shot generalizado	**VICTOR modificado**	0.80508	0.79593	0.78926	0.80508
	Decisões STJ	0.90018	0.89793	0.89665	0.90018
5-shot generalizado	**VICTOR modificado**	0.80558	0.79808	0.79149	0.80558
	Decisões STJ	0.90108	0.89887	0.89794	0.90108

Fonte: O autor

A Tabela 25 também compara os resultados obtidos pelo modelo JurisBERT, só que em relação a aplicação dos paradigmas ZSL e FSL (1-*shot* e 5-*shot*). Para o *dataset* Votos TCU, os resultados obtidos na aplicação do método 5-*shot* são melhores que os obtidos na aplicação do 1-*shot*, que, por sua vez, traz rendimento melhor que o *zero-shot*. Isso é o esperado, devido às respectivas quantidades de exemplos para treinamento em cada método, com vantagem para o método com mais exemplos vistos.

Já em relação ao *dataset* Decisões STJ, as métricas de acurácia e F1-*score* tiveram valores menores na aplicação do 5-*shot*, evidenciando uma limitação da aplicação do ZSL e do FSL convencionais nos moldes deste trabalho.

Tabela 25 - Resultados dos métodos ZSL e FSL (1-*shot* e 5-*shot*) sobre os *datasets* Votos TCU e Decisões STJ usando o modelo JurisBERT

Modelo JurisBERT	Dataset	Acurácia	F1-score	Precisão	Recall
Zero-shot	**Votos TCU**	0.10714	0.02074	0.01148	0.10714
	Decisões STJ	0.33123	0.18943	0.13270	0.33123
1-shot	**Votos TCU**	0.33929	0.20314	0.23798	0.33929
	Decisões STJ	0.36094	0.19993	0.20294	0.36094
5-shot	**Votos TCU**	0.41071	0.30382	0.24107	0.41071
	Decisões STJ	0.09271	0.11650	0.34445	0.09271

Fonte: O autor

8 CONSIDERAÇÕES FINAIS

Este trabalho explorou o potencial do Processamento de Linguagem Natural (PLN), em especial as técnicas de *Few-shot Learning* (FSL) e *Zero-shot Learning* (ZSL), na classificação de elementos jurídicos a partir de peças processuais. A pesquisa se concentrou em analisar a viabilidade dessas abordagens em cenários com poucos ou nenhum exemplo de treinamento, visando auxiliar na automatização e gestão de processos no âmbito jurídico.

O objetivo inicial do trabalho era pesquisar diferentes modelos, algoritmos e metodologias para serem usadas em contexto de ZSL e FSL em tarefas de classificação de dados jurídicos em português. Este trabalho abordou o conceito de *Transfer Learning* usando modelos baseados em *transformers*. Explorar metodologias diferentes como *Prompt Learning* e suas diferentes abordagens, tais como ICL e CoT, e *Meta-learning*, esse com o uso de diferentes algoritmos, a exemplos do MAML e Prototypical Networks, são de grande relevância para expandir conhecimento acerca da aplicação de DL na automação da classificação de documentos jurídicos e é indicado para trabalhos futuros. Há também muita

pertinência em desvendar, em dados jurídicos em português no contexto de ZSL e FSL, o uso de metodologias de *ensemble learning*, em que múltiplos LLMs são combinados para obter melhor desempenho preditivo do que poderia ser obtido com apenas um dos modelos de constituintes. Melhorar o suporte semântico dos modelos com a utilização de sistemas de recuperação de informações (*Retrieval-Augmented Generation* - RAG) ou com a adição de mais informações sobre a classe não vista ou pouco vista no treinamento, incorporando, por exemplo, a descrição dessa classe, em uma abordagem transdutiva, também podem favorecer trabalhos posteriores.

A partir da análise exploratória de dados contextualizados do campo jurídico e da aplicação de diferentes algoritmos e modelos de PLN, foi possível observar o desempenho e as nuances do FSL e ZSL. Os resultados obtidos, avaliados quantitativamente, demonstraram o potencial e as limitações de cada modelo utilizado.

Observou-se que o GFSL e GZSL apresentaram desempenho semelhante se comparado à utilização dos modelos em contextos em que todas as classes possuem grandes quantidades de exemplos para treinamento na classificação dos documentos jurídicos, demonstrando a capacidade dos modelos de generalização a partir de poucos exemplos ou mesmo nenhum exemplo de treinamento. É interessante avaliar em pesquisas posteriores o desempenho dos modelos em tarefas de classificação com mais de 1 classe não vista ou pouco vista. Além disso, pode ser útil fazer testes com diferentes subamostragens do dataset VICTOR ou mesmo investigar a classe “outros”, buscando identificar e agrupar as possíveis subclasses a partir de técnicas de aprendizagem não-supervisionadas e a posterior análise e anotação manual de cada subcategoria.

Para abordagens ZSL e FSL convencionais, sem classes vistas e com quantidade de amostras semelhante das classes pouco vistas, respectivamente, o método de *transfer learning* com os modelos usados demonstrou baixo desempenho, dada a diferença do vocabulário ou da estrutura diferente dos dados originais do pré-treinamento.

Embora os modelos de PLN tenham apresentado bons resultados, em especial nos métodos GFSL e GZSL, identificou-se a necessidade de aprimoramento em relação ao tamanho do conjunto de dados e tamanhos dos modelos. Os modelos possuem limitações quanto à entrada de *tokens*, prejudicando a classificação, considerando que informações cruciais para a classificação podem estar em porções textuais não processadas pelos modelos. Nesse contexto, faz sentido pesquisar no futuro a aplicação de algumas técnicas para adequação dos dados textuais ao tamanho dos modelos menores, como a divisão dos textos maiores que o limite de *tokens* em textos menores, com alguma sobreposição de parte do texto

em pedaços consecutivos, para mitigar perda de contexto. Outro caminho possível seria usar LLMs para sumarizar os textos maiores que 512 *tokens*, adequando-os ao limite de processamento dos modelos que criam as *embeddings* para a classificação de documentos.

Por fim, este estudo contribui para o avanço da pesquisa em PLN aplicado ao Direito, demonstrando a viabilidade e os desafios da aplicação de FSL e ZSL na classificação de elementos textuais jurídicos em português. As descobertas aqui apresentadas podem auxiliar na criação de ferramentas automatizadas, que facilitem a análise de documentos no âmbito jurídico. Além disso, foi feita extensa pesquisa para a fundamentação teórica do trabalho, que pode ser útil para embasamento de pesquisas futuras.

Há muitas oportunidades de trabalhos a serem desenvolvidos posteriormente, dentre elas: Análise com diferentes bases de dados, especialmente com conjunto de dados de treinamento maiores, além do aprimoramento das bases textuais usadas nesta pesquisa; testes com outros modelos, sobretudo modelos maiores, ou mesmo com classificadores mais robustos; uso de diferentes hiperparâmetros e a avaliação do uso de outras técnicas para aplicação em abordagens ZSL e FSL.

REFERÊNCIAS

BROWN, Tom et al. **Language models are few-shot learners. Advances in neural information processing systems**, v. 33, p. 1877-1901, 2020.

NADKARNI, Prakash M.; OHNO-MACHADO, Lucila; CHAPMAN, Wendy W. **Natural language processing: an introduction**. Journal of the American Medical Informatics Association, v. 18, n. 5, p. 544-551, 2011.

KAVZOGLU, T., **Increasing the accuracy of neural network classification using refined training data, Environmental Modelling & Software**, Volume 24, Issue 7, Pages 850-858, 2009, ISSN 1364-8152, https://doi.org/10.1016/j.envsoft.2008.11.012

POLO, Felipe M.; MENDONÇA, Gabriel C. F.; PARREIRA, Kauê C. J.; GIANVECHIO, Lucka; CORDEIRO, Peterson; FERREIRA, Jonathan B.; LIMA, Leticia M. P. de; MAIA, Antônio C. do A.; VICENTE, Renato. **LegalNLP: Natural language processing methods for the brazilian legal language**. 2021, Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional, p. 763-774. Porto Alegre: SBC, 2021.

AGUIAR, André Wescley Oliveira de. **Um modelo de classificação multiclasse de processos judiciais em língua portuguesa**. 2023. 55f Fortaleza. Disponível em: https://biblioteca.sophia.com.br/terminal/9575/acervo/detalhe/583405. Acesso em: 11 set. 2024.

SCHOPF, Tim; BRAUN, Daniel; MATTHES, Florian. **Evaluating unsupervised text classification: zero-shot and similarity-based approaches**. In: Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval. 2022. p. 6-15.

DE ARAUJO, Pedro Henrique Luz et al. **VICTOR: a dataset for Brazilian legal documents classification**. In: Proceedings of the Twelfth Language Resources and Evaluation Conference. 2020. p. 1449-1458.

SILVA, F. C. E. ; CHAVES, A. G. S. ; BARBOSA, BRUNO H. G. ; Ferreira, D.D. . **Classificador Fuzzy-genético aplicado ao processamento de linguagem natural**. In: XXIII Congresso Brasileiro de Automática, 2020, Virtual. Anais do XXIII Congresso Brasileiro de Automática, 2020. v. XXIII. p. 1-8.

BROWNLEE, Jason. **Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems**. Machine Learning Mastery, 2017. 414 p.

BIRD, Steven; KLEIN, Ewan; LOPER, Edward. **Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit**. 1. ed. Sebastopol, CA: O'Reilly Media, 2009.

KOCHMAR, Ekaterina. **Getting Started with Natural Language Processing**. Manning, 2022. 456 p.

RATHOD, Shubhangi; GOVILKAR, Sharvari. **Survey of various POS tagging techniques for Indian regional languages**. Int. J. Comput. Sci. Inf. Technol, v. 6, n. 3, p. 2525-2529, 2015.

ALKAOUD, M.; SYED, M. **On the Importance of Tokenization in Arabic Embedding Models**. In: Proceedings of the Fifth Arabic Natural Language Processing Workshop, Barcelona, Spain (Online), 2020. p. 119–129. Association for Computational Linguistics.

HABERT, Benoit et al. **Towards tokenization evaluation**. In: Lrec. 1998. p. 427-432.

ALMEIDA, Felipe; XEXÉO, Geraldo. **Word embeddings: A survey**. arXiv preprint arXiv:1901.09069, 2019.

Wang, S., Zhou, W. & Jiang, C. **A survey of word embeddings based on deep learning**. Computing 102, 717–740 , 2020. https://doi.org/10.1007/s00607-019-00768-7

INCITTI, F., URLI, F., SNIDARO, L., **Beyond word embeddings: A survey**. Information Fusion, Volume 89, Pages 418-436, 2023, ISSN 1566-2535, https://doi.org/10.1016/j.inffus.2022.08.024

WANG, Congcong; NULTY, Paul; LILLIS, David. **A comparative study on word embeddings in deep learning for text classification**. In: Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval. 2020. p. 37-46.

LI, Qian et al. **A survey on text classification: From traditional to deep learning**. ACM Transactions on Intelligent Systems and Technology (TIST), v. 13, n. 2, p. 1-41, 2022.

MIKOLOV T., CHEN K., CORRADO G., DEAN J., **Efficient Estimation of Word Representations in Vector Space**. In Proceedings of Workshop at ICLR, arXiv; 2013. p. 1301-3781, 2013.

ETHAYARAJH, K., **How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings**. arXiv preprint arXiv:1909.00512, 2019.

WANG, Zhiqiang; PANG, Yiran; LIN, Yanbin. **Large Language Models Are Zero-Shot Text Classifiers**. arXiv preprint arXiv:2312.01044, 2023.

ALBRECHT, Jens; RAMACHANDRAN, Sidharth; WINKLER, Christian. **Blueprints for Text Analysis Using Python**. Sebastopol, CA: O'Reilly Media, 2021.

ZHAO, Wayne Xin et al. **A survey of large language models**. arXiv preprint arXiv:2303.18223, 2023.

Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. **Improving language understanding by generative pre-training**. 2018.

POURPANAH, Farhad et al. **A review of generalized zero-shot learning methods. IEEE transactions on pattern analysis and machine intelligence**, v. 45, n. 4, p. 4051-4070, 2022.

VIEGAS, Charles Felipe Oliveira. **JurisBERT: Transformer-based model for embedding legal texts**. 2022.

KIM, Seong-Woong; CHOI, Dong-Wan. **Better generalized few-shot learning even without base data**. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2023. p. 8282-8290.

DA SILVA JUNIOR, Daniel et al. **Datasets for Portuguese Legal Semantic Textual Similarity**. Journal of Information and Data Management, v. 15, n. 1, p. 206-215, 2024.

FIELDS, John; CHOVANEC, Kevin; MADIRAJU, Praveen. **A survey of text classification with transformers: How wide? How large? How long? How accurate? How expensive? How safe?**. IEEE Access, 2024.

GIRAY, Louie. **Prompt engineering with ChatGPT: a guide for academic writers**. Annals of biomedical engineering, v. 51, n. 12, p. 2629-2633, 2023.

LUO, Hengyu. **Prompt-learning and Zero-shot Text Classification with Domain-specific Textual Data**. Uppsala University, 2023.

KIM, Seungone et al. **The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning**. arXiv preprint arXiv:2305.14045, 2023.

WEI, Jason et al. **Chain-of-thought prompting elicits reasoning in large language models**. Advances in neural information processing systems, v. 35, p. 24824-24837, 2022.

KOJIMA, Takeshi et al. **Large language models are zero-shot reasoners**. Advances in neural information processing systems, v. 35, p. 22199-22213, 2022.

LI, Yinheng. **A practical survey on zero-shot prompt design for in-context learning**. arXiv preprint arXiv:2309.13205, 2023.

CAMPESATO, Oswald. **Natural Language Processing and Machine Learning for Developers**. Dulles: Mercury Learning and Information, 2021.

KULKARNI, Akshay; SHIVANANDA, Adarsha; KULKARNI, Anoosh. **Natural Language Processing Projects: Build Next-Generation NLP Applications Using AI Techniques**. Apress, 2022.

MINAEE, Shervin et al. **Large language models: A survey**. arXiv preprint arXiv:2402.06196, 2024.

PENNINGTON, J., SOCHER, R., MANNING, C., **GloVe: Global Vectors for Word Representation**. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics, 2014.

KHATTAK, Faiza Khan et al. A survey of word embeddings for clinical text. **Journal of Biomedical Informatics**, v. 100, p. 100057, 2019.

SINGH, Jyotika. **Natural Language Processing in the Real World: Text Processing, Analytics, and Classification**. 1. ed. Nova York: Chapman and Hall/CRC, 2023. 388 p. ISBN 9781003264774

TUNSTALL, Lewis; VON WERRA, Leandro; WOLF, Thomas. **Natural language processing with transformers**. O'Reilly Media, Inc., 2022.

CAMPESATO, Oswald. **Transformer, BERT, and GPT: Including ChatGPT and Prompt Engineering**. Boston, MA: Mercury Learning and Information, 2024.

LEE, Raymond S. T. **Natural Language Processing: A Textbook with Python Implementation**. 1. ed. Singapore: Springer Nature Singapore Pte Ltd., 2024.

JANIESCH, C., ZSCHECH, P. & HEINRICH, K. **Machine learning and deep learning**. Electron Markets 31, 685–695, 2021. https://doi.org/10.1007/s12525-021-00475-2

GÉRON, Aurélien. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2. ed. Sebastopol, CA: O'Reilly Media, 2019.

KELLEHER, John D. **Deep Learning**. The MIT Press, 2019. DOI: 10.7551/mitpress/11171.001.0001.

RAFF, E. **Inside Deep Learning: Math, Algorithms, Models**. New York: Manning Publications Co., 2022. ISBN 978-1617298639.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4. ed. United Kingdom: Pearson Education, 2021.[6]

GERSHEHSON, C.. **Artificial Neural Networks for Beginners.** Formal Computational Skills Teaching Package, COGS, University of Sussex, 2001.

HAGAN, Martin T.; DEMUTH, Howard B.; BEALE, Mark Hudson; DE JESUS, Orlando. **Neural Network Design**. 2. ed. [S.l.]: Martin Hagan, 2014. 800 p. ISBN 0971732116, 9780971732117

HAYKIN, Simon. **Neural Networks and Learning Machines**. 3. ed. Upper Saddle River, NJ: Pearson, 2009.

KROGH, A. **What are artificial neural networks?**. Nat Biotechnol 26, 195–197, 2008. https://doi.org/10.1038/nbt1386

QIU, Xipeng; SUN, Tianxiang; XU, Yige; SHAO, Yunfan; DAI, Ning; HUANG, Xuanjing. **Pre-trained models for natural language processing: A survey**. Science China Technological Sciences, v. 63, n. 10, p. 1872-1897, 2020.

BISHOP, Christopher M. **Reconhecimento de padrões e aprendizagem de máquina**. New York: Springer-Verlag New York, Inc., 2006. (Information Science and Statistics).

[6] Adaptação autorizada da edição dos Estados Unidos, intitulada Inteligência Artificial: Uma Abordagem Moderna, 4ª Edição, ISBN 978-0-13-461099-3 por Stuart J. Russell e Peter Norvig.

WEISS, K., KHOSHGOFTAAR, T.M. & WANG, D. **A survey of transfer learning**. *J Big Data* 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

AZUNRE, Paul. **Transfer Learning for Natural Language Processing**. Manning Publications, 2021.

SOUZA, F.C., NOGUEIRA, R.F., LOTUFO, R.A., **BERT models for Brazilian Portuguese: Pretraining, evaluation and tokenization analysis**, Applied Soft Computing, Volume 149, Part A, 2023 https://doi.org/10.1016/j.asoc.2023.110901

PAN, Sinno Jialin. Transfer Learning. **Data Classification: Algorithms and Applications**, v. 21, p. 537-570, 2014.

GHOJOGH, Benyamin; GHODSI, Ali. **Attention mechanism, transformers, BERT, and GPT: tutorial and survey**. 2020.

KITAEV, Nikita; KAISER, Lukasz; LEVSKAYA, Anselm. **Reformer: The efficient transformer**. arXiv preprint arXiv:2001.04451, 2020.

BELTAGY, Iz; PETERS, Matthew E.; COHAN, Arman. **Longformer: The long-document transformer**. arXiv preprint arXiv:2004.05150, 2020.

FEDUS, William; ZOPH, Barret; SHAZEER, Noam. **Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity**. Journal of Machine Learning Research, v. 23, n. 120, p. 1-39, 2022.

CLARK, Kevin; LUONG, MT; LE, Q. V.; MANNING, C. D. **Electra: Pre-training text encoders as discriminators rather than generators**. arXiv preprint arXiv:2003.10555, 2020.

BISHOP, Christopher M.; BISHOP, Hugh. **Deep Learning: Foundations and Concepts**. 1. ed. Switzerland: Springer, 2024. 649 p. ISBN 978-3-031-45467-7.

PATWARDHAN, N.; MARRONE, S.; SANSONE, C. **Transformers in the Real World: A Survey on NLP Applications**. Information 2023, 14, 242. https://doi.org/10.3390/info14040242

GALASSI, Andrea; LIPPI, Marco; TORRONI, Paolo. **Attention in natural language processing**. IEEE transactions on neural networks and learning systems, v. 32, n. 10, p. 4291-4308, 2020.

ISLAM, S.; ELMEKKI, H.; ELSEBAI, A.; BENTAHAR, J.; DRAWEL, N.; RJOUB, G.; Pedrycz, W. **A comprehensive survey on applications of transformers for deep learning tasks**. Expert Systems with Applications, p. 122666, 2023.

KAMATH, Uday; GRAHAM, Kenneth; EMARA, Wael. **Transformers for machine learning: a deep dive**. Chapman and Hall/CRC, 2022.

BAHDANAU, D.; CHO, K.; BENGIO, Y. **Neural machine translation by jointly learning to align and translate.** In: International conference on learning representations (ICLR), 2015, San Diego. San Diego: ICLR, 2015. p. 1-15.

ZHANG, Yue; TENG, Zhiyang. **Natural language processing: a machine learning perspective**. Cambridge University Press, 2021.

JURAFSKY, D., & MARTIN, J. H., **Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition**. Prentice-Hall, 2009.

Manning, C. D., & Schütze, H., **Foundations of statistical natural language processing**. MIT press Cambridge, 1999.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J., **Distributed representations of words and phrases and their compositionality**. In Advances in neural information processing systems, pp. 3111-3119, 2013.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K., **BERT: Pre-training of deep bidirectional transformers for language understanding**. arXiv preprint arXiv:1810.04805, 2018.

MARTINS, V. S.; SILVA, C. D. **Text classification in law area: a systematic review**. Journal of Information and Data Management, v. 13, n. 6, 2022.

LIU, Yinhan et al. **Roberta: A robustly optimized bert pretraining approach**. arXiv preprint arXiv:1907.11692, 2019.

LAN, Zhenzhong et al. **Albert: A lite bert for self-supervised learning of language representations**. arXiv preprint arXiv:1909.11942, 2019.

PALANIAPPAN, M.; VEDHAMANI, A.; SUNDHARAKUMAR K., **Zero-Shot Learning For Text Classification: Extending Classifiability Beyond Conventional Techniques**. In: 2023 IEEE Region 10 Symposium (TENSYMP). IEEE, 2023. p. 1-6.

WANG, Zhengbo et al. **Exploiting Semantic Attributes for Transductive Zero-Shot Learning**. arXiv preprint arXiv:2303.09849, 2023.

SUN, Xiaohong; GU, Jinan; SUN, Hongying. **Research progress of zero-shot learning. Applied Intelligence**, v. 51, p. 3600-3614, 2021.

LAKE, Brenden et al. **One shot learning of simple visual concepts**. In: Proceedings of the annual meeting of the cognitive science society. 2011.

IŞIK, Gültekin; PAÇAL, İshak. **Few-shot classification of ultrasound breast cancer images using meta-learning algorithms**. Neural Computing and Applications, p. 1-13, 2024.

GRIVA, Aikaterini I. et al. **Model-Agnostic Meta-Learning Techniques: A State-of-The-Art Short Review**. In: 2023 12th International Conference on Modern Circuits and Systems Technologies (MOCAST). IEEE, 2023. p. 1-4.

FINN, Chelsea; ABBEEL, Pieter; LEVINE, Sergey. **Model-agnostic meta-learning for fast adaptation of deep networks**. In: International conference on machine learning. PMLR, 2017. p. 1126-1135.

FE-FEI, L. et al. **A bayesian approach to unsupervised one-shot learning of object categories**. In: IEEE. proceedings ninth IEEE international conference on computer vision. [S.l.], 2003. p. 1134–1141.

GOMES, Jacó Cirino. **Uma arquitetura de few-shot learning para classificação de insetos na agricultura usando poucas amostras**. 2022. xii, 63 f., il. Dissertação (Mestrado em Sistemas Mecatrônicos) — Universidade de Brasília, Brasília, 2022.

KOCH, G.; ZEMEL, R.; SALAKHUTDINOV, R. et al. **Siamese neural networks for one-shot image recognition**. In: LILLE. ICML deep learning workshop. [S.l.], v. 2, p. 0, 2015.

ZHOU, Chunpeng et al. **Less is more: A closer look at semantic-based few-shot learning. Information Fusion**, v. 114, p. 102672, 2024.

PALATUCCI, M.; POMERLEAU, D.; HINTON, G.; MITCHELL, T. **Zero-shot learning with semantic output codes**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 22. 2009. p. 1410-1418.

LAMPERT, C. H.; NICKISCH, H.; HARMELING, S. **Learning to detect unseen object classes by between class attribute transfer**. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2009. p. 951-958.

SARKAR, Rajdeep; OJHA, Atul Kr.; MEGARO, Jay; MARIANO, John; HERARD, Vall; MCCRAE, John P. **Few-shot and Zero-shot Approaches to Legal Text Classification: A Case Study in the Financial Sector**. In: Proceedings of the natural legal language processing workshop, Punta Cana, Dominican Republic. Punta Cana: Association for Computational Linguistics, 2021. p. 102-106.

WEI, J.; HUANG, C.; VOSOUGHI, S.; CHENG, Y.; XU, S**. Few-shot text classification with triplet networks, data augmentation, and curriculum learning**. In: NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, 2021, Online, June 6-11, 2021. p. 5493-5500.

KUKLEVA, Anna; KUEHNE, Hilde; SCHIELE, Bernt. **Generalized and incremental few-shot learning by explicit learning and calibration without forgetting**. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021. p. 9020-9029.

NAIR, V.; HINTON, G. E. **Rectified linear units improve restricted Boltzmann machines**. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 27., 2010, Haifa, Israel. p. 807-814.

LEWIS, M.; LIU, Y.; GOYAL, N.; GHAZVININEJAD, M.; MOHAMED, A.; LEVY, O.; STOYANOV, V.; ZETTLEMOYER, L. **BART: denoising sequence-to-sequence**

pretraining for natural language generation, translation, and comprehension. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 58., 2020, Online, July 5-10, 2020. p. 7871-7880.

SAVELKA, Jaromir. **Unlocking practical applications in legal domain: Evaluation of gpt for zero-shot semantic annotation of legal texts**. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law. 2023. p. 447-451.

HECK, Amabyle Rabeche. **Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro**. 2022. 71 f. TCC (Graduação) - Curso de Engenharia de Controle e Automação, Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2022. Disponível em: https://repositorio.ufsc.br/handle/123456789/233250. Acesso em: 16 out. 2023.

SOUZA, Fábio; NOGUEIRA, Rodrigo; LOTUFO, Roberto. **Portuguese Named Entity Recognition using BERT-CRF**. arXiv preprint arXiv:1909.10649, 2019. Disponível em: http://arxiv.org/abs/1909.10649.

LUZ DE ARAUJO, Pedro Henrique. **lener-br**. [S.l.]: GitHub, 2020. Disponível em: https://github.com/peluz/lener-br.

NILC. **Repositório de Word Embeddings do NILC**. [S.l.: s.n.], 2017. Disponível em: http://www.nilc.icmc.usp.br/embeddings

TJONG KIM SANG, Erik F.; DE MEULDER, Fien. **Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition**. In: PROCEEDINGS of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003. [S.l.: s.n.], 2003. P. 142–147. Disponível em: https://aclanthology.org/W03-0419.

HARTMANN, Nathan; FONSECA, Erick R.; SHULBY, Christopher; TREVISO, Marcos Vinıcius; RODRIGUES, Jéssica S.; ALUISIO, Sandra M. **Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks**. CoRR, abs/1708.06025, 2017. arXiv: 1708.06025. Disponível em: http://arxiv.org/abs/1708.06025.

G. S. CHAVES, ALESON ; H. G. BARBOSA, BRUNO ; D. FERREIRA, DANTON. **Extração de Entidades de Produtos Utilizando Técnicas de Few-Shot Learning**. In: XV Simpósio Brasileiro de Automação Inteligente, 2021, Online, 2022. v. 01

RIZINSK, Maryan; JANKOV, Andrej; SANKARADAS, Vignesh; PINSK, EugeneI; MISKOVSKI, Igor and TRAJANOV, Dimitar. **Company classification using zero-shot learning**. arXiv preprint arXiv:2305.01028, 2023. Disponível em: https://arxiv.org/abs/2305.01028v2

COLOMBO, Pierre et al. **Transductive learning for textual few-shot classification in API-based embedding models**. arXiv preprint arXiv:2310.13998, 2023.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. **Attention is all you need**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2017. p. 6000-6010.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding with unsupervised learning**. Technical report, OpenAI, 2018.

LIN, T; WANG, Y; LIU, X; QIU, X. **A survey of transformers**. AI open, v. 3, p. 111-132, 2022.

SOUZA, M.; SOUZA, R. R. **Modelagem de tópicos: resumir e organizar corpus de dados por meio de algoritmos de aprendizagem de máquina.** Disponível em: http://hdl.handle.net/20.500.11959/brapci/137081. Acesso em: 26 nov. 2023.

CORDEIRO, Bernardo Cardoso. **Bert e Word2vec: Uma análise inferencial e computacional na classificação de textos com redes neurais convolucionais**. 2019. 51 f. TCC (Graduação) - Curso de Engenharia de Computação e Informação, Escola Politécnica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2019. Disponível em: https://pantheon.ufrj.br/handle/11422/18239. Acesso em: 26 nov. 2023.

APÊNDICE A – ARTIGO DO TRABALHO

Zero-shot learning e Few-shot learning no desenvolvimento de modelos inteligentes para classificação e análise de documentos jurídicos

Fernando Cesar da Costa Junior[1], Elder Rizzon Santos[1]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 88.040-370 – Florianópolis, SC – Brasil

fernando.cesar.c.j@grad.ufsc.br, elder.santos@ufsc.br

Abstract. *This work aims to explore the development of NLP and the role that this area can play in the legal field, investigating the potential of Zero-shot Learning (ZSL) and Few-shot Learning (FSL) in the classification of legal documents. Deep Learning (DL) models, based on Transformers, were implemented and applied to Law, based on the text contained in legal documents. The performance of the models in different ZSL and FSL methods was compared. The results were promising for generalized ZSL and FSL, especially with the JurisBERT and BERTimbau models, reaching values around 90% for accuracy and F1-score, as well as difficulties when applying the traditional ZSL and FSL methods.*

Resumo. *Este Trabalho busca explorar o desenvolvimento do PLN e o papel que essa área pode exercer no campo jurídico, investigando o potencial do Zero-shot Learning (ZSL) e Few-shot Learning (FSL) na classificação de documentos do âmbito legal. Foram implementados modelos de Deep Learning (DL), baseados em Transformers, aplicando-os ao Direito, a partir do texto contido em peças judiciais. Foram comparados o desempenho dos modelos em diferentes métodos de ZSL e FSL. Os resultados foram promissores para ZSL e FSL generalizados, em especial com os modelos JurisBERT e BERTimbau, alcançando-se valores em torno de 90% para acurácia e F1-score, assim como foram encontradas dificuldades quando aplicados os métodos de ZSL e FSL tradicionais.*

1. Introdução

O Processamento de Linguagem Natural (PLN) tem desempenhado um papel cada vez mais importante na área da Inteligência Artificial, permitindo que as máquinas compreendam e processem a linguagem humana (Nadkarni; Ohno-Machado; Chapman, 2011). Com o avanço das técnicas de *Machine Learning* (ML), surgiram abordagens inovadoras, como o *Few-Shot Learning* (FSL) e o *Zero-Shot Learning* (ZSL), que têm sido exploradas em diversas esferas e desempenham um papel importante no desenvolvimento do campo do PLN.

Conforme Brown et al. (2020), em "*Language Models are Few-Shot Learners*", o *Few-Shot Learning* refere-se à capacidade de um modelo aprender com um número limitado de exemplos de treinamento. O mesmo autor também explica que o *Zero-Shot Learning* possui o enfoque no aprendizado com classes ou conceitos para os quais o modelo não foi explicitamente treinado. Essas abordagens são significativas, pois, os modelos usados em tarefas de aprendizado tradicionais, como, por exemplo, classificadores baseados em Redes Neurais Artificiais, costumam requerer grandes volumes de dados para generalizar bem para novos exemplos (Kavzoglu, 2009). No entanto, em muitos cenários do mundo real, a coleta de dados pode ser cara, demorada ou simplesmente inviável.

Os desafios para a obtenção de uma grande quantidade de dados qualificados envolvem extensas anotações manuais para rotulá-los em categorias e a dependência de expertise de domínio para uma classificação precisa. Nessa conjuntura, o domínio jurídico apresenta um terreno fértil para a aplicação de estratégias de ZSL e FSL, pois aquela área tem como uma de suas características o dispendioso trabalho de análise e interpretação de documentos textuais e processos formais, demandando um esforço expressivo por parte dos profissionais do Direito. Além disso, há situações em que o número de documentos jurídicos pode ser limitado, levando a uma pouca ou nenhuma disponibilidade de exemplos para treinamento para as classes ou categorias de interesse. Então, a aplicação de técnicas avançadas de *Machine Learning*, especificamente o ZSL e o FSL, no âmbito do PLN, surge como uma solução capaz de proporcionar benefícios substanciais a partir da contribuição para a otimização dos processos legais e da abertura de caminho para a automação de tarefas complexas e o acesso mais rápido e efetivo ao conhecimento jurídico, mesmo com poucos dados.

A utilização de modelos de ML em cenários ZSL e FSL com o intuito de aprimorar a eficiência, a precisão e a acessibilidade na análise de textos jurídicos requer a execução de tarefas de PLN como a classificação de texto, que representa uma tarefa chave neste trabalho. A classificação de texto envolve atribuir rótulos a documentos ou partes de documentos, com base em seu conteúdo, tornando-se útil para organizar os dados em categorias específicas.

Logo, o propósito deste trabalho é pesquisar, analisar e comparar diferentes modelos e estratégias de ML, aplicando-os ao uso das técnicas FSL e ZSL na execução de tarefas de classificação de documentos, dentro do contexto jurídico, nos casos em que a disponibilidade de dados é restrita, e como tais abordagens podem ampliar a aplicabilidade de modelos de PLN em ocorrências do mundo real.

2. Fundamentação teórica

Few-Shot Learning (FSL) é uma subárea do aprendizado de máquina que lida especificamente com a capacidade de um modelo de aprender com um número limitado de exemplos por classe. O FSL busca imitar a capacidade humana de aprender com poucas amostras, uma vez que as pessoas são capazes de compreender novos conceitos rapidamente e depois reconhecer variações desses conceitos em percepções futuras (Lake et al., 2011). Wang, Pan e Lin (2023) afirmam que o FSL é vantajoso, mesmo com acesso a conjuntos de dados maiores, porque rotular dados é demorado e o treinamento em conjuntos de dados extensos pode ser computacionalmente caro.

O *Zero-Shot Learning* é uma estratégia de aprendizado em que um modelo é treinado para generalizar para classes não vistas durante o treinamento. ZSL usa a compreensão contextual e as habilidades de codificação de modelos de linguagem pré-treinados para permitir a classificação sem a necessidade de entrada rotulada das classes invisíveis (Palaniappan; Vedhamani; Sundharakumar, 2023).

Conforme Wang et al. (2023), o ZSL pode ser implementado em duas configurações de treinamento: A transdutiva e a indutiva. No ZSL indutivo, os dados de origem que aparecem no treinamento (ou pré-treinamento) podem ter rótulos disponíveis nessa fase, mas esses dados não correspondem às classes invisíveis que são alvos na tarefa de predição. Já no ZSL transdutivo, dados vistos rotulados e dados não vistos não rotulados estão disponíveis no treinamento. Ou seja, embora não tenham os rótulos das classes invisíveis durante o treinamento, os modelos utilizam informações descritivas ou representações vetoriais dessas classes para auxiliar na classificação.

Outra metodologia para configuração de ZSL utilizada é a que separa a aprendizagem em classes invisíveis entre o ZSL convencional e o *Generalized* ZSL (GZSL). De acordo com Pourpanah et. al (2022), o GZSL visa treinar um modelo para classificar amostras de dados sob a condição de que algumas (mas não todas) classes de saída sejam desconhecidas durante o aprendizado supervisionado. Com isso, o GZSL aproveita as informações semânticas das classes vistas no treinamento para preencher a lacuna entre as classes vistas e não vistas. Por outro lado, no ZSL convencional, o treinamento (ou pré-treinamento) ocorre sem que nenhuma rotulagem para as classes de destino apareça no treinamento.

3. Trabalhos Correlatos

O artigo de Sarkar et al. (2021) investiga o uso de *Few-shot* e *Zero-shot learning* para classificar textos legais, buscando alternativas ao PLN tradicional que exige grandes conjuntos de dados. Usando dados do setor financeiro e uma arquitetura tripla com *Sentence*-BERT, os autores compararam seu modelo com métodos supervisionados e *Zero-shot* (BART). O modelo *Few-shot* obteve bom *recall*, superando outros em alguns casos, mostrando-se promissor para construir rapidamente sistemas de classificação com poucos dados, especialmente útil como um primeiro filtro em revisões legais.

O artigo de Savelka et al. (2023) avaliou o GPT-3.5 para anotação semântica *zero-shot* de textos legais, comparando-o com *Random Forest* e RoBERTa. Usando definições concisas, o GPT-3.5 apresentou bom desempenho em contratos (F1=0,86), decisões judiciais (0,73) e textos regulatórios (0,54). RoBERTa e, principalmente, *Random Forest*, necessitaram de mais dados para atingir desempenho similar. O estudo concluiu que o GPT-3.5 tem potencial para reduzir custos em tarefas de anotação legal, mas pode exigir ajustes finos dependendo da complexidade do domínio.

Heck (2022) desenvolveu um modelo de Reconhecimento de Entidades Nomeadas (REN) para textos jurídicos em português, usando BiLSTM e BERT com e sem CRF, treinados com LeNER-Br. O modelo BERT-CRF criado superou o modelo anterior de Luz de Araújo et al. (2018), principalmente nas classes PESSOA e LOCAL, demonstrando a eficácia do uso de *Transformers* para REN em textos legais.

O trabalho de Chaves et al. (2022) investigou a extração de entidades de produtos usando *Few-Shot Learning* (FSL) no contexto do comércio eletrônico, onde a adição constante de novos produtos por diversos vendedores resulta em classes com poucas amostras. O estudo comparou os classificadores FSL *Matching Networks* e Redes Neurais Siamesas com o algoritmo KNN em um *marketplace* com 34 classes e 394 amostras. Os algoritmos FSL apresentaram melhor desempenho que o KNN no teste de validação cruzada *leave-one-out*, atingindo acurácia de 96,85%, mesmo com poucas amostras por classe.

O trabalho de Rizinski et al. (2023) aborda a classificação de empresas, comum em pesquisas financeiras, buscando alternativas aos métodos tradicionais. O estudo propõe o uso de aprendizado de máquina (ML) e processamento de linguagem natural (NLP), focando na classificação de texto com modelos *Transformer* pré-treinados, especificamente a abordagem de aprendizado *zero-shot*. Utilizando o modelo valhalla/distilbart-mnli-12-3 e dados da WRDS com classificação GICS, os resultados alcançaram um F1-score de 0.56 com as categorias originais e 0.64 com modificações nos nomes dos setores. O método demonstra potencial para automatizar a classificação, sendo útil em finanças, marketing e inteligência de negócios.

O trabalho de Viegas (2022) apresenta o JurisBERT, uma extensão do modelo BERT treinada especificamente para o domínio jurídico, com foco em Similaridade Semântica Textual (SST). O JurisBERT foi treinado do zero com textos legais (leis, doutrinas e precedentes) e 24 mil pares de ementas com graus de similaridade pré-definidos, considerando o jargão jurídico ("sub linguagem"). A criação deste dataset especializado é um dos contributos do trabalho. Os experimentos demonstraram que o JurisBERT superou outros modelos (BERT multilíngue e BERTimbau) em até 22% (sem ajuste fino) e 20% (com ajuste fino) em precisão (F1), além de reduzir as etapas de treinamento em 5 vezes com hardware acessível.

4. Desenvolvimento

Este capítulo descreve o desenvolvimento da pesquisa, que analisa o uso de diferentes arquiteturas, metodologias e algoritmos de *Deep Learning* na classificação de documentos jurídicos em português brasileiro, avaliando o desempenho de modelos em tarefas de *Zero-Shot Learning* e *Few-Shot Learning* no campo de PLN. A pesquisa utilizou aprendizado indutivo e três datasets: VICTOR small, Votos TCU e Decisões STJ, todos com textos legais publicamente disponíveis.

As etapas incluíram pré-processamento (subamostragem no VICTOR para balanceamento de classes e redução a duas features para todos os *datasets*). Para o VICTOR, foram aplicados ZSL generalizado (GZSL) e FSL generalizado (GFSL), com uma classe definida como não vista (GZSL) ou pouco vista (GFSL) durante o treinamento. Para o STJ e Votos TCU, além de GFSL e GZSL, foram analisadas tarefas de FSL e ZSL puros, onde todas as categorias eram pouco vistas ou não vistas, respectivamente. Configurações de 1-*shot* e 5-*shot* foram usadas para GFSL e FSL. As amostragens foram aleatórias.

Os modelos de DL utilizados, da plataforma *Hugging Face*, foram ALBERT, RoBERTa, BERTimbau e JurisBERT (versões menores, com limite de 512 *tokens* e truncamento automático quando necessário), devido a restrições computacionais. O desenvolvimento e treinamento ocorreram em Jupyter *Notebooks* no Google Colab com GPUs. Os hiperparâmetros fixos foram: 3 épocas, taxa de aprendizado de 0,00002 e decaimento de peso de 0,01. A validação ocorreu ao final de cada época, com a escolha automática do melhor modelo para teste. Não houve ajuste fino manual de hiperparâmetros. O *fine tuning* foi executado em GFSL, GZSL e FSL convencional, mas não em ZSL convencional (onde apenas os dados de teste são usados). A camada de classificação dos modelos foi ajustada para o número de classes de cada dataset. A avaliação foi feita com acurácia, F1-*score*, precisão e *recall*. As principais ferramentas utilizadas foram: Python, Google Colab e as bibliotecas *Transformers* (*Hugging Face*), Pytorch, Pandas e Scikit-learn.

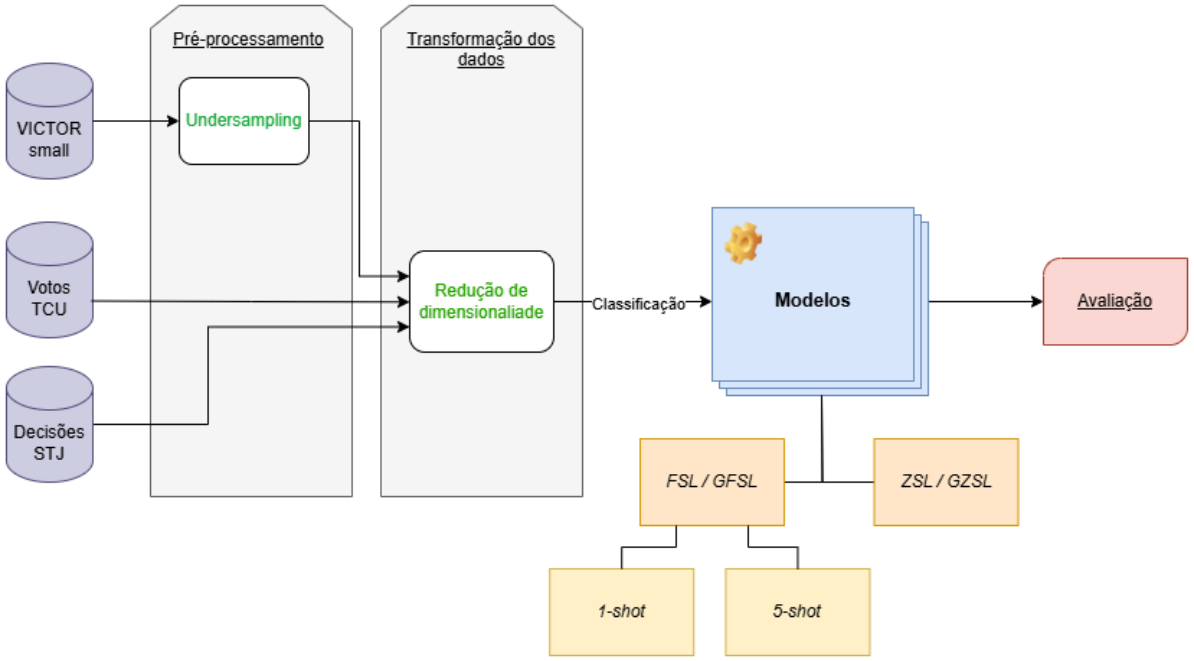


Figura 1. Etapas gerais do desenvolvimento do trabalho

4.1. Conjuntos de Dados

Neste trabalho, foi utilizado o dataset VICTOR (Luz de Araújo et al., 2020), derivado de documentos do STF, em sua versão "*small*" (SVic), contendo 6.510 Recursos Extraordinários, 94.267 documentos e 339.478 páginas, com anotações de tipo de documento (Acórdão, RE, ARE, Despacho, Sentença e Outros) e tema de processo. O *dataset* foi reduzido às *features* "*body*" (texto) e "*document_type*" (categoria). O *dataset* original possui divisão em treinamento, validação e teste (70/15/15%). Devido ao desbalanceamento da classe "Outros", foi aplicado *undersampling*, reduzindo-a para 3000 exemplos no treinamento e 2000 na validação e teste. Para as abordagens GZSL e GFSL, a classe "Despacho" foi definida como não vista/pouco vista, respectivamente, devido à sua menor frequência. No GZSL, exemplos de "Despacho" foram excluídos do treinamento e validação. No FSL, *datasets* de treinamento com 1 e 5 exemplos de "Despacho" foram criados. Uma quantidade considerável de documentos excedeu o limite de 512 *tokens* dos modelos, tendo sido realizado, na implementação dos modelos, o truncamento automático da parte excedente.

Os outros *datasets* usados foram o Decisões do STJ (7403 instâncias) e Votos do TCU (371 instâncias), obtidos dos sites dos respectivos órgãos e disponibilizados por Da Silva et al. (2024). O *dataset* do TCU contém os atributos “Área”, “tema”, “subtema”, “enunciado”, “processo”, “ano”, “tipo_processo”, “relator” e “voto”. Enquanto o do STJ possui ”matéria”, “natureza”, “tema”, “processo”, “relator”, “órgão”, “data_julgamento”, “data_publicação” e “ementa”. Para a classificação, foram usadas as classes "matéria" (STJ) e "Área" (TCU), reduzindo os *datasets* para "matéria" e "ementa" (STJ) e "área" e "voto" (TCU). As matérias do STJ incluem “Direito Administrativo”, “Direito Civil”, “Direito da Criança e do Adolescente”, “Direito do Consumidor”, “Direito Empresarial”, “Direito Ambiental”, “Direito do Consumidor” e “Direito Penal”. E as áreas do TCU são “Responsabilidade”, “Licitação”, “Direito Processual” e “Pessoal”.

Os dados foram divididos em treinamento, validação e teste (70/15/15%). Para GZSL/GFSL no *dataset* do STJ, "Direito Ambiental" foi escolhida como a classe não vista/pouco vista, seguindo o critério de menor frequência. Conjuntos de dados de

treinamento com 1 e 5 exemplos das classes pouco vistas foram criados a partir dos *datasets* descritos para avaliação do GFSL, além de conjuntos de dados sem as classes não vistas para GZSL. Ambos os *datasets* apresentam documentos com número de *tokens* excedendo o limite dos modelos (512 *tokens*), que também foram truncados automaticamente na execução dos modelos.

5. Resultados

O JurisBERT, modelo treinado especificamente em textos jurídicos, demonstrou consistentemente um desempenho superior no *dataset* Decisões STJ, especialmente nas abordagens GZSL e GFSL, alcançando acurácias próximas de 90%. O GZSL superou a classificação tradicional (com todos os dados de treinamento) no mesmo dataset, sugerindo que a abordagem GZSL mitigou um possível viés dos modelos para as classes majoritárias presente no treinamento convencional. No *dataset* VICTOR modificado, o BERTimbau, modelo treinado em textos genéricos em português, apresentou desempenho competitivo em relação ao JurisBERT, mostrando que modelos pré-treinados em linguagem geral podem ter boa performance em domínios específicos, dependendo das características do *dataset*.

A comparação entre as configurações de *few-sho*t (1-*shot* e 5-*shot*) mostrou uma ligeira vantagem para o 5-*shot*, principalmente no dataset VICTOR, o que é esperado, já que o modelo tem acesso a mais exemplos da classe pouco vista. As abordagens ZSL e FSL "puras", sem nenhum ou com poucos exemplos para treinamento, respectivamente, apresentaram desempenho consideravelmente inferior às abordagens GFSL e GZSL. No entanto, a comparação entre 1-*shot* e 5-*shot* em FSL puro demonstrou que poucos exemplos adicionais podem gerar um aumento notável no desempenho.

Um fator que influenciou os resultados foi o limite de 512 *tokens* para processamento das entradas nos modelos. A presença de palavras-chave descritivas do conteúdo no início dos documentos do *dataset* Decisões STJ provavelmente beneficiou os modelos, especialmente o JurisBERT. O truncamento dos documentos que excederam esse limite pode ter afetado negativamente o desempenho em outros casos, especialmente no *dataset* VICTOR, que não possuem os mesmos tipos de informações que o *dataset* Decisões STJ tem no início dos seus documentos.

De forma geral, o JurisBERT mostrou ser o modelo mais adequado para a classificação de documentos jurídicos em português, principalmente quando combinado com abordagens GFSL ou GZSL. O BERTimbau demonstrou ser uma alternativa viável para alguns casos. As características dos datasets, como a presença de palavras-chave e o tamanho dos documentos em relação ao limite de *tokens*, também tiveram um papel relevante nos resultados.

6. Considerações finais

Este trabalho explorou o uso de Processamento de Linguagem Natural, especificamente *Few-Shot Learning* e *Zero-Shot Learning*, para classificação de elementos jurídicos. O objetivo era analisar a viabilidade dessas abordagens em cenários com poucos ou nenhum exemplo de treinamento. A pesquisa utilizou *Transfer Learning* com modelos baseados em *transformers*.

Trabalhos futuros sugeridos incluem: Explorar *Prompt Learning* (ICL e CoT), *Meta-learning* (MAML e *Prototypical Networks*) e *Ensemble Learning*; melhorar o suporte

semântico com *Retrieval-Augmented Generation* (RAG) ou adição de informações sobre classes não vistas/pouco vistas; e avaliar o desempenho com mais de uma classe não vista/pouco vista. Também sugere-se investigar a classe "outros" do *dataset* VICTOR com aprendizagem não supervisionada.

Os resultados quantitativos demonstraram o potencial e as limitações dos modelos. GFSL e GZSL apresentaram desempenho semelhante a contextos com muitos exemplos de treinamento, mostrando capacidade de generalização. As abordagens ZSL e FSL convencionais tiveram baixo desempenho, dada a diferença de vocabulário/estrutura dos dados de pré-treinamento. A limitação do tamanho da entrada dos modelos foi identificada como uma área para melhoria, sendo sugerida uma futura implementação de técnicas para adequar os textos ao limite de *tokens*, como divisão com sobreposição ou sumarização com LLMs.

Referências

NADKARNI, Prakash M.; OHNO-MACHADO, Lucila; CHAPMAN, Wendy W. Natural language processing: an introduction. Journal of the American Medical Informatics Association, v. 18, n. 5, p. 544-551, 2011.

BROWN, Tom et al. Language models are few-shot learners. Advances in neural information processing systems, v. 33, p. 1877-1901, 2020.

KAVZOGLU, T., Increasing the accuracy of neural network classification using refined training data, Environmental Modelling & Software, Volume 24, Issue 7, Pages 850-858, 2009, ISSN 1364-8152, https://doi.org/10.1016/j.envsoft.2008.11.012

LAKE, Brenden et al. One shot learning of simple visual concepts. In: Proceedings of the annual meeting of the cognitive science society. 2011.

WANG, Zhiqiang; PANG, Yiran; LIN, Yanbin. Large Language Models Are Zero-Shot Text Classifiers. arXiv preprint arXiv:2312.01044, 2023.

PALANIAPPAN, M.; VEDHAMANI, A.; SUNDHARAKUMAR K., Zero-Shot Learning For Text Classification: Extending Classifiability Beyond Conventional Techniques. In: 2023 IEEE Region 10 Symposium (TENSYMP). IEEE, 2023. p. 1-6.

WANG, Zhengbo et al. Exploiting Semantic Attributes for Transductive Zero-Shot Learning. arXiv preprint arXiv:2303.09849, 2023.

POURPANAH, Farhad et al. A review of generalized zero-shot learning methods. IEEE transactions on pattern analysis and machine intelligence, v. 45, n. 4, p. 4051-4070, 2022.

SARKAR, Rajdeep; OJHA, Atul Kr.; MEGARO, Jay; MARIANO, John; HERARD, Vall; MCCRAE, John P. Few-shot and Zero-shot Approaches to Legal Text Classification: A Case Study in the Financial Sector. In: Proceedings of the natural legal language processing workshop, Punta Cana, Dominican Republic. Punta Cana: Association for Computational Linguistics, 2021. p. 102-106.

SAVELKA, Jaromir. Unlocking practical applications in legal domain: Evaluation of gpt for zero-shot semantic annotation of legal texts. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law. 2023. p. 447-451.

HECK, Amabyle Rabeche. Processamento de Linguagem Natural Aplicado a Reconhecimento de Entidades Nomeadas em Textos Legais em Português Brasileiro. 2022.

71 f. TCC (Graduação) - Curso de Engenharia de Controle e Automação, Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2022. Disponível em: https://repositorio.ufsc.br/handle/123456789/233250. Acesso em: 16 out. 2023.

LUZ DE ARAUJO, Pedro Henrique. lener-br. [S.l.]: GitHub, 2020. Disponível em: https://github.com/peluz/lener-br.

G. S. CHAVES, ALESON ; H. G. BARBOSA, BRUNO ; D. FERREIRA, DANTON. Extração de Entidades de Produtos Utilizando Técnicas de Few-Shot Learning. In: XV Simpósio Brasileiro de Automação Inteligente, 2021, Online, 2022. v. 01

RIZINSK, Maryan; JANKOV, Andrej; SANKARADAS, Vignesh; PINSK, EugeneI; MISKOVSKI, Igor and TRAJANOV, Dimitar. Company classification using zero-shot learning. arXiv preprint arXiv:2305.01028, 2023. Disponível em: https://arxiv.org/abs/2305.01028v2

VIEGAS, Charles Felipe Oliveira. JurisBERT: Transformer-based model for embedding legal texts. 2022.

DA SILVA JUNIOR, Daniel et al. Datasets for Portuguese Legal Semantic Textual Similarity. Journal of Information and Data Management, v. 15, n. 1, p. 206-215, 2024.