Robô do processo: utilizando técnicas de reconhecimento de entidades nomeadas para treinamento de Inteligência Artificial para a área juridica

1st Giovanna Ily Farias Ramalho
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
gifr@cin.ufpe.br

I. Introdução

Nos escritórios de advocacia, a utilização de Inteligência Artificial (IA) e automações tem se tornado cada vez mais comum. Sendo frequentemente referida como "Robô do Processo", esses sistemas são treinados para identificar clientes que estão sendo intimados em um novo processo judicial ou que tiveram movimentações em processos já existentes. Essa tecnologia, apesar de aparentemente simples, revela-se extremamente útil e com grande potencial de aplicação.

No contexto jurídico, o volume de processos é avassalador. Em 2023, mais de 36 milhões de novas ações foram ajuizadas no Brasil, somando-se aos 83 milhões de processos já em andamento nas várias instâncias judiciais. Isso resultou em quase 100 milhões de processos pendentes de julgamento apenas naquele ano, de acordo com dados do Conselho Nacional de Justiça (JUSTIÇA EM NÚMEROS, 2024). Esses números evidenciam um gargalo significativo, destacando a urgência de buscar soluções inovadoras para a consecução de uma Justiça mais célere e econômica, entre elas a Inteligência Artificial.

A IA possibilita a automação de tarefas repetitivas e demoradas, como a revisão de documentos, a pesquisa de jurisprudência e a análise de contratos. Ferramentas de aprendizado de máquina, como o Contract Intelligence (COIN) de J.P. Morgan e o ROSS Intelligence, exemplificam como a IA pode reduzir o tempo de trabalho de milhares de horas para meros segundos. Esse aumento na eficiência permite que advogados se concentrem em atividades mais estratégicas e complexas, aumentando a produtividade geral do escritório. Além disso, a automação não só economiza tempo, mas também reduz custos operacionais. A capacidade da IA de processar grandes volumes de dados com precisão minimiza a necessidade de mão de obra intensiva, resultando em uma operação mais econômica. Por exemplo, a utilização do COIN pela JP Morgan Chase & Co. substitui o trabalho de 360 mil horas de advogados anualmente, gerando economia significativa e reduzindo erros humanos.

No trabalho de Schuch(2021) [1], é proposta uma arquitetura de aprendizagem profunda voltada para a identificação e classificação de possíveis interessados em procedimentos extrajudiciais. Schuch utiliza técnicas avançadas de Processamento de Linguagem Natural (PLN), especificamente o Word Embedding (WE), que cria vetores de representações de palavras, incorporando significados semânticos e sintáticos extraídos de um vasto conjunto de dados. O código desenvolvido por Schuch emprega bibliotecas como SpaCy e NLTK para reconhecimento de entidades nomeadas (NER) em português e para manipulação de textos. Além disso, o trabalho abrange funcionalidades como limpeza e manipulação de textos, tratamento de *stopwords*, carregamento e gerenciamento de dados via Google Drive, e a implementação de um modelo de NER para identificação de entidades em textos.

Apesar das vantagens, o código de Schuch pode ser aprimorado com o treinamento de uma biblioteca mais robusta, que elimine a necessidade de web scraping e, consequentemente, evite potenciais erros de URL. Proponho, portanto, o uso de uma base de dados pública em conjunto com a biblioteca BERTimbau. Essa abordagem não apenas atualizaria o sistema, como também possibilitaria sua expansão, garantindo uma maior abrangência e precisão na identificação e classificação de entidades em textos jurídicos.

A implementação dessas tecnologias de IA nos escritórios de advocacia está transformando a prática jurídica, oferecendo soluções inovadoras para problemas antigos e permitindo que os profissionais do direito trabalhem de maneira mais eficiente e precisa.

II. Replicação do código

Utilizando Processamento de Linguagem Natural, o código aplica uma técnica de reconhecimento de entidades nomeadas (NER), identificando objetos e atribuindo classificações. Schuch exemplifica:

> Por exemplo, a frase "Thiago fala inglês e estuda na Ulbra" tem três objetos que poderiam ser identificados: Thiago como pessoa, inglês como linguagem e Ulbra como organização. (SCHUCH, 2021, p. 14)

Para facilitar a identificação e classificação de nomes de pessoas e organizações dentro dos textos trabalhados, foi utilizada a biblioteca *SpaCy* no Python, que possui modelos prontos para diversos idiomas, inclusive o português. Assim, o trabalho de identificar quem é interessado ou investigado dentro do texto fica apenas com a rede neural que foi desenvolvida.

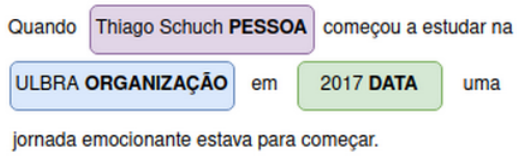

Figura 1. Exemplo de visualizador *SpaCy*.

Schuch explica:

> Na Figura 1 mostra um exemplo de como é trabalhar com os dados do *SpaCy*, pode-se notar três itens destacados: "Thiago", "ULBRA" e "2016". Esses itens são chamados de entidades por fazerem parte de um grupo de classificação, no caso "Pessoa", "Organização" e "Data", respectivamente. (SCHUCH, 2021, p. 15)

Para ajudar no entendimento dos textos trabalhados, que se tratam de *strings* em português brasileiro, o código começa com a instalação do modelo de língua portuguesa (**'pt_core_news_lg'**) para o *SpaCy*. Também utiliza a função **nltk.download('stopwords')** para obter as *stopwords* em português, ajudando na limpeza do texto.

São realizados procedimentos importantes para a preparação de dicionários de palavras antes e depois das entidades, além de listas para indexadores e entidades detectadas. Também são utilizadas funções para adicionar palavras e entidades às listas e dicionários apropriados, assim como para limpar *strings* de caracteres não desejados.

Os dados são carregados em JSON com informações de palavras e entidades já processadas, para treinamento ou ajuste do modelo. É aplicado o modelo de NER para identificação das entidades em textos específicos e armazenamento dos resultados.

A coleta de dados foi feita em um artigo .CSV contendo três colunas: "Interessado", "Investigado" e "Texto". Em seguida, foram feitos o tratamento e a limpeza dos dados.

O texto captado foi convertido em dados binários utilizando o método *One-Hot Encoding*, onde para cada palavra do *dataset* existe um número, e se ele existe no trecho analisado, este número será um, caso contrário, será zero.

A rede neural foi implementada utilizando Keras e TensorFlow dentro do ambiente do Google Colaboratory, e os dados convertidos. A rede neural do tipo Long Short Term Memory foi escolhida para o projeto por possuir "memória". Esta rede precisa saber o que já foi passado ou não por ela e também tem a capacidade de esquecer certos elementos para assim tentar prever qual será o próximo elemento.

Foram realizados experimentos com as camadas e parâmetros da rede, manipulando as gerações de cada processamento, adicionando e removendo diferentes tipos de camadas, analisando os gráficos de perda e acurácia gerados, bem como o conjunto de validação. Os dados de validação foram extraídos dos dados totais, numa quantia de 20%, deixando os outros 80% para treino.

Por fim, a saída da rede exibe ao usuário um campo de entrada em que ele pode digitar ou colar um texto e a rede neural tentará fazer a predição e retornará o que ela conseguiu extrair.

O código carece de uma melhor organização em termos de modularização, dividindo claramente as partes de processamento de dados, configuração de modelo e execução. Também faltam comentários para melhor entendimento.

Na replicação do código conseguimos resultados semelhantes aos do autor ao longo de 50 épocas.

A acurácia tanto no conjunto de treinamento quanto no de validação melhorou consistentemente ao longo das épocas. Começando com uma acurácia de treinamento de 82.02% e de validação de 90.04% na primeira época, o modelo alcançou uma acurácia de 97.24% no treinamento e 95.22% na validação nas últimas épocas. Isso indica um bom aprendizado por parte do modelo.

O modelo também apresenta diminuição da perda, começando com uma perda de treinamento de 0.9956 e de validação de 0.6615 na primeira época, alcançando reduções para 0.0479 e 0.2629, respectivamente. Isso demonstra que o modelo está ficando mais eficiente em suas predições com o avanço do treinamento.

Contudo, a perda de validação começou a mostrar sinais de variabilidade e aumento leve nas últimas 20 épocas. Isso pode indicar o início de sobreajuste (*overfitting*), onde o modelo se ajusta demais aos dados de treinamento e perde a capacidade de generalizar para dados novos. Há também uma discrepância entre a acurácia de treinamento e a de validação. Embora ambas sejam altas, a de treinamento é consistentemente maior. Isso também pode ser um sinal de sobreajuste, especialmente se a diferença entre elas começar a aumentar.

III. Melhorias que podem ser aplicadas no código

1. Organização e Modularização O código carece de uma organização clara e modularização. Isso torna difícil a manutenção e a compreensão do fluxo de dados e das etapas de processamento. Dividir o código em funções ou módulos distintos para processamento de dados, configuração do modelo e execução pode melhorar significativamente a legibilidade e a manutenção.

2. Falta de Comentários A ausência de comentários explicativos torna o código menos acessível para outros desenvolvedores ou para futuras revisões. Comentários detalhados são essenciais para descrever a funcionalidade de blocos de código, especialmente em seções complexas como o pré-processamento de dados e a configuração da rede neural.

3. Dependência de Web Scraping O código depende de web scraping para coletar dados, o que pode levar a erros de URL

e problemas de manutenção a longo prazo. Utilizar fontes de dados públicas e confiáveis, como datasets disponíveis em plataformas como o Kaggle, pode evitar esses problemas e garantir a consistência dos dados.

4. Possível Sobreajuste (Overfitting) Os resultados indicam sinais de sobreajuste, como a discrepância crescente entre a acurácia de treinamento e a de validação nas últimas épocas. Isso sugere que o modelo está se ajustando excessivamente aos dados de treinamento, perdendo a capacidade de generalizar para novos dados. Técnicas como regularização, uso de dropout, ou early stopping podem ser implementadas para mitigar o sobreajuste.

5. Preparação de Dados A preparação dos dados inclui etapas como a limpeza de texto, tokenização e codificação one-hot. No entanto, o processo de preparação de dicionários de palavras antes e depois das entidades pode ser otimizado utilizando métodos mais robustos para representação de texto, como embeddings pré-treinados (ex: BERTimbau), que capturam melhor os contextos semânticos das palavras.

6. Implementação de NER A implementação do modelo de NER utiliza a biblioteca SpaCy, que é adequada para tarefas de PLN. No entanto, a utilização de modelos mais avançados, como BERTimbau, pode melhorar a precisão e a robustez do reconhecimento de entidades, especialmente em contextos jurídicos complexos.

7. Configuração do Ambiente A execução do código no Google Colaboratory é adequada para desenvolvimento e teste, mas pode ser benéfico documentar e fornecer instruções para a replicação do ambiente, incluindo a instalação de bibliotecas e configuração do ambiente de desenvolvimento, para facilitar a reprodução e a colaboração.

IV. Melhoramento com BERTimbau

1. Dataset Para treinar seu projeto, Schuch utiliza uma planilha fornecida por um especialista do Sistema do Tribunal de Justiça de Tocantins (TJTO). A planilha contém três colunas: Interessado, Investigado e Texto. Sendo a coluna texto composta por URLs que dão acesso às páginas de processos judiciais, utilizados para a realização de web scraping. Desta maneira o código desenvolvido por Schuch acessa essas URLs, busca palavras-chave pré-definidas pela biblioteca SpaCy e armazena essas palavras, juntamente com as cinco palavras anteriores e as cinco posteriores, em bibliotecas e arquivos JSON para realizar o treinamento do modelo.

O primeiro problema que propus solucionar, então, é o problema com o dataset que além de ser composto por dados sensíveis, o que limita a expansão do treinamento, também depende de webscrapin. Portanto, optei por utilizar dados públicos de um dataset disponível no Kaggle [2], que contém três colunas: id, tokens e nertags, conforme ilustrado na Figura 2. Além disso, o treinamento foca na identificação de entidades nomeadas sem a necessidade de armazenar as cinco palavras anteriores e posteriores, tornando o processo mais eficiente e seguro, na função de identificar as Entidades Nomeadas (NER).

2. Preparação de dados

id	tokens	ner_tags
0	['Número' 'do' 'Acórd… 'AUGUSTO' 'NARDE… 'TOMADA' 'DE' 'CON… '11/05/2016' 'Númer… 'recorrida' 'Ministra' …	[0 0 0 11 12 0 1 … 0 5 0 0 0 0 0 0
1	['Interessado' '/' 'Resp…	[0 0 0 0 0 0 0]
2	['Interessados/Respor…	[0 0 0 0]
3	['Interessado' ':' 'Supe… '00.497.560/0001-0…	[0 0 1 2 2 0 0 0 0 0]
4	['Responsáveis' ':' 'Al… ')' ':' 'Antonio' 'Carlos… 'Carlos' 'Aureliano' 'M… 'Carlos' 'de' 'Almeida… 'Alves' 'Mey' '(' '025.… 'Empreendimentos' … 'Oliveira' 'Alves' '(' '1… 'Biangolino' '(' '290.…	[0 0 3 4 4 4 0 0 0 0 3 … 4 0 0 0 0 1 2 2 2 2 2

Figura 2. legal data set from kaggle

Schuch inicia a preparação de dados com o download das stopwords em português e a carga do modelo de linguagem pt_core_news_lg do SpaCy. Em seguida, define-se conjuntos e dicionários para armazenar palavras antes e depois das entidades, indexadores e entidades. Funções como addToListBefore e addToListAfter são usadas para adicionar palavras aos dicionários apropriados, enquanto addEntity adiciona entidades a uma lista e addIndexer cria indexadores para as palavras antes e depois das entidades. A função removeString limpa strings, removendo caracteres indesejados.

O código lê dados de arquivos JSON e um CSV, processando texto para encontrar entidades nomeadas (pessoas e organizações). Adiciona palavras antes e depois das entidades identificadas nos respectivos dicionários e cria indexadores para as entidades e palavras relacionadas. Os dados processados são então salvos em arquivos JSON.

Os principais problemas desse treinamento incluem a falta de modularização clara, dificultando a manutenção e a leitura do código. A dependência de web scraping para coletar dados pode resultar em erros de URL e problemas de manutenção a longo prazo. Além disso, há sinais de sobreajuste, onde o modelo se ajusta excessivamente aos dados de treinamento, perdendo a capacidade de generalizar para novos dados. A preparação de dados, incluindo a limpeza de texto e tokenização, pode ser otimizada usando métodos mais robustos para representação de texto, como embeddings pré-treinados.

Na minha preparação de dados começo montando o Google Drive para acessar o arquivo CSV contendo os dados. Em seguida, os dados são carregados em um DataFrame do Pandas, com tratamento para garantir que os valores estejam no formato correto e sem valores nulos. As colunas de tokens e rótulos de entidades são divididas em listas, e os dados são armazenados em um novo DataFrame.

Os rótulos únicos são identificados e uma lista de rótulos é atualizada. O comprimento das frases é verificado e truncado para um máximo de 128 tokens. O tokenizer do BERTimbau é carregado e uma função é definida para tokenizar e alinhar os rótulos, garantindo que os rótulos fiquem corretamente alinhados com os tokens após a tokenização. O DataFrame é

convertido em um Dataset do Hugging Face, e a tokenização e alinhamento dos rótulos são aplicados em lote.

Os dados tokenizados são convertidos para tensores PyTorch e armazenados em um TensorDataset. Argumentos de treinamento são definidos, incluindo a taxa de aprendizado, tamanho do batch, número de épocas e decaimento de peso. O modelo pré-treinado BERTimbau para classificação de tokens é carregado com o número de rótulos adequado.

Aotilizar o BERTimbau, um modelo de linguagem pré-treinado específico para o português, proporciona alta precisão na tokenização e no alinhamento de rótulos, essencial para o reconhecimento de entidades nomeadas (NER) em textos jurídicos. O uso de embeddings pré-treinados permite capturar melhor os contextos semânticos das palavras, aumentando a eficácia do modelo. A técnica de tokenização com truncamento e preenchimento garante consistência no tamanho das sequências, facilitando o processamento eficiente e preciso dos dados durante o treinamento do modelo. Essa abordagem resulta em um modelo robusto capaz de generalizar bem em diferentes conjuntos de dados.

V. Análise dos resultados

Alguns pontos positivos puderam ser observados nesse treinamento:

Redução da Perda: Observa-se uma redução contínua e significativa tanto na perda de treinamento quanto na perda de validação ao longo das épocas. A perda de treinamento reduziu de 0.145900 na primeira época para 0.040800 na terceira, enquanto a perda de validação reduziu de 0.070816 para 0.028358 no mesmo período. Esta tendência positiva indica que o modelo está aprendendo eficazmente a partir dos dados fornecidos, melhorando sua capacidade de generalização.

Eficiência do Modelo: O uso do BERTimbau, que é um modelo robusto e pré-treinado para o português, possibilitou uma tokenização precisa e um alinhamento eficiente dos rótulos, resultando em um treinamento mais eficaz. A função tokenize_and_align_labels desempenhou um papel crucial, garantindo que os rótulos fossem corretamente alinhados com os tokens, mesmo após a aplicação de truncamento e preenchimento, preservando a integridade das informações.

Desempenho do Treinamento: Com um tempo total de treinamento de aproximadamente 3 horas e 18 minutos (11914.9039 segundos), o modelo demonstrou uma eficiência considerável, processando cerca de 1.971 amostras por segundo e 0.493 etapas por segundo. O volume total de operações de ponto flutuante (FLOPs) realizadas foi de aproximadamente $1.534 \times 10\hat{1}5$, indicando a complexidade e a profundidade das operações envolvidas.

Precisão na Tokenização: O uso do tokenizer do BERTimbau permitiu uma tokenização precisa, essencial para a correta interpretação e processamento das sequências de texto em português. Alinhamento Eficiente de Rótulos: A função personalizada para alinhar os rótulos garantiu que os dados mantivessem sua integridade, melhorando a qualidade do treinamento e a precisão do modelo. Redução Significativa da Perda: A diminuição contínua das perdas de treinamento e validação ao longo das épocas evidencia a eficácia do processo de treinamento e a capacidade do modelo de generalizar bem a partir dos dados de entrada.

Melhoria da Generalização: A baixa perda de validação indica que o modelo está generalizando bem e não apenas memorizando os dados de treinamento, o que é crucial para o desempenho em dados não vistos.

Em resumo, a implementação e otimização do processo de tokenização e treinamento utilizando o BERTimbau resultaram em melhorias substanciais na eficiência e precisão do modelo, demonstrando o potencial dessa abordagem para aplicações em reconhecimento de entidades nomeadas em textos jurídicos em português.

Referências

[1] Schuch, T. F. (2021). Rede neural para identificação de interessados e investigados em procedimentos extrajudiciais (Trabalho de Conclusão de Curso, Bacharelado em Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas, Tocantins). Recuperado em 12 de maio de 2024, de https://ulbra-to.br/bibliotecadigital/publico/home/documento/2497

[2] TheDevastator. (n.d.). LeNER-Br: Portuguese Legal NER Dataset. Kaggle. Retrieved July 9, 2024, from https://www.kaggle.com/datasets/thedevastator/lener-br-portuguese-legal-ner-dataset

[3] JUSTIÇA EM NÚMEROS 2023: (ano-base 2023). Brasília: Conselho Nacional de Justiça, 2017. Anual. disponível em: https://justica-em-numeros.cnj.jus.br/painel-estatisticas/. Acessado em 06/07/2024

[4] Felipe, B. F. da C., & Perrota, R. P. C. (2018). Inteligência Artificial no Direito – Uma Realidade a Ser Desbravada. Revista de Direito, Governança e Novas Tecnologias, 4(1), 01-16.