Uma proposta de IA para classificador de documentos

Kaio Mitsuharu Lino Aida[1], Bruno Nogueira Magalhães[2] (orientador)

Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)

Campo Grande – MS – Brasil

Resumo. A classificação de documentos textuais é uma ferramenta crucial para grandes instituições, visando a organização eficiente da informação, aprimoramento operacional, segurança e conformidade regulatória. A aplicação de Machine Learning (ML) automatiza esse processo, aumentando a precisão e reduzindo tempo e erros humanos. A Lei Geral de Proteção de Dados (LGPD) impõe a necessidade de classificação para proteger dados pessoais, assegurar os direitos dos titulares e cumprir regras de retenção e descarte seguro. Estas práticas são essenciais para evitar vazamentos de dados, acessos não autorizados e garantir que as informações sejam mantidas pelo tempo necessário, promovendo a minimização e a segurança dos dados sensíveis.

Palavras-chave: Automação, Classificação, Documentos, Machine Learning.

Lista de abreviaturas e siglas

ML	Machine Learning
LGPD	Lei Geral de Proteção de Dados
OCR	Optical Character Recognition
NLP	Natural Language Processing
IA	Inteligência Artificial
NER	Reconhecimento de Entidades Nomeadas

Sumário

Introdução

Este projeto propõe o desenvolvimento de uma Inteligência Artificial (IA) destinada à classificação de documentos textuais, tais como contratos e e-mails, utilizando imagens extraídas por Reconhecimento Óptico de Caracteres (OCR) e processadas através de técnicas de Processamento de Linguagem Natural (PLN) para uso em Redes Neurais.

Objetivo

O objetivo deste projeto é proporcionar uma solução eficiente para a classificação de grandes volumes de documentos textuais em organizações, atendendo tanto às necessidades internas de gestão quanto às exigências externas, como as impostas pela LGPD. A proposta visa classificar documentos com um nível aceitável de confiabilidade, oferecendo uma classificação inicial que facilite o manuseio e a gestão dos mesmos.

Requisitos

1. Importação de Documentos:
    - O sistema deve permitir a importação de documentos em diversos formatos, incluindo PDF, PNG e JPEG
2. Processamento de Imagem e OCR:
    - O sistema deve converter documentos de imagem (PNG, JPEG) em texto editável utilizando OCR.
    - O sistema deve pré-processar as imagens para melhorar a qualidade do OCR.
3. Classificação de Documentos:
    - O sistema deve classificar documentos em categorias predefinidas.

Referencial Teórico

Trabalhos Relacionados

- https://ieeexplore.ieee.org/document/7333933
- https://ieeexplore.ieee.org/abstract/document/6977258
- https://ieeexplore.ieee.org/document/7333933/references#references
- https://www.rationalenterprise.com/wordpress/wp-content/uploads/2021/04/document-classification-with-support-vector-machines.pdf

Natural Language Processing (NLP)

Natural Language Processing (NLP) é uma sub-área da IA focada na interação entre computadores e linguagens humanas. O objetivo do NLP é permitir que os computadores compreendem, interpretam e respondem à linguagem humana de maneira valiosa. Aqui estão alguns conceitos e técnicas importantes em NLP, aplicações de NLP:

- Análise de Sentimento: Determinação do sentimento expresso em um texto (positivo, negativo, neutro).

- Tradução Automática: Tradução de texto de um idioma para outro.
- Chatbots e Assistentes Virtuais: Sistemas que interagem com usuários em linguagem natural.
- Reconhecimento de Entidades Nomeadas (NER): Identificação de entidades como nomes de pessoas, locais, datas em textos.
- Sumarização Automática: Criação de resumos de textos longos.

Tokenização

Definição: O processo de dividir um texto em unidades menores, como palavras ou frases (tokens). Exemplo: Transformar a frase "Eu gosto de aprender." em ["Eu", "gosto", "de", "aprender", "."].

Lematização e Stemming

Definição: Reduzir palavras às suas formas base ou raiz.
Stemming: Remove sufixos, como "aprender" de "aprendendo". Lematização: Reduz palavras à forma dicionária, como "aprender" de "aprendendo".

TF-IDF (Term Frequency-Inverse Document Frequency)

Definição: Uma técnica para avaliar a importância de uma palavra em um documento em relação a um corpus de documentos. Aplicação: Melhoria na busca de informações e recuperação de textos.

Kaggle

Kaggle é uma plataforma online que oferece uma comunidade para cientistas de dados e entusiastas de machine learning. Ela permite que os usuários participem de competições de ciência de dados, acessem datasets públicos, e compartilhem seus próprios códigos e notebooks.

OCR

OCR, ou Reconhecimento Óptico de Caracteres (Optical Character Recognition), é uma tecnologia que converte diferentes tipos de documentos, como imagens digitalizadas, fotos de documentos, e arquivos PDF, em texto editável e pesquisável.

NLTK

NLTK, abreviação para Natural Language Toolkit, é uma biblioteca em Python amplamente utilizada para processamento de linguagem natural (PLN). Foi desenvolvida para suportar pesquisa e ensino em PLN, oferecendo uma gama de ferramentas e recursos que facilitam tarefas como tokenização, stemming, lematização, análise sintática, análise semântica, entre outras.

Stopwords

Stopwords são palavras comuns que geralmente não contribuem significativamente para o significado de um texto e são frequentemente removidas durante a análise de texto para melhorar a precisão e a eficiência de algoritmos de processamento de linguagem natural.

Exemplos de palavras que são consideradas stopwords (palavras vazias) para o idioma inglês: a, an, the, and, or, but, if, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very.

Aprendizado Supervisionado

O aprendizado supervisionado é uma técnica de machine learning onde um modelo é treinado utilizando um conjunto de dados rotulados, ou seja, cada exemplo de treinamento é composto por uma entrada e uma saída desejada conhecida. Durante o treinamento, o modelo aprende a mapear entradas para saídas com base nos exemplos fornecidos, ajustando seus parâmetros internos para minimizar o erro de previsão. Após o treinamento, o modelo pode generalizar seu aprendizado para prever a saída de novos dados não vistos anteriormente. Esse método é amplamente utilizado em tarefas como classificação, onde o objetivo é atribuir rótulos a novas instâncias, e regressão, que visa prever valores contínuos.

Naive Bayes Multinomial

O algoritmo Naive Bayes Multinomial é um método de classificação baseado no Teorema de Bayes, adaptado para dados discretos, especialmente eficaz em problemas de categorização de texto, como filtragem de spam e análise de sentimentos. Ele assume que as características (palavras ou termos) de cada documento são independentes entre si, uma suposição simplificadora mas poderosa. O modelo calcula a probabilidade de cada classe com base na frequência das características observadas, proporcionando uma rápida e eficiente forma de classificar novos documentos. Apesar da suposição de independência condicional ser muitas vezes violada em dados reais, o Naive Bayes Multinomial frequentemente apresenta uma performance robusta, tornando-o uma escolha popular em tarefas de processamento de linguagem natural.

Random Forest com Busca em Grade

O Random Forest é um poderoso algoritmo de aprendizado de máquina que utiliza uma combinação de várias árvores de decisão para melhorar a precisão de classificação e reduzir o overfitting. A Busca em Grade (Grid Search) é um método de otimização que ajusta os hiperparâmetros do modelo para encontrar a configuração que maximiza a performance. Em uma Random Forest, a Busca em Grade pode ajustar parâmetros como o número de árvores, a profundidade máxima de cada árvore, e o número mínimo de amostras para dividir um nó. Essa abordagem sistemática permite que o modelo seja afinado para obter a melhor performance possível, proporcionando uma solução altamente precisa e robusta para problemas de classificação complexos.

Material e métodos

Link do projeto

- Vídeo explicando o projeto: https://drive.google.com/file/d/17MJjfM1wfNiX1IPHEeT9Onpm0Grm54Bf/view?usp=drive_link[1]
- Repositório github: https://github.com/kaiomudkt/projeto_Machine_Learning

Funcionalidades mapeadas do projeto

1. Ambiente em container com Docker
    a. Dependências instaladas automaticamente
    b. Ambiente python com Jupyter
2. ETL para download
3. ETL de Imagem com OCR
4. ETL para processamento de texto para NLP
5. Dataframe pandas armazenado em arquivos seralizados para otimizar procedimentos já realizados

Principal limitação do projeto

A principal limitação deste projeto foi a dificuldade em encontrar *datasets* públicos adequados. O *dataset* ideal seria composto por uma combinação de arquivos de imagens, como PNG ou JPEG, tanto em alta quanto em baixa resolução, além de arquivos PDF já processados por OCR. No entanto, não foi possível localizar um dataset que atendesse a todos esses critérios. As opções mais próximas do ideal encontradas foram:

- https://www.kaggle.com/datasets/ritvik1909/document-classification-dataset/data
- https://www.kaggle.com/datasets/shaz13/real-world-documents-collections

Características do dataset escolhido

1. Diversidade de Documentos: O dataset abrange uma variedade de tipos documentais, incluindo artigos, relatórios, recibos, cartas, entre outros. Esta diversidade possibilita a análise de uma ampla gama de categorias documentais.
2. Formato de Arquivo: Os documentos estão disponibilizados em formatos de imagem, como PNG e JPEG, o que facilita a aplicação de técnicas de processamento de imagem e reconhecimento óptico de caracteres (OCR).
3. Qualidade da Imagem: O dataset inclui documentos em múltiplas resoluções, tanto em alta quanto em baixa, permitindo a avaliação de algoritmos em diferentes condições de qualidade de imagem.

---

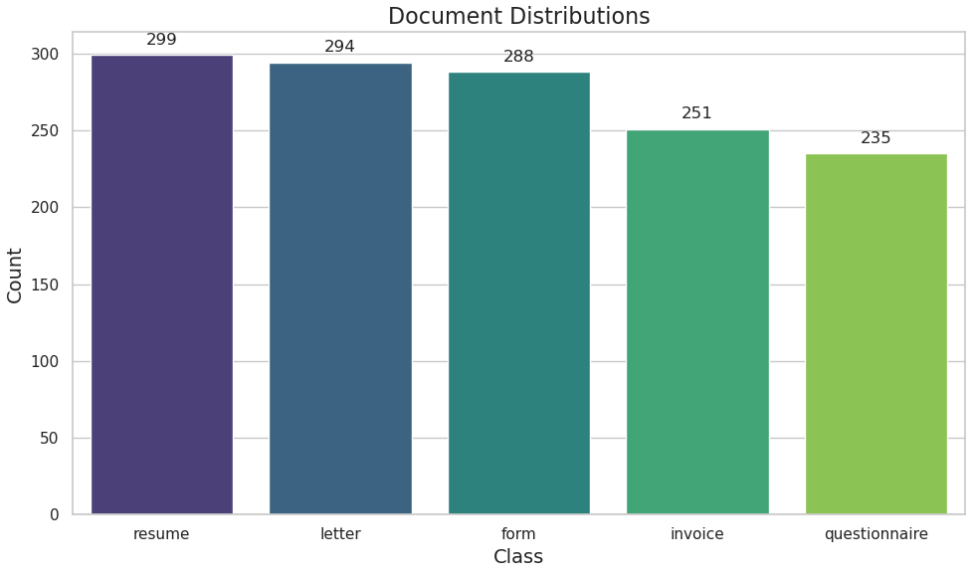
[1] https://drive.google.com/file/d/17MJjfM1wfNiX1IPHEeT9Onpm0Grm54Bf/view?usp=drive_link

4. Classificação Prévia: Os documentos do dataset são previamente classificados em diversas categorias, o que facilita a criação de modelos de aprendizado de máquina para tarefas de classificação supervisionada.
5. Tamanho e Variedade: Com um número considerável de documentos, o dataset oferece uma base robusta para o treinamento e validação de modelos de classificação.
6. Metadados: O dataset inclui metadados que fornecem informações adicionais sobre cada documento, como a categoria à qual pertence, auxiliando na análise e no processamento dos dados.

Outra grande limitação encontrada no projeto foi o baixo poder computacional para poder tudo em um tempo razoavel para poder

Análise Exploratória de Dados (EDA)

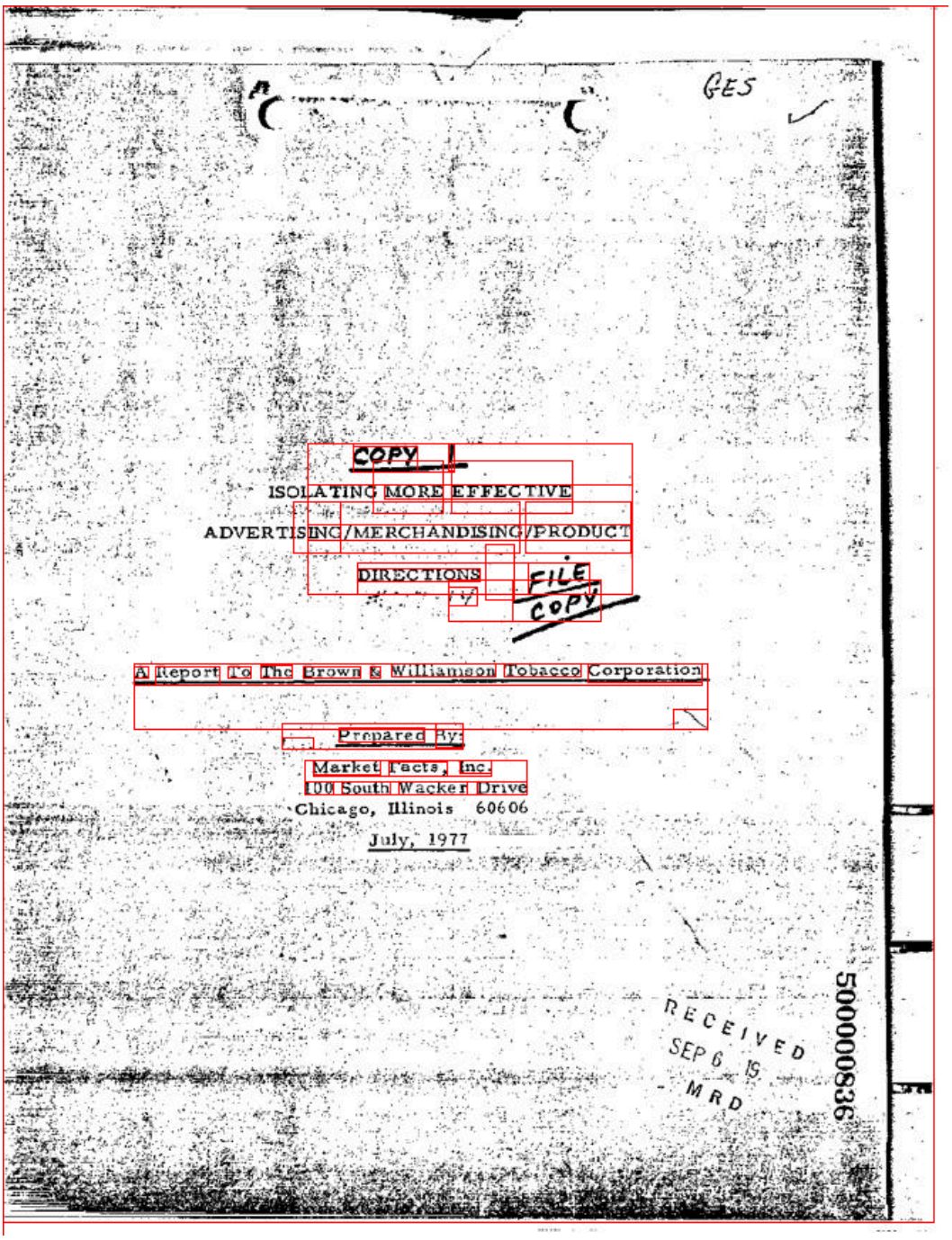
Foi verificado que a classificação dos dados resultou na seguinte distribuição: "resume" com 299 imagens, "letter" com 294 imagens, "form" com 288 imagens, "invoice" com 251 imagens e "questionnaire" com 235 imagens.



Gráfico

Coletando dados de dentro da imagem com OCR

Os dados foram coletados a partir de imagens utilizando a técnica de Reconhecimento Óptico de Caracteres (OCR). Para realizar o OCR, foi empregada a biblioteca "pytesseract", uma interface em Python para o Tesseract OCR Engine. O Tesseract é uma ferramenta de código aberto amplamente utilizada para a execução de OCR em imagens, permitindo a extração precisa de informações textuais.



Preparação dos Dados para a modelagem

A preparação dos dados é uma etapa crucial para garantir que o modelo de aprendizado de máquina possa interpretar e aprender efetivamente a partir dos dados textuais. O processo inicia-se com a conversão para minúsculas, assegurando que todas as palavras sejam tratadas de forma consistente. Em seguida, realizamos a substituição de quebras de linha e tabulações por espaços, mantendo a uniformidade no texto. A remoção de espaços extras elimina os espaços em branco desnecessários, contribuindo para a normalização da estrutura do texto.

Para evitar que os números interfiram no modelo, procede-se à remoção de números. A remoção de pontuação garante que os sinais de pontuação não sejam tratados como palavras. A tokenização é realizada utilizando a função "*word tokenize*" do NLTK, que divide o texto em tokens individuais (palavras e pontuações). Este passo é fundamental para trabalhar com o texto em nível de palavra.

Subsequentemente, ocorre a remoção adicional de pontuação e "*stopwords*", eliminando palavras comuns que geralmente não contribuem significativamente para o significado do texto.

A lematização é executada usando o "*WordNetLemmatizer*" do NLTK, reduzindo cada palavra à sua forma base ou lema. Isso ajuda na normalização do texto, reduzindo variações morfológicas e simplificando o vocabulário.

Finalmente, todos os tokens "lematizados" são concatenados em uma única string, onde cada token é separado por um espaço em branco. Esta string processada é então retornada como a saída da função, pronta para ser utilizada na modelagem.

K-Fold Cross Validation (Validação Cruzada)

A divisão dos dados em conjuntos de treinamento e teste foi realizada utilizando a função "*traintest split*" da biblioteca Scikit-learn. A divisão foi efetuada com uma proporção de 80% para treinamento e 20% para teste, garantindo a reprodutibilidade dos resultados com a definição do parâmetro "*random state*" como 678. Essa metodologia assegura a validação robusta do modelo de classificação a ser desenvolvido.

TF-IDF Vectorizer

O TF-IDF Vectorizer do scikit-learn é uma ferramenta que converte uma coleção de documentos de texto em uma matriz de características TF-IDF (Term Frequency-Inverse Document Frequency). Esta técnica é crucial para a modelagem de dados textuais, pois permite extrair características que refletem a importância de palavras ou n-gramas em relação a um documento dentro de um corpus. O TF-IDF combina duas métricas: a Frequência do Termo (TF), que conta o número de vezes que uma palavra aparece em um documento, e a Frequência Inversa do Documento (IDF), que diminui o peso de palavras comuns que aparecem em muitos documentos. Essa combinação resulta em uma pontuação que destaca termos mais relevantes e específicos para cada documento.

A criação de uma matriz TF-IDF resulta em uma matriz esparsa, que é eficiente para armazenar dados quando a maioria dos valores são zeros. Esta eficiência é alcançada ao armazenar apenas os índices e valores dos elementos não-zero, economizando memória em comparação com uma matriz densa, que armazena todos os elementos, incluindo os zeros. Essa característica é particularmente valiosa ao lidar com grandes volumes de texto, permitindo que o processamento e a análise sejam realizados de maneira mais rápida e com menor consumo de recursos.

Utilizar o TF-IDF Vectorizer em um contexto de modelagem de texto oferece várias vantagens. Ele ajuda a capturar a relevância semântica das palavras, filtrando termos comuns e destacando aqueles que são informativos, o que pode melhorar significativamente a precisão de classificadores de texto. Além disso, ao manter a matriz esparsa, ele facilita o

manuseio de grandes conjuntos de dados, tornando o processo de treinamento de modelos de aprendizado de máquina mais eficiente e eficaz.

Ajuste de Hiperparâmetros

Hiperparâmetros são parâmetros do modelo que não são aprendidos diretamente dos dados durante o treinamento. Eles são configurados antes do treinamento e influenciam como o modelo é treinado. O ajuste de hiperparâmetros é o processo de encontrar a melhor combinação de hiperparâmetros para maximizar o desempenho do modelo. Existem várias técnicas para isso, incluindo:

O ajuste de hiperparâmetros com "*GridSearchCV"* é uma técnica que busca encontrar a configuração ideal de parâmetros para cada modelo. Para cada modelo, definimos um dicionário de "*param grid*" que contém os parâmetros a serem ajustados utilizando "*GridSearchCV"*. Esses parâmetros são selecionados com base em práticas comuns e podem ser ajustados conforme necessário. Busca pela configuração ideal de parâmetros para cada modelo. Esta abordagem sistemática garante que exploremos um espaço abrangente de parâmetros, maximizando a performance dos modelos ao identificar as configurações de hiperparâmetros mais eficazes.

Construindo o Modelo

Foram utilizados dois modelos de aprendizado de máquina para a classificação de dados textuais: Naive Bayes Multinomial e Random Forest com Busca em Grade. O modelo Naive Bayes foi implementado utilizando a técnica de vetorização TF-IDF, seguido pela aplicação do classificador MultinomialNB.

Em paralelo, o modelo Random Forest foi otimizado por meio de uma busca em grade, onde diferentes combinações de hiperparâmetros foram avaliadas utilizando validação cruzada com três divisões. Os parâmetros otimizados incluíram o número de estimadores, o critério de divisão e a utilização ou não de bootstrap.

Resultados encontrado

Previsões

O modelo Naive Bayes alcançou uma acurácia de treinamento de 95,22% e uma acurácia de teste de 95.13%. A matriz de confusão revelou a distribuição das classificações corretas e incorretas, indicando a eficácia do modelo em diferenciar entre as classes de dados.

O modelo Random Forest, após otimização através de busca em grade, obteve os melhores parâmetros com 200 estimadores, critério "gini" e bootstrap habilitado. Este modelo apresentou uma acurácia de treinamento de 99,15% e uma acurácia de teste de 99,25%. A matriz de confusão para o Random Forest também demonstrou uma alta precisão na classificação, com menor incidência de classificações incorretas em comparação ao Naive Bayes.

Os resultados indicam que, embora ambos os modelos tenham performado bem, o Random Forest mostrou-se superior em termos de precisão de teste. A análise detalhada das

matrizes de confusão sugere que o Random Forest teve um desempenho mais consistente na correta classificação das instâncias, tornando-o uma escolha robusta para tarefas de classificação de dados textuais neste contexto.

Previsões com dados de Treinamento para Detectar Overfitting

No Naive Bayes a diferença entre a acurácia de treinamento e a acurácia de teste é pequena (95.13%), o que indica que o modelo está se generalizando bem e não há sinais claros de overfitting. As matrizes de confusão mostram que o modelo está fazendo poucas confusões entre as classes, o que também sugere um bom desempenho geral.

Já no Random Forest a diferença entre a acurácia de treinamento e a acurácia de teste pode indicar algum nível de overfitting. O modelo tem uma acurácia de 99% no treinamento, o que é um indicativo clássico de overfitting, pois significa que ele está memorizando os dados de treinamento. As matrizes de confusão mostram que o modelo está fazendo poucas confusões entre as classes no conjunto de teste, o que é bom, mas a discrepância na acurácia sugere que o modelo pode não estar generalizando tão bem quanto poderia.

Para o Futuro

É pretendido treinar uma nova rede neural usando este projeto como base, mas utilizando outro dataset privado. Aplicando mais técnicas para aumentar acurácia, como sabemos que o tipo de documento é determinado por muitas especificações, como o design do documento, o cabeçalho e rodapé, o corpo do documento e como a escrita é formatada dentro do documento. Todos esses fatores ajudam no processo de identificação do tipo de documento.

Reconhecimento de Entidades Nomeadas (NER): Identificação de entidades como nomes de pessoas, locais, datas em textos. Por exemplo: após classificar o tipo do documento, é relevante extrair sua data de emissão, pois a LGPD exige que cada tipo de documento seja descartado após determinado período de emissão.

Criar um sistema que contemple os seguintes requisitos, já integrado com o classificador de documentos em produção:

1. Interface de Usuário:
    a. O sistema deve ter uma interface amigável para o usuário, permitindo a visualização e a verificação dos documentos classificados.
    b. O sistema deve fornecer opções de busca e filtragem para localizar documentos específicos.
2. Feedback e Correção:
    a. O sistema deve permitir que o usuário forneça feedback sobre a precisão da classificação.
    b. O sistema deve permitir a correção manual da classificação incorreta e aprender com essas correções para melhorar a precisão futura.
3. Armazenamento e Gerenciamento de Documentos:
    a. O sistema deve armazenar documentos e suas classificações de forma segura.

    b. O sistema deve permitir a organização e o gerenciamento de documentos classificados.
4. Relatórios e Análises:
    a. O sistema deve gerar relatórios sobre a quantidade de documentos classificados por categoria.
    b. O sistema deve fornecer estatísticas de desempenho do modelo de IA, como taxas de precisão e erro.
5. Integração com Outros Sistemas:
    a. O sistema deve permitir a integração com outras ferramentas e sistemas usados pela instituição financeira, como sistemas de gerenciamento de documentos (DMS) e software de CRM.
6. Segurança e Privacidade:
    a. O sistema deve garantir que todos os documentos sejam armazenados e processados de maneira segura, com controles de acesso apropriados.
    b. O sistema deve cumprir regulamentações de privacidade e proteção de dados.
7. Escalabilidade e Performance:
    - O sistema deve ser escalável para lidar com um grande volume de documentos sem degradação de desempenho.
    - O sistema deve processar e classificar documentos de forma eficiente, minimizando o tempo de espera para o usuário.

Referências

- https://arxiv.org/pdf/1502.07058v1
- https://www.kaggle.com/code/sunilthite/document-classification-project-ocr
- https://www.sciencedirect.com/science/article/pii/S1319157821003013