Utilização de Modelos de ML para classificação de Solicitações de Serviços de TIC

Leonardo Melo Costa da Silva
Centro Universitário SENAI CIMATEC
Salvador, Brasil
leonardo.silva6@senaicimatec.edu.br

Milena de Carvalho Cordeiro
Centro Universitário SENAI CIMATEC
Salvador, Brasil
milena.cordeiro@senaicimatec.edu.br

Márcio Freire Cruz
Centro Universitário SENAI CIMATEC
Salvador, Brasil
marciofreire@gmail.com

***Resumo*—Os usuários do MPBA abrem seus chamados de serviços de TI via sistema, cuja descrição corresponde a um texto de livre edição em linguagem natural. Nesse processo, surge um tipo de problema que consiste na possibilidade de classificação indevida desses chamados por parte dos atendentes humanos, acarretando no estabelecimento de SLA em desacordo com a demanda.**

Este trabalho pretende apresentar uma prova de conceito da viabilidade de utilização dos modelos de aprendizado de máquina supervisionado para a classificação automática das descrições dos chamados entre os macrosserviços do sistema Central de Serviços de TI. Em virtude da natureza do problema mencionado, serão apresentadas técnicas de Processamento de linguagem natural, bem como os modelos de machine learning Random Forest, LSTM e BERT, entre outros. Esses modelos experimentados tiveram cerca de 80% de acurácia, que, em conjunto com outras métricas, revelaram um bom desempenho.

Assim, constituindo-se como uma possível solução de inteligência artificial para o problema de classificação automática desses textos das solicitações dos usuários.

***Palavras-chave*- Random Forest, RNN, Aprendizado de Máquina, Classificação, BERT, LSTM, Random Forest, PLN**

I. INTRODUÇÃO

A diretoria de tecnologia da informação do MPBA está estruturada, internamente, em várias coordenações especializadas que prestam serviços a toda estrutura organizacional, objetivando oferecer o suporte necessário para o funcionamento do MPBA, além de se integrar aos objetivos estratégicos do Órgão, agregando valor no cumprimento das suas obrigações institucionais perante à sociedade.

Nesse contexto, dentre outras atribuições, a CAAU (Coordenação de Atendimento e Apoio ao Usuário) é responsável por prestar o atendimento de primeiro nível em demandas de TI, previstas no catálogo de serviços de TI, para os usuários do MPBA.

Para tanto, o usuário deve abrir seus chamados, através do sistema CSTI(Central de Serviços de TI), para o devido atendimento. Ressalta-se que esses chamados podem ser abertos por outros canais, tais como telefone e e-mail, no entanto os atendentes de Nível 1 devem cadastrá-los no referido sistema.

Dentre os diversos atributos associados a cada chamado, destaca-se o tipo de serviço que é classificado manualmente pelo atendente N1.

Essa classificação associada a outros atributos, tal como o perfil do usuário, determinam o seu prazo de atendimento, conforme o estabelecido no Acordo de nível de serviço (Service Level Agreement - SLA). Existem alguns pontos críticos nessa classificação realizada manualmente, a saber:

1) Com a elevada demanda de chamados, advindos tanto da capital quanto do interior, essa atividade manual estaria mais suscetível ao erro humano, podendo acarretar no comprometimento do nível de serviço prestado, em virtude da determinação equivocada do SLA.
2) Sobrecarga de atividades dos atendentes humanos que poderiam concentrar mais sua atenção na solução do problema apresentado pelo usuário, ao invés da classificação do seu chamado.
3) Tendo em vista que os indicadores servem de base para a tomada de decisões gerenciais, a classificação errada do tipo de serviço e, consequentemente, de seus indicadores devem acarretar em avaliações e direcionamentos de ações em desacordo com a realidade por parte do gestor. Destaca-se que os indicadores de SLA mensurados, entre outros, são utilizados para avaliação contratual, já que parte do atendimento de Nível 1 é prestado por um HelpDesk terceirizado. Por exemplo, a quantidade de chamados por categoria equivocada pode fazer com que sejam direcionados, desnecessariamente, mais recursos para uma área de maior quantidade de chamados, a fim de reduzi-los.
4) A determinação de SLA´s errados pode induzir o gestor a não priorizar solicitações com maior impacto nas atividades do órgão.

Esse trabalho foi motivado por uma demanda de classificação automática de chamados de atendimento de TI no sistema CSTI e como os modelos de machine learning poderiam ser experimentados, diante desse problema, no sentido de encontrar uma melhor solução para tal fim.

Essa solução agilizaria o processo de classificação dos chamados, que atualmente é realizado manualmente, e, por conseguinte, a determinação dos SLAs e priorização mais célere das solicitações, entre outros.

Vale salientar que foi definido como escopo deste trabalho os atendimentos direcionados para a CAAU.

Nas seções subsequentes, este artigo encontra-se estruturado da seguinte forma:

Seção II: Foram elencados trabalhos relacionados e demais referenciais teóricos que embasaram este experimento;

Seção III: Detalhamento acerca da métodos aplicados, dentre eles: análise exploratória de dados, limpeza, seleção de atributos, tokenização das palavras, anonimização de dados, construção de modelos de classificação;

Seção IV: Foram discutidos os resultados dos treinamentos e dificuldades encontradas;

Seção V: Por fim, têm-se as conclusões, com a indicação dos modelos avaliados, e possíveis trabalhos futuros.

II. TRABALHOS RELACIONADOS / REFERENCIAL TEÓRICO

Alguns estudos experimentais para a classificação automática de textos de demandas de TI já foram realizados anteriormente. Em um deles foi proposta uma solução de inteligência artificial para resolver o problema de classificações equivocadas das solicitações de serviços para o SERPRO (Serviço Federal de Processamento de Dados), abertas pelo atendente, chatbot e próprio usuário ou cliente, em um sistema de acionamentos.

Segundo os autores, esses erros de classificação, geralmente, provocam "desvios" durante o direcionamento das solicitação ao seu grupo de atendimento que precisam ser corrigidas e reencaminhadas para o grupo de atendimento correto [11].

Um outro trabalho realizado nesse sentido foi um experimento que utilizou as técnicas de Aprendizado de Máquina para detecção e recuperação de tickets duplicados na base de dados de sistemas de gerenciamento de solicitações de usuários (tickets). Segundo o autor, essa duplicidade consistia na existência de tickets nos repositórios, criados por vários usuários, para reportarem o mesmo incidente, solicitação e/ou dúvida.

Mais especificamente, os autores utilizaram Naive Bayes, SVM, LSTM, BERT e XML-RoBERTa em uma dataset de 132.703 tickets, contendo registros de Incidentes/Problemas, Perguntas/Dúvidas e Solicitações de Tarefas/Serviços, registrados no período compreendido entre Out/2019 e Set/2020 [3].

Em uma outra iniciativa de estudo, foi apresentada a classificação automática de tickets não estruturados de help desk de TI. Nesse estudo, os autores classificaram 10.742 tickets em 18 classes. Contudo, eles utilizaram uma abordagem diferente, ou seja, o estudo explora o uso de classificadores em conjunto (ensemble learning), como Bagging, Boosting e Voting, que combinam as previsões de vários modelos de machine learning para melhorar a precisão da classificação [9].

Vale ressaltar que, no atual cenário mundial, é comum a geração de um grande volume de dados não estruturados diariamente. Dentre esses, existem os textos em linguagem natural que contêm padrões de conhecimento e informações relevantes, acarretando em uma necessidade, cada vez maior, de extração automatizada dessas informações para diversos fins.

Essas demandas se enquadram em uma área de conhecimento denominada processamento de linguagem natural. Conforme indicado por Martin [8], o processamento de linguagem natural (PLN) é um campo interdisciplinar que envolve ciência da computação, inteligência artificial e linguística, focado em programar computadores para processar e analisar grandes quantidades de dados de linguagem natural.

Existem diversos modelos ou algoritmos de aprendizado que podem ser utilizados nas demandas de PLN. No entanto, o presente artigo abordará somente Random Forest, LSTM e BERT.

As escolhas dos modelos LSTM e BERT devem-se ao fato dos mesmos serem adequados para lidar com a natureza sequencial e contextual das linguagens naturais. Por outro lado, decidiu-se utilizar também um outro algoritmo de machine learning, que não se enquadra-se no grupo de deep learning, como um contraponto no experimento, o Random Forest. Além disso, os principais motivos dessa escolha pelo Random Forest são sua conhecida robustez contra o overfitting e por sua capacidade de capturar relações complexas nos dados.

O conceito de Random Forest, um método de aprendizado de máquina que combina múltiplas árvores de decisão para melhorar a acurácia e a generalização de modelos preditivos, foi introduzido por Breiman [2]. Cada árvore na floresta é treinada em uma amostra aleatória do conjunto de dados original e em um subconjunto aleatório de características, o que promove diversidade e reduz o *overfitting*.

O Long Short-Term Memory (LSTM) é um tipo de rede neural recorrente capaz de aprender dependências de longo prazo em dados sequenciais, segundo Hochreiter e Schmidhuber [6]. Essa capacidade deve-se ao fato desse tipo de rede ter sido projetada para mitigar o problema do desvanecimento do gradiente das RNNs tradicionais, utilizando mecanismos de portas que controlam o fluxo de informações e gradientes através do tempo, mantendo informações relevantes por longos períodos.

Vale ressaltar que o desvanecimento do gradiente dificulta o aprendizado das relações de longo prazo em sequências pelas redes neurais, tendo em vista que os gradientes das funções de perda em relação

aos pesos se tornam muito pequenos à medida que o algoritmo de backpropagation avança para as camadas anteriores da rede, impactando nos ajustes dos pesos destas camadas.

A utilização do BERT deve-se, principalmente, a sua capacidade de realizar o pré-treinamento bidirecional de representações de linguagem.

De forma complementar, os modelos de linguagem unidirecionais limitam a escolha de arquiteturas que podem ser usadas durante o pré-treinamento. O BERT usa abordagem fine tunning que faz com que poucos parâmetros precisem partir do zero, como discutido no estudo de vários autores [4].

Acrescenta-se que o BERT, por ser um tipo de transformer, tem a capacidade de capturar dependências de longo alcance em sequências de tokens, sem depender de estruturas recorrentes, tendo como principal inovação o mecanismo de atenção que permite ao modelo gerar representações mais ricas e complexas das palavras em contexto.

Vale salientar que a qualidade e quantidade dos dados é fundamental para a performance de um modelo de machine learning. Em outras palavras, a capacidade de generalizar de modelo depende, entre outros, de treiná-lo com mais dados.

Nesse sentido, o Data Augmentation é uma técnica utilizada para aumentar a quantidade e a diversidade dos dados de treinamento sem coletar novos dados. No contexto de processamento de linguagem natural (PLN), data augmentation envolve a criação de novas amostras de texto a partir das existentes, introduzindo variações que ajudam o modelo a generalizar melhor [10].

No contexto de avaliação de modelos, pode-se utilizar o cross-validation, entre outras técnicas e métricas.

A cross-validation é uma técnica de validação de modelos de aprendizado que consiste em dividir o dataset de treinamento em várias partes (ou folds), garantindo que cada parte seja usada tanto para treinamento quanto para validação, conforme conceito abordado pelos autores [7].

Essa técnica mostra-se eficiente em detectar e prevenir o overfitting, além de maximixar o uso dos dados disponíveis; o que é especialmente útil quando o conjunto de dados é pequeno.

Sendo que o overfitting é um fenômeno pelo qual o modelo perde sua capacidade de generalizar diante de novos dados, ajustando-se excessivamente bem aos dados de treinamento.

III. METODOLOGIA

Para alcançar o objetivo de pesquisa, que consiste em experimentar os diversos algoritmos de machine learning para a classificação automática dos macrosserviços do chamados de TI de nível 1, foram seguidas as seguintes etapas do processo:

1) Extrair e realizar a análise exploratória dos conjuntos de dados dos chamados de TI, no sentido de identificar os padrões dos dados, os possíveis problemas e pré-tratamentos para experimentação nas arquiteturas de modelos de PLN.
2) Desenvolver, treinar, testar e refinar os modelos de aprendizagem. Esse refinamento consiste em ajustes de arquitetura e de seus hiperparâmetros.

Tendo em vista que refinar modelos e ajustar hiperparâmetros exige grande poder computacional, as limitações de ambiente enfrentadas restringiram a profundidade das otimizações no experimento. Inclusive, ocorreram problemas de estouro de memória e uso de GPU no próprio ambiente do Google Colab.

Esse foi o motivo pelo qual foram executadas, no máximo, 10 épocas no treinamento dos modelos. Sendo que para o modelo BERT só foi possível realizar o treinamento com uma época. Mais um outro exemplo, seria a necessidade de divisão do dataset em lotes, na fase de pré-processamento, para as rotinas de anonimização e Data Augmentation. Em outras palavras, o dataset, referente às classes REPARO E CORREÇÃO e DÚVIDAS E ORIENTAÇÕES, foi divido em lotes e, para cada lote, foi aplicada a rotina de Data Augmentation de forma iterativa. Ademais, foi estabelecido um intervalo de tempo entre o processamento desses lotes, já que estavam ocorrendo erros de timeout quando existiam muitas requisições seguidas de traduções ao serviço do Google.

Da mesma forma, o processamento das anonimizações foram realizados por lotes de dados.

Nesse contexto, optou-se pelas ferramentas Visual Studio Code e o Google Colab como ambientes de desenvolvimento e execução dos scripts escritos em Python;

A. ANÁLISE EXPLORATÓRIA DE DADOS

As descrições dos chamados foram extraídas do banco de dados do sistema CSTI, gerando-se um json correspondente. Sendo que, além dos dados chaves para o projeto (descrição do chamado, tipo de serviço e Grupo_Atendimento), foram recuperados outros campos de indicadores tais como: se houve Violacao_SLA, Impacto, tempo_sla, urgencia_chamado, etc.

Embora esses campos não tenham sido usados no treinamento, eles foram extraídos com o objetivo de realizar estatísticas desses indicadores em trabalhos futuros. Mais especificamente, a intenção seria comparar esses indicadores antes e depois da aplicação da classificação automática pelos modelos de machine learning.

No entanto, essa comparação e aplicação do modelo em predições de chamados, de forma integrada ao sistema CSTI, não está contemplado no escopo do presente projeto e problema de pesquisa e, portanto, poderão ser abordados em trabalhos futuros. De acordo com o escopo definido, foram filtradas, do dataset extraído, as solicitações direcionados

à CAAU, bem como os campos de descrição do chamado e tipo de serviço.

Como o texto com a descrição do chamado é de livre edição pelo usuário, poderiam existir dados pessoais e/ou sensíveis, tais como nome do servidor ou membro, bem como seu CPF, RG e etc. Dessa forma, foi necessário seguir as diretrizes de proteção desses dados contidas em normativos, dentre eles a Lei Geral de Proteção de Dados (LGPD) [1].

Diante dessa possibilidade, foi criada uma rotina prévia de anonimização dos dados. Essa rotina utilizou expressões regulares para anonimização de RG, Matrícula do servidor ou membro, telefone e CPF.

Sendo que, na busca no texto por padrões desses dados, foram considerados erros de digitação ou variações incomuns. Por exemplo, troca de separadores "ponto" por traços ou espaços entre os conjuntos de números de um CPF.

A identificação de nomes de pessoas nos textos é uma atividade mais complexa e, para resolver esse problema, utilizou-se uma técnica de PLN denominada NER (**Named Entity Recognition**). Essa técnica trabalha com o conceito de identificação e classificação de entidades mencionadas em um texto.

Assim, foi utilizado um modelo pré-treinado da biblioteca Spacy [5], denominado pt_core_news_lg, cuja entidade categorizada como PER equivale ao nome de uma pessoa. No entanto, esse modelo não apresentou um desempenho satisfatório, deixando de identificar vários nomes de servidores e membros do MPBA.

Em uma tentativa de contornar esse problema, foram realizados retreinamentos, no modelo base do Spacy, com mais exemplos de nomes brasileiros. No entanto, essa solução ainda não se mostrou eficiente na identificação correta desses nomes. Por fim, optou-se por uma solução composta para mitigar o referido problema de anonimização dos nomes das pessoas, cuja primeira etapa consistia em utilizar o modelo pré-treinado do Spacy e, em seguida, aplicou-se refinamentos adicionais com a busca de nomes completos ou parciais de servidores ou membros, extraídos de bases do MPBA, e sua substituição por tokens de anonimização.

Vale ressaltar que essas rotinas de anonimização tem por objetivo, tão somente, resguardar vazamentos de possíveis dados pessoais/sensíveis nos textos. No entanto, a sua presença explícita ou respectivos tokens de anonimização não impactam no aprendizado dos modelos de classificação, objetos do estudo, tendo em vista que não representam padrões significativos para determinação dos tipos de macrosserviços presentes no catálogo, segundo as regras de negócio estabelecidas.

Nessa fase, um outro problema identificado foi a quantidade de 251 tipos de serviços, o que demandaria um modelo de classificação supervisionado mais complexo com 251 targets.

Ademais, a quantidade de exemplos por serviço é pequena ou pouco significativa, o que aumentaria, consideravelmente, a complexidade desse classificador multiclasse.

Por outro lado, o campo serviço é uma composição do macrosserviço com os serviços. No sentido de simplificar e viabilizar a solução do problema, optou-se por trabalhar com a classificação dos chamados por macrosserviços, de acordo com o catálogo oficial de serviços da CSTI.

Sendo que esse catálogo prevê a existência de três macrosserviços, a saber: SERVIÇO TÉCNICO, REPARO E CORREÇÃO e DÚVIDAS E ORIENTAÇÕES.

Assim, a partir do campo de serviço, foi derivado o campo Macrosserviço. Inicialmente, foram extraídos 192698 chamados, no entanto, com a exclusão de chamados duplicados, esse número passou para 35059.

Um outro ponto, observado no dataset, seria a existência de macrosserviços não previstos no catálogo de serviços, a saber: Fora de escopo, AUTORIZAÇÃO, SEGURANÇA E VULNERABILIDADE, SEM CATEGORIA, vide tabela I

Tabela I: Quantidade por Macrosserviço

Macrosserviço	Quantidade
SERVIÇO TÉCNICO	23472
REPARO E CORREÇÃO	5415
DÚVIDAS E ORIENTAÇÕES	5409
Fora de escopo	455
AUTORIZAÇÃO	204
SEGURANÇA E VULNERABILIDADE	89
SEM CATEGORIA	19

Assim, deliberou-se por transformar todos os macrosserviços não previstos no Catálogo de Serviços para a classe OUTROS, gerando a nova tabela II.

Tabela II: Quantidade por Macrosserviço

Macrosserviço	Quantidade
SERVIÇO TÉCNICO	23472
REPARO E CORREÇÃO	5415
DÚVIDAS E ORIENTAÇÕES	5409
OUTROS	767

Observou-se, também, um desbalanceamento significativo entre as classes REPARO E CORREÇÃO e DÚVIDAS E ORIENTAÇÕES em relação a classe SERVIÇO TÉCNICO, além da classe OUTROS. Por isso, após a limpeza dos dados, foram analisadas técnicas possíveis para contornar esse problema.

A primeira opção foi a Synthetic Minority Oversampling Technique (SMOTE), que permite criar dados sintéticos após criar-se a representação vetorial dos textos. Essa abordagem foi descartada, pois não leva em consideração o contexto, especialmente se os textos forem curtos ou se os embeddings usados não capturarem bem as relações semânticas, podendo levar a criação de ruídos indesejados e assim influenciar o modelo para dados que não possuem relação com a classe.

Foram consideradas outras técnicas de data augmentation, sendo que quatro foram elegidas como candidatas: substituição por sinônimos, troca de posição, remoção de palavras ou tradução. Escolheu-se usar a tradução ou *back-translation* que consiste em traduzir para um idioma intermediário e retornar para o original. A escolha dá-se por essa técnica manter o sentido original da frase. Já que a substituição por sinônimos, troca de posição ou remoção de palavras podem incluir ruídos que não favoreçam os modelos, visto que há palavras com múltiplos significados, e que sinônimos e posições diferentes podem acarretar mudança no contexto.

Outra técnica que foi pensada para utilizar em conjunto com o *back-translation* foi o ajuste dos pesos na função de perda. Porém foi encontrado um desbalanceamento severo dos dados, além de uma baixa quantidade de palavras na maioria dos registros, devido a essa característica, os textos tendem a apresentar pequenas variações após a técnica de tradução, resultando em amostras bastante semelhantes aos originais. Dessa forma, não poderíamos afirmar que essa semelhança não tenderia ao overfitting decorrente do aprendizado de padrões repetidos, assim optamos em não usar essa técnica.

Adicionalmente, em virtude da impossibilidade da área gestora realizar a reclassificação da classe OUTROS manualmente, nesse momento, e por conta de seu percentual pouco representativo de cerca de 2,1%, decidiu-se desconsiderá-lá no pré-processamento, inclusive Data Augmentation, e treinamentos.

Antes do tratamento dos textos, foram apuradas algumas estatísticas acerca do seu tamanho, ou seja, a quantidade de palavras por descrição de chamado. Essas estatísticas indicam uma considerável variabilidade em relação ao tamanhos dos textos. De fato, evidencia-se um desvio padrão de 88,88, refletido também na distância entre entre a média de 51,97 para a mediana que é de 14.

Ademais, constatou-se a seguinte distribuição da quantidade de palavras por texto do chamado no dataset: 25% tem 9 ou menos , 50% tem 14 ou menos , enquanto 75% têm 52 ou menos.

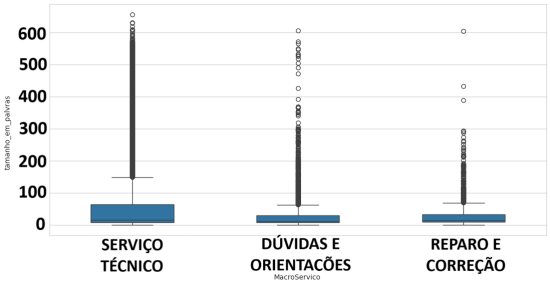

Figura 1: 3 categorias sem outros

Analisando também o boxplot para os três tipos de serviço, vide Figura 1, percebe-se que a mediana está bem próxima do limite inferior, indicando que a maioria dos textos tem um tamanho relativamente pequeno. A presença de muitos outliers indica que alguns textos são significativamente mais longos que a maioria, o que influencia na alta variabilidade dos dados, já mencionada.

Também, acrescentamse os possíveis erros de classificação dos chamados. Vale ressaltar que a revisão e, possível, correção dos dados de classificação exigem a ação humana do especialista do negócio em conjunto com a equipe técnica.

B. ENGENHARIA DE DADOS

O dataset é composto de dados não estruturados e semi-estruturados, pois o sistema permite que usuários preencham as informações dos chamados através de campos abertos de texto livre. Apesar dessa abordagem permitir uma maior flexibilidade na inserção das informações, os textos, eventualmente, podem ter sua qualidade comprometida, tais como ausência de dados importantes, inclusão de informações desnecessárias, falta de clareza, entre outros.

No entanto, quando se trata de utilizarmos linguagem natural como recurso de comunicação, esse tipo de situação é comum. Um outro ponto importante é a existência de informações vindas de outros sistemas, por exemplo o serviço de e-mail, que, ao ter seus dados copiados, geram informações de metadados.

Esses tipos de informações podem dificultar o aprendizado de padrões, realmente, relevantes para o negócio pelos modelos de NLP, já que existe a possibilidade dos algoritmos de aprendizado estabelecerem correlações equivocadas entre os dados, sem o seu devido pré-processamento.

Com o objetivo de constatar os efeitos do desbalanceamento na performance dos modelos, foram realizados treinamentos com as classes desbalanceadas, conforme os resultados apresentados nas tabelas III e IV.Tendo sido obtido uma performance mais baixa em relação ao desempenho dos modelos treinados com um dataset mais balanceado, como será visto nas próximas seções

Tabela III: Tabela de Resultados

Modelo	Acurácia	Perda	Recall
Random Forest	74,32%	-	74,32%
LSTM1(cat=3)	77,72%	0,554	64%
LSTM2(cat=3)	77,34%	0,557	62%
BERT	78,57%	0,555	78,57%

Tabela IV: Tabela de Resultados

Modelo	F1-score	Precisão
Random Forest	73,19%	72,9%
LSTM1(cat=3)	67%	71%
LSTM2(cat=3)	65%	72%
BERT	77,14%	77,71%

De fato, o F1-score mais baixos dos modelos constitui um indício dessa influência, aliado a análise das outras métricas.

Dessa forma, optou-se por aplicar Data Augmentation nas classes REPARO E CORREÇÃO e

DÚVIDAS E ORIENTAÇÕES em dois ciclos. Para tanto, utilizou-se a ideia de tradução do grupo de textos dos chamados, na qual é escolhida uma ou mais línguas intermediárias para a tradução dos textos, construindo-se as frases segundo as regras destas línguas.

Na sequência a frase é retornada para a linguagem inicial, esperando-se que ocorram mudanças na estrutura ou nas palavras utilizadas originalmente, gerando dessa forma novos exemplos, contudo permanecendo inalterado o sentido da frase.

Na prática, adotou-se no primeiro ciclo o coreano como língua intermediária, já no segundo ciclo, a língua intermediária foi o alemão. No final de ambos os ciclos, os textos traduzidos retornaram para a língua original, o português.

Essa decisão de executar dois ciclos de *Data Augmentation* deve-se ao fato das classes REPARO E CORREÇÃO e DÚVIDAS E ORIENTAÇÕES ainda apresentarem um desbalanceamento significativo ao final do primeiro ciclo. De fato, cerca de 23,6% em cada macrosserviço que sofreu *data augmentation*, enquanto a classe SERVIÇO TÉCNICO representava 51,2%.

Assim, aplicou-se um segundo ciclo de *data augmentation* sobre os novos chamados resultantes do ciclo anterior. Chegando-se ao seguinte quadro, vide gráfico 2:

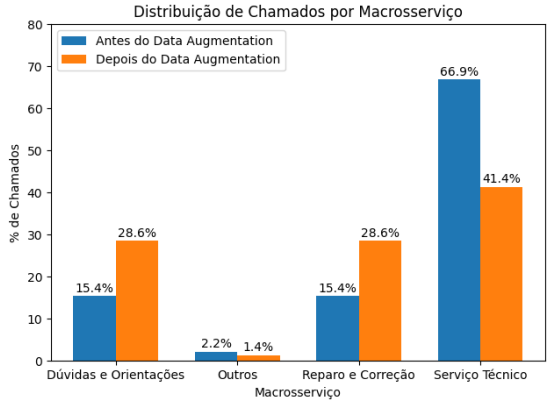

Figura 2: Distribuição de Chamados por Macrosserviço Antes/Depois do Data Augmentation

Tabela V: Quantidade por Macrosserviço - Depois do Data Augmentation

classe	Quantidade
SERVIÇO TÉCNICO	23472
REPARO E CORREÇÃO	16245
DÚVIDAS E ORIENTAÇÕES	16227
OUTROS	767

Observação: Não foram executados mais ciclos de *data augmentation*, em virtude das limitações de recursos computacionais já exposta.

Depois do *Data Augmentation*, foram aplicados alguns tratamentos de limpeza nos chamados, tais como: exclusão de *stopwords*, exclusão de informações irrelevantes para determinação dos macrosserviços.

De fato, cabeçalhos e rodapés de e-mail, cargos dos solicitantes, assinaturas, números de telefones, datas e o próprio nome do MPBA.

Além disso, foram removidas mensagens automáticas, geradas pelo próprio sistema de chamados do MPBA, e links para páginas WEB e e-mails. Por fim, os textos tratados foram submetidos ao processo de lematização.

TfidfVectorizer foi usado para converter o conjunto de chamados, pré-processados no passo anterior do pipeline,em uma representação numérica vetorizada.

Durante a *tokenização* foram realizadas técnicas de *padding* e *truncation*, limitando o tamanho dos *tokens* para que não houvesse influência da diferença de tamanho dos textos no desempenho do modelo.

C. Modelo LSTM

Diante da natureza do problema de pesquisa, que se enquadra na área de PLN, optou-se por, também, experimentar o LSTM. Em virtude do fato desse modelo ser um tipo de RNN que trata o problema da questão de degradação do gradiente (vanishing gradient) quando a palavra está muito distante.

Essa característica, dentre outras, faz com que esse modelo seja adequado à demanda de aprendizado acerca dos textos livres dos chamados, com tamanhos diversos, cuja a questão de contexto é relevante. Para codificação dos textos dos chamados pré-processados em vetores, foi utilizada a técnica de vetorização.

Para o treinamento de todas arquiteturas LSTM apresentadas, abaixo, utilizou-se 10 épocas, tendo como otimizador o Adam.

Para as camadas densas, optou-se pela função de ativação *relu,* pois ela ajuda a mitigar o problema do gradiente desaparecendo, que é comum em funções de ativação como a sigmoid e a tanh, além do fato de sua simplicidade e eficiência computacional.

Já para a camada de saída foi utilizada a função de ativação *softmax*, adequada para as tarefas de classificação multi classe.

Vale ressaltar que a escolha por LSTM bidirecional deve-se à sua capacidade de considerar o contexto completo, permitindo capturar informações contextuais tanto do passado quanto do futuro no texto. Seguem, abaixo, as arquiteturas de LSTM testadas, bem como os seus indicadores de performances:

a) Primeira Arquitetura (LSTM01): Nessa arquitetura foram utilizadas:

- Camada de entrada com dimensão 64 para os vetores de *embedding*
- Uma camada *LSTM* bidirecional com 64 unidades
- Uma camada densa de 64 neurônios
- Uma camada de saída com 3 neurônios

b) Segunda Arquitetura(LSTM02): Foram realizados ajustes na arquitetura em relação ao mo-

delo *LSTM01*, objetivando melhorar a sua performance e evitar possíveis problemas de *overfitting*.

Assim, foi incluída uma camada de *SpatialDropout1D*, tendo em vista que é uma técnica eficaz para regularizar modelos que lidam com dados de sequência, ajudando a prevenir o *overfitting*.

Para tanto, um percentual definido das unidades inteiras ao longo da dimensão de entrada(vetores de *embedding*) são desligadas(zeradas) para todos os passos de tempo. Essa abordagem é particularmente útil em modelos *LSTM*.

A arquitetura *LSTM02* foi estruturada da seguinte forma:

- Camada de entrada com dimensão 64 para os vetores de *embedding*
- Uma camada SpatialDropout1D com rate de 20%
- Uma camada LSTM bidirecional com 64 unidades e recursos de dropout com rate de 0,2
- Uma Camada densa de 64 neurônios
- Uma Camada de saída com 3 neurônios

Para ambos os modelos do LSTM, Random Forest e o BERT, foram utilizados 80% do dataset para treino e 20% foram separados para o conjunto de teste. Além disso, foi extraída uma parte do dataset de treino para realizar a validação.

D. Modelo RANDOM FOREST

Para o Random Forest, foi utilizado o GridSearchCV para o processo de tuning, com a identificação dos seus melhores hiperparâmetros, a saber:

- 400 árvores para a floresta(n_estimators=400)
- Limite máximo de profundidade de cada árvore igual a 100(max_depth=100)
- Número mínimo de amostras necessárias para dividir um nó igual a 2 (min_samples_split=2)
- max_features igual a None, possibilitando que todas as features fossem consideradas para cada divisão
- bootstrap com reposição, o que possibilita a escolha da mesma amostra mais de uma vez no treinamento de uma árvore (bootstrap=True).

IV. RESULTADOS E DISCUSSÕES

A. Modelo RANDOM FOREST

Esse modelo obteve uma boa performance, sendo próxima aos resultados dos modelos de *deep learning*. De fato, esse modelo apresentou uma acurácia de 78,86% para o dataset de teste. Ademais, evidencia-se esse relativo bom desempenho mediante análise de outras métricas no quadro de resultados VI , ou seja, da precisão, recall , f1-score e support.

Sendo que a precisão é a proporção de exemplos corretamente identificados como positivos em relação ao total de exemplos identificados como positivos (verdadeiros positivos mais falsos positivos).

O recall é a proporção de exemplos corretamente identificados como positivos em relação ao total de exemplos que realmente são positivos (verdadeiros positivos mais falsos negativos).

Já o f1-score é a média harmônica da precisão e do recall, proporcionando uma única métrica que leva em consideração ambas as medidas. A métrica support corresponde à contagem de instâncias de cada classe, fornecendo um contexto adicional sobre a distribuição das classes no conjunto de dados.

Dessa forma, para a classe 0 (SERVIÇO TÉCNICO), observou-se uma alta recuperação, com um recall de 85%, significando que a maioria das instâncias dessa classe foram corretamente identificadas.

No entanto, a precisão é um pouco menor, indicando que, dentre as previsões para a classe 0, ele está acertando 77%. Em outras palavras, a precisão é um pouco menor, indicando que há um número um pouco maior de falsos positivos dentre as previsões positivas do que em outras classes. Além dessas métricas, foi obtida F1-Score com um valor de 81%.

As Classes 1 (REPARO E CORREÇÃO) e 2 (DÚVIDAS E ORIENTAÇÕES) tiveram precisão e f1-score bem próximos. Mais precisamente, f1-score de 78% e 77% respectivamente para as classes 1 e 2, além da precisão de 80% e 81% respectivamente para as as classes 1 e 2.

Isso indica que a capacidade do modelo acertar sobre as previsões positivas das classes 1 e 2 é bem parecida, tendo a classe 2 uma pequena vantagem.

Já o recall de 76% da classe 1 indica que esta está identificando mais casos realmente positivos do que a classe 2, que, por sua vez, tem recall de 72%.

Ressalta-se que os F1-score para todas as classes revela um bom equilíbrio entre métricas de precisão e recall. Embora a Classe "DÚVIDAS E ORIENTAÇÕES"e "REPARO E CORREÇÃO tenham uma precisão melhor do que as classe SERVIÇO TÉCNICO, o recall das mesmas poderia ser melhorado se comparada com a classe SERVIÇO TÉCNICO.

Tabela VI: Relatório de Classificação

classe	precision	recall	f1-score	support
0	0.77	0.85	0.81	4785
1	0.80	0.76	0.78	3111
2	0.81	0.72	0.77	3293
accuracy			0.79	11189
macro avg	0.79	0.78	0.78	11189
weighted avg	0.79	0.79	0.79	11189

1) MATRIZ DE CONFUSÃO: Ao longo da diagonal principal da matriz, representa-se os casos onde o modelo previu corretamente a classe.

Conforme os dados dessa matriz, infere-se que o modelo teve um bom desempenho, embora tenha errado a previsão de 502 e 700 chamados de REPARO E CORREÇÃO e DÚVIDAS E ORIENTAÇÕES, classificando como SERVIÇO TÉCNICO todos esses casos. Vide Figura 3.

B. Modelo LSTM

O modelo LSTM01 apresentou uma acurácia no dataset de testes de 80,45% e uma perda de

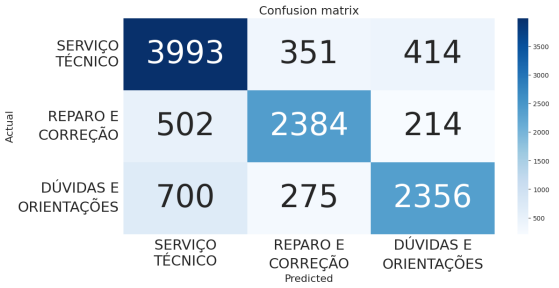

Figura 3: Random Forest - Matriz de Confusão

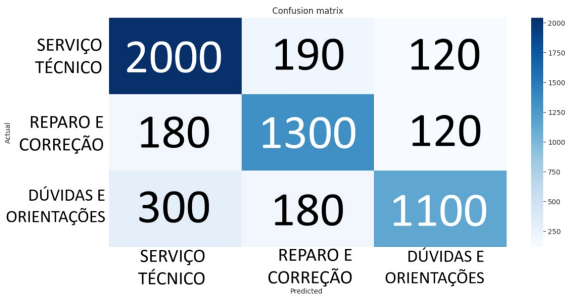

Figura 4: Matriz de Confusão (LSTM01 - 3 Categorias)

0.4799, tendo seus resultados retratados, abaixo, na matriz de confusão e gráfico de desempenho do treinamento, vide figura 4, 5 e 6 respectivamente:

Da análise do gráfico 5 e 6, infere-se que a acurácia de treinamento aumenta ao longo das épocas, enquanto, a acurácia de validação flutua um pouco, mas permanece relativamente constante após um aumento inicial.

A perda de treinamento diminui consistentemente, enquanto a perda de validação diminui rapidamente nas primeiras épocas e depois se estabiliza.

A pequena divergência entre as curvas de treinamento e validação indica que o modelo ainda pode estar aprendendo algumas nuances dos dados de treinamento.

No entanto, a acurácia e a perda, no conjunto de validação são, relativamente estáveis após as primeiras épocas, o que pode indicar que o modelo está generalizando e não está sofrendo de algum *overfitting*.

No treinamento desse modelo, obteve-se as seguintes métricas: Precisão = 81%, Recall e F1-Score igual a 80%. Isso indica que 81% das previsões positivas estavam corretas e 80% dos casos realmente positivos foram identificados. O valor de F1-Score do modelo indica um bom equilíbrio entre precisão e *recall*.

O modelo *LSTM01* também foi treinado com *cross validation*. Tendo obtido uma melhora da acurácia para 82,11% no *dataset* de teste e da perda para 0,44458. Sendo que teve a sua melhor performance com k = 4, conforme pode ser observado nos gráficos 7, 8, 9, 10 e 11.

No entanto, as suas métricas de precisão, *recall* e *F1-Score* não melhoram em relação à arquitetura *LSTM02*.

Ambas as arquiteturas apresentaram uma precisão de 0,83, *recall* de 0.81 e *F1-Score* de 0,81, isso indica que 83% das previsões positivas estavam corretas e 81% dos casos realmente positivos foram identificados. O F1-Score do modelo foi de 81%, indicando um bom equilíbrio entre precisão e *recall*.

O modelo LSTM02 apresentou uma acurácia no dataset de testes de 80,41% e uma perda de 0.4853, tendo seus resultados retratados, abaixo, na matriz de confusão e gráfico de desempenho do treinamento, vide figuras 12, 13 e 14 respectivamente:

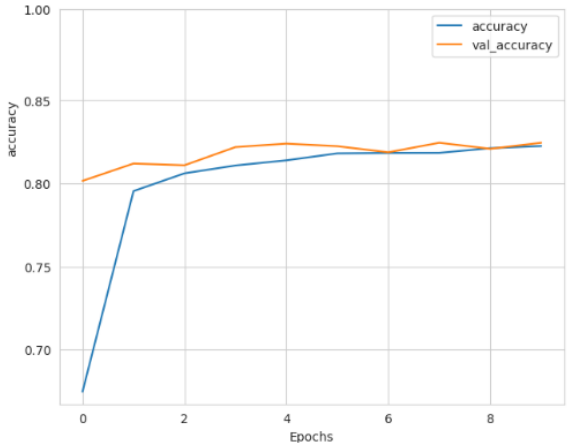

Figura 5: LSTM01 - 3 Categorias - Evolução da Acurácia

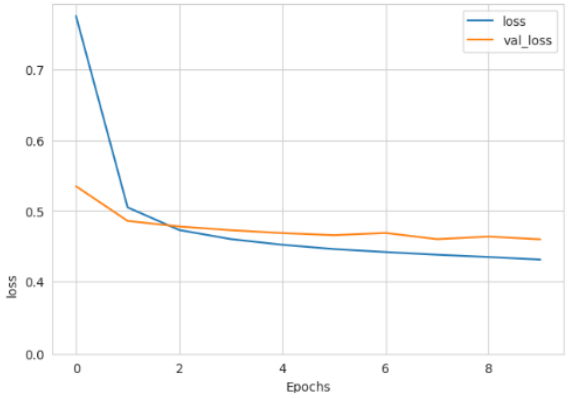

Figura 6: LSTM01 - 3 Categorias - Evolução da Perda

Embora não tenha sido observada uma melhora da acurácia em relação ao dataset de teste e à perda, percebeu-se uma leve melhora nas curvas de aprendizado, vide figuras 13 e 14.

De fato, na matriz de confusão, ilustrada na figura 12, a quantidade de acertos para a categoria SERVIÇO TÉCNICO foi de 2100 chamados, enquanto na arquitetura LSTM01 foi de 2000.

Na arquitetura LSTM01, o modelo errou 120 previsões para a classe DÚVIDAS E CORREÇÕES, quando na verdade era SERVIÇO TÉCNICO. Já no modelo LSTM02, ele errou menos, ou seja, previu 84 ao invés de 120. Esse modelo apresentou uma precisão de 0,83, recall de 0,81 e F1-Score de 0,81.

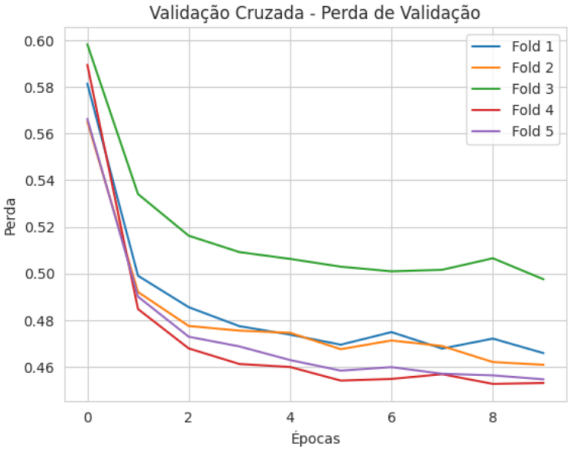

Figura 7: LSTM01 - 3 cat - Kfold - perda - validacao

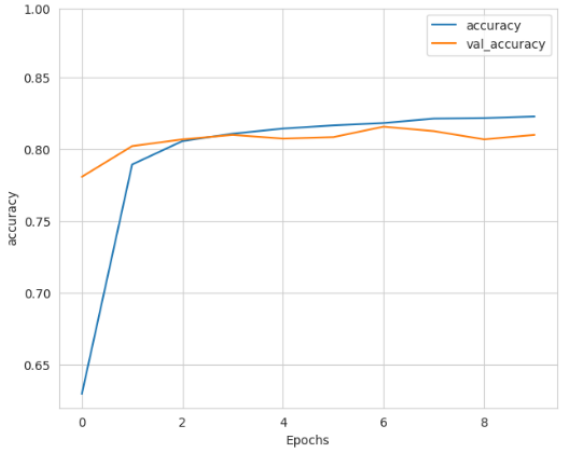

Figura 8: LSTM01 - 3 categorias - k=4 - Acurácia

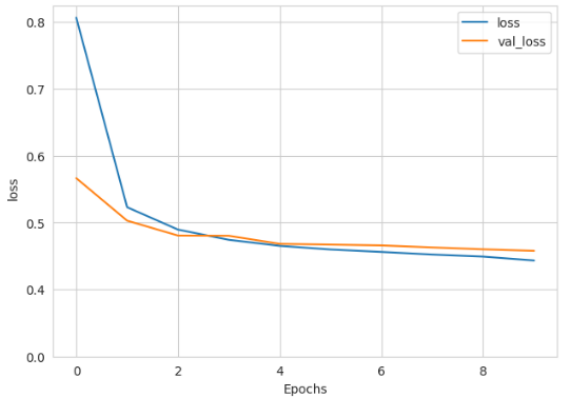

Figura 9: LSTM01 - 3 categorias - k=4 - Perda

Embora a classe OUTROS tenha sido desconsiderada pelos motivos supramencionados, foi experimentado o treinamento da arquitetura *LSTM02* com a categoria OUTROS, objetivando provar a tese que sua inclusão poderia provocar um problema de *overfitting* no modelo final. Sendo que foi utilizada a técnica de EarlyStopping sob a métrica perda no conjunto de validação, com patience=3 e min_delta=0.0001. Isso significa que a variação mínima para considerar que houve uma melhora foi de 0.0001 e a tolerância para o treinamento de mais 3 épocas sem nenhuma melhora em val_loss. Se após 3 épocas a val_loss não

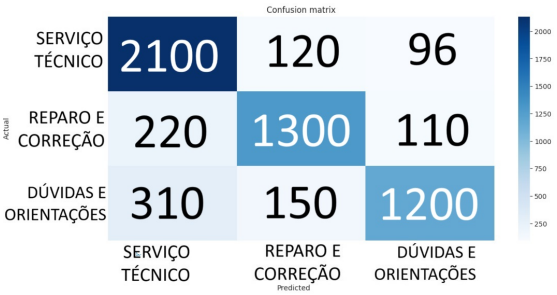

Figura 10: LSTM01 - três categorias - Matriz de Confusão ao utilizar Cross Validation

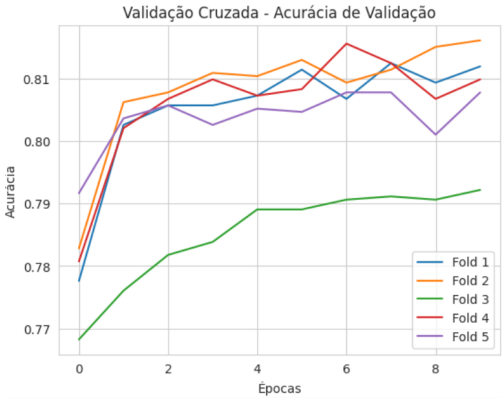

Figura 11: LSTM01 - três categorias - Evolução da acurácia para cada Fold ao utilizar Cross validation

melhorasse, o treinamento foi configurado para ser interrompido De fato, o gráfico das Figuras 15 e 16 demonstra que o modelo apresenta dificuldade de generalizar diante do grupo de validação. Em outras palavras, a perda de validação (val_loss) é praticamente constante, o que sugere que o modelo atingiu um ponto de saturação, ou seja, parou de melhorar no conjunto de validação. Ademais, ele apresenta um decréscimo do *recall* e *F1-Score* mais considerável. Tendo a precisão = 78%, *Recall* = 62% e *F1-Score*=63% Vale ressaltar que, embora as métricas mencionadas, demonstram uma piora da performance do modelo, ele ainda indica uma performance de acurácia em cerca de 80%. Mais especificamente, 80,48 % de acurácia no dataset de teste.

C. Modelo BERT

Apesar de ser um modelo poderoso em problemas relacionados a *PLN*, não foi possível explorar muito de suas potencialidades devido a falta de hardware apropriado para sua execução.

Desta forma, trabalhou-se com uma arquitetura mais simples e com a redução de épocas para experimentar o seu potencial retratado na literatura[4], obtendo-se bons resultados parciais.

Foi realizado seu treinamento com 3 categorias, removendo a categoria OUTROS, pois essa formação apresentou melhores resultados nos demais modelos. Ressalta-se que os textos utilizados tiveram o mesmo tratamento dos demais modelos.

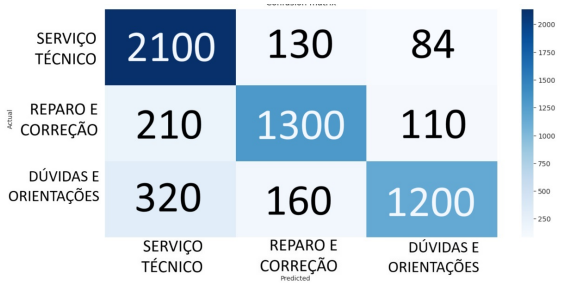

Figura 12: LSTM02 - 3 categorias - matriz de confusão

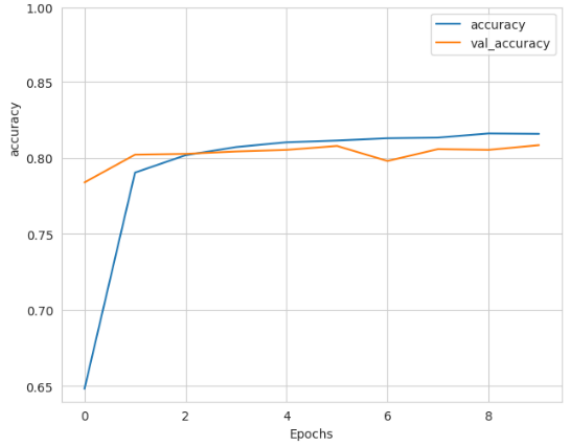

Figura 13: LSTM02 - 3 categorias - Evolução da Acurácia

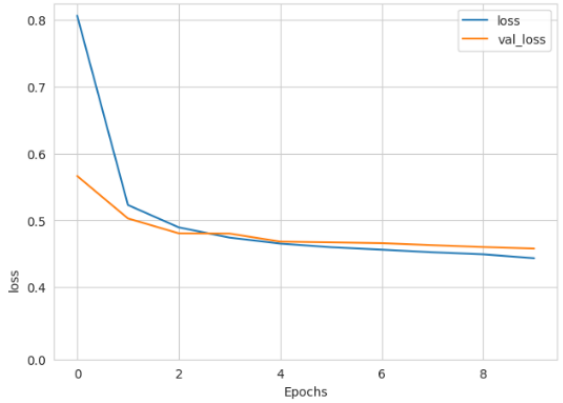

Figura 14: LSTM02 - 3 categorias - Evolução da perda

Visando um desempenho mais rápido da tokenização, foi utilizada a função *map* que divide o processamento em lotes, agilizando a execução por aplicar operações simultâneas em um grupo, ao invés de repeti-las diversas vezes para cada unidade de dado.

Devido à robustez do modelo, só foi possível rodá-lo no Google *colab* utilizando *GPUs*, as demais tentativas não prosperaram devido às restrições do ambiente em permanecer longos períodos processando.

A arquitetura foi montada utilizando uma *seed* fixa para possibilitar a reprodutibilidade dos resultados. Foi utilizada somente uma época e 4476 passos durante o treinamento. Ademais, foram utilizados os seguintes parâmetros:

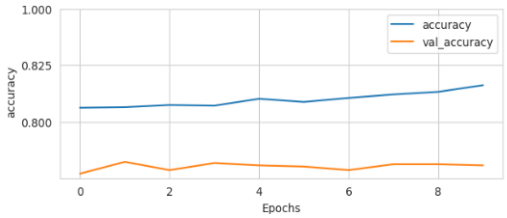

Figura 15: LSTM02 - 4 categorias - Evolução da Acurácia

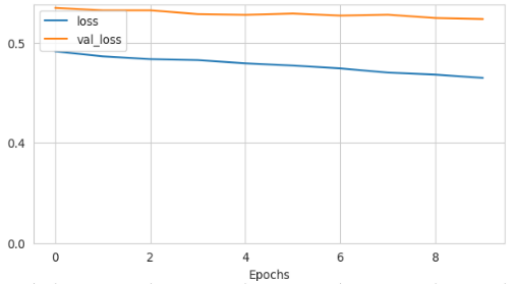

Figura 16: LSTM02 - 4 categorias - Evolução da Perda

- *per_device_train_batch_size* e *per_device_eval_batch_size* são parâmetros que dizem a quantidade de exemplos processados por lote, foram configurados em 8 para não consumir tanta memória, devido às restrições do ambiente.
- *Max_steps* igual a 700, eval_steps definido para avaliar o modelo a cada 500 passos, monitorando o desempenho e evitando o overfititng
- save_steps para salvar o modelo a cada 1000 passos, permitindo encontrar versões do modelo com melhores desempenhos e para permitir continuar o processamento em casos de interrupção indevida
- *Warmup_steps*, que aumenta gradualmente a taxa de aprendizado no início do treinamento, foi configurado para 500, esperando contribuir com um treinamento mais suave e eficiente
- "decadência de pesos"(*weight_decay*) igual a 0.01, essa técnica de regularização ajuda a evitar sobreajuste (overfitting)
- *evaluation_strategy* e *logging_strategy* utilizaram estratégia de “passos”, taxa de aprendizado igual a 5 x 10^5 (ou 5e-5)
- *logging_steps* guardando log a cada 1 passo, guardando o melhor modelo ao fim, utilizando a acurácia para selecionar o melhor modelo

Também foram medidas as seguintes métricas: f1-score, acurácia, recall e precisão.

Devido ao comportamento diferente do modelo que não registra todos os dados de métricas para todo o treinamento por padrão, foi utilizada uma abordagem diferente para montar o gráfico, montando Perda por Passos.

Não foi possível montar o gráfico de acurácia para o treinamento, pois somente a acurácia final foi registrada e a acurácia de alguns passos. No

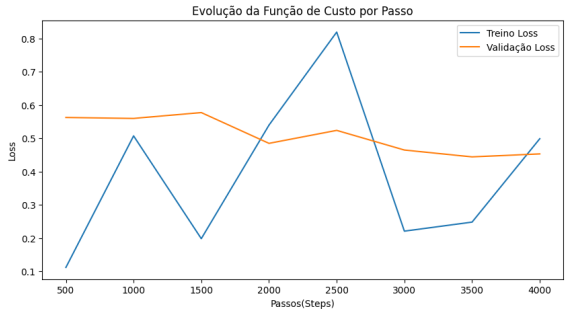

Figura 17: BERT - Evolução da Função de Custo do Treino e Validação

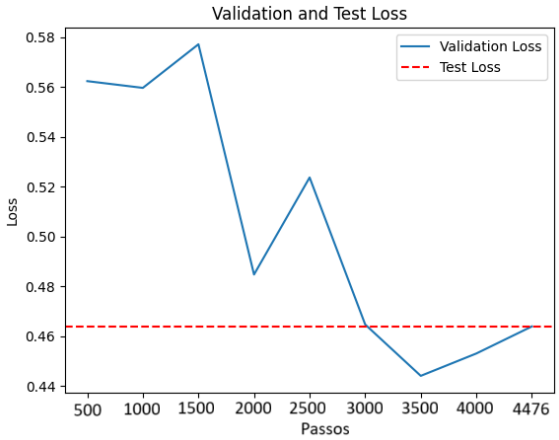

Figura 18: BERT - Evolução da Função de Custo do conjunto de Validação comparado com o valor de teste médio

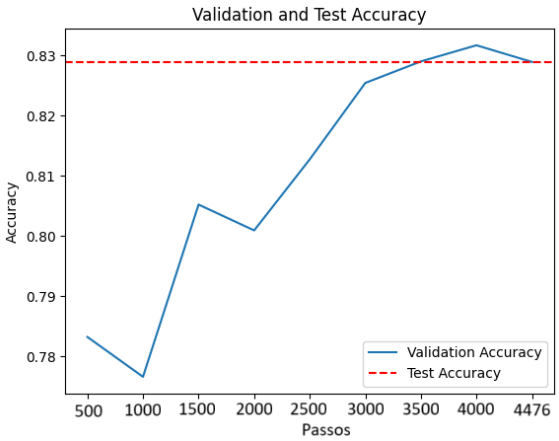

Figura 19: BERT - Evolução da Acurácia do conjunto de Validação comparado com o valor de teste médio

entanto, a acurácia medida foi alta nos pontos registrados.

Foi possível montar um gráfico comparando a perda na validação com a acurácia média de teste. Por esse motivo, foi traçada uma linha comparando a evolução da validação com o valor médio de teste.

Da mesma forma foi possível montar o gráfico com a acurácia da validação e da medição média no conjunto de teste. Assim, comparamos a evolução da acurácia no conjunto de validação com o valor médio de teste.

Devido às dificuldades de execução não foi possível fazer novos testes a fim de explorar melhor o modelo e obter dados melhores para montar os gráficos.

As métricas obtidas dos modelos estão nas Tabelas VII e VIII.

Tabela VII: Tabela de Resultados

Modelo	Acurácia	Perda	Recall
Random Forest	78,86%	-	79%
LSTM1(cat=3)	80,45%	0,479	80%
LSTM1(cat=3/cv)	82,11%	0,445	81%
LSTM2(cat=3)	80,41%	0,485	81%
LSTM2(cat=4)	80,48%	0,489	62%
BERT	83%	0,453	83%

Tabela VIII: Tabela de Resultados

Modelo	F1-score	Precisão
Random Forest	79%	79%
LSTM1(cat=3)	80%	81%
LSTM1(cat=3/cv)	81%	83%
LSTM2(cat=3)	81%	83%
LSTM2(cat=4)	63%	78%
BERT	83%	83%

V. CONCLUSÃO

Diante dos experimentos realizados, chegou-se ao entendimento de que os algoritmos de machine learning utilizados apresentaram-se como uma boa solução para a classificação automática dos macrosserviços dos chamados de TI, em reposta ao problema de pesquisa proposto. Tendo em vista que esses classificadores se mostraram capazes de categorizar corretamente uma quantidade relevante de chamados. Consequentemente, eles também têm potencial para alcançar os citados objetivos de negócio, tais como: auxiliar na devida determinação dos SLAs e na priorização mais célere das solicitações. Além disso, podem contribuir para a redução do erro humano na classificação e para a diminuição da sobrecarga dos atendentes. Por fim, têm-se como mais benefícios melhoria na correta priorização de demandas e indicadores gerenciais mais precisos.

Contudo, foram encontradas algumas dificuldades que podem ter influenciado na obtenção de resultados ainda melhores. De fato, os textos possuíam muitas informações desnecessárias, alguns chamados possuíam erros de classificação, algumas classes estavam muito desbalanceadas, existindo classes com poucos exemplos reais. Além de poder computacional abaixo do necessário, o que inviabilizou, por exemplo, a execução de mais de uma época para o BERT.

Ainda assim, o bert, que é um tipo de transformer, apresentou maior potencial para a resolução do problema. De fato, no passo 4000, ele já estava apresentando uma acurácia, precisão, recall e F1-Score de cerca de 83% em uma única época.

Por outro lado, a arquitetura final do LSTM também obteve uma boa performance, consumindo menos recursos, se comparado ao BERT.

O Random Forest foi o que teve a menor performance, no entanto, próximo dos demais, tendo suas métricas cerca de 2% abaixo em relação ao LSTM.

Vale ressaltar que existe uma demanda futura de melhoria da qualidade dos dados junto ao gestor, o que, potencialmente, deve incrementar a performance desses modelos.

Assim, sanados alguns problemas inerentes ao negócio e aos dados, bem como os respectivos refinamentos no modelo, sugere-se, futuramente, a sua integração com o atual Sistema da Central de Serviços de TI

Outra barreira nessa integração seria a mensuração do desempenho do modelo com relação ao tempo necessário para as previsões diante do alto fluxo que o ambiente de produção pode demandar.

Uma outra preocupação seria a implementação progressiva e sistemática da solução na classificação automática no ambiente de produção, sem intervenção humana. Em outras palavras, sugere-se que a solução seja aplicada em fases, com algum monitoramento inicial do ser humano, avaliando seu comportamento no ambiente de produção.

Adicionalmente, torna-se necessário estabelecer uma estratégia de atualização do modelo periodicamente para que o mesmo não se torne obsoleto, refletindo às atualizações ou mudanças do ambiente

Também, como trabalhos futuros podem ser explorados: a criação de um chatbot, integrado ao modelo, como mais uma opção de abertura de chamado e sua classificação automática, além de direcionamento destes chamados para a área correta de forma tempestiva; a investigação de mais técnicas de data augmentation e balanceamento de classes; a experimentação dos modelos com outros parâmetros; utilização de outros modelos; avaliação com outras métricas; utilização IA generativa para reclassificação os serviços de TI; reclassificação manual dos serviços de TI; avaliação da viabilidade de utilização do agrupamento, um tipo de algoritmos de aprendizado não supervisionado, para automatizar o processo de verificação e/ou reclassificação dos chamados;

Cabe observar que torna-se imprescindível a melhoraria na configuração das máquinas e ambientes para o processamento de modelos que demandam mais recursos computacionais

Referências

[1] Brasil. Lei n° 13.709, de 14 de agosto de 2018, Lei Geral de Proteção de Dados Pessoais (LGPD). Diário Oficial da União, Brasília, DF, 15 ago. 2018, 2018. Acesso em: Ago. 2024.

[2] L. Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[3] M. A. Cruz. *Detecção de solicitações de usuários duplicadas utilizando aprendizado de máquina*. PhD thesis, Mestrado em Sistemas de Informação e Gestão do Conhecimento, 2021.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Explosion AI. spaCy: Industrial-strength Natural Language Processing in Python. https://spacy.io, 2020. Acesso em: jan. 2024.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[7] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[8] J. H. Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (3rd ed.)*. Prentice Hall, 2021.

[9] S. Paramesh, C. Ramya, and K. Shreedhara. Classifying the unstructured it service desk tickets using ensemble of classifiers. In *2018 3rd international conference on computational systems and information technology for sustainable solutions (CSITSS)*, pages 221–227. IEEE, 2018.

[10] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *Proc. 54th Annu. Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 86–96, 2016.

[11] F. A. Weber. Técnicas de processamento de linguagem natural aplicadas à gestão de serviços de ti. *Especialização em Data Science e Big Data Universidade Federal do Paraná*, 2020.

Sistema FIEB | SENAI CIMATEC
PELO FUTURO DA INOVAÇÃO

CENTRO UNIVERSITÁRIO SENAI CIMATEC
ESPECIALIZAÇÃO EM DATA SCIENCE & ANALYTICS

ATA DE APRESENTAÇÃO DE PROJETO FINAL DE CURSO

Ata de apresentação do Projeto Final de Curso "**Utilização de Modelos de ML para classificação de Solicitações de Serviços de TIC"**, submetido pelo aluno **Leonardo Melo Costa da Silva** como parte dos requisitos para obtenção do Certificado de **Especialista em Data Science & Analytics** pelo Centro Universitário SENAI CIMATEC, às 14h30 do dia 21 de agosto de 2024. Reuniu-se remotamente pela plataforma Microsoft Teams, a Banca Examinadora designada pelo Prof. Dr. Márcio Freire Cruz - Orientador, constituída pelo Prof. Dr. Márcio Freire Cruz e pelo Prof. Esp. Tácito Henrique da Silva Graça. O orientador deu início aos trabalhos com as devidas orientações, e a exposição foi realizada pelo estudante dentro do prazo de tempo estabelecido. Ao final da apresentação a banca reuniu-se atribuindo a seguinte nota: 9,0 (Nove pontos).

A banca de avaliadores decidiu pela:

(X) Aprovação do trabalho
Caberá ao aluno apresentar em no máximo em 30 (trinta) dias a contar da data de assinatura desta Ata, uma cópia do trabalho em PDF, constando as considerações pontuadas pela banca. A Ata de Apresentação do Projeto Final de Curso deve ser digitalizada e inserida na última página do artigo.

( ) Reprovação do trabalho
O aluno terá que se matricular novamente no TCC – Trabalho de Conclusão de Curso e ser submetido a uma banca avaliadora no semestre seguinte**.**

As ações consequentes ao status de Aprovação deverão obedecer ao prazo proposto acima sob pena do parecer final ser modificado para o status de Reprovado automaticamente e sem possibilidade de recurso.

Para constar, lavrou-se a presente ata que vai assinada por todos os membros da Banca. Por estarem cientes de suas obrigações estão de acordo com os termos desse documento:

Salvador, 21 de agosto de 2024

Assinado eletronicamente por:
Marcio Freire
CPF: ***.235.375-**
Data: 16/09/2024 16:53:41 -03:00

Márcio Freire Cruz
Professor Orientador

Assinado eletronicamente por:
Tácito Henrique da Silva Graça
CPF: ***.007.225-**
Data: 16/09/2024 17:23:02 -03:00

Tácito Henrique da Silva Graça
Membro da banca

Assinado eletronicamente por:
Patricia Freitas Tourinho
CPF: ***.733.265-**
Data: 16/09/2024 16:37:00 -03:00

Patrícia Freitas Tourinho
Coordenadora do Pós-Graduação Lato Sensu