ISSN: 1983 7402 ITA, 29 a 30 SET 2020

Classificação automática de temporalidade de documentos usando processamento de linguagem natural e Redes Neurais *LSTM*

Edney da Silva Santos[1] e Carlos Henrique Costa Ribeiro[2]
[1]Instituto Tecnológico de Aeronáutica, São José dos Campos/SP - Brasil
[2]Instituto Tecnológico de Aeronáutica, São José dos Campos/SP - Brasil

***Resumo*— Este artigo apresenta uma proposta para classificação automática de temporalidade de documentos oficiais da Força Aérea Brasileira (FAB) por meio de uma Rede Neural Recorrente do tipo LSTM (*Long short-term memory*) e técnicas de processamento de linguagem natural (NLP). A correta classificação do horizonte de tempo para o qual um documento deve ter a sua salvaguarda garantida permite a racionalização administrativa, a agilidade, a transparência no acesso às informações e a adequada preservação do patrimônio documental de uma instituição. A partir da obtenção de aproximadamente 16 mil documentos do SIGADAER (Sistema Informatizado de Gestão Arquivística de Documentos da Aeronáutica), com as respectivas temporalidades preenchidas, foi implementado um classificador automático de documentos, com aprendizado supervisionado, o qual obteve valores de acurácia média acima de 87% em um modelo de *Deep Learning* com duas camadas LSTM. Em trabalhos futuros deseja-se aperfeiçoar este modelo considerando a aplicação de outras técnicas de aprendizado de máquina.**

***Palavras-Chave*— Processamento de Linguagem Natural, Redes Neurais, Documentos.**

I. Introdução

No âmbito do Governo Federal e por consequência na Força Aérea Brasileira, conforme preconiza a Lei Federal nº 8.159, de 8 de janeiro de 1991, art. 1º, *"É dever do Poder Público a gestão documental e a proteção especial a documentos de arquivos, como instrumento de apoio à administração, à cultura, ao desenvolvimento científico e como elementos de prova e informação"*. Visando cumprir esta determinação de proteger e gerir o patrimônio documental da nação, o CONARQ (Conselho Nacional de Arquivos), órgão vinculado ao Arquivo Nacional do Ministério da Justiça e Segurança Pública, através da Resolução nº 14, de 24 de outubro de 2001, decidiu criar o Código de classificação de documentos de arquivo para a administração pública [1].

A classificação é essencial para a organização dos arquivos em suas fases corrente, intermediária e permanente, e permite, além do acesso eficaz, a transparência na gestão documental. A produção de documentos controlada e classificada, e a racionalização do seu fluxo, por meio de um SIGAD (Sistema de Gestão Arquivística de Documentos), traz grandes benefícios para qualquer instituição e é um dos objetivos de um programa de gestão de documentos, melhorando a qualidade nos serviços arquivísticos e consequentemente dando suporte para as decisões político-administrativas.

E. da Silva Santos, edneyess@fab.mil.br; C. Henrique Costa Ribeiro, carlos@ita.br.

Em seu *software* de gestão arquivistica, o SIGADAER, a Aeronáutica permite aos seus usuários, de forma não obrigatória, a realização da classificação de temporalidade dos documentos com base na citada resolução do CONARQ. O usuário escolhe o código da classificação de temporalidade, o qual representa, de acordo com o assunto, tema e área, por quanto tempo este documento deverá ser mantido em uma base de arquivos correntes e em que momento ele migrará para uma base de arquivos nas fases intermediária, permanente ou até mesmo para descarte. O volume de documentos produzidos por todas as unidades organizacionais da Aeronáutica, espalhadas em todo o território nacional, e a complexidade em atribuir a correta classificação de temporalidade conforme a tabela de temporalidade prevista na resolução, tem como consequência a geração de enormes quantidades de documentos físicos e digitais sem os seus limites de tempo de salvaguarda definidos. Em um cenário mais abrangente, ou seja, levando em conta todos os documentos oficiais do governo federal, a quantidade de documentos não classificados ou com classificação inadequada é da ordem de milhões.

Para mitigar a grande quantidade de documentos com problemas na classificação de temporalidade, surgem basicamente duas opções: promover comissões de militares e civis especialistas para realizar mutirões de classificação manual dos documentos, ou prover um sistema automatizado, possivelmente baseado em técnicas de Inteligência Artificial, para prover ou auxiliar na classificação.

II. Referencial Teórico

A classificação automática de textos vem sendo estudada e utilizada em diversas aplicações reais nas últimas décadas. Trata-se de uma subárea do Processamento de Linguagem Natural (NLP — *Natural Language Processing*), cujas técnicas computacionais tornam possível realizar tradução automática de textos, reconhecimento de voz, conversão de voz para texto escrito, análise de sentimentos e até mesmo prover sistemas de atendimento telefônico por voz automatizada, também conhecidos como *chatbots*.

Normalmente os sistemas para classificação automática de textos, baseados em NLP, são compostos pelas seguintes fases: coleta de dados, pré-processamento, extração de características, seleção do classificador e avaliação do modelo [2]. Neste artigo serão a seguir discutidas técnicas e implementações para cada uma dessas fases, dando ênfase para aquelas relacionadas diretamente com a proposta deste trabalho.

A. Coleta de dados

A primeira fase no processo de classificação é obter as bases de dados que serão utilizadas tanto no treinamento quanto nos testes do modelo. Nesta fase, é muito comum a utilização de técnicas para converter os dados em formatos como html, doc e pdf para texto ASCII.

B. Pré-processamento

Nesta fase, algoritmos são aplicados ao conjunto de textos obtidos da fase de coleta de dados, promovendo uma limpeza de informações irrelevantes para a classificação. Técnicas muito utilizadas na fase de pré-processamento são a tokenização, *padding* [3], remoção de *stop words*, remoção de pontuação e caracteres especiais e remoção de espaços em branco duplicados entre as palavras [2]. Após a aplicação de todos os processos nesta fase de pré-processamento, os dados limpos e no formato texto puro estão prontos para a fase subsequente.

C. Extração de características

A extração de características consiste em aplicar modificações no texto não-estruturado, visando permitir que o classificador receba os dados em um formato padronizado e organizado. Nesta fase, os dados normalmente são transformados em algum tipo de representação em um espaço vetorial de atributos, em que cada documento é composto por um vetor de palavras que busca descrevê-lo de forma unívoca. As transformações aplicadas promovem a redução da dimensionalidade e são conhecidas como técnicas de *Word Embedding*. As técnicas mais comuns são *Term Frequency-Inverse Document Frequency (TF-IDF)* [4], *Term Frequency (TF)* [4], *Word2Vec* [5], e *Global Vectors for Word Representation (GloVe)* [6].

Além das técnicas citadas, em modelos de aprendizado profundo baseados em Redes Neurais Artificiais, a extração de características pode ser realizada na primeira camada do modelo através do aprendizado de características, que se dá da seguinte forma:

1) ***Lookup Table*:** criação de uma tabela onde cada linha possui um índice para uma palavra do vocabulário e uma representação vetorial, formada inicialmente por valores reais aleatórios;
2) **Mapeamento da entrada da Rede Neural:** A entrada da rede, já no formato de vetor de inteiros, é mapeada na *Lookup Table*;
3) **Aprendizado:** Durante o treinamento da rede, na fase de *back propagation*, o vetor de números reais, ou seja os pesos, são atualizados, de forma a diminuir o erro atual da rede. Este processo de atualização dos pesos é o mesmo que acontece para as demais camadas da rede.

A Figura 1 ilustra uma rede com *Embedding Layer*.

D. Seleção do classificador

A escolha do tipo de classificador que melhor se adequa ao problema proposto é um dos passos mais importantes em um sistema de classificação automática de textos. As abordagens mais populares são *Naïve Bayes* [7], *Regressão Logística* [8], *K-Nearest Neighbor (KNN)* [9], *Support Vector Machine (SVM)* [10], além daquelas baseadas em estrutura de dados do tipo árvore como *Random Forest* [11] e Árvores de Decisão [12].

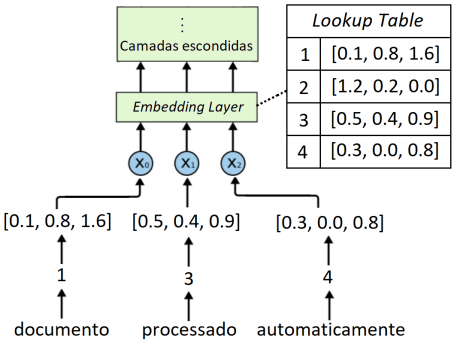

Fig. 1. Entrada da rede com *Embedding Layer*

Nos últimos anos surgiram soluções de classificação baseadas em *Deep Learning* [13], produzindo resultados frequentemente melhores do que os classificadores tradicionais citados. O sucesso destas novas abordagens se deve a sua grande capacidade de modelar complexidade e não-linearidade nos relacionamentos dos dados.

As Redes Neurais Recorrentes [14], mais especificamente do tipo *Long Short-Term Memory* [15], utilizadas em modelos de *Deep Learning*, têm mostrado ótimos resultados na área de processamento de linguagem natural. Este tipo de Rede Neural Artificial permite modelar a dependência temporal que existe nos dados por meio de realimentação em suas células de processamento. Além da capacidade de reter informações como nas Redes Neurais Recorrentes tradicionais, as redes LSTM podem processar séries temporais com intervalos de tempo irregulares.

Diferente das Redes Neurais *Feed Forward* [16], as Redes Neurais Recorrentes têm como entrada não só os dados das amostras correntes, mas também a contribuição dos dados das amostras anteriores. Este artifício é alcançado por meio da realimentação das unidades de processamento. A Figura 2 ilustra uma rede recorrente desdobrada no tempo [17].

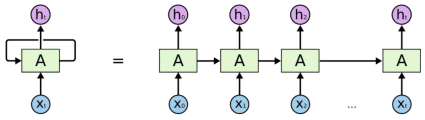

Fig. 2. Rede Neural Recorrente desdobrada no tempo [17]

Nas Redes Neurais Recorrentes do tipo LSTM, a presença de três portas (*gates*), nomeadamente *input gate*, *output gate* e *forget gate*, permite a modelagem da dependência de longa duração, evitando assim o problema clássico de *vanish gradient* [18] das Redes Neurais Recorrentes tradicionais. A Figura 3 mostra uma rede LSTM desdobrada no tempo, destacando a estrutura interna da célula no segundo passo.

Além das redes recorrentes do tipo LSTM, existem muitas opções de classificadores usados para a área de processamento de textos, e tem-se observado um grande crescimento em pesquisas relacionadas ao tema. Contudo, especial atenção deve ser dada à escolha da abordagem mais adequada para o processamento de linguagem natural, e para a parametrização do classificador escolhido.

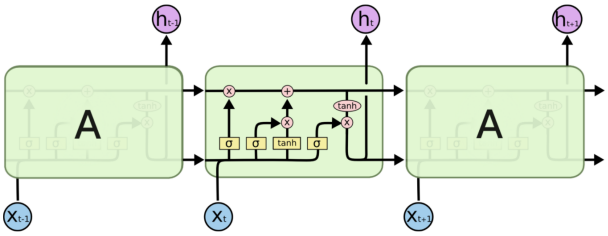

Fig. 3. Rede Neural LSTM desdobrada no tempo, com destaque para a estrutura interna da célula no segundo passo [17]

E. Avaliação do Modelo

Após a limpeza dos dados, extração de características, treinamentos e testes utilizando o modelo de classificador escolhido, tem-se a fase de avaliação da qualidade dos resultados obtidos. As métricas mais utilizadas para avaliar modelos de classificadores são [19]: *Precision*, *Recall*, *F1-Score* e *Accuracy*. Estas métricas podem ser calculadas a partir dos valores apresentados na Matriz de Confusão [20], uma estrutura tabular onde as linhas são representadas pelas classes preditas e as colunas pelas classes reais. A partir desta tabela, pode-se identificar quatro sub-conjuntos, essenciais para a obtenção das métricas estatísticas citadas: Verdadeiro Positivo (VP), Falso Positivo (FP), Verdadeiro Negativo (VN) e Falso Negativo (FN).

Os valores das quatro métricas podem então ser obtidos da seguinte forma:

- ***Precision*:** métrica que indica a quantidade de amostras positivas que foram classificadas corretamente em relação ao total de amostras classificadas como positivas:

$$Precision = \frac{VP}{VP + FP} \tag{1}$$

- ***Recall*:** medida da quantidade de amostras classificadas corretamente como positivas em relação a quantidade real de amostras positivas. De forma resumida, indica a qualidade do classificador para classificar corretamente a classe de interesse:

$$Recall = \frac{VP}{VP + FN} \tag{2}$$

- ***F1-Score*:** média harmônica entre os valores de *Precision* e *Recall*, com o objetivo de identificar um balanceamento entre as duas métricas anteriores:

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

- ***Accuracy*:** corresponde à quantidade de amostras classificadas corretamente em relação ao total de amostras:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{4}$$

III. Materiais e Métodos

Para atingir o objetivo deste trabalho, as atividades foram divididas em duas etapas consecutivas: a) Obtenção e pré-processamento dos dados; e b) Modelagem, treinamento e teste da Rede Neural LSTM. Em seguida, em tópicos dedicados, serão apresentados os experimentos e os respectivos resultados obtidos, com uma avaliação detalhada do modelo de classificador escolhido.

A primeira etapa foi voltada à obtenção e tratamento da base de dados utilizada para a realização dos experimentos. O conteúdo e os metadados dos documentos do SIGADAER são armazenados em um banco de dados relacional do tipo *PostgreSQL*. É importante informar que os metadados necessários aos experimentos estavam distribuídos em mais de uma tabela do banco de dados, sendo necessária a construção de consultas SQL para a seleção dos dados relevantes.

Com os dados selecionados, foi produzido um arquivo no formato CSV, posteriormente utilizado como entrada para os *scripts* desenvolvidos durante a etapa seguinte.

Após a obtenção da base de dados, foram realizadas inicialmente as seguintes tarefas de pré-processamento: eliminação de *tags* HTML no conteúdo de texto; filtragem de palavras com pouco valor semântico, como preposições, artigos e conjunções; e exclusão de pontuações e símbolos especiais, como exclamações, interrogações, entre outras. Além das filtragens citadas, foram eliminados também acentos, números e espaços duplicados, assim como palavras com menos do que dois ou com mais que 40 caracteres.

Ainda com o intuito de preparar o conjunto de dados para os experimentos, tornou-se necessária a unificação de alguns valores presentes no conjunto de classes de temporalidade para os documentos. Por exemplo, os documentos classificados com as temporalidades “5 anos a partir da desativação do meio”, “5 anos a partir da data de baixa do armamento” e “5 anos a partir da desativação do sistema ou equipamento” tiveram seus rótulos unificados, passando todos eles a ter o mesmo rótulo de “5 anos”. Assumiu-se portanto que documentos com os rótulos citados podem ter seus rótulos unificados, com pouca perda semântica.

Já a segunda etapa teve como principal objetivo a modelagem da Rede Neural LSTM para o treinamento e realização de inferências de classes para temporalidades dos documentos. Nesta etapa, os dados já filtrados anteriormente sofreram um processo de *tokenização*, transformando cada palavra em um número inteiro que a representa dentro de um dicionário formado pelo conjunto das palavras mais frequentes.

Assim como o conteúdo de texto, as classes que representam as temporalidades, ainda no formato de categorias, também foram transformadas através de um processo conhecido como *one-hot-encoding*. Nesse processo de transformação, as classes em formato texto são mapeadas em uma sequencia formada por um conjunto de zeros e apenas um digito com valor 1 em posição específica. A Tabela I apresenta alguns exemplos do mapeamento.

TABELA I
EXEMPLO DE MAPEAMENTO DAS CATEGORIAS PARA *one-hot-encoding*

temporalidade (ds_fase_corrente)	*one-hot-encoding*
1 ano	10000000
2 anos	01000000
3 anos	00100000
5 anos	00010000
...	...

Ainda na segunda etapa, foi criada a arquitetura de Rede Neural LSTM; foram separados os conjuntos de dados de treinamento e testes; e foram realizados treinamentos da Rede

Neural, avaliando-se, durante o processo, os hiperparâmetros do modelo.

Por fim, foram realizados experimentos com o conjunto de dados de testes e obtenção de medidas estatísticas para verificar a qualidade do aprendizado da referida Rede Neural.

A. Conjunto de Dados

A base de dados para a realização dos experimentos foi obtida junto à equipe de desenvolvimento de *software* do Centro de Computação da Aeronáutica de São José dos Campos (CCA-SJ). Trata-se de uma base de dados de testes do sistema SIGADAER, em sua versão 6.0, com informações sensíveis descartadas.

Apesar da base de dados possuir somente informações de carácter ostensivo, observou-se o cuidado em eliminar nomes de pessoas, patentes militares, datas, nomes de organizações militares, etc, antes da sua utilização nos experimentos.

Através de consultas SQL, foi feita a união de informações das tabelas de documentos e temporalidades, dando origem ao conjunto de dados inicial, utilizado como entrada para a fase de pré-processamento dos dados. A Tabela II apresenta a forma inicial de parte dos dados.

TABELA II
EXEMPLO DE PARTE DO CONJUNTO DE DADOS INICIAL

id	html	id_temporalidade	ds_fase_corrente
989	... 1.Em atenção aos entendimentos anteriores havidos (...)	496	5 anos
974	... 1.Em atenção aos documentos supracitados, o Centro (...)	507	1 ano

É importante observar que os atributos "id" e "id_temporalidade" não são utilizados na fase de pré-processamento, restando somente as colunas "html" e "ds_fase_corrente" para compor o conjunto de dados de entrada.

B. Pré-processamento

A partir da obtenção da base de dados, exportada para um arquivo no formato CSV, e com o intuito de preparar o texto para o treinamento da Rede Neural na etapa subsequente, foram realizadas as seguintes etapas de pré-processamento:

- **Extração de texto em HTML:** Foram eliminadas todas as *tags* HTML e seus respectivos atributos e classes, e todas as declarações de folha de estilo CSS, e foi selecionado somente o conteúdo de texto que encontrava-se dentro da seção com *id* "conteudoDocumento";
- **Remoção de *stop words* e pontuações:** Utilizando a biblioteca *NLTK - Natural Language Toolkit* [21], que é um conjunto de classes e programas para processamento natural de linguagem simbólica e estatística, foram eliminadas do texto original todas as ocorrências de palavras com pouco valor semântico, como: preposições, artigos, conjunções, etc. Além das *stop words*, foram removidas também todas as pontuações contidas no texto, tais como vírgulas, parágrafos, exclamações, etc;
- **Eliminação de acentuação, caracteres especiais, números, letras maiúsculas e texto vazio:** Após o processo de remoção de *stop words*, observou-se que o texto ainda possuía palavras com acentuação, números, letras maiúsculas, espaços duplicados entre palavras, texto contendo somente a *string* vazia e palavras menores que dois e maiores que 40 caracteres. Foi implementada uma função que percorre todo o conteúdo de texto buscando e eliminando estes tipos de ocorrência;
- **Tokenização:** Consistiu em transformar o texto livre, obtido das fases anteriores, em uma sequência de números inteiros. Cada palavra do texto é representada por um número inteiro, onde o valor é obtido em um dicionário das palavras mais frequentes. Por exemplo, a partir do dicionário de palavras frequentes {'processado': 1, 'documento': 2, 'automaticamente': 3}, pode-se mapear o seguinte texto "documento processado automaticamente" na sequência de números inteiros [2, 1, 3];
- ***Padding:*** Devido a necessidade de fixar o tamanho da entrada para a Rede Neural, foi necessário realizar um processo de *padding* no texto dos documentos, de forma a garantir que todos tivessem o mesmo tamanho, ou seja, a mesma quantidade de palavras. Foi calculado o tamanho médio dos documentos e promovido o truncamento daqueles maiores que a média, assim como o preenchimento com zeros no final daqueles que tinham o tamanho menor que a média. Após esta transformação, cada documento é composto por um vetor de números inteiros de mesmo tamanho [3];
- **Tratamento do atributo meta:** Assim como o texto livre, a coluna "ds_fase_corrente", que foi utilizada como atributo meta, também necessitou ser pré-processada. Cada tipo de texto contido nesta coluna foi transformado em um número inteiro distinto. Esta coluna possuía informações do tipo "5 anos", "7 anos", "1 ano", as quais foram transformadas em 5, 7 e 1 respectivamente, totalizando 8 categorias distintas. Após a fase de transformação do atributo meta, aplicou-se o *one-hot-enconding*, mapeando os números inteiros em representações binárias e facilitando a sua utilização como *label* de classe nos conjuntos de dados para a Rede Neural.

C. Modelo de Rede Neural LSTM

Para os experimentos realizados neste trabalho foi considerada apenas a utilização da técnica de aprendizado de máquina baseada em Redes Neurais. Foi escolhida a Rede Neural Recorrente do tipo LSTM, pois esta é amplamente utilizada para aplicações na área de processamento de linguagem natural.

A implementação da Rede Neural se deu através da utilização do *framework* de aprendizado profundo Keras [22], em sua versão 2.3.0. O modelo sequencial foi utilizado para a montagem da arquitetura da Rede Neural LSTM.

O Keras oferece a possibilidade de realizar a extração de características do vetor de entrada por meio da configuração da primeira camada do modelo sequencial. A camada *Embedding* é inicializada com pesos aleatórios, e durante o treinamento converte o vetor de índices inteiros em um tensor de

dimensões parametrizáveis. De forma resumida, cada palavra do vocabulário é mapeada em uma *lookup table* composta pelos vetores de pesos a serem treinados [23].

A Tabela III apresenta um exemplo de arquitetura de Rede Neural e seus respectivos hiperparâmetros, configurados para a execução dos experimentos.

TABELA III
EXEMPLO DE ARQUITETURA SEQUENCIAL IMPLEMENTADA

Parâmetro	Valor
Tipo de arquitetura no Keras	Sequencial
Tamanho do vocabulário	50000 palavras
Quantidade de camadas	4
1ª camada	*Embedding*
2ª camada	LSTM com 256 neurônios
3ª camada	LSTM com 256 neurônios
4ª camada	*Dense* com ativação *softmax*
Otimizador	Adam
Quantidade de classes	8
Máximo de épocas	50
Critério de parada	*Early stopping - max accuracy*

Na Tabela III observa-se também a instanciação do modelo sequencial e as seguintes camadas:

- **1ª camada - *Embedding*:** Realiza a extração de características através da transformação do texto de entrada, já no formato vetorizado, em um tensor no formato definido pelos hiperparâmetros. Nesta camada foram definidos o tamanho do vocabulário utilizado no dicionário da tokenização e a quantidade de saídas para a próxima camada. Opcionalmente pode-se configurar esta camada para utilizar um modelo de extração de características pré-treinado como: *Word2Vec*, *GloVe* e *TF-IDF*;
- **2ª camada - *LSTM*:** Composta por células LSTM e as configurações dos hiperparâmetros relativos à função de ativação, fator de *dropout* e habilitação do retorno em formato de sequência para a camada seguinte;
- **3ª camada - *LSTM*:** Mesmo tipo da camada anterior, alterando somente a habilitação do retorno em formato de sequência para falso;
- **4ª camada - *Dense*:** Camada de Rede Neural totalmente conectada. Permite a configuração da função de ativação para *softmax*, cujo objetivo é normalizar um conjunto de classes de entrada em uma distribuição de probabilidades de acordo com as exponenciais destes valores de entrada;
- **Compilação do modelo:** Após as configurações das camadas, a compilação do modelo indica qual algoritmo de otimização será utilizado pela rede e quais métricas de desempenho serão extraídas;
- **Treinamento:** Configura quais os dados de entrada e saída da rede e permite definir parâmetros relativos à quantidade de épocas de treinamento, entre outros;
- **Avaliação:** Após o treinamento do modelo, realiza-se a avaliação de medidas como função de perda, acurácia, etc.

IV. EXPERIMENTOS

Após o pré-processamento dos dados e a configuração do modelo de Rede Neural LSTM, a base de dados foi dividida em dois grupos: 70% conjunto de treinamento e 30% conjunto de testes.

Na fase de treinamento, os dados foram submetidos ao modelo por diversas épocas, cada época equivale a uma rodada completa com todo o conjunto de treinamento, e o critério de parada baseou-se na função de *callback* oferecida pelo *framework* Keras chamada *Early Stopping*, que interrompe o treinamento quando a medida de desempenho escolhida para de apresentar melhora de resultados sobre o conjunto de testes.

Com os conjuntos de dados já separados, diversos experimentos foram conduzidos, durante os quais foram feitos vários ajustes nos valores dos hiperparâmetros da Rede Neural. A Tabela IV apresenta os melhores resultados dos experimentos. Os valores de função de perda e acurácia foram obtidos sobre o conjunto de dados de testes.

TABELA IV
EXPERIMENTOS COM O MODELO IMPLEMENTADO

Exp.	Arquitetura	Épocas	Função de Perda	Acurácia Média
1	*camadas LSTM* = 1, *hidden_size* = 256, *activation* = 'relu', *optimizer* = 'adam'	6	0.5199	0.8665
2	*camadas LSTM* = 1, *hidden_size* = 256, *activation* = 'tanh', *optimizer* = 'adam'	6	0.5672	0.8681
4	*camadas LSTM* = 2, *hidden_size* = 128, *activation* = 'tanh', *optimizer* = 'adam'	4	0.4918	0.8559
6	*camadas LSTM* = 2, *Tipo* = *Bi-direcional*, *hidden_size* = 128, *activation* = 'relu', *optimizer* = 'adam'	10	0.6342	0.8663
7	*camadas LSTM* = 2, *hidden_size* = 256, *activation* = 'tanh', *optimizer* = 'adam'	11	0.4632	0.8713

V. AVALIAÇÃO DOS RESULTADOS

Analisando os resultados após os experimentos, observou-se que a alteração dos parâmetros relacionados à função de ativação e tipo de otimizador, não provocaram resultados com melhora expressiva em relação ao primeiro experimento. A utilização de camadas LSTM bidirecionais também não causou impactos significativos no sentido de melhora dos valores de função de perda e acurácia média.

Experimentos adicionais foram conduzidos, alterando-se o número de células LSTM para 8, 16 e 32, porém apresentaram resultados com acurácia média inferiores a 60%.

A melhor configuração de hiperparâmetros foi obtida no experimento 7, onde foram utilizadas duas camadas LSTM, 256 células de LSTM nas camadas escondidas, ativação do tipo tangente hiperbólica e otimizador Adam. Para este experimento, foram obtidos os seguintes valores de média ponderada: *Precision:* 87%, *Recall:* 87% e *F1-Score:* 87%.

A partir da matriz de confusão gerada no experimento 7, apresentada na Figura 4, observa-se nas células destacadas em cinza, que 81 de um total de 3.373 documentos do conjunto de testes, foram classificados com temporalidade menor que aquela prevista para a respectiva amostra. Classificar incorretamente um documento com temporalidade menor do que a ideal tem grande impacto na qualidade do classificador proposto, podendo ocasionar o descarte prematuro de documentos oficiais.

É importante destacar que a classe "Enquanto Vigora" não foi considerada no somatório das células destacadas em cinza, por não ter uma quantidade de anos exata para a temporalidade.

Classe Predita \ Classe Real	Enquanto Vigora	1 ano	2 anos	3 anos	4 anos	5 anos	6 anos	7 anos
Enquanto Vigora	881	27	23	3	3	94	0	0
1 ano	16	288	15	0	0	25	0	0
2 anos	15	5	748	0	3	20	0	1
3 anos	0	0	0	2	0	1	0	0
4 anos	8	3	2	0	14	13	0	1
5 anos	68	33	41	0	2	992	0	2
6 anos	1	0	0	0	0	3	5	0
7 anos	0	0	0	0	0	7	0	8

Fig. 4. Matriz de Confusão para o experimento 7.

Os resultados obtidos demonstram que o modelo implementado pode ser utilizado como sistema de recomendação, auxiliando o usuário no momento da classificação do documento e possivelmente exibindo o nível de confiança da classificação proposta. Comparando com outros modelos considerados estado da arte em classificação de documentos [24] e [25], os resultados sugerem um potencial de uso para aplicações em produção.

VI. Conclusão

Neste artigo foi apresentada uma proposta para classificação de temporalidade de documentos do SIGADAER utilizando processamento de linguagem natural e um modelo de *Deep Learning* baseado em Rede Neural Recorrente do tipo LSTM.

Foram apresentados também os procedimentos adotados em cada uma das fases para a implementação do classificador. Primeiramente, a coleta de dados em um banco de dados relacional. Em seguida, na fase de pré-processamento, os documentos foram transformados e padronizados por meio de algoritmos de tokenização, remoção de *stop words*, remoção de pontuação e caracteres especiais, etc. Os documentos tiveram então seus tamanhos padronizados através de um algoritmo de *padding*. E por fim, os dados foram submetidos ao modelo de Rede Neural LSTM, passando pelas fases de treinamento, teste e avaliação dos resultados do classificador.

A utilização da rede LSTM produziu bons resultados para a classificação de temporalidade dos documentos. Com valores de acurácia média acima de 87%, este modelo tem grande potencial para servir de sistema de recomendação para os usuários no momento da escolha da classificação de temporalidade no SIGADAER. Cabe salientar, que a recomendação proposta não substitui a decisão do usuário e tem apenas o intuito de auxiliá-lo.

Os resultados obtidos nos experimentos mostram que a aplicação de técnicas de NLP e modelos de *Deep Learning* podem auxiliar os órgãos responsáveis pela salvaguarda de documentos oficiais da Aeronáutica de modo a garantir a eficácia na gestão documental, a racionalização e a transparência do seu fluxo.

Em trabalhos futuros, a fim de obter valores mais altos de acurácia e consequentemente ter um modelo com menor quantidade de documentos classificados com temporalidade abaixo da prevista, deseja-se explorar melhor as combinações de hiperparâmetros e até mesmo outros modelos de Redes Neurais, como as redes convolucionais e outras técnicas de aprendizado de máquina.

Referências

[1] A. N. B. C. N. de Arquivos, *Classificação, temporalidade e destinação de documento de arquivo: relativos às atividades-meio da administração pública*. Arquivo Nacional, 2001.

[2] J. Kaur and J. Saini, "A study of text classification natural language processing algorithms for indian languages," *VNSGU Journal of Science and Technology*, vol. 4, pp. 162–167, 07 2015.

[3] M. Dwarampudi and N. Reddy, "Effects of padding on lstms and cnns," *arXiv preprint arXiv:1903.07288*, 2019.

[4] L. Havrlant and V. Kreinovich, "A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation)," *International Journal of General Systems*, vol. 46, no. 1, pp. 27–36, 2017. [Online]. Available: https://doi.org/10.1080/03081079.2017.1291635

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[6] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[7] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[8] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[9] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[10] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[11] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[12] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997.

[17] C. Olah, "Understanding lstm networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[18] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[19] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[20] S. Visa, B. Ramsay, A. L. Ralescu, and E. Van Der Knaap, "Confusion matrix-based feature selection." *MAICS*, vol. 710, pp. 120–127, 2011.

[21] E. Loper and S. Bird, "Nltk: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.

[22] K. Ramasubramanian and A. Singh, "Deep learning using keras and tensorflow," in *Machine Learning Using R*. Springer, 2019, pp. 667–688.

[23] C. Olah, "Deep learning, nlp, and representations," *GitHub blog, posted on July*, vol. 7, 2014.

[24] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: Bert for document classification," 04 2019.

[25] G. Nikolentzos, A. J.-P. Tixier, and M. Vazirgiannis, "Message passing attention networks for document understanding," *arXiv preprint arXiv:1908.06267*, 2019.