SENADO FEDERAL

Secretaria de Tecnologia da Informação - Prodasen

Brasília, 20 de dezembro de 2019.

Ao Serviço de Fomento à Pesquisa da Coordenação de Educação Superior do Instituto Legislativo Brasileiro,

Na qualidade de Facilitador do Grupo de Estudos e Pesquisas Acadêmicas da Área de Conhecimento 07 (Tecnologia da Informação), instituído pelo processo seletivo do Edital ILB/COESUP nº 9/2018, encaminho os produtos acadêmicos gerados, conforme previsão das alíneas "i" e "j" do Termo de Compromisso do Facilitador (Anexo VIII do referido edital).

A proposta de estudo e pesquisa acadêmica, aprovada no processo seletivo, previa a entrega dos seguintes produtos: a) um artigo científico; b) código fonte do protótipo em software; c) apresentação do resultado da pesquisa no evento Conversando Tecnologia.

Encaminhamos o artigo científico no primeiro anexo a este documento; o código fonte na linguagem Python do protótipo em software, no segundo anexo; e a notícia da cobertura do evento Conversando Tecnologia, realizado no dia 19 de dezembro de 2019, às 10h00 no Prodasen, no terceiro anexo.

Dessa maneira, o GEPA atingiu todos os objetivos previstos, gerando os produtos planejados.

Agradecemos todo o apoio da COESUP no suporte ao desenvolvimento da pesquisa.

Atenciosamente,

_____

João Alberto de Oliveira Lima
Analista de Informática Legislativa – 10550-0

Aplicação da Inteligência Artificial na classificação de textos: o caso da triagem de solicitações à Consultoria Legislativa do Senado Federal

João Alberto de Oliveira Lima
Doutor em Direito e em Ciência da Informação pela Universidade Brasília.
<joaolima@senado.leg.br>

Fernando Luiz Brito de Melo
Pós-graduado em Ciência de Dados pela Johns Hopkins University (Baltimore, Estados Unidos) e Análise de Sistemas pela Universidade Católica de Brasília.
<fernando@senado.leg.br>

Francisco Edmundo de Andrade
Mestre em Ciência da Computação pela Universidade Federal do Ceará e MBA em Governança em TI no Setor Público pelo Centro Universitário do Distrito Federal.
<francisco.edmundo@camara.leg.br>

Klause Alvarenga do Nascimento
Pós-graduado em Gestão de Projeto de TI pela Faculdade AVM e Tecnólogo em Processamento de Dados pelo Centro de Ensino Superior Unificado de Brasília.
<klause@senado.leg.br>

Lauro César Araujo
Doutor em Ciência da Informação pela Universidade de Brasília.
<lauro.araujo@senado.leg.br>

Marcos Fragomeni Padron
Mestre em Ciência da Informação pela Universidade de Brasília.
<fragomeni@senado.leg.br>

Wagner Rodrigues Teixeira
MBA em Gestão Integrada de Projetos pelo Instituto Legislativo Brasileiro.
<wagner@senado.leg.br>

Pesquisadores do Instituto Legislativo Brasileiro do Senado Federal (ILB/Interlegis)
Edital COESUP Nº 9/2018, Seleção de Projetos para Formação de Grupos de Estudos e Pesquisas Acadêmicas. Área de Conhecimento: 07 - Tecnologia da Informação

Resumo

Somos testemunhas do rápido crescimento da Inteligência Artificial (IA) em vários cenários. No presente trabalho, apresentamos um estudo de como a IA pode realizar a classificação de textos no âmbito do Senado Federal. Como estudo de caso, pretendemos testar a IA no processo de triagem de solicitações à Consultoria Legislativa. Será testada a abordagem *Universal Language Model Fine-tuning* (ULMFiT), proposta por Howard e Ruder (2018), investigando variações na criação do *Language Model*, com o *corpus* da Wikipédia e a base de discursos de parlamentares, bem como variações de ajuste fino desse modelo, com *datasets* de solicitações enriquecidos de metadados. Dessa forma, pretende-se investigar se a utilização de textos específicos do domínio legislativo para o treinamento do *Language Model* é mais eficiente que o uso dos textos da Wikipédia, e se vale a pena enriquecer os dados de treinamento com metadados na etapa de *fine tuning*.

Palavras-chave: Informação legislativa. Processamento de linguagem natural. Classificação de Textos em Português. Inteligência Artificial. ULMFiT.

Application of AI in the classification of texts: the case of triage of requests to the Legislative Consulting of the Brazilian Federal Senate

Abstract

We are witnessing the rapid growth of Artificial Intelligence (AI) in various scenarios. In this paper, we present a study of how AI can perform the classification of texts within the Federal Senate. As a case study, we intend to test the AI in the process of triage of requests to the Legislative Consultancy. The Universal Language Model Fine-tuning (ULMFiT) method, proposed by Howard e Ruder (2018), will be tested, investigating variations in the creation of the Language Model, with the Wikipedia corpus and the parliamentary speech base, as well as variations in the fine-tuning of this model, with datasets of requests enriched by metadata. Thus, it is intended to investigate whether the use of texts specific to the legislative domain for the training of the Language Model is more efficient than the use of Wikipedia texts, and whether it is worth enriching the training data with metadata in the fine tuning stage.

Keywords: *Legislative Information. Natural Language Processing. Portuguese Text Classification. Artificial Intelligence. ULMFiT.*

1 Introdução

Apesar de possuir uma longa história na Ciência da Computação, que pode ser rastreada até a década de 1950[1], apenas recentemente a área de Inteligência Artificial (IA) tem experimentado um rápido avanço e desenvolvido aplicações para resolver tarefas cotidianas em diversos cenários, conforme os exemplos a seguir:

a) tradução de textos em múltiplos idiomas, tais como, *Google Translator*[2] e DeepL[3] que se utilizam de *Deep Neural Networks* (DNN) (SINGH et al., 2017);

[1] O termo "Inteligência Artificial" (*Artificial Intelligence*) foi cunhado por John McCarthy e Marvin Minksy em 1956 na conferência "*Dartmouth Summer Research Project on Artificial Intelligence*" ocorrida no Dartmouth College em New Hampshire (Estados Unidos) (HAENLEIN; KAPLAN, 2019, p. 2).
[2] https://translate.google.com/
[3] https://www.deepl.com/translator

b) reconhecimento e classificação de imagens com uma precisão maior que a do ser humano (GUO et al., 2016), inclusive na Medicina (LIU et al., 2017);

c) aplicação de *Deep Learning* na indústria farmacêutica na descoberta de novos medicamentos (CHEN et al., 2018).

Nos últimos anos, a aplicação de *Deep Learning* em várias tarefas de Processamento de Linguagem Natural (PLN) tem evoluído rapidamente (YOUNG et al., 2018, p. 55). De forma similar ao que ocorreu em 2012, quando as redes neurais convolucionais, combinadas com o método de transferência de aprendizado (*transfer learning*) estabeleceram um novo patamar de acurácia na classificação de imagens (RAZAVIAN et al., 2014), estamos testemunhando, desde 2018, similares avanços nas tarefas de PLN.

No contexto legislativo, em que a palavra se manifesta em documentos escritos em linguagem natural, são muitas as oportunidades de aplicação da IA. Por exemplo, uma atividade recorrente é a classificação de documentos de acordo com o seu conteúdo, tal como no caso das proposições legislativas em temas e assuntos, dos discursos parlamentares e da triagem das centenas de requisições realizadas à Consultoria Legislativa semanalmente.

O presente estudo irá tratar especificamente das Solicitações de Trabalhos à Consultoria (STCs). Essas requisições incluem pedidos de estudos, propostas de projetos de lei, discursos, minuta de pareceres, entre outros. A Consultoria Legislativa do Senado Federal é dividida em quatro núcleos temáticos para os quais são direcionados os pedidos, de acordo com o tema principal. Dessa forma, a triagem das solicitações é parte crítica para a agilidade do atendimento dos pedidos à consultoria.

Nesse contexto, este artigo apresenta os resultados da aplicação de aplicação de técnicas de PLN por meio da abordagem de aprendizado profundo (*deep learning*) da IA para a classificação de solicitações à Consultoria Legislativa do Senado Federal.

O restante deste artigo é organizado conforme a seguir: a seção 2 apresenta aspectos metodológicos englobando a justificativa, as hipóteses, o objetivo geral, os objetivos específicos e a abordagem metodológica; a seção 3 apresenta um resumo da arquitetura definida pela abordagem ULMFiT, detalhando os passos do processo de treinamento do modelo de linguagem; a seção 4 detalha a preparação dos *datasets* utilizados na criação do modelo de linguagem, bem como dos *datasets* utilizados na etapa de ajuste do modelo; a seção 5 apresenta os resultados obtidos na criação e ajuste dos modelos; a seção 6 analisa os resultados apresentados na seção anterior; e, por fim, na seção 7, apresentam-se as conclusões e sugestões de trabalhos futuros.

2 Aspectos Metodológicos

2.1 Justificativa

Em todas as áreas do conhecimento, os computadores são empregados de forma determinante nas atividades-fim. Por exemplo, na Engenharia, os engenheiros utilizam os computadores não apenas como editores de textos, mas também como máquinas capazes de realizar cálculos estruturais complexos e projetos gráficos com alta precisão. Essa configuração não se observa no processo legislativo. Apesar dos avanços, a utilização da capacidade computacional é ainda tímida e oferece muitas oportunidades.

As limitações impostas pela Emenda Constitucional nº 95/2016, conhecida também como a Emenda do Teto dos Gastos Públicos (BRASIL, 2016), dificultam a reposição do quadro de servidores nos vários órgãos da administração pública. Além disso, o atual cenário tende a se tornar mais crítico nos próximos

anos: segundo relatório do TCU, no acórdão nº 2779/2017, "chama a atenção a situação do Senado Federal, que nos vinte anos de vigência da Emenda, poderá ver seus quadros serem reduzidos até atingirem 22% do quantitativo de pessoal observado em 2016" (BRASIL, 2017, p. 22).

O princípio da eficiência na administração pública, inscrito no *caput* do art. 37 da Constituição Federal, é considerado por Hely Lopes Meirelles et al. (2007, p. 98) como "o mais moderno princípio da função administrativa, que já não contenta em ser desempenhada apenas com legalidade". Esforços empreendidos no sentido de contribuir com a eficiência na administração pública, pela otimização do uso dos recursos computacionais e consequente economia de tempo no tratamento de documentos legislativos nos sistemas de informação, justificam a presente investigação. A experiência adquirida neste estudo de caso será um modelo aplicável a outras necessidades de classificação de documentos, propiciando maior produtividade, qualidade e padronização, com melhor aproveitamento dos recursos humanos.

Diversos esforços foram dispendidos no aprimoramento da capacidade de tratamento da informação legislativa no Senado Federal, tais como resultados de pesquisa de seus colaboradores (LIMA, 2008; ARAUJO, 2017; LIMA, 2019) e de produção do Grupo de Estudo em Tecnologia da Informação Legislativa, instituído pelo Edital ILB 4/2016 (MARTIM; LIMA; ARAUJO, 2018; TEIXEIRA et al., 2019). Essas iniciativas têm influenciado a definição dos padrões especificados pelo Projeto LexML[4] e ferramentas derivadas desses padrões, como o Linker e o Parser, estão presentes no dia-a-dia da Casa, e são incorporados em diversos sistemas, tais como o Legis, Sedol, LexEdit, LexComp, LexOr e o Portal LexML. No entanto, nenhuma dessas iniciativas empregam técnicas de IA.

2.2 Hipóteses

Ao explorar os caminhos para implementar soluções que tornem mais eficiente atividades de classificação de textos do processo legislativo por meio da aplicação de técnicas de PLN com IA, deseja-se validar as seguintes hipóteses:

a) A utilização de textos específicos do domínio para o treinamento do *Language Model* no método ULMFiT (Passo 1) é mais eficiente que o uso dos textos em língua portuguesa da Wikipédia;

b) O enriquecimento de dados de treinamento com metadados relacionados ao processo legislativo pode melhorar a performance do classificador no método ULMFiT (Passos 2 e 3).

2.3 Objetivos

Estabelecemos um objetivo geral e seis objetivos específicos a serem atingidos para a consecução do objetivo geral e verificação das hipóteses, conforme a seguir.

2.3.1 Objetivo Geral

Investigar como a IA pode realizar o trabalho de classificação de textos no domínio legislativo.

2.3.2 Objetivos Específicos

a) Preparar *datasets* para criação do *Language Model* com textos provenientes da Wikipédia em português e das bases de dados de discursos dos parlamentares;

b) Preparar *datasets* para *fine tuning* e treinamento do classificador de texto utilizando variações de metadados das solicitações de trabalhos técnicos à Consultoria Legislativa;

[4] http://www.lexml.gov.br

c) Aplicar o método *Universal Language Model Fine-tuning* (ULMFiT) para classificar as solicitações de trabalhos técnicos à Consultoria Legislativa com o *Language Model* criado a partir de artigos em português da Wikipédia;

d) Aplicar o método *Universal Language Model Fine-tuning* (ULMFiT) para classificar solicitações de trabalhos técnicos à Consultoria Legislativa com o *Language Model* criado a partir da base de discursos dos parlamentares;

e) Elaborar estudo comparativo dos resultados da aplicação do método ULMFiT, ressaltando a performance dos modelos em função tanto do *corpus* do Passo 1 (pré-treinamento do modelo de linguagem) como do Passo 2 (*fine-tuning* do modelo de linguagem);

f) Produzir protótipo de classificação automática de solicitações à Consultoria Legislativa.

2.4 Abordagem Metodológica

Para a consecução do objetivo geral e o alcance dos objetivos específicos, será utilizada a abordagem metodológica Pesquisa-Ação, que se caracteriza por aliar um objetivo de pesquisa a um objetivo de ação (LIMA, 2007). Thiollent (1998, p. 14) define Pesquisa-Ação como "um tipo de pesquisa social com base empírica que é concebida e realizada em estreita associação com uma ação ou com a resolução de um problema coletivo". É importante ressaltar que a Pesquisa-Ação "está comprometida com a produção de novo conhecimento através da procura de solução ou melhoramentos de problemas práticos da vida real" (MCKAY; MARSHALL, 2001, p. 42).

Uma das principais características da Pesquisa-Ação é a repetição das suas etapas em um processo cíclico que transita ora no domínio da pesquisa, ora no da ação. Existe "um ciclo de retroalimentação no qual os achados iniciais geram possibilidades para mudanças, as quais são então implementadas e avaliadas como um início para mais uma investigação" (DENSCOMBE, 2014, p. 73).

Ainda segundo Denscombe (2014, p. 73-74), além de ter caráter prático e aplicado, a Pesquisa-Ação é participativa: as pessoas que estão envolvidas com o problema no mundo real são cruciais no processo de pesquisa. Para cada objetivo específico, são identificados os *stakeholders* que podem contribuir na solução do problema de pesquisa. Por exemplo, os servidores que realizam a triagem na Consultoria Legislativa foram consultados para melhor entendimento da tarefa de triagem de solicitações.

3 Arquitetura do ULMFiT e passos do processo de treinamento

De acordo com Howard e Ruder (2018), o método ULMFiT consiste na consecução de três passos, conforme a seguir:

a) Pré-treinamento do modelo de linguagem a partir de um *corpus* de domínio geral (Wikipédia, Discursos etc.);

b) *Fine-tuning* do modelo de linguagem a partir de um *corpus* específico para a tarefa alvo;

c) *Fine-tuning* do classificador especializado na tarefa alvo.

A Figura 1 apresenta esquematicamente o fluxo preconizado pelo método ULMFiT para geração do classificador de solicitações de trabalho, considerando experimentos baseado no *corpus* de artigos da Wikipédia. O mesmo fluxo foi aplicado para experimentos baseados no *corpus* de discursos parlamentares.

A arquitetura do ULMFiT (Figura 2), implementada na versão 1.0.58 da biblioteca *fast.ai*, é composta de três grupos de camadas:

Figura 1 – Passos do método ULMFiT - experimento com dados da Wikipédia em português.

Fonte: adaptado de fast.ai (2019).

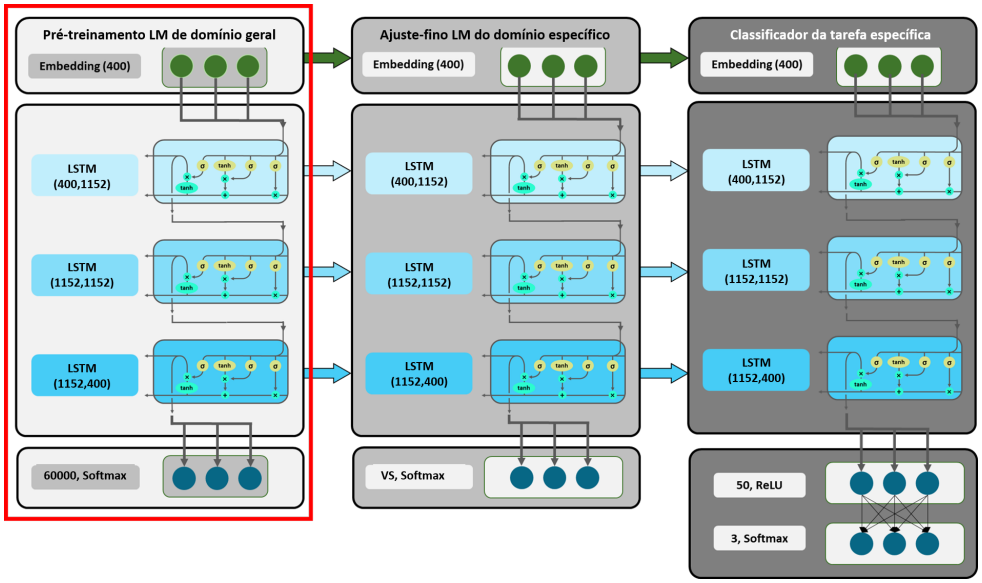
a) *embedding*, responsável por representar cada *token* do vocabulário em um vetor numérico de 400 dimensões;

b) *AWD-LSTM*, conforme Merity, Keskar e Socher (2017), com três camadas *LSTM*: LSTM-1 (400 x 1152), LSTM-2 (1152 x 1152), LSTM-3 (1152 x 400);

c) camada de saída, específica de cada passo.

Em todas as camadas, são aplicados módulos *dropout* conforme modelo AWD-LSTM (MERITY; KESKAR; SOCHER, 2017), para regularização do modelo e prevenção de *overfitting*.

Figura 2 – Arquitetura do método ULMFiT.



Fonte: adaptado de Faltl, Schimpke e Hackober (2019).

O treinamento de cada passo é realizado conforme a seguir:

a) Passo 1 (*language model pre-training*) – a camada de saída é um decodificador linear (400 x VS), onde VS é o tamanho do vocabulário (60.000 *tokens* para o presente trabalho).

b) Passo 2 – (*language model fine-tuning*): a camada de *embedding* é reconstruída de acordo com o vocabulário específico da tarefa alvo (entre 45.000 e 55.000 *tokens*, no caso das solicitações de trabalho). A camada de saída é um decodificador linear (400 x VS), onde VS é o tamanho do vocabulário específico.

c) Passo 3 – (*classifier fine-tuning*): a camada de *embedding* é a mesma do passo 2. A camada de saída é um classificador composto de um módulo de *batch normalization* (1200) com *dropout*, um módulo linear (1200 x 50) com ativação ReLU, outro módulo de *batch normalization* (50) com *dropout* e mais um módulo linear (50 x NC), onde NC é o número de classes utilizadas para classificar as solicitações de trabalho.

A transferência de aprendizado do passo 1 para o passo 2, e do passo 2 para o passo 3, se dá pela inicialização dos pesos com base nos valores obtidos pelo treinamento do modelo no passo anterior, com exceção da última camada. Durante o *fine-tuning*, as camadas finais são treinadas primeiro. Gradualmente, o treinamento se estende às camadas anteriores à medida que o *learning rate* vai sendo reduzido. Dessa forma, preserva-se boa parte do conhecimento apreendido nos passos anteriores.

4 Preparação dos datasets

4.1 Preparação dos Datasets para o Passo 1 (*language model pre-training*) - Wikipédia e Discursos

O passo 1 do ULMFiT requer um *corpus* de domínio geral extenso para que as camadas do *language model* resultante possam capturar as características gerais da linguagem alvo. Neste passo, o presente estudo estabelece como requisito mínimo para o tamanho do corpus de textos em língua portuguesa: 120 milhões de *tokens*[5]. A grandeza deste valor equipara-se à adotada pelo WikiText-103, um corpus de língua inglesa com 103.690.236 de *tokens*, amplamente utilizado para treinamento de *language models* (MERITY et al., 2016; HOWARD; RUDER, 2018).

Uma das principais hipóteses desta pesquisa é verificar, na abordagem descrita pela ULMFiT, se o *language model* gerado a partir dos textos da Wikipedia pode ser substituído de forma eficaz por um outro gerado a partir de textos de um domínio mais específico, baseado nos discursos parlamentares.

O *dataset* de textos da Wikipedia foi obtido a partir da opção de *bulk download*[6] de artigos existentes em português no *site*. O arquivo original, com data de 1 de agosto de 2019, possui 1.011.673 documentos com 249.110.165 *tokens*. Com o objetivo de ajustar o tamanho do *corpus* ao patamar estabelecido no presente estudo (120 milhões de *tokens*), eliminou-se os artigos mais curtos (menos de 500 caracteres) e os mais longos (mais de 6.050 caracteres). Verificou-se empiricamente que os artigos de menor tamanho não possuíam conteúdo relevante. Da mesma forma, os artigos de maior tamanho foram eliminados para prevenir algum tipo de viés relativo a domínios técnicos específicos[7]. Por esse motivo, o critério de seleção por tamanho de artigo foi escolhido como forma de excluir os extremos, priorizando os artigos com tamanho mais próximo à média. O *corpus* resultante totaliza 120.511.120 *tokens* distribuídos em 444.158 documentos com tamanho médio de 1.696 caracteres.

O *dataset* de discursos foi obtido a partir das bases de dados do Senado Federal e da Câmara dos Deputados. Foram selecionados 105.507 arquivos nos formatos RTF, DOC e DOCX com discursos do Senado Federal de 1994 até 2019, que, em seguida, foram processados por um programa na linguagem Java, utilizando a biblioteca Apache TIKA[8], para extração do texto sem formatação, remoção das quebras de linha e agregação em um único arquivo CSV com um discurso por linha em ordem cronológica. Eliminadas as duplicidades, chegou-se a 104.887 discursos distintos proferidos no Senado totalizando 133.200.472 *tokens*. No caso da Câmara dos Deputados, discursos proferidos entre 1988 e 2019 foram obtidos por meio

[5] Segmento de texto ou símbolo utilizado para representar a menor porção de texto a ser analisado ou processado
[6] https://dumps.wikimedia.org/ptwiki/20190801/ptwiki-20190801-pages-articles.xml.bz2
[7] De acordo com a Wikipédia, a página "Lista de exoplanetas descobertos usando a sonda Kepler" era composta por 837.839 bytes, em novembro de 2019, sendo considerada a mais longa em língua portuguesa.
[8] <https://tika.apache.org/>

da API (*application programming interface*) de dados abertos[9]. Eliminadas as duplicidades, chegou-se a 376.353 discursos distintos proferidos por Deputados Federais totalizando 180.243.176 *tokens*.

A junção dos discursos parlamentares da Câmara dos Deputados e do Senado Federal gerou uma base com 481.240 documentos distintos totalizando 313.443.648 *tokens*. Com o objetivo de ajustar o tamanho do *corpus* ao patamar estabelecido no presente estudo (120 milhões de *tokens*), eliminou-se os discursos mais curtos (menos de 900 caracteres) e os mais longos (mais de 5.630 caracteres). O *corpus* resultante totaliza 120.560.341 *tokens* distribuídos em 283.360 documentos com tamanho médio de 2.646 caracteres.

A especificidade do domínio do processo legislativo é percebida ao compararmos a quantidade de *tokens* únicos nos *corpora* resultantes da seleção de documentos relatada acima. O *corpus* de discursos possui 1.658.546 *tokens* únicos e o *corpus* da Wikipédia possui 3.920.685 *tokens* únicos, para uma quantidade similar de *tokens* em cada *corpus* (cerca de 120 milhões).

4.2 Preparação dos Datasets para os Passos 2, 3 e Teste (Solicitações de Trabalho à Consultoria Legislativa)

4.2.1 Contextualização

Servidores de gabinetes parlamentares, comissões e outros órgãos do Senado Federal solicitam trabalhos à Consultoria Legislativa por meio de sistema informatizado (SAC - Sistema de Apoio às Consultorias). Ao solicitá-los, o usuário informa, além do tipo de trabalho desejado, um texto descritivo e, opcionalmente, matérias legislativas relacionadas. São exemplos de tipos de trabalhos: Discurso, Estudo, Parecer, Projeto de Lei, Emenda a Projeto de Lei, Requerimento, entre outros.

Cada solicitação recebida pela Consultoria é triada por uma equipe e encaminhada a um núcleo temático especializado, que será o responsável pelo atendimento da demanda. A triagem baseia-se no assunto da solicitação. Os gestores dos núcleos podem ainda redirecionar para outro núcleo uma solicitação triada erroneamente. Este estudo considera a triagem de solicitações para os núcleos de Direito, Economia e Social, aos quais foram atribuídas as classes 0, 1 e 2 respectivamente. Adicionalmente aos núcleos mencionados, existe um tipo de trabalho específico que lida com as solicitações de redação de discursos. Este caso não é alvo deste estudo, pois a triagem é desnecessária por existir um núcleo específico para o atendimento desse tipo de demanda.

4.2.2 Procedimento

Para os *datasets* utilizados no treinamento e validação dos passos 2 e 3, foram selecionadas 88.239 solicitações de trabalho requisitadas entre 1º de janeiro de 2007 e 11 de setembro de 2019. Dentre essas solicitações, verificou-se que apenas 35.767 possuíam alguma proposição legislativa associada no momento da solicitação, o que equivale a 40,53% do total.

No caso dos *datasets* do passo de teste, foram selecionadas 1.184 solicitações entre 12 de setembro de 2019 e 8 de outubro de 2019.

Em todos os *datasets*, foram desconsideradas as solicitações canceladas bem como as triadas para o Núcleo de Pronunciamentos.

O enriquecimento dos metadados básicos de uma solicitação com dados adicionais relacionados à proposição informada pode afetar o desempenho do classificador. Para testar essa hipótese e identificar a

[9] https://dadosabertos.camara.leg.br/swagger/api.html

melhor composição de metadados de solicitações, foram criados os seguintes *datasets* de solicitações de trabalho à Consultoria Legislativa:

a) *Dataset* 1 – Tipo de solicitação e texto descritivo;

b) *Dataset* 2 – Tipo de solicitação, texto descritivo e indexação da primeira proposição relacionada;

c) *Dataset* 3 – Tipo de solicitação, texto descritivo e, para a primeira proposição informada, a ementa e explicação da ementa;

d) *Dataset* 4 – Tipo de solicitação, texto descritivo e, para a primeira proposição informada, a ementa e explicação da ementa apenas no caso do texto descritivo possuir menos de 50 caracteres;

e) *Dataset* 5 – Tipo de solicitação, texto descritivo e, para a primeira proposição informada, assunto geral e assunto específico relacionados à proposição.

Os *datasets* de solicitação foram produzidos em arquivos no formato CSV (*Comma Separated Values*) com duas colunas, sendo a primeira composta do código da classe correspondente ao núcleo de triagem e, a segunda, dos metadados concatenados.

A indexação informada no *Dataset* 2 é composta de descritores de assuntos atribuídos de forma manual provenientes do sistema de indexação de proposições legislativa, com o apoio do Tesauro do Senado Federal.

A explicação da ementa, presente nos *datasets* 3 e 4, é um texto cadastrado pela Assessoria Técnica da Secretaria-Geral da Mesa do Senado Federal que descreve, de forma mais detalhada do que a própria ementa, o conteúdo da proposição.

Já os campos Assunto Geral e Específico são também atribuídos às proposições legislativas pela Assessoria Técnica da Secretaria Geral da Mesa do Senado Federal e tem por objetivo definir categoria e subcategoria importantes para o processo legislativo em geral.

5 Resultados da Execução

A presente seção descreve as versões de software utilizadas, os hiperparâmetros escolhidos e os resultados obtidos após a execução dos passos do ULMFiT em um computador com Linux Ubuntu instalado, equipado com 32 GB de memória RAM e uma GPU (*graphics processing unit*) NVidia modelo GeForce GTX 1080 Ti com capacidade de armazenamento de 11 GB.

5.1 Versões dos Softwares Utilizados

Na execução de todos os procedimentos, utilizou-se as versões de *software* mais recentes, disponíveis à época:

a) Python : 3.7.4

b) Fast.ai : 1.0.57 (passo 1) e 1.0.58 (passos 2, 3 e teste)

c) Pytorch : 1.2.0

5.2 Hiperparâmetros

A Tabela 1 apresenta os valores dos hiperparâmetros de treinamento adotados na execução dos passos 1, 2 e 3 do ULMFiT. A política *1cycle* (SMITH, 2018) adotada acelera a convergência de treinamento por

meio da variação cíclica do *learning rate*[10] em sincronia com a variação cíclica do *momentum*[11]. Em cada época de treinamento, o *momentum* completa um ciclo de variação que começa com seu valor máximo (0,8), vai reduzindo gradualmente até seu valor mínimo (0,7) e passa a aumentar de valor até atingir novamente seu valor máximo (0,8).

No passo 1, utiliza-se 0,002 de *learning rate*, 64 para o *batch size*[12], zero para o hiperparâmetro *drop mult*[13] e 0,1 para *weight decay*[14]. Após dez épocas, o modelo pré-treinado resultante é salvo para futuras operações de *fine-tuning*.

No passo 2, o modelo pré-treinado no passo 1 é utilizado para *fine-tuning* com base no *corpus* do domínio alvo (solicitações à Consultoria), adotando-se um *batch size* de 128, com 1,0 para o *drop mult* e zero para o *weight decay*. Primeiramente, o *fine-tuning* opera sobre o último grupo de camadas por duas épocas, com *learning rate* máximo definido de acordo com a Tabela 1. Em seguida, o *fine-tuning* é aplicado em todas as camadas do modelo por oito épocas, com *learning rate* máximo reduzido para 10% do valor estabelecido no início do passo.

No passo 3, grupos de camadas do modelo resultante do passo 2 são utilizados para *fine-tuning* com base nos conjuntos de dados anotados do domínio alvo (solicitações à Consultoria), adotando-se um *batch size* de 32, com 0,5 para o *drop mult* e zero para o *weight decay*. Primeiramente, o *fine-tuning* opera sobre o último grupo de camadas por duas épocas, com *learning rate* máximo definido de acordo com a Tabela 1. Em seguida, o *fine-tuning* é aplicado apenas nos dois últimos grupos de camadas por duas épocas, com *learning rate* máximo distinto para cada grupo de camadas de acordo com o método *Discr* – *fine-tunning* discriminativo (HOWARD; RUDER, 2018). Depois, o método *Discr* é aplicado sobre os três últimos grupos de camadas por duas épocas, reduzindo-se o *learning rate* máximo pela metade do valor estabelecido no início do passo. Por fim, aplica-se o *Discr* sobre todas as camadas por uma época, reduzindo-se o *learning rate* máximo pela décima parte do valor estabelecido no início do passo.

Tabela 1 – Hiperparâmetros adotados para os passos 1, 2, 3 do ULMFiT

	Passo 1	Passo 2	Passo 3
Batch size (BS)	64	128	32
Maximum learning rate (MLR)	0.002	$0.01 \cdot \frac{BS}{48}$	$0.02 \cdot \frac{BS}{48}$
Learning rate for 1cycle policy	$\frac{MLR}{25} \nearrow MLR \searrow \frac{MLR}{250000}$	$\frac{MLR}{25} \nearrow MLR \searrow \frac{MLR}{250000}$	$\frac{MLR}{25} \nearrow MLR \searrow \frac{MLR}{250000}$
Momentum for 1cycle policy	$0.8 \searrow 0.7 \nearrow 0.8$	$0.8 \searrow 0.7 \nearrow 0.8$	$0.8 \searrow 0.7 \nearrow 0.8$
Drop mult	0.0	1.0	0.5
Weight decay	0.1	0.0	0.0
Backpropagation through time	70	70	70
Epochs training only last layer group	-	2	2
Epochs training only last two groups	-	-	2
Epochs training only last three groups	-	-	2
Epochs training all layer groups	10	8	1

Fonte: Produzido pelos autores.

[10] Hiperparâmetro que controla o ritmo de aprendizagem do modelo durante o treinamento, dimensionando o tamanho do passo de atualização dos pesos da rede neural.
[11] *Momentum* permite ponderar o efeito do gradiente de iterações anteriores sobre a atualização dos pesos na iteração atual.
[12] Hiperparâmetro que controla a quantidade de exemplos do *dataset* considerados a cada atualização dos pesos da rede neural.
[13] Fator multiplicador aplicado sobre todos os parâmetros de *dropout* do modelo. *Dropout* é uma das técnicas utilizadas para regularização do modelo com o objetivo de evitar *overfitting* e melhorar seu nível de generalização.
[14] *Weight decay* é uma técnica de regularização que restringe a magnitude dos pesos para prevenir *overfitting*.

5.3 Resultados Obtidos

A Tabela 2 apresenta os resultados da geração dos modelos de linguagem da Wikipédia e da base de discursos com os seguintes indicadores de desempenho: *Train loss*, *Valid loss*, *Accuracy* e *Perplexity*.

Tabela 2 – Resultados da execução do passo 1 do ULMFiT

	Passo 1			
ML	Train loss	Valid loss	Accuracy	Perplexity
Wikipédia	2.947	2.970	0.443	19.499
Discurso	2.742	2.789	0.447	16.270

Fonte: Produzido pelos autores.

A Tabela 3 apresenta os resultados das execuções dos passos 2, 3 e teste para cada um dos cinco *datasets* e para cada um dos modelos de linguagem (Wikipédia e Discursos). São apresentados os seguintes indicadores de desempenho: *Train loss* (Passo 2 e 3), *Valid loss* (Passo 2 e 3), *Accuracy* (Passo 2, 3 e teste), *Perplexity* (Passo 2) e F1-score[15] (teste).

Tabela 3 – Resultados da execução dos passos 2, 3 do ULMFiT e teste

		Passo 2				Passo 3			Teste	
ML	DS	Train loss	Valid loss	Accuracy	Perplexity	Train loss	Valid loss	Accuracy (Validação)	Accuracy	F1-score
Wikipédia	1	1.883	2.013	0.581	7.486	0.368	0.457	0.820	0.735	0.735
	2	1.718	1.781	0.631	5.934	0.304	0.387	0.854	0.805	0.800
	3	1.799	1.939	0.596	6.951	0.338	0.442	0.824	0.758	0.760
	4	1.854	2.014	0.582	7.492	0.328	0.453	0.818	0.757	0.760
	5	1.854	1.989	0.586	7.310	0.324	0.432	0.837	0.763	0.760
Discursos	1	1.827	1.992	0.583	7.334	0.328	0.451	0.825	0.747	0.750
	2	1.685	1.762	0.632	5.826	0.296	0.382	0.858	0.802	0.801
	3	1.765	1.918	0.598	6.808	0.343	0.444	0.826	0.766	0.766
	4	1.817	1.994	0.583	7.346	0.330	0.450	0.822	0.762	0.761
	5	1.816	1.969	0.587	7.166	0.321	0.414	0.841	0.774	0.773

Fonte: Produzido pelos autores.

A Figura 3 apresenta a Matriz de Confusão referente à execução do modelo de classificação (Passo 3) que obteve o melhor resultado *F1-score*, em que o eixo vertical apresenta os valores reais dos dados de validação para as classes 0 (Direito), 1 (Economia) e 2 (Social) e o eixo horizontal apresenta os valores de predição do modelo de classificação.

5.4 Protótipo do Classificador

O protótipo do classificador de solicitações à Consultoria Legislativa está disponível no endereço <https://classificador-consultoria.herokuapp.com/>.

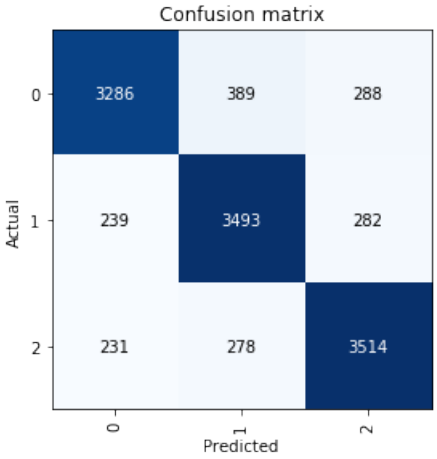
No desenvolvimento, utilizou-se a linguagem Python e biblioteca Flask. A interface solicita um texto de solicitação de trabalho e, como resultado, apresenta as probabilidades de cada classe, conforme exemplo apresentado na Figura 4.

6 Discussão sobre os Resultados

A acurácia superior a 82% das diferentes versões de modelos de classificação (Passo 3), apresentada na Tabela 3, confirma resultados de Howard e Ruder (2018) segundo os quais são necessários poucos

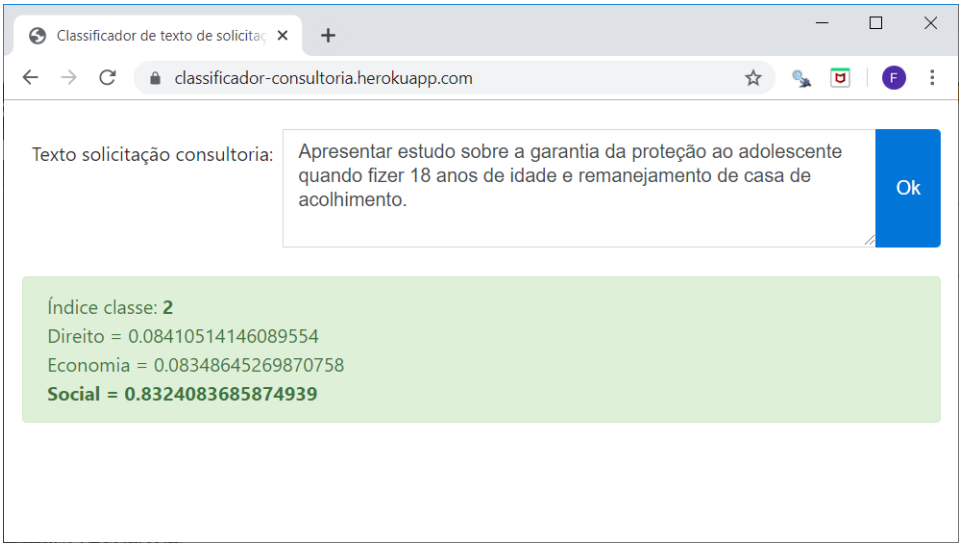
[15] Também conhecida como *F-Measure*, a *F1-score* é a média harmônica de precisão (*precision*) e revocação (*recall*).

Figura 3 – Matriz de Confusão da melhor performance (Discursos - DS2)



Fonte: Produzido pelos autores.

Figura 4 – Protótipo do classificador de solicitações



Fonte: Produzido pelos autores.

exemplos rotulados para se obter uma taxa de classificação aceitável utilizando o método ULMFiT. Comparativamente, o ser humano treinado realiza o mesmo trabalho com acurácia de 91%[16], ou seja, os testes nos modelos treinados alcançaram pelo menos 90% do desempenho humano.

Observa-se que o treinamento dos modelos segundo o método ULMFiT pode ser realizado com recursos de hardware convencionais, bastando uma placa GPU moderna, a exemplo do sistema utilizado nesta pesquisa (seção 5). A criação de cada modelo de linguagem (Passo 1) consome cerca de 20 horas, a execução do *fine-tuning* dos modelos (Passo 2) consome cerca de 70 minutos e o treinamento do classificador (Passo 3) consome cerca de 15 minutos.

[16] A acurácia humana é calculada pelo total de solicitações não redirecionadas a outros núcleos após triagem inicial. Dados obtidos do sistema de informação SAC referente ao período dos *datasets*.

Os resultados demonstraram que o pré-treinamento (Passo 1) realizado com um *corpus* específico do domínio, os discursos de parlamentares, não leva a resultados significativamente melhores do que o uso de *corpus* genérico, a Wikipédia, considerando todos os *datasets* de teste. Ao comparar as acurácias da validação (Passo 3), confirma-se a hipótese de que a utilização de textos específicos do domínio para o treinamento do *Language Model* no método ULMFiT (Passo 1) é mais eficiente que o uso dos textos em língua portuguesa da Wikipédia, uma vez que o modelo de linguagem específico do domínio (Discursos Parlamentares) teve uma melhoria de performance entre 0.2% e 0.5%. No entanto, a pequena diferença não é significativa para justificar o custo adicional da obtenção de um grande *corpus* para treinamento de um modelo de linguagem específico de domínio, uma vez que a Wikipédia é um *corpus* disponível, gratuito e cobre amplo espectro de domínios e assuntos, o que é suficiente para capturar propriedades gerais da linguagem. Portanto, isso novamente confirma a proposta de Howard e Ruder (2018), em que a abordagem *universal* do ULMFiT funciona adequadamente, utilizando uma mesma arquitetura e processo de treinamento, sem a necessidade de um *corpus* específico de domínio.

Em relação à segunda hipótese, de que o enriquecimento de dados de treinamento com metadados relacionados ao processo legislativo pode melhorar a performance do classificador no método ULMFiT (Passos 2 e 3), os resultados apresentados na Tabela 3 demonstram uma significativa diferença entre o pior e o melhor resultados de acurácia (Passo 3) considerando os dois modelos de linguagem. Tanto no caso da Wikipédia como no caso de discursos, o enriquecimento de dados com metadados adicionais proporcionou uma variação positiva de 3,6%, o que confirma a hipótese. O *dataset* 2, enriquecido com metadados de indexação da proposição legislativa relacionada, apresentou o melhor resultado. Vale a pena ressaltar que esse resultado poderia ter sido ainda melhor considerando que o enriquecimento foi aplicado em apenas cerca de 40% das requisições, conforme explicitado no item 4.2.2. Outros enriquecimentos utilizando metadados foram tentados utilizando-se ementa, explicação de ementa e assunto relacionado à proposição legislativa eventualmente associada ao texto da solicitação (*Datasets* 3 a 5). Em todos os casos, os resultados foram inferiores aos do *dataset* enriquecido com base na indexação da primeira proposição relacionada (*Dataset 2*). Isso ressalta a importância do uso de indexação das proposições.

7 Conclusão

Este trabalho demonstrou a viabilidade da aplicação de técnicas de PLN por meio da abordagem de aprendizado profundo (*deep learning*) da IA no processo legislativo por meio da classificação de solicitações à Consultoria Legislativa do Senado Federal. Dessa forma, atingiu-se o objetivo geral do estudo.

De forma semelhante, os objetivos específicos foram atingidos, conforme a seguir:

a) A preparação dos *datasets* previstos nos itens a) e b) da subseção 2.3.2 foi realizada conforme exposto na seção 4;

b) A execução do método ULMFiT, utilizando os dois modelos de linguagem, como previsto nos itens c) e d) da subseção 2.3.2 foi realizada conforme exposto na subseção 4.2.2, com os resultados apresentados na seção 5;

c) O estudo comparativo da aplicação do método ULMFiT, previsto no Item e) da subseção 2.3.2, foi apresentado na seção 5;

d) O protótipo do classificador de solicitações de trabalho à Consultoria Legislativa, previsto no Item f) da subseção 2.3.2, foi apresentado na subseção 5.4.

O presente estudo confirmou as duas hipóteses apresentadas na subseção 2.2. Há de se ressaltar que, no caso da primeira hipótese, o ganho do uso de um corpus específico do domínio para criação do modelo

de linguagem (Passo 1) não apresentou resultados significativos. Já no caso da segunda hipótese, os resultados demonstram que o enriquecimento com dados relacionados ao processo legislativo melhora a performance do classificador no método ULMFiT. Especificamente, o uso da indexação das proposições acrescenta ganho de performance de 3,6% em relação ao uso de textos de solicitações sem enriquecimento. O uso de outros metadados, como ementa, explicação da ementa e assunto, não apresentaram melhoras significativas. Esses resultados demonstram a importância do conhecimento tácito de especialistas para identificação dos elementos do domínio capazes de otimizar a qualidade dos classificadores automáticos.

Considerando a precisão de 90% do modelo em relação ao trabalho humano, observa-se que há potencial para o protótipo construído realize o trabalho repetitivo como suporte à tarefa classificação e triagem de solicitações à consultoria.

7.1 Limitações e trabalhos futuros

Identificamos algumas oportunidades para investigação futura, conforme abaixo:

a) Verificar a aplicação da mesma abordagem para outros casos do domínio legislativo: classificação de proposições, normas e discursos parlamentares;

b) Treinar os modelos de forma bi-direcional e calcular a predição com base na média aritmética de ambos. Verifica-se que as palavras de uma frase possuem relação entre elas, tanto com as palavras posteriores quanto com as anteriores. Esta técnica simples resulta na melhora da performance do classificador.

c) Treinar os modelos usando o método MultFiT (EISENSCHLOS et al., 2019), uma evolução do método ULMFiT. O método MultFiT foi divulgado recentemente, dia 10 de setembro de 2019, e destaca-se pelo uso do *tokenizador Sentence Piece* que trabalha a nível de sub-palavras e utiliza uma arquitetura de rede neural diferente, *Quasi-Recurrent Neural Networks (QRNNs). QRNNs* são mais rápidas para treinar e conseguem performance similar.

Agradecimentos

Agradecemos à Consultoria Legislativa do Senado Federal, pela cessão dos dados utilizados e apoio durante o desenvolvimento desta pesquisa.

Agradecemos ao Instituto Legislativo Brasileiro, pela oportunidade de desenvolver trabalho pioneiro no Senado Federal em IA no processo legislativo.

Agradecemos ao Prodasen, à Quarta-Secretaria do Senado Federal, ao Gabinete do Senador Izalci Lucas e à Câmara dos Deputados, por possibilitar a participação dos servidores no Grupo de Estudos e Pesquisa Acadêmica da área de Tecnologia Informação.

Agradecemos a contribuição de André Ferrari, Francisco Alberto e Fabrício Santana na discussão técnica e sugestões ao texto.

Referências

ARAUJO, L. C. *Uma Linguagem para Formalização de Discursos com base em Ontologias*. Brasília: Senado Federal, 2017. (Coleção de Teses, Dissertações e Monografias de Servidores do Senado Federal).

ISBN 978-85-7018-902-8.

BRASIL. Emenda Constitucional nº 98, de 6 de dezembro de 2017. Altera o Ato das Disposições Constitucionais Transitórias, para instituir o Novo Regime Fiscal, e dá outras providências. *Diário Oficial [da] República Federativa do Brasil*, Poder Executivo, Brasília, p. 2, 16 dez. 2016.

BRASIL. Acórdão número 2779/2017, plenário. tc 014.133/2017-2. Tribunal de Contas da União, Brasília, 2017. Disponível em: <https://contas.tcu.gov.br/sagas/SvlVisualizarRelVotoAcRtf?codFiltro=SAGAS-SESSAO-ENCERRADA&seOcultaPagina=S&item0=606182>. Acesso em: 22 ago. 2019.

CHEN, H. et al. The rise of deep learning in drug discovery. *Drug discovery today*, Elsevier, v. 23, n. 6, p. 1241–1250, 2018.

DENSCOMBE, M. *The good research guide: for small-scale social research projects*. [S.l.]: McGraw-Hill Education (UK), 2014.

EISENSCHLOS, J. et al. Multifit: Efficient multi-lingual language model fine-tuning. *CoRR*, abs/1909.04761, 2019. Disponível em: <http://arxiv.org/abs/1909.04761>. Acesso em: 6 dez. 2019.

FALTL, S.; SCHIMPKE, M.; HACKOBER, C. *Universal Language Model Fine-Tuning (ULMFiT): State-of-the-Art in Text Analysis*. 2019. Disponível em: <https://humboldt-wi.github.io/blog/research/information_systems_1819/group4_ulmfit/#2-2-example-of-a-forward-pass-through-the-lm-a-class-anchor-id-forwardpass-a>. Acesso em: 6 dez. 2019.

FAST.AI. *fast.ai NLP*. 2019. Disponível em: <http://nlp.fast.ai/>. Acesso em: 6 dez. 2019.

GUO, Y. et al. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 87–102.

HAENLEIN, M.; KAPLAN, A. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, SAGE Publications Sage CA: Los Angeles, CA, v. 61, n. 4, p. 5–14, 2019.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. *CoRR*, abs/1801.06146, 2018. Disponível em: <http://arxiv.org/abs/1801.06146>. Acesso em: 22 ago. 2019.

LIMA, J. A. d. O. Pesquisa-ação em ciência da informação. *Métodos para a pesquisa em Ciência da Informação. Brasília: Thesaurus*, p. 63–82, 2007.

LIMA, J. A. de O. *Modelo Genérico de Relacionamentos na Organizaçao da Informaçao Legislativa e Jurıdica*. Tese (Doutorado) — PhD thesis, Departamento de Ciência da Informaçao e Documentaçao , Universidade de Brasília., 2008.

LIMA, J. A. de O. *Consolidação de Normas Jurídicas: encontro entre Direito, Ciência da Informação, Filosofia da Linguagem e Lógica, a convite do neoinstitucionalismo*. Tese (Doutorado) — PhD thesis, Faculdade de Direito, Universidade de Brasília., 2019.

LIU, Y. et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

MARTIM, H. de; LIMA, J. A. de O.; ARAUJO, L. C. Base de normas jurídicas brasileiras: uma iniciativa de open government data. *Perspectivas em Ciência da Informação*, v. 23, n. 4, p. 133, 2018. Disponível em: <http://portaldeperiodicos.eci.ufmg.br/index.php/pci/article/view/3567>. Acesso em: 24 jan. 2019.

MCKAY, J.; MARSHALL, P. The dual imperatives of action research. *Information Technology & People*, MCB UP Ltd, v. 14, n. 1, p. 46–59, 2001.

MEIRELLES, H. L. et al. *Direito administrativo brasileiro*. [S.l.]: Revista dos Tribunais, 2007. v. 3.

MERITY, S.; KESKAR, N. S.; SOCHER, R. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182, 2017. Disponível em: <http://arxiv.org/abs/1708.02182>. Acesso em: 6 dez. 2019.

MERITY, S. et al. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. Disponível em: <http://arxiv.org/abs/1609.07843>. Acesso em: 6 dez. 2019.

RAZAVIAN, A. S. et al. Cnn features off-the-shelf: an astounding baseline for recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. [S.l.: s.n.], 2014. p. 806–813.

SINGH, S. P. et al. Machine translation using deep learning: An overview. In: IEEE. *2017 International Conference on Computer, Communications and Electronics (Comptelix)*. [S.l.], 2017. p. 162–167.

SMITH, L. N. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. Disponível em: <http://arxiv.org/abs/1803.09820>. Acesso em: 11 dez. 2019.

TEIXEIRA, W. R. et al. Exemplo de extração de definições em textos articulados de normas jurídicas com o apoio do processamento de linguagem natural. *Cadernos de Informação Jurídica*, v. 6, n. 1, p. 49–64, 2019. Disponível em: <https://www.cajur.com.br/index.php/cajur/article/view/212/240>. Acesso em: 19 ago. 2019.

THIOLLENT, M. *Metodologia da pesquisa-ação*. [S.l.]: São Paulo: Cortez, 1998.

YOUNG, T. et al. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, IEEE, v. 13, n. 3, p. 55–75, 2018.

Código fonte do protótipo em software

```
import asyncio
import uvicorn
from fastai import *
from fastai.text import *
from io import BytesIO
from starlette.applications import Starlette
from starlette.middleware.cors import CORSMiddleware
from starlette.responses import HTMLResponse, JSONResponse,
   PlainTextResponse
from starlette.staticfiles import StaticFiles
from clftriagemstc import model
import logging

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)-15s %(levelname)-8s %(message)s"
)

path = Path(__file__).parent

app = Starlette()
app.add_middleware(CORSMiddleware, allow_origins=['*'], allow_headers=['X-
   Requested-With', 'Content-Type'])
app.mount('/static', StaticFiles(directory=path / 'static'))

app_prefix = Starlette()
app_prefix.mount("/triagemstc", app=app)

async def setup_learner():
    learn = model.load_classifier()
    logging.info("Classes: %s", str(learn.data.classes))
    return learn

loop = asyncio.get_event_loop()
tasks = [asyncio.ensure_future(setup_learner())]
learn = loop.run_until_complete(asyncio.gather(*tasks))[0]
loop.close()

@app.route('/')
async def homepage(request):
    html_file = path / 'static' / 'index.html'
    return HTMLResponse(html_file.open().read())


@app.route('/predict', methods=['POST'])
async def analyze(request):
    body = await request.body()
    text_data = body.decode()

    prediction = learn.predict(text_data)

```

```
    idx_classe = prediction[1].item()

    probs = [{ 'classe': learn.data.classes[i], 'probabilidade':
   prediction[2][i].item() } for i in range(len(prediction[2]))]

    result = {
        'idx_classe': idx_classe,
        'nome_classe': learn.data.classes[idx_classe],
        'probabilidade': prediction[2][idx_classe].item(),
        'lista_prob': probs
    }
    return JSONResponse({'result': result})

@app.route('/sfstatus/ping')
async def status_ping(request):
    return PlainTextResponse('ok')

if __name__ == '__main__':
    port = int(os.getenv('PORT', 5042))
    uvicorn.run(app=app_prefix, host='0.0.0.0', port=port)
```

Código 1: Código em *python* da parte servidora

```
<!DOCTYPE html>
<html lang="pt">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>Classificador para triagem de solicitações à Consultoria
   Legislativa</title>

  <meta name="description" content="Página para teste do serviço do
   classificador de solicitações à Consultoria Legislativa">
  <meta name="author" content="Senado Federal">

  <link rel="icon" href="data:;base64,iVBORw0KGgo=">

  <link href="static/css/bootstrap.min.css" rel="stylesheet">
  <link href="static/css/style.css" rel="stylesheet">

  <script src="static/js/jquery.min.js"></script>
  <script src="static/js/tether.min.js"></script>
  <script src="static/js/bootstrap.min.js"></script>

  <script type="text/javascript">

    function getInferSuccess(rsp) {
      $('#progressBar').hide();

      var html_result = "Índice classe: <strong>" + rsp.result.idx_classe
   + "</strong>";
```

```
29
30      for (i=0; i<rsp.result.lista_prob.length;i++) {
31        tmp = rsp.result.lista_prob[i].classe + " = " + rsp.result.
    lista_prob[i].probabilidade;
32        if (i == rsp.result.idx_classe) {
33          tmp = "<strong>" + tmp + "</strong>";
34        }
35        tmp = "</br>" + tmp;
36        html_result += tmp;
37      }
38
39      $("#divResultado").html(html_result);
40      $('#divResultado').show();
41    }
42
43    function getInferError(rsp, status, errorThrown) {
44      $('#divAlertErro').html("Erro! Status " + status);
45      $('#divAlertErro').show();
46      $('#progressBar').hide();
47    }
48
49    function btnOkClick() {
50
51      $('#divAlertErro').hide();
52      $('#divResultado').hide();
53      $('#progressBar').show();
54
55      param_texto = $('#textoReqConsultoria').val();
56
57      $.ajax('predict', {
58        type: "POST",
59        data: param_texto,
60        success: getInferSuccess,
61        error: getInferError
62      });
63
64      return false;
65    }
66  </script>
67
68 </head>
69
70 <body>
71
72   <div class="container-fluid">
73     <div class="row mb-4">
74       <h1 class="bd-title">Classificador para triagem de solicitações à
    Consultoria Legislativa</h1>
75     </div>
76     <div class="row with-margin mb-4">
77
78       <div class="col-lg-8">
79         <form id="formPredict" action="/none" onsubmit="return false;">
80           <div class="input-group">
```

```
            <label for="textoReqConsultoria" class="mx-sm-2 pt-2">
              Texto da solicitação:
            </label>
            <textarea class="form-control" id="textoReqConsultoria" rows="
    4"></textarea>
            <span class="input-group-btn">
              <button type="submit" class="btn btn-primary" id="btnOk"
    onclick="btnOkClick()">Ok</button>
            </span>
          </div>
        </form>

      </div>

    </div>
    <div class="alert alert-danger" role="alert" id="divAlertErro" style="
    display: none">
      Erro
    </div>
    <div class="alert alert-success" role="alert" id="divResultado" style=
    "display: none">

    <div class="progress">
      <div id="progressBar" class="progress-bar progress-bar-striped
    progress-bar-animated" role="progressbar" aria-valuenow="100" aria-
    valuemin="0" aria-valuemax="100" style="width: 100%; display: none"></
    div>
    </div>

    </div>
  </div>

</body>

</html>
```

Código 2: Código em *html* da parte cliente

Joao Alberto de Oliveira Lima

De: Comunicação Interna do Senado Federal
Enviado em: quinta-feira, 19 de dezembro de 2019 17:33
Para: Comunicação Interna do Senado Federal
Assunto: Comunicados e Notícias

Boletim Diário

Comunicação Interna

Quinta-feira, 19 de dezembro de 2019

Software de Inteligência Artificial faz triagem de pedidos à Consultoria

Equipe de Relações Públicas visita setores para agradecer parcerias ao longo do ano

COMUNICADOS

O restaurante Peixe na Rede, no Espaço do Servidor, informa que nesta sexta (20) encerrará suas atividades às 15h para confraternização de final de ano da equipe.

Liga do Bem entrega presentes em lar de idosos de Planaltina

Palavra do Chef: Restaurante dos Senadores oferece o cardápio 'Feliz Natal'

CARDÁPIOS | ANIVERSÁRIOS | CLASSIFICADOS

Software de Inteligência Artificial faz triagem de pedidos à Consultoria Legislativa

19/12/2019, 15h34 - ATUALIZADO EM 19/12/2019, 15h40

O evento Conversando Tecnologia contou com a presença de representantes do Prodasen, co[illegible]mação Legislativa (Sinfleg) e de outros órgãos públicos

Jonas Araújo/Núcleo de Intranet

Um software de Inteligência Artificial (IA) que faz a triagem inicial das solicitações de serviço que chegam à Consultoria Legislativa (Conleg) foi apresentado nesta quinta-feira (19) no evento Conversando Tecnologia, realizado na Secretaria de Tecnologia da Informação (Prodasen). A ferramenta, prevista para entrar em operação em fevereiro do ano que vem, foi desenvolvida por colaboradores do Senado que formam o Grupo de Estudo e Pesquisa Acadêmica da Área de Tecnologia da Informação (GEPA/TI).

Por semana, centenas de solicitações de serviço chegam à Conleg, que as direciona para três grandes núcleos: Social, Direito e Economia. O software desenvolvido pelo GEPA analisa a ementa da solicitação para definir, em termos percentuais, para qual núcleo a solicitação está mais relacionada. Para tanto, a ferramenta faz uso do ULMFit, um método de processamento de textos por IA surgido em maio do ano passado nos meios acadêmicos dos Estados Unidos.

Conforme Fernando Melo, servidor da Quarta Secretaria e participante do GEPA, esse método revolucionou a forma como a IA pode ser utilizada, oferecendo uma metodologia muito mais operacional.

— Estudos internacionais indicam que a área de processamento de textos é a maior demanda do poder público —, acentuou Fernando.

Segundo João Alberto Lima, facilitador do GEPA, a intenção do grupo foi trabalhar com propostas exequíveis, a fim de em etapas futuras migrar para projetos mais ambiciosos. Ele ressaltou que avanços recentes permitiram que a IA deixasse de ser apenas uma possibilidade para agora entregar resultados práticos com mais

frequência.

— O grupo de pesquisa permite fazer essa integração entre o meio acadêmico e a atividade legislativa, resultando em benefícios para as duas áreas —, argumentou João Alberto, que é chefe do Serviço de Soluções para Informação Legislativa e Jurídica (Seleju), da Coordenação de Informática Legislativa e Parlamentar (Colep).

Iniciativa

A reunião do Conversando Tecnologia contou com a presença de representantes do Prodasen, Conleg, Serviço de Informação Legislativa (Sinfleg), Câmara dos Deputados, Tribunal de Contas da União, Tribunal Superior do Trabalho, Agência Nacional de Aviação Civil e Presidência da República. Presente ao evento, o diretor do Prodasen, Alessandro Albuquerque, elogiou a iniciativa dos servidores em investirem esforços em IA.

— É um setor que passa por grande evolução nos últimos anos e cada vez mais está presente no dia a dia das pessoas e também no Poder Legislativo — afirmou Alessandro.

O GEPA foi instituído por processo seletivo do edital nº 9/2018, da Coordenação de Educação Superior (Coesup), do Instituto Legislativo Brasileiro (ILB). O grupo de pesquisa é formado por servidores do Prodasen, de gabinetes parlamentares, da Quarta Secretaria e da Câmara dos Deputados.

EDITORIAS:

Tecnologia