

LEONARDO AUGUSTO DA SILVA PACHECO

**CLASSIFICAÇÃO AUTOMÁTICA DE ACÓRDÃOS A PARTIR
DE ENUNCIADOS DA JURISPRUDÊNCIA SELECIONADA:
Análise comparativa de técnicas e fontes de dados**

Brasília

2020

LEONARDO AUGUSTO DA SILVA PACHECO

**CLASSIFICAÇÃO AUTOMÁTICA DE ACÓRDÃOS A PARTIR
DE ENUNCIADOS DA JURISPRUDÊNCIA SELECIONADA:
*Análise comparativa de técnicas e fontes de dados***

Trabalho de conclusão do curso de pós-graduação
lato sensu de Especialização em Análise de Dados
para o Controle, realizado pela Escola Superior do
Tribunal de Contas da União como requisito para a
obtenção do título de especialista.

Orientador: Prof. Dr. Thiago Paulo Faleiros

Brasília

2020

REFERÊNCIA BIBLIOGRÁFICA

PACHECO, Leonardo. **Classificação Automática de Acórdãos a Partir de Enunciados da Jurisprudência Seleccionada**: análise comparativa de técnicas e fontes de dados. 2020. Trabalho de Conclusão de Curso (Especialização em Análise de Dados para o Controle) – Escola Superior do Tribunal de Contas da União, Instituto Serzedello Corrêa, Brasília DF. 76 fl.

CESSÃO DE DIREITOS

NOME DO AUTOR: Leonardo Augusto da Silva Pacheco

TÍTULO: Classificação Automática de Acórdãos a Partir de Enunciados da Jurisprudência Seleccionada

GRAU/ANO: Especialista/2020

É concedido ao Instituto Serzedello Corrêa (ISC) permissão para reproduzir cópias deste Trabalho de Conclusão de Curso e emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Do mesmo modo, o ISC tem permissão para divulgar este documento em biblioteca virtual, em formato que permita o acesso via redes de comunicação e a reprodução de cópias, desde que protegida a integridade do conteúdo dessas cópias e proibido o acesso a partes isoladas desse conteúdo. O autor reserva outros direitos de publicação e nenhuma parte deste documento pode ser reproduzida sem a autorização por escrito do autor.

Leonardo Augusto da Silva Pacheco
leonardopacheco@tcu.gov.br

Ficha catalográfica

Pacheco, Leonardo Augusto da Silva

Classificação Automática de Acórdãos a Partir de Enunciados da Jurisprudência Seleccionada: análise comparativa de técnicas e fontes de dados / Leonardo Augusto da Silva Pacheco; orientador, Thiago Paulo Faleiros, 2020.

76 p.

Monografia (especialização) - Escola Superior do Tribunal de Contas da União, Curso de Especialização em Análise de Dados para o Controle, Brasília, 2020.

Inclui referências.

1. Análise de Dados. 3. Mineração de Dados. 4. Classificação Textual. 5. Aprendizado de Máquina. I. Faleiros, Thiago. II. Escola Superior do Tribunal de Contas da União. Especialização em Análise de Dados para o Controle. III. Título.

LEONARDO AUGUSTO DA SILVA PACHECO

**CLASSIFICAÇÃO AUTOMÁTICA DE ACÓRDÃOS A PARTIR
DE ENUNCIADOS DA JURISPRUDÊNCIA SELECIONADA:**

Análise comparativa de técnicas e fontes de dados

Trabalho de conclusão do curso de pós-graduação *lato sensu* de Especialização em Análise de Dados para o Controle, realizado pela Escola Superior do Tribunal de Contas da União como requisito para a obtenção do título de especialista.

Brasília, 24 de março de 2020.

Banca Examinadora:

Prof. Dr. Thiago Paulo Faleiros
Orientador
Universidade de Brasília

Prof. Me. Saul Campos Berardo
Avaliador
Tribunal de Contas da União

AGRADECIMENTOS

O desafio de encarar uma nova pós-graduação e, em particular, de seguir com este projeto de TCC, foi associado a um conjunto de belas conquistas, por um lado, e alguns sacrifícios, de outro. Àqueles que estiveram presentes comigo, em diversos momentos dessa caminhada, minha mais sincera gratidão.

À Guaíra, amor de uma vida inteira, agradeço pelo enorme apoio, desde o início, até a revisão deste texto, pela paciência, em especial, pelas ausências em finais de semana, pelas férias que não viajamos, pelas tarefas domésticas que vez ou outra tive que postergar. Obrigado por mais essa conquista em uma jornada tão bonita que construímos juntos.

A meus garotos, Tales e Theo, que também tiveram a paciência de esperar o pai terminar as tarefas, quase sempre... e que muito me apoiaram, e me inspiraram a ser cada vez melhor. À Yasmin, que me abriu os olhos para a relatividade dos problemas do dia a dia, e mostrou, de supetão, quão incondicional é o amor de pai.

À minha mãe, que foi mãe, pai, exemplo de coragem, responsabilidade e bondade, e minha primeira professora, por me incentivar desde as primeiras conquistas, e inspirar o melhor sempre. À meu pai, mesmo deixando-nos prematuramente, sempre presente em minha vida.

A meus geniais colegas de TI, sempre inspiradores, em especial meus colegas de equipe, que seguraram o leme tantas vezes para que atravessássemos os momentos mais difíceis da travessia. Em especial, companheiros Borela, Victor, Alessandra, Luís, Leandro, pelas excelentes discussões, insights e enorme apoio para a execução deste projeto. A meus chefes, Robson e Rodrigo, pelo suporte necessário, apoio nas decisões, e as cobranças certas.

A meus colegas de pós, companheiros de tantas dificuldades que enfrentamos, penso que todos muito mais preparados para os novos desafios.

A meus professores, pelas lições valiosas e pelas tarefas desafiadoras. Em particular, meu professor e orientador Thiago, pelas ideias luminosas, sem as quais este projeto caminharia bem menos, pela tranquilidade e paciência de sempre.

À equipe de apoio da pós, solícitos e muito pacientes com toda a ansiedade da turma, e todo o pessoal do ISC, pelo ambiente espetacular que proporcionam a todos que participam dos excelentes eventos.

RESUMO

O presente trabalho visa verificar a viabilidade da classificação automática de enunciados de jurisprudência e acórdãos do Tribunal de Contas da União em áreas de jurisprudência, a partir de textos de enunciados, de excertos de acórdãos e de inteiros teores de acórdãos, empregando redes neurais. Técnicas tradicionais de aprendizado de máquina e baseadas em redes neurais são comparadas a fim de identificar a mais adequada para cada tarefa de classificação. Além disso, diferentes técnicas de geração de *word embeddings* e seu impacto nas tarefas de classificação são verificados. Os experimentos consistiram em testes com diferentes arquiteturas e hiperparâmetros, e comparação estatística de alguns modelos e configurações selecionados por meio de validação cruzada e medidas F1. No caso de textos mais curtos de entrada, como enunciados, excertos, e acórdãos reduzidos, os resultados apontaram melhor desempenho em redes recorrentes com *embeddings* pré-treinados a partir de textos do mesmo domínio do conhecimento. Para textos mais longos, redes convolucionais e modelos não neurais apresentaram melhor desempenho, e o pré-treinamento de *embeddings* foi menos relevante.

Palavras-chave: Mineração de texto. Classificação. Aprendizado de máquina. Redes Neurais. Word embeddings.

ABSTRACT

The present work aims to verify the feasibility of automatic classification of statements of jurisprudence and rulings from Federal Court of Accounts in areas of jurisprudence, based on texts of statements, excerpts from rulings and entire contents of rulings, using neural networks. Using traditional, non-neural machine learning techniques as basis for comparison, it verifies which types of neural networks are most suitable for each classification task. It also compares different techniques for generating word embeddings and their impact on classification tasks. The experiments consisted of tests with different architectures and hyperparameters, and statistical comparison of some selected models and configurations through cross-validation and F1 scores. For shorter input texts, such as statements, excerpts, and reduced rulings, recurrent networks containing embeddings pre-trained from texts inside the same domain of knowledge had the best results. For longer texts, convolutional networks and non-neural models provided the best performance, and pre-training of embeddings had less relevance.

Keywords: Text mining. Classification. Machine learning. Word embeddings. Neural networks.

LISTA DE ILUSTRAÇÕES

Figura 1 - Exemplo de acórdão publicado no STF.....	13
Figura 2 - Evolução da quantidade de acórdãos nos últimos anos	14
Figura 3 - Etapas principais de pré-processamento de um texto.	19
Figura 4 - Representação de uma rede neural clássica	24
Figura 5 - Representações de uma unidade recorrente simples (RNN).....	25
Figura 6 - Representações de unidades LSTM e GRU.....	26
Figura 7 - Funcionamento da convolução em uma dimensão.	26
Figura 8 - Fluxo dos dados para a classificação dos textos empregando redes neurais	31
Figura 9 - Distribuição de enunciados por área de jurisprudência	42
Figura 10 - Distribuição de acórdãos referenciados por área única de jurisprudência.....	44
Figura 11 - Distribuição de acórdãos por áreas combinadas de jurisprudência.	45

LISTA DE TABELAS

Tabela 1 - Matriz de confusão para classificação binária.....	27
Tabela 2 - Alguns valores comuns para a variável t, na distribuição t-Student bicaudal.....	30
Tabela 3 - Enunciados de jurisprudência por acórdão.....	33
Tabela 4 - Exemplos de registros de jurisprudência.....	41
Tabela 5 - Exemplos de excertos de acórdãos.....	43
Tabela 6 - Informações sobre o pré-processamento de enunciados e excertos.....	46
Tabela 7 - Informações sobre o pré-processamento de acórdãos.....	46
Tabela 8 - Treinamentos sobre enunciados com embeddings sem pré-treino.....	48
Tabela 9 - Treinamentos sobre enunciados com embeddings pré-treinados do NILC-USP....	49
Tabela 10 - Treinamentos sobre enunciados com embeddings pré-treinados sobre acórdãos.....	50
Tabela 11 - Treinamentos sobre excertos com embeddings sem pré-treino.....	52
Tabela 12 - Treinamentos sobre excertos com embeddings pré-treinados do NILC-USP.....	53
Tabela 13 - Treinamentos sobre excertos com embeddings pré-treinados sobre acórdãos.....	54
Tabela 14 - Treinamentos sobre excertos filtrados com embeddings sem pré-treino.....	55
Tabela 15 - Treinamentos sobre excertos filtrados com embeddings do NILC-USP.....	56
Tabela 16 - Treinamentos sobre excertos filtrados com embeddings de acórdãos.....	57
Tabela 17 - Treinamentos sobre acórdãos associados a única área de jurisprudência.....	58
Tabela 18 - Treinamentos sobre todos os acórdãos associados a áreas de jurisprudência.....	60
Tabela 19 - Treinamentos sobre acórdãos filtrados (6000), única área de jurisprudência.....	61
Tabela 20 - Treinamentos sobre acórdãos filtrados (6000), múltiplas áreas de jurisprudência.....	62
Tabela 21 - Treinamentos sobre acórdãos filtrados (500), única área de jurisprudência.....	64
Tabela 22 - Treinamentos sobre acórdãos filtrados (500), múltiplas áreas de jurisprudência.....	65
Tabela 23 - Comparação das estratégias de embeddings – Scores F1 Globais.....	66
Tabela 24 - Comparação das estratégias de embeddings – Scores F1 por classe.....	67
Tabela 25 - Modelos neurais de classificação de excertos – Scores F1 Globais.....	67
Tabela 26 - Modelos neurais de classificação de excertos – Scores F1 por classe.....	68
Tabela 27 - Modelos neurais de classificação de acórdãos – Scores F1 Globais.....	69
Tabela 28 - Modelos neurais de classificação de acórdãos – Scores F1 por classe.....	69
Tabela 29 – Classificação de enunciados por modelos não neurais – Scores F1 Globais.....	70
Tabela 30 – Classificação de excertos por modelos não neurais – Scores F1 Globais.....	70
Tabela 31 – Classificação de acórdãos por modelos não neurais – Scores F1 Globais.....	70
Tabela 32 - Classificação de acórdãos por Regressão Logística – Scores F1 por Classe.....	71
Tabela 33 - Alguns dos melhores resultados com classificações exploratórias.....	72

SUMÁRIO

1	INTRODUÇÃO.....	12
1.1	Contextualização	12
1.1.1	Descrição do problema.....	12
1.1.2	Processo de geração do acórdão	15
1.1.3	Processo de geração da jurisprudência selecionada	15
1.1.4	Motivação.....	16
1.2	Objetivos	16
1.2.1	Objetivo geral	16
1.2.2	Objetivos específicos.....	17
2	REFERENCIAL TEÓRICO.....	18
2.1	Mineração de texto	18
2.2	Bag of Words.....	19
2.3	Classificação.....	20
2.4	Modelos de classificação para texto.....	21
2.5	Word Embeddings	23
2.6	Arquiteturas de redes neurais	24
2.7	Avaliação.....	27
3	PREPARAÇÃO.....	31
3.1	Obtenção dos dados.....	31
3.2	Pré-processamento.....	33
3.2.1	Classificadores não neurais	33
3.2.2	Classificação por redes neurais	34
3.2.3	Redução dos dados	36
3.2.4	Geração de embeddings pré-treinados sobre acórdãos.....	36
3.3	Modelos de classificação.....	38
3.3.1	Modelos não neurais.....	38
3.3.2	Redes neurais densas	38

3.3.3	Redes neurais recorrentes	39
3.3.4	Redes neurais convolucionais	40
4	RESULTADOS	41
4.1	Descrição dos dados	41
4.1.1	Enunciados de jurisprudência.....	41
4.1.2	Excertos de acórdãos	43
4.1.3	Inteiros teores de acórdãos	43
4.2	Pré-processamento.....	45
4.2.1	Pré-processamento para redes neurais.....	45
4.2.2	Geração de embeddings a partir de acórdãos	47
4.3	Classificação – treinamentos exploratórios	47
4.3.1	Classificação de enunciados	47
4.3.2	Classificação de excertos.....	51
4.3.3	Classificação de excertos filtrados	54
4.3.4	Classificação de acórdãos.....	57
4.3.5	Acórdãos filtrados até 6000 palavras	60
4.3.6	Acórdãos filtrados até 500 palavras	63
4.4	Classificação – comparações com validação cruzada	65
4.4.1	Estratégias de embeddings para enunciados	65
4.4.2	Modelos de classificação de excertos.....	67
4.4.3	Modelos de classificação de acórdãos.....	68
4.4.4	Modelos não neurais.....	70
4.5	Resumo	71
5	CONCLUSÃO.....	74
	REFERÊNCIAS	76

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

1.1.1 Descrição do problema

O Tribunal de Contas da União (TCU) disponibiliza suas decisões para o público por meio do Sistema de Pesquisa de Jurisprudência¹. O sistema é composto por cinco bases principais de pesquisa:

- **Acórdãos:** inteiro teor (relatório, voto e acórdão) de acórdãos e decisões;
- **Jurisprudência selecionada:** enunciados elaborados a partir de deliberações selecionadas sob o critério de relevância jurisprudencial;
- **Publicações:** boletins elaborados (Jurisprudência, Pessoal, Licitações e contratos);
- **Súmulas:** enunciados de súmulas.
- **Respostas a consultas:** respostas emitidas a consultas realizadas ao Tribunal.




A única pesquisa completa, que abrange todas as decisões colegiadas, é a pesquisa de acórdãos. Porém, como o texto do inteiro teor, em geral, é muito grande, sua pesquisa se torna um tanto imprecisa. Isso cria dificuldade para que o usuário consiga encontrar facilmente o que procura ao realizar uma pesquisa, a não ser que providencie um termo de busca extremamente preciso e elaborado, o que não está ao alcance do usuário comum. Há exemplos de inteiros teores com mais de trezentas páginas².

Em outros tribunais superiores, para cada acórdão, uma equipe de jurisprudência elaborava uma ementa, que resume as informações consideradas mais relevantes em cada decisão, o tema, um resumo da decisão tomada, legislação empregada, termos para indexação, outras decisões citadas. Uma pesquisa baseada apenas nos textos das ementas, indexadores e outras informações preparadas traria provavelmente maior precisão nas pesquisas realizadas. Podemos visualizar o exemplo de um acórdão do STF na Figura 1, a seguir.

¹ Pesquisa de Jurisprudência - localizada no endereço: <https://pesquisa.apps.tcu.gov.br/#/pesquisa/jurisprudencia>

² Acórdão TCU nº 2121/2017 – Plenário: <https://contas.tcu.gov.br/sagas/SvlVisualizarRelVotoAcRtf?codFiltro=SAGAS-SESSAO-ENCERRADA&seOcultarPagina=S&item0=603589>

Figura 1 - Exemplo de acórdão publicado no STF.

Pesquisa de Jurisprudência   

Acórdãos

Documentos encontrados: 684 (1 / 69) pági

<< | < | > | >> | Nova Pesquisa 1 | 2 | 3 | 4 | 5 | 6 | 7 | Próximo

Expressão de busca: (PRISAO EM SEGUNDA INSTANCIA)

Acompanhamento Processual	Inteiro Teor	DJ/DJe	Ementa sem Formatação
---------------------------	--------------	--------	-----------------------

HC 163718 AgR / SP - SÃO PAULO
AG.REG. NO HABEAS CORPUS
Relator(a): Min. LUIZ FUX
Julgamento: 30/11/2018 **Órgão Julgador: Primeira Turma**

Publicação

PROCESSO ELETRÔNICO
 DJe-264 DIVULG 07-12-2018 PUBLIC 10-12-2018

Parte(s)

AGTE.(S) : ANGELO APARECIDO SIVIERO
 ADV.(A/S) : HELIO DA SILVA SANCHES
 AGDO.(A/S) : RELATOR DO HC Nº 473.576 DO SUPERIOR TRIBUNAL DE JUSTIÇA

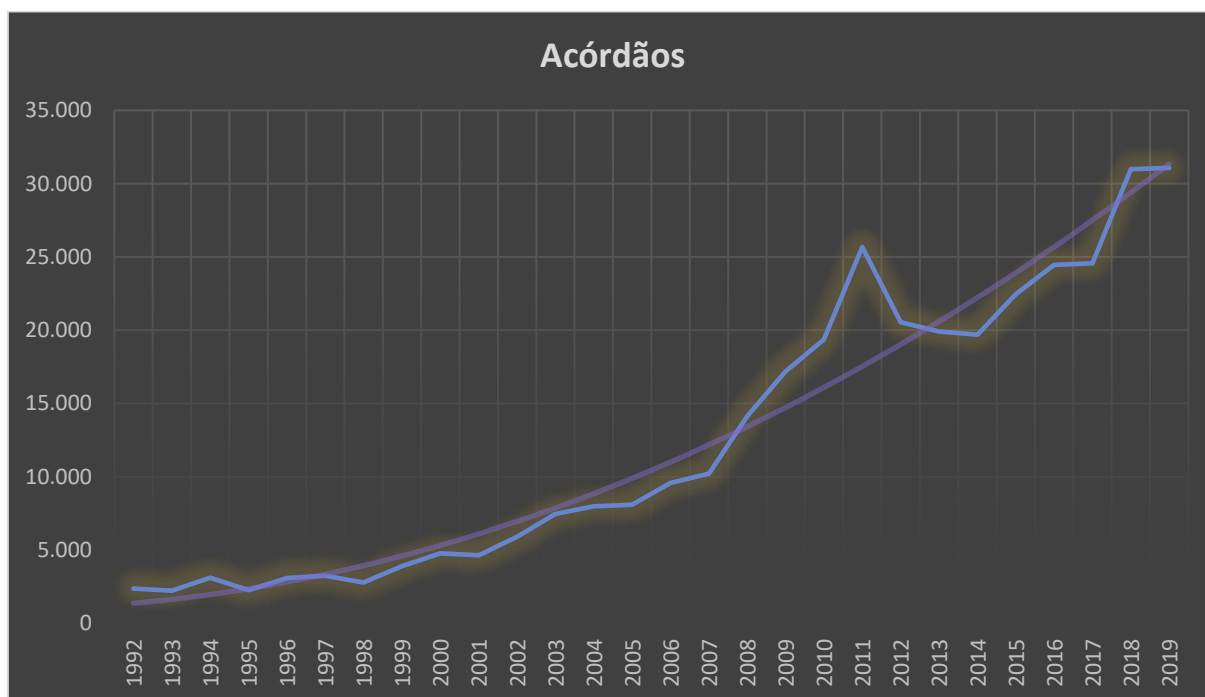
Ementa

Ementa: AGRAVO REGIMENTAL NO HABEAS CORPUS. PENAL E PROCESSO PENAL. CRIMES DE TRÁFICO ILÍCITO DE ENTORPECENTES, ASSOCIAÇÃO PARA O TRÁFICO E PORTE ILEGAL DE ARMA DE FOGO DE USO RESTRITO. ARTIGOS 33 E 35 DA LEI 11.343/06 E ARTIGO 16 DA LEI 10.826/03. ALEGADA NULIDADE PROCESSUAL. REDISCUSSÃO DOS CRITÉRIOS DE DOSIMETRIA DA PENA. PLEITO DE REVOGAÇÃO DA CUSTÓDIA CAUTELAR. ALEGADA NULIDADE PROCESSUAL. TEMAS NÃO DEBATIDOS PELAS **INSTÂNCIAS** PRECEDENTES. SUPRESSÃO DE **INSTÂNCIAS**. IMPOSSIBILIDADE. INEXISTÊNCIA DE ABUSO DE PODER OU FLAGRANTE ILEGALIDADE. REVOLVIMENTO DO CONJUNTO FÁTICO-PROBATÓRIO. INADMISSIBILIDADE NA VIA ELEITA. AGRAVO REGIMENTAL DESPROVIDO. 1. Inexiste excepcionalidade que permita a concessão da ordem de ofício ante a ausência de flagrante ilegalidade ou abuso de poder na decisão da Corte Superior que assentou ser inviável a análise do mérito da questão recorrida, sob pena de supressão de **instâncias**. 2. In casu, o recorrente foi condenado, pelo juízo natural, à pena de 15 (quinze) anos e 04 (quatro) meses de reclusão, **em** regime inicial fechado, pela prática dos crimes previstos nos artigos 33 e 35 da Lei 11.343/06 e 16 da Lei 10.826/03. 3. A dosimetria da pena, bem como os

Fonte: <http://stf.jus.br/portal/jurisprudencia/pesquisarJurisprudencia.asp>

O tratamento prévio de acórdãos para fins de jurisprudência, por outro lado, exigiria um trabalho muito maior do que temos hoje: maior equipe, maior suporte de ferramentas e maior tempo de elaboração. Outra dificuldade é que poderia atrasar a produção dos acórdãos e a publicação. Hoje, tão logo assinado, um acórdão pode ser publicado imediatamente, facilitando o acesso às partes e à sociedade. O tratamento jurisprudencial de todos os acórdãos traria o risco de que esse prazo fosse ampliado. Conforme podemos visualizar na Figura 2, a quantidade de decisões do Tribunal tem crescido ano a ano (a linha de tendência traçada é uma parábola), o que torna essa opção cada vez mais difícil.

Figura 2 - Evolução da quantidade de acórdãos nos últimos anos



Fonte: <https://pesquisa.apps.tcu.gov.br/#/pesquisa/acordao-completo>

O Tribunal vem adotando uma solução intermediária, que é a jurisprudência selecionada. A Secretaria das Sessões seleciona não todos, mas alguns acórdãos, após o julgamento, a partir do qual geram um texto de enunciado, classificado em área, tema e subtema, outros termos indexadores e metadados. Com isso, possibilita uma pesquisa temática mais precisa, sobre uma base restrita, que permite um entendimento jurisprudencial aproximado do Tribunal acerca de um determinado tema, e indica alguns acórdãos que demonstram aquela linha de entendimento.

Não sendo uma pesquisa completa, reduz os custos e riscos mencionados, mas possui limitações. Não conseguimos, para um acórdão qualquer, encontrar facilmente toda informação de suporte necessária, da mesma forma que se acha em acórdãos de outros tribunais superiores. Além disso, ao realizar determinadas pesquisas, o usuário comumente encontra dificuldades em encontrar esse acórdão relevante, conforme é demonstrado pelo registro dos logs do sistema de pesquisa.

1.1.2 Processo de geração do acórdão

A origem do acórdão é uma auditoria, monitoramento ou outro trabalho realizado pelo corpo técnico (Secretaria do Tribunal), cujo resultado é um relatório, que explica a situação analisada e traz uma proposta de encaminhamento da decisão do Tribunal àquele respeito. É autuado em um processo pelo sistema e-TCU, e geralmente assinado pelo auditor ou equipe de auditoria, diretor e secretário.

É designado um ministro relator, que produz um relatório final a partir do relatório técnico, um voto e um rascunho do acórdão (decisão). O ministro revisor poderá seguir o voto do relator, emitir voto contrário ou complementar. Os ministros, reunidos em uma sessão do Tribunal, discutem, votam e produzem a decisão (o acórdão). O sistema Sagas faz todo o gerenciamento desde o momento de designação do relator do processo, passando pela sessão até a assinatura do acórdão.

Uma vez assinado e disponibilizado no Sagas, o acórdão é publicado pelo Sistema de Pesquisa de Jurisprudência. Ao final, a ata da sessão também é publicada. O acórdão pode depois ser objeto de recurso ao Tribunal, mandado de segurança ou outra ação judicial.

1.1.3 Processo de geração da jurisprudência selecionada

Dentro da Secretaria das Sessões, existe a Diretoria de Jurisprudência, responsável pela elaboração da jurisprudência selecionada. O trabalho ocorre utilizando o sistema e-Juris.

Primeiramente, são selecionados acórdãos que denotam linhas de entendimento do Tribunal acerca de diversos assuntos. Por exemplo, em 2017, foram proferidos 24.558 acórdãos, dos quais menos de mil foram selecionados. Do inteiro teor são extraídos excertos, ou seja, parágrafos e trechos do texto dos quais melhor se pode depreender o entendimento. Em seguida, produz-se um enunciado, que é o resumo do entendimento realizado. Foram 993 enunciados

produzidos a partir de acórdãos de 2017. É possível, de um único acórdão, extrair mais de um enunciado.

O enunciado de jurisprudência é acompanhado de uma série de termos de indexação, organizados em área, tema e subtema. Tais termos pertencem a um vocabulário controlado, o VCE (Vocabulário de Controle Externo), produzido pela área de gestão documental do TCU. Além disso, são registradas a legislação empregada e outras informações relevantes.

1.1.4 Motivação

Trazendo o conhecimento gerado pelo trabalho da jurisprudência selecionada para a pesquisa de acórdãos, poderemos acrescentar informações relevantes, facilitar o processo de busca e a associação de acórdãos afins, provendo, enfim, uma pesquisa mais útil para o usuário.

O próprio processo de produção da jurisprudência selecionada poderia ser facilitado a partir do processamento de decisões pretéritas, com o preenchimento automatizado de informações ou sugestões.

O presente projeto visa, então, verificar a viabilidade de classificação automática de enunciados de jurisprudência e acórdãos por áreas de jurisprudência, a partir de textos produzidos ou extraídos nos sistemas de pesquisa de Jurisprudência e do sistema de Jurisprudência (e-Juris).

A opção mais simples é extraí-la a partir do enunciado, que já contém a síntese do que deseja exprimir como entendimento jurisprudencial. Desejamos verificar também a possibilidade de obtenção da área a partir do excerto do acórdão, que é uma informação menos tratada, ou diretamente do inteiro teor.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Efetuar a classificação automática de acórdãos, por áreas de jurisprudência, a partir de seu inteiro teor, comparando diferentes modelos, técnicas, hiperparâmetros por meio de sua acurácia.

1.2.2 Objetivos específicos

- Efetuar a classificação automática de enunciados de jurisprudência por área, a partir de seu texto, comparando diferentes modelos, técnicas, hiperparâmetros por meio de sua acurácia.
- Efetuar a classificação automática enunciados de jurisprudência por área, a partir dos excertos dos acórdãos associados, comparando diferentes modelos, técnicas, hiperparâmetros por meio de sua acurácia.
- Avaliar os resultados encontrados. A classificação a partir do inteiro teor do acórdão é viável, ou a perda de acurácia em relação à classificação sobre enunciado é demasiada?

2 REFERENCIAL TEÓRICO

2.1 MINERAÇÃO DE TEXTO

Mineração de texto consiste na extração de informação significativa de um texto (Allahyari et al, 2017). Uma tarefa típica é a classificação.

Para a realização dessa extração, técnicas de Aprendizado de Máquina foram ganhando popularidade em relação a técnicas tradicionais de definição manual de regras particularmente a partir da década de 90 (Sebastiani, 2001). Aprendizado de Máquina é um ramo da Inteligência Artificial que tenta definir um conjunto de abordagens para encontrar padrões nos dados, de forma a prever os padrões em dados futuros (Allahyari et al, 2017).

Os modelos de predição, em geral, e de classificação, em particular, são normalmente desenhados para construir modelos sobre dados estruturados (Miner et al, 2012). Estes estão prontos para processamento pela máquina, ao passo que dados não estruturados precisam antes ser convertidos em formato estruturado.

O texto é um exemplo típico de informação não estruturada, facilmente reconhecida e processada por humanos, mas de entendimento muito mais difícil para as máquinas (Allahyari et al, 2017). Apesar de ser uma fonte preciosa de informação e conhecimento, precisamos de métodos e algoritmos de processamento para que máquina possa utilizar essas informações em uma determinada aplicação.

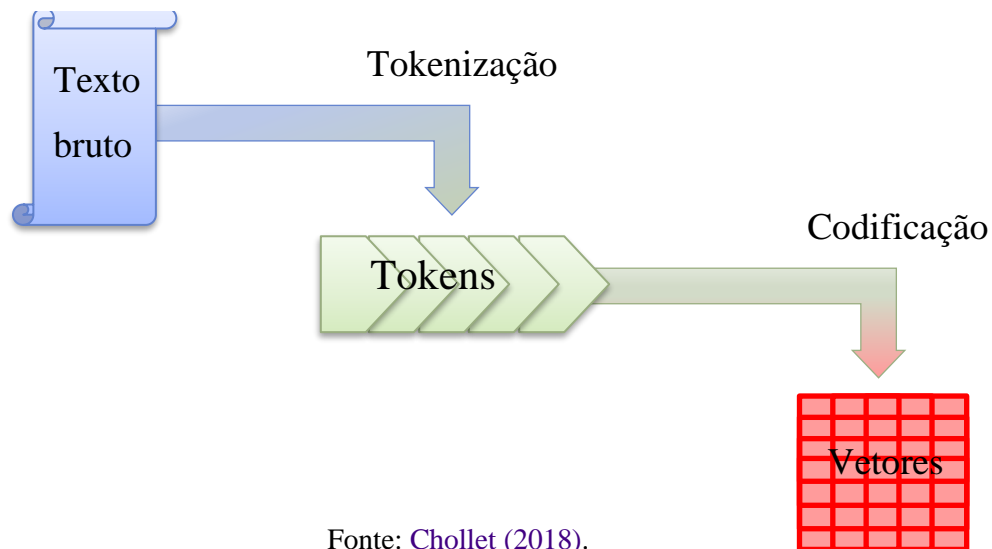
Para conversão de texto em formato estruturado, em geral, há duas etapas principais de pré-processamento (Chollet, 2018):

- **Tokenização (*tokenization*):** quebra do texto em uma lista de unidades menores, os tokens. Esses tokens podem ser baseados em palavras, caracteres, subpalavras (Mikolov et al, 2013), sequências de palavras (n-gramas de palavras), entre outros.
- **Codificação (*encoding ou numericalization*):** transformação do token em informação numérica, normalmente vetorial, que pode ser então processada por um modelo. As duas principais técnicas de codificação, ou seja, de transformação de uma sequência de tokens em um vetor numérico, são *bag of words* (ou *bag of chars*, ou *bag of n-grams*, de acordo com o tipo de tokenização utilizado) e *word embeddings* (também pode variar com o tipo de tokenização utilizado) (Chollet, 2018).

Esse fluxo mínimo de pré-processamento do texto é representado na Figura 3. Adicionalmente, outras técnicas de pré-processamento podem ser utilizadas, tais como (Allahyari et al, 2017):

- **Filtragem (*filtering*):** remoção de alguns tokens do texto. Um exemplo típico é a remoção de *stopwords*, que são palavras que aparecem frequentemente nos textos, mas carregam pouca informação de conteúdo (artigos, preposições, conjunções, etc).
- **Lematização (*lemmatization*):** agrupamento de formas flexionadas de uma palavra, a partir da análise morfológica das palavras. Por exemplo, transformar um verbo em sua forma no infinitivo, ou o plural de um substantivo em singular.
- **Stemização (*stemming*):** aplicação de regras para tentativa de obtenção de uma raiz (*stem*) a partir da palavra derivada, não necessariamente relacionada à raiz morfológica da palavra. É uma alternativa à lematização.

Figura 3 - Etapas principais de pré-processamento de um texto.



Fonte: Chollet (2018).

2.2 BAG OF WORDS

Dada uma coleção de textos $\mathcal{D} = \{ d_1, d_2, d_3, \dots, d_D \}$, **vocabulário** é o conjunto de termos (palavras, por exemplo) distintos encontrados na coleção: $\mathcal{V} = \{ t_1, t_2, t_3, \dots, t_V \}$. D é o tamanho da coleção, e V é o tamanho do vocabulário. Seja $f_d(t)$ a frequência do termo $t \in \mathcal{V}$ em um texto $d \in \mathcal{D}$, e $f_{\mathcal{D}}(w)$ o número de documentos da coleção que possuem o termo $t \in \mathcal{V}$ (Allahyari et al, 2017).

A forma mais simples de codificação *bag of words* é o **modelo booleano** (*one-hot*): para um subconjunto de documentos \mathcal{D}' , com D' documentos, associa um documento d_i a um vetor $\omega = (\omega_1, \omega_2, \omega_3, \dots, \omega_V)$. Cada posição ω_j corresponde a um termo distinto do vocabulário:

terá o valor 1 se o termo t_j está presente no texto ($f_d(t) > 0$), e 0 caso contrário ($f_d(t) = 0$). Formase, portanto, uma matriz de zeros e uns de documentos por termos, de dimensão $D' \times V$:

$$W = [\omega_{ij}] D' \times V,$$

$$\text{onde } \omega_{ij} = 0 \text{ ou } 1$$

A variação mais popular é o *term frequency-inverse document frequency (TF - IDF)*, em que o peso que associa o termo ao documento é calculado pela fórmula:

$$w(t,d) = f_d(t) \times \log (D / f_{\mathcal{D}} (t))$$

Na forma TF-IDF, a frequência do token (*TF*) no documento é normalizada pelo inverso da frequência nos documentos (*IDF*). Com isso, pesos maiores ocorrerão para termos mais frequentes no documento relativamente à frequência média na coleção. Da mesma forma que no modelo booleano, termos ausentes em um documento terão peso zero (pois $f_d(t) = 0$), assim como tokens que apareçam em todos os documentos, como *stopwords* ($f_{\mathcal{D}} (w) = D \Rightarrow \log (D / f_{\mathcal{D}} (w) = 0$). A matriz, em geral, também será esparsa.

Como, em geral, a vasta maioria dos termos aparece em um conjunto pequeno de documentos ($f_{\mathcal{D}} (w) \ll D$), a matriz conterá mais zeros que outros valores, sendo, portanto, uma matriz esparsa. Será também de grande porte, se a quantidade de documentos for grande e o vocabulário também. As técnicas de filtragem, citadas anteriormente, podem reduzir o tamanho do vocabulário, reduzindo assim o tamanho da matriz. Técnicas de redução de dimensionalidade também podem reduzir as dimensões da matriz a ser utilizada pelo modelo (Sebastiani, 2001), por exemplo a técnica PCA – *principal component analysis* (Zaki et al, 2014). Outra alternativa é o uso de estruturas de dados especiais para matrizes esparsas que, embora não modifiquem a informação na matriz, reduzem o uso de recursos de memória da máquina (Pisanezky, 2008).

Outra desvantagem do *bag of words* é o descarte da informação de ordem e proximidade entre os *tokens* em cada documento, embora mantenha a informação sobre a distribuição dos termos nos documentos (Chollet, 2018).

2.3 CLASSIFICAÇÃO

A tarefa de classificação visa designar classes pré-definidas a documentos de texto (Allahyari et al, 2017). Dado um conjunto de documentos $\mathcal{D} = \{ d_1, d_2, d_3, \dots, d_D \}$, e um subconjunto de treino $\mathcal{D}_T = \{ d_1, d_2, d_3, \dots, d_N \}$, no qual cada documento d_i está associado a um

rótulo ℓ_i , do conjunto de classes $\mathcal{L} = \{ \ell_1, \ell_2, \ell_3, \dots, \ell_K \}$, o objetivo é encontrar um modelo de classificação f onde:

$$f: \mathcal{D} \rightarrow \mathcal{L}$$

$$f(d) = \ell$$

Esse modelo de classificação associa, portanto, um documento a uma única classe (*hard*). Alternativamente, pode-se construir um modelo de classificação que associa a cada documento todas as classes, cada qual com uma probabilidade (*soft*):

$$f_P: \mathcal{D} \times \mathcal{L} \rightarrow [0,1]$$

$$f_P(d, \ell) = p, \text{ onde } 0 \leq p \leq 1$$

Com relação ao número de classes (K), podemos assumir que $K > 1$ pois um conjunto de classes unitário tornaria trivial o problema de classificação. Se temos $K = 2$, a classificação é binária, e se $K > 2$ temos uma classificação multi-classe (Sebastiani, 2001). Além disso, para o caso da função f descrita, denominamos *single-label* pois cada documento está associado a apenas um rótulo. Se, ao contrário, tivermos um documento associado a um subconjunto dos rótulos:

$$\mathcal{P}(\mathcal{L}) = \{ \emptyset, \{\ell_1\}, \{\ell_2\}, \{\ell_3\}, \dots, \{\ell_K\}, \{\ell_1, \ell_2\}, \dots \}$$

denominamos a classificação como *multi-label*:

$$f_M: \mathcal{D} \rightarrow \mathcal{P}(\mathcal{L})$$

$$f(d) = L \subset \mathcal{L}$$

2.4 MODELOS DE CLASSIFICAÇÃO PARA TEXTO

Há diversos modelos de classificação, que se amparam em propriedades e técnicas distintas. Descreveremos alguns deles.

Naive Bayes (Bayes “ingênuo”) é um classificador probabilístico que assume que a distribuição de diferentes termos é independente dos demais. Apesar dessa assunção ser claramente falsa em casos reais, o modelo costuma funcionar bem (Allahyari et al, 2017).

Logistic Regression (regressão logística) também é um classificador probabilístico, também simples e versátil (Chollet, 2018). Define a saída como função da entrada, cujos coeficientes devem ser calculados, de forma similar a uma rede neural de uma camada (perceptron).

Nearest Neighbor (vizinho mais próximo) é um classificador baseado em proximidade / distância. Para cada documento, toma-se os k vizinhos mais próximos no conjunto de treinamento, e a classe mais comum entre eles é retornada como a classe do documento (Allahyari et al, 2017).

Decision Tree (árvore de decisão) é uma árvore na qual uma condição sobre os valores de atributos é empregada para dividir os dados hierarquicamente. No caso de documentos de texto, essas condições podem se concretizar, por exemplo, na presença ou ausência de determinada palavra no documento, ou intervalos de valores de frequência relativa TF-IDF.

Support Vector Machine (SVM) é um classificador que tenta encontrar um hiperplano (ou conjunto deles, no caso de classificação multi-classes), formado pela combinação dos elementos dos vetores associados aos documentos, que melhor separa os documentos em classes distintas, de forma a maximizar a margem entre pontos de classes distintas.

Linear Discriminant Analysis (LDA) é mais conhecida como técnica para redução de dimensionalidade, mas também pode ser empregada como classificador. Modela a dependência estocástica entre termos por meio de matrizes de covariância das classes (Sebastiani, 2001).

Rede neural (neural network – NN) é uma rede formada por unidades de processamento, os neurônios, composto por uma combinação linear entre suas entradas e uma função de ativação (as mais comuns: sigmoide, Tanh e ReLU). A entrada da rede é o vetor, matriz ou tensor codificado para um documento e a saída é a classe atribuída para aquele documento. Os coeficientes da combinação linear em todos os neurônios formam os pesos da rede. Normalmente, esses pesos são inicializados com zeros ou números aleatórios, e são otimizados por meio de *backpropagation*, que consiste em fazer pequenos ajustes nos pesos camada a camada, da saída para a entrada no caso de erro em uma classificação no conjunto de treino (valor predito difere do valor rotulado). Emprega-se uma função de erro ou de custo, e o objetivo da rede é achar seu valor mínimo (Sebastiani, 2001). Há diversas formas de construir redes neurais, segundo diversas arquiteturas, algumas das quais serão discutidas mais adiante.

Um **comitê de classificadores (ensemble)** baseia-se na ideia de que uma combinação de especialistas pode se sair melhor do que um especialista individual. Se um conjunto de classificadores diferentes é aplicado a um documento, temos uma lista das classes que cada um deles atribuiu ao documento, e usa-se a lista para produzir uma classificação final. Pode-se escolher a classe mais frequente, ou moda (votação), montar uma combinação linear das probabilidades das classes (no caso de classificação soft, por exemplo), ou montar um modelo de classificação sobre as saídas dos classificadores (*stacking*).

2.5 WORD EMBEDDINGS

As técnicas de *word embeddings* associam a cada termo do vocabulário um vetor de dimensão “d”, contendo valores de ponto flutuante. Como, em geral, essa dimensão é relativamente pequena, e os valores são significativos, esta representação é dita densa, em contraste com a representação esparsa, citada anteriormente (Chollet, 2018). Outra diferença é que esses valores são aprendidos a partir dos dados, a partir da rede neural, enquanto para *bag of words* há uma rotina de construção da matriz com valores calculados a partir dos textos.

A forma de inicialização e ajuste dos vetores pode variar:

- Os vetores são inicializados com valores aleatórios e vão se ajustando durante o treinamento do modelo.
- Os vetores são inicializados com valores produzidos a partir de tarefa anterior de aprendizado de máquina, e aproveitados na tarefa corrente. Neste caso, os *embeddings* são denominados pré-treinados (*pretained word embeddings*). Esses valores podem permanecer fixos ou se ajustarem durante o treinamento.

Nas aplicações de aprendizado de máquina, a adaptação ao idioma e ao domínio de conhecimento são importantes. Nesse aspecto, conjuntos de dados em língua portuguesa e modelos pré-treinados em domínios específicos são recursos valiosos. A forma de utilização de *word embeddings* influencia diretamente os resultados dos treinamentos. O uso de um corpus no mesmo domínio melhora significativamente o desempenho de uma determinada tarefa, mais do que o tamanho do corpus, por exemplo (Lai et al, 2016).

Os *embeddings* pré-treinados, em geral, são gerados a partir de grandes coleções de documentos não rotulados (corpus) e podem capturar conhecimento sintático, semântico e morfológico desses textos (Hartmann et al, 2017). Os algoritmos empregados se dividem em duas famílias: baseados em matrizes de co-ocorrência de termos, caso do Global Vectors (GloVe), e métodos preditivos, como o Word2Vec. Este comporta duas estratégias de treinamento: *Continuous Bag-of-Words Model* (CBOW) e *Continuous Skip-gram Model* (Skip-gram), propostas por Mikolov et al (2013).

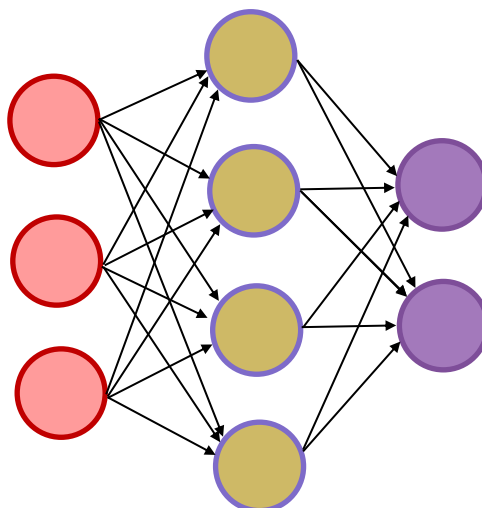
Para sua utilização dos embeddings na tarefa de classificação, é usual homogeneizar as sequências de *tokens* de um mesmo lote de treinamento com um mesmo tamanho s . Sequências com menos tokens são preenchidas (*padding*) com zeros, e com mais *tokens* são truncadas (Chollet, 2018). A sequência de *tokens* é então transformada em uma matriz $d \times s$, denominada matriz de *embeddings*.

Ao utilizar *word embeddings* pré-treinados, eventualmente, alguma palavra do texto empregado no treinamento do modelo pode não existir no vocabulário gerado. São comumente denominadas OOV (*Out-of-Vocabulary*) words (Mikolov, 2011). Neste caso, são transformadas em um vetor nulo. Se os *embeddings* são ajustáveis, essas palavras podem ser aprendidas durante o treinamento. Modelos baseados em subpalavras, como o FastText e métodos de lematização (Hartmann et al, 2017) também podem ser empregados para reduzir o impacto das *OOV words*.

2.6 ARQUITETURAS DE REDES NEURAIIS

Redes neurais *feed-forward* (FFNNs) apresentam neurônios organizados em camadas sucessivas, em que os neurônios de uma camada servem de sinal de entrada para a camada seguinte. A primeira camada é a de entrada da rede, a última é a de saída da rede, e as demais são as camadas escondidas (*hidden*). A Figura 4 ilustra uma rede FFNN típica, também chamada de rede densa.

Figura 4 - Representação de uma rede neural clássica



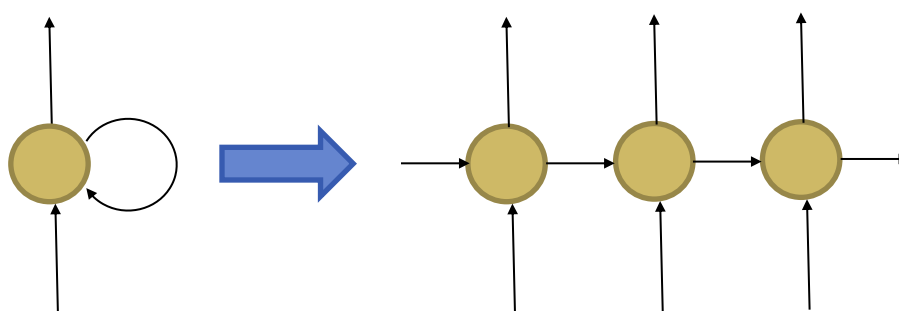
Aprendizado profundo (*deep learning*) é a construção de redes com grandes quantidades de camadas escondidas, com o objetivo de formar hierarquias de funcionalidades, compreendendo funcionalidades de mais alto nível, aprendidas nas últimas camadas, sendo formadas pela composição de funcionalidades de mais baixo nível, aprendidas nas primeiras camadas (Bengio et al, 2009). Essas representações complexas vão sendo desenvolvidas no treinamento da rede de forma incremental, e todos esses níveis de representação se formando ao mesmo

tempo, de forma conjunta. Isso se tornou possível pela melhoria na propagação de gradiente através das camadas da rede, por meio de evoluções algorítmicas como novas funções de ativação, novos esquemas de inicialização de parâmetros da rede, e novos métodos de otimização (Chollet, 2018).

Por outro lado, há uma dificuldade em representar, por meio dessas redes *feed-forward*, padrões temporais relativos, assim como outros padrões sequenciais em geral. Para tanto, é necessário que a rede consiga, de alguma maneira, reter informações entre um evento e outro na sequência, manter alguma memória (Elman, 1990).

Nas **redes neurais recorrentes (RNNs)**, as camadas escondidas conseguem enxergar as suas prévias saídas, de forma que o comportamento subsequente pode ser influenciado por respostas anteriores, empregando assim uma memória. A Figura 5 possui maneiras formas de representar uma unidade recorrente.

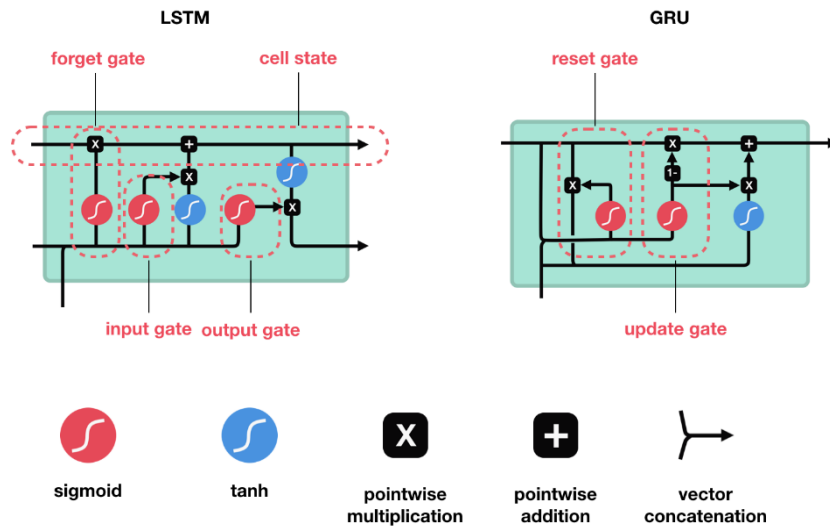
Figura 5 - Representações de uma unidade recorrente simples (RNN)



Por sua vez, RNNs tradicionais muito longas muitas vezes apresentam problemas de explosão dos pesos ou fenecimento dos gradientes (Hochreiter et al, 1997). Em função disso, foram propostas arquiteturas derivadas que possuem portas de controle nas unidades recorrentes, as redes Long Short-Term Memory (LSTM) e, posteriormente, as Gated Recurrent Unit (GRU). O funcionamento interno dessas unidades está representado na Figura 6.

As **redes neuronais convolucionais (CNNs)**, tradicionalmente associadas ao processamento de imagens, também podem ser empregadas para tarefas de análise de texto. Definido o tamanho da janela de entrada, e o número de canais de saída, uma convolução transforma uma sequência de entrada em uma sequência de saída, em que cada elemento da saída será formado pela aplicação de um produto entre uma subsequência de entrada, com o tamanho da janela, e um filtro (*kernel*) formado pelos pesos da rede. É possível visualizar o funcionamento da rede convolucional no tratamento de texto por meio da Figura 7.

Figura 6 - Representações de unidades LSTM e GRU.

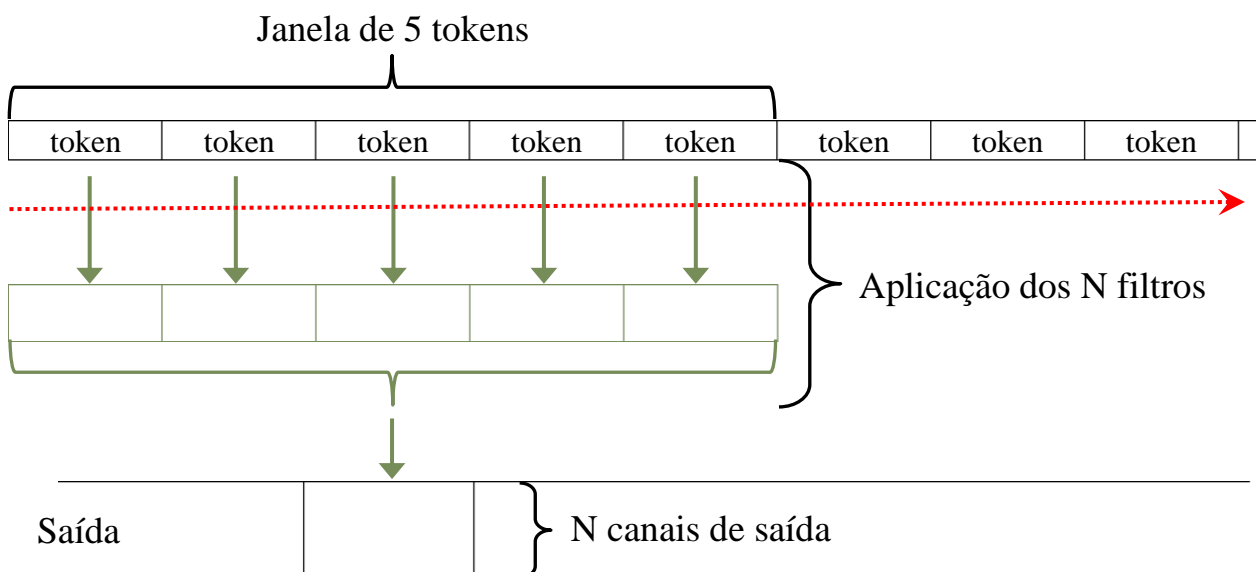


Fonte: [Nguyen, 2018](#).

Comumente, a cada camada convolucional se acrescenta uma operação de redução de dados (MaxPooling ou AveragePooling).

Redes convolucionais aprendem padrões posicionais dentro de um texto, e camadas sucessivas podem aprender hierarquias de padrões, construindo assim modelos mais complexos de interpretação. Redes convolucionais unidimensionais tem sido uma alternativa às redes recorrentes em tarefas de classificação de texto de treinamento mais rápido ([Chollet, 2018](#)).

Figura 7 - Funcionamento da convolução em uma dimensão.



Fonte: [Chollet, 2018](#).

2.7 AVALIAÇÃO

Uma vez construído o classificador, é desejável medir sua efetividade, ou seja, sua habilidade de tomar as decisões corretas de classificação. Assim, antes da construção do classificador, a coleção de documentos (corpus) é dividida em dois conjuntos (Sebastiani, 2001):

- **Conjunto de treino:** o classificador é construído pela observação das características desses documentos;
- **Conjunto de teste:** empregado para testar a efetividade do classificador. Para cada documento do conjunto, a decisão do classificador é comparada com a decisão dos especialistas (rótulo dos dados), e a medida da efetividade da classificação se baseia na frequência com a qual as decisões coincidem.

Para visualização dessa comparação entre as decisões de classificadores, podemos empregar a **matriz de confusão**, representada na Tabela 1. Para um conjunto de K classes, em uma classificação single-label, é uma matriz $K \times K$, em que as linhas representam classificações efetuadas pelo modelo (classe prevista), nas colunas temos rótulos dos dados (classe real) e, em cada célula, temos a quantidade de ocorrências daquela combinação de linha e coluna.

Tabela 1 - Matriz de confusão para classificação binária

		Classe real	
		Sim	Não
Classe prevista	Sim	verdadeiro positivo (VP)	falso positivo (FP)
	Não	falso negativo (FN)	Verdadeiro negativo (VN)

Fonte: Sebastiani, 2001.

No caso de uma classificação multi-classes, a matriz de confusão tem várias linhas e colunas, mas a tabela 1 também é útil, para uma análise classe a classe, se considerarmos a classe em estudo versus todas as demais: VP_i fornece a quantidade de coincidência de classificação prevista e real para a classe i . Já VN_i fornece a quantidade de vezes em que a classificação prevista não foi a classe i e a classificação real também não. FP_i conta as previsões para a classe em que a classe real é outra, e FN_i traz os documentos em que a classe real é i mas a previsão foi outra. $VP_i + FP_i$ nos dá as previsões para a classe i , e $VP_i + FN_i$ traz as ocorrências reais da classe i .

Precisão de um classificador para uma classe é dada pela fração de predições corretas sobre todas as previsões para aquela classe (Zaki et al, 2014):

$$precision_i = \frac{VP_i}{VP_i + FP_i}$$

Cobertura de um classificador para uma classe é dada pela fração de predições corretas sobre todas as ocorrências reais daquela classe (Zaki et al, 2014):

$$recall_i = \frac{VP_i}{VP_i + FN_i}$$

A **função $F\beta$** , ou $F\beta$ por classe, é dada por (Sebastiani, 2001):

$$F\beta_i = \frac{(\beta^2 + 1) \cdot precision_i \cdot recall_i}{\beta^2 \cdot precision_i + recall_i}$$

onde β pode ser visto como o grau relativo de importância entre a precisão e a cobertura, de acordo com o problema de classificação sendo tratado. $F0$ ($\beta=0$) coincide com a precisão, $F\infty$ coincide com o *recall*. $F1$, dada pela média harmônica entre as duas medidas, corresponde a uma mesma importância dada às duas medidas.

A **acurácia** é a fração de predições corretas sobre todo o conjunto (Zaki et al, 2014). O numerador corresponde à soma dos verdadeiros positivos de cada classe, em um problema multi-classe, ou a $VP+VN$, em um problema binário. O denominador corresponde ao número de documentos do conjunto de teste:

$$accuracy = \frac{\sum_{i=1}^K VP_i}{D_{Test}}$$

A **taxa de erro** é a fração de predições incorretas sobre todo o conjunto. Enquanto a acurácia traz a probabilidade de uma predição correta, a taxa de erro traz a probabilidade de uma predição incorreta (Zaki et al, 2014):

$$error\ rate = 1 - accuracy = \frac{\sum_{i=1}^K FP_i}{D_{Test}}$$

Para o caso de classificações multi-classe, precisão e cobertura podem também ser medidos globalmente, por meio de médias macro, ponderada e micro (Sebastiani, 2001 e Sokolova et al, 2009):

- As **médias macro** (*macroaveraging*) da precisão e da cobertura correspondem às médias aritméticas das medidas por classe. Desconsideram a prevalência entre as classes.
- As **médias ponderadas** (*weighted-averaging*) da precisão e da cobertura correspondem às médias aritméticas das medidas por classe ponderadas pelos tamanhos reais das classes.
- As **médias micro** (*macroaveraging*) da precisão e da cobertura são obtidas pela soma de todas as decisões por classe:

$$precision_{\mu} = \frac{\sum_{i=1}^K VP_i}{\sum_{i=1}^K (VP_i + FP_i)} \quad recall_{\mu} = \frac{\sum_{i=1}^K VP_i}{\sum_{i=1}^K (VP_i + FN_i)}$$

As medidas **F1 macro, ponderada e micro** são obtidas pelas médias harmônicas das precisões e coberturas macro, ponderada e micro, respectivamente. Para o caso das médias micro, temos que $\sum_{i=1}^K (VP_i + FP_i) = D_{test} = \sum_{i=1}^K (VP_i + FN_i)$, portanto:

$$\text{precisão micro} = \text{cobertura micro} = \text{F1 micro} = \text{acurácia}.$$

No caso de grande desbalanceamento entre as classes, as médias micro e ponderadas tendem a se aproximar dos resultados das classes mais prevalentes, enquanto as macro dão o mesmo peso para classes menos representativas.

Todas essas medidas podem ser calculadas também sobre o conjunto de treino, porém, como o modelo é otimizado observando as funções de erro ou custo sobre os dados de treino, é possível que os valores fiquem artificialmente altos. Ou seja, uma avaliação sobre documentos fora do conjunto de treino tenderia a ser pior. Como normalmente o que importa é a efetividade do classificador fora do conjunto de treino, as medidas neste conjunto não são consideradas para avaliação do modelo.

O conjunto de teste, por outro lado, sendo um conjunto de documentos disjuncto do conjunto de treino, tende a se aproximar mais de um desempenho real do modelo. Porém, sendo uma única escolha aleatória, o conjunto de teste pode também não apresentar a mesma distribuição que ocorreria em um uso real do classificador. O ideal seria testarmos sobre vários ou sobre todos os conjuntos possíveis de teste, de forma a neutralizar uma diferença da distribuição de uma medida de teste apenas, aproximando assim do desempenho real do classificador (Zaki et al, 2014).

A **validação cruzada** (*cross-validation*) divide a coleção de documentos (\mathcal{D}) em um determinado número N de partes iguais (*folds*). Cada parte (\mathcal{D}_i) uma por vez, é tratada como o

conjunto de teste, e as demais ($\mathcal{D} - \mathcal{D}_i$), em conjunto, formam o conjunto de treino. Um modelo M_i é treinado (usando o conjunto de treino) e avaliado (usando o conjunto de teste), produzindo uma métrica θ_i . O valor esperado para a métrica (μ_θ) pode então ser estimado como a média das medições, assim como a variância (σ_θ^2). Essa divisão pode ainda ser repetida múltiplas vezes, se for feita de forma aleatória e gerando particionamento diferente em cada repetição, produzindo assim mais valores (**validação cruzada repetida**).

Usualmente, o número de partes iguais é 5 ou 10. Um caso especial $N=D$, chamado **leave-one-out cross-validation**, funciona em cada passo com um único documento de teste e todos os demais para treinamento, gerando assim tantos valores de medida quanto documentos na coleção.

Uma vez estimadas a média e variância de uma métrica, podemos construir um **intervalo de confiança**. Para número de amostras pequenos ($N=5$ ou $N=10$, por exemplo), a distribuição t-Student traz uma estimativa do valor real da métrica a partir dos valores observados para os testes, com o grau de confiança α (Zaki et al, 2014):

$$\mu \in \left(\mu_\theta - t_{\alpha/2, N-1} \cdot \frac{\sigma_\theta}{\sqrt{N}}, \mu_\theta + t_{\alpha/2, N-1} \cdot \frac{\sigma_\theta}{\sqrt{N}} \right)$$

onde μ_θ e σ_θ são calculados a partir das medições, e t é uma variável cujo valor pode ser obtido a partir de tabelas como a Tabela 2, abaixo.

Tabela 2 - Alguns valores comuns para a variável t , na distribuição t-Student bicaudal

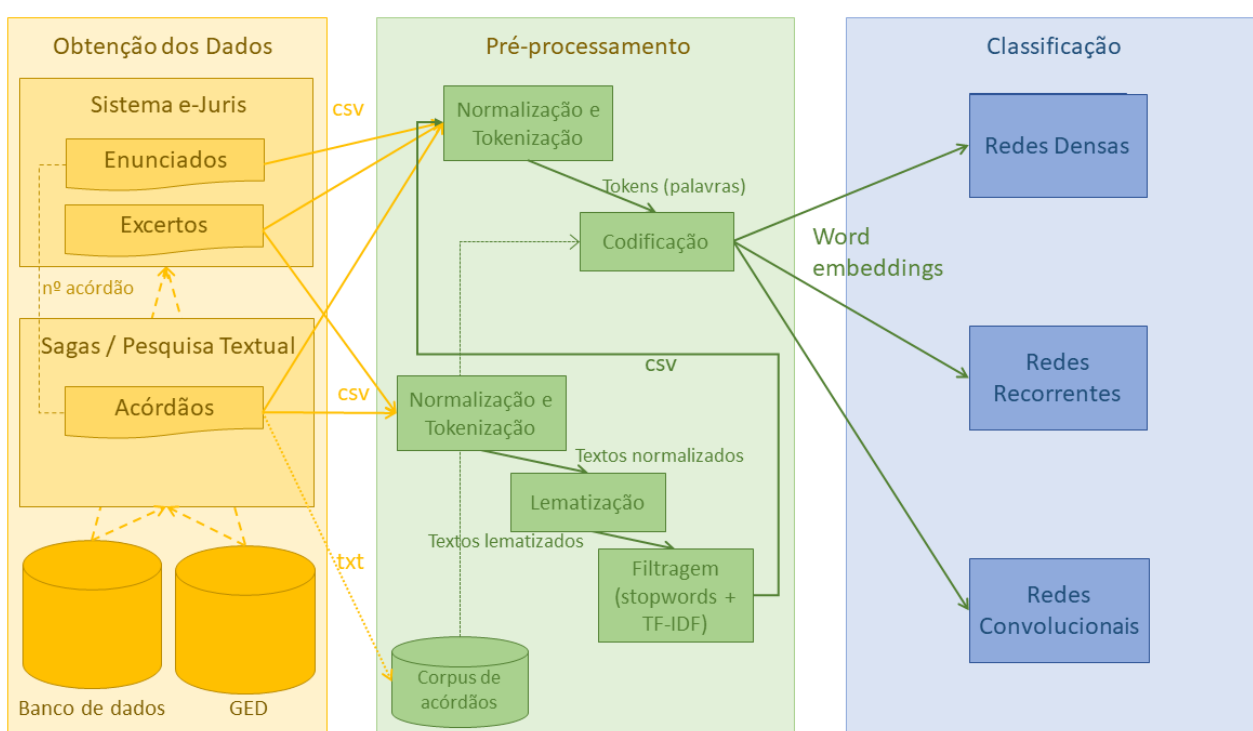
$t_{\alpha/2, N-1}$	$\alpha = 95\%$	$\alpha = 99\%$
$N - 1 = 4$	2,776	4,604
$N - 1 = 9$	2,262	3,250

3 PREPARAÇÃO

Para o atingimento da classificação dos textos de enunciados, excertos de acórdãos e inteiros teores de acórdãos, diversas atividades de obtenção e preparação dos dados se mostraram necessárias a fim de disponibilizar os dados para classificação pelas redes neurais.

O esquema geral das atividades realizadas e do fluxo dos dados desde os sistemas de origem até as redes que fazem a classificação está esboçado na Figura 8. Na sequência, serão detalhadas essas atividades.

Figura 8 - Fluxo dos dados para a classificação dos textos empregando redes neurais



Os testes com classificadores não neurais empregaram algumas etapas diferentes de pré-processamento, e não estão representadas na figura. Porém, a etapa de obtenção de dados é a mesma, e as especificidades de pré-processamento serão detalhadas a seguir.

3.1 OBTENÇÃO DOS DADOS

Ao invés de dados primários obtidos diretamente dos sistemas, foram obtidos preferencialmente dados secundários manipulados pelo sistema de pesquisa de jurisprudência, por já apresentarem algum tratamento de qualidade e cruzamento de informações, permitindo obter os resultados mais rapidamente.

As informações produzidas e manipuladas por meio dos sistemas estão disponíveis na base de dados corporativa de produção (Oracle), e os textos estão contidos em documentos, armazenados em um sistema de Gestão Eletrônica de Documentos (GED), denominado Sisdoc.

Os metadados principais da jurisprudência são obtidos da visão JURISPRUDENCIA.VW_JURISPRUDENCIA_SELECIONADA, entre eles, os códigos de documento do enunciado e do excerto, os códigos e descrições de área, tema e subtema, a chave do acórdão (número, ano e colegiado) e o tipo do processo (processo de instrução da auditoria).

A pesquisa de Jurisprudência contém uma tabela de *staging*, BUSCA_TEXTUAL.ACORDAO, que possui diversos metadados, bem como os identificadores dos documentos referentes ao acórdão. Entre eles, obtemos o número do documento que contém o inteiro teor do acórdão.

Empregando uma *servlet* pública do Sisdoc³, que recebe como entrada o código do documento, baixou-se cada documento utilizado (enunciados, excertos e inteiros teores dos acórdãos), em formato Word. O conteúdo textual do documento foi extraído por meio da ferramenta Tika⁴, obtendo assim os textos de enunciados, excertos e acórdãos.

As informações obtidas de enunciados⁵ e excertos⁶ são armazenadas em arquivos separados por vírgula (CSV).

O cruzamento da *view* da Jurisprudência Seleccionada com a tabela de *staging* da Busca Textual por meio da chave do acórdão (número + ano + colegiado), produz a relação de acórdãos com enunciados de jurisprudência e, conseqüentemente, com áreas de jurisprudência. De cada enunciado é extraído um único acórdão, porém enunciados diferentes podem ser extraídos de pontos diferentes (excertos) de um mesmo acórdão. Como cada enunciado é classificado em uma área de jurisprudência, um acórdão pode estar relacionado a uma ou mais áreas, conforme ilustrado na Tabela 3. Por consequência, foram criados dois arquivos para pré-processamento e classificação: um contendo apenas os acórdãos com uma única área associada⁷, e outro contendo todos os acórdãos, inclusive os com mais de uma área associada (*multi-label*)⁸.

No caso de registros de acórdãos com mais de uma área, pode-se considerar uma combinação de áreas como uma classe separada, convertendo assim um problema *multi-label* em

³ Servlet do Sisdoc para download de documento: <https://contas.tcu.gov.br/egestao/ObterDocumentoSisdoc>

⁴ Apache Tika™ toolkit: <https://tika.apache.org>

⁵ Enunciados: https://github.com/leonardo3108/clacjur/blob/master/dados/jurisprudencia_seleccionada_enunciados.csv

⁶ Excertos (arquivo compactado): https://github.com/leonardo3108/clacjur/blob/master/dados/jurisprudencia_seleccionada_excertos.zip

⁷ Acórdãos com área única: <https://github.com/leonardo3108/clacjur/blob/master/dados/acordaos-unicos.csv>

⁸ Todos os acórdãos: <https://github.com/leonardo3108/clacjur/blob/master/dados/acordaos-seleccionada.csv>

single-label. A desvantagem é que algumas novas classes derivadas serão pouco representativas, dificultando a tarefa do classificador.

Além disso, arquivos com os inteiros teores de quase 314 mil acórdãos foram armazenadas em arquivos do Google Drive⁹, formando um corpus para construção de *embeddings*.

Tabela 3 - Enunciados de jurisprudência por acórdão

Enunciados	Quantidade	Parte	Acumulado
1	7.236	82,8%	82,8%
2	1.113	12,7%	95,5%
3	230	2,63%	98,2%
4	89	1,02%	99,2%
5	39	0,45%	99,6%
6	12	0,14%	99,8%
7	7	0,08%	99,9%
8	4	0,05%	99,9%
9	2	0,02%	99,9%
10	1	0,01%	99,9%
11	2	0,02%	100,0%
12	1	0,01%	100,0%
13	2	0,02%	100,0%

3.2 PRÉ-PROCESSAMENTO

3.2.1 Classificadores não neurais

Os textos de entrada – enunciados de jurisprudência, excertos de acórdãos e inteiros teores de acórdãos – são pré-processados, com os seguintes passos:

- Normalização: caracteres minúsculos e remoção de pontuações
- Remoção de *stopwords*: conforme lista do português da biblioteca nltk
- Stemização: remoção da terminação das palavras conforme PorterStemmer

As informações intermediárias são armazenadas em arquivos separados por vírgula (CSV), e em memória, mantidas em dataframe pandas¹⁰ e matrizes NumPy¹¹.

Os textos stemizados são transformados em vetores esparsos por meio de técnica de *bag of words* (BOW) com TFIDF, com vocábulos de diferentes tamanhos.

⁹ <https://drive.google.com/drive/folders/1ElAzFzUtG7WTqx5VUhOLpSmOfQwVZ0v4?usp=sharing>

¹⁰ Pandas data analysis and manipulation tool: <https://pandas.pydata.org>

¹¹ NumPy package for scientific computing in Python: <https://docs.scipy.org/doc/numpy/index.html>

Os algoritmos de mineração utilizam com dados de saída as classificações em áreas, e como dados de entradas as BOW de enunciados, excertos e acórdãos. Eventualmente, são utilizados também os tipos de processo (transformados em variáveis *dummy*), BOW de excertos de votos e temas e subtemas.

3.2.2 Classificação por redes neurais

Os textos de entrada – enunciados de jurisprudência, excertos de acórdãos e inteiros teores de acórdãos – são tokenizados em palavras, retirando ainda pontuações e transformando as letras em minúsculas, meio da biblioteca de pré-processamento de textos do framework Keras¹².

Os textos de entrada, tokenizados, são transformados em uma matriz onde cada linha corresponde a um texto de entrada, e cada coluna corresponde a um token, até um limite definido de tokens. Textos com menos tokens são completados, e com mais são recortados. Foi empregada a biblioteca de pré-processamento de sequências do Keras¹³.

Para codificação, foi empregada a técnica de *word embeddings*, provida como primeira camada da rede neural do Keras¹⁴, em cinco variações:

1. *Embeddings* inicializados randomicamente, sem pré-treino, com os valores se ajustando junto com a rede neural de classificação.
2. *Embeddings* fornecidos¹⁵ pelo Núcleo Interinstitucional de Linguística Computacional da Universidade de São Paulo (NILC-USP), pré-treinados a partir de textos gerais em Língua Portuguesa, com 50 ou 100 dimensões, construídos por meio de técnica Word2Vec Continuous Bag of Words (CBOW). Valores mantidos fixos durante o treinamento.
3. Idem ao anterior, mas os valores podendo se ajustar junto com o resto da rede.
4. *Embeddings* construídos a partir dos textos de todos os acórdãos, utilizando a biblioteca Gensim¹⁶, com 50 ou 100 dimensões, construídos por meio de técnica Word2Vec CBOW. Valores mantidos fixos durante o treinamento.
5. Idem ao anterior, mas os valores podendo se ajustar junto com o resto da rede.

¹² Keras Documentation - Text Preprocessing: <https://keras.io/preprocessing/text/>

¹³ Keras Documentation - Sequence Preprocessing: <https://keras.io/preprocessing/sequence/>

¹⁴ Keras Documentation - Embedding Layers: <https://keras.io/layers/embeddings/>

¹⁵ Repositório de Word Embeddings do NILC-USP: <http://nilc.icmc.usp.br/embeddings>

¹⁶ Gensim - Python framework for Vector Space Modelling: <https://pypi.org/project/gensim/>

A diferença dos *embeddings* com e sem pré-treino está no comando de criação da camada de *embeddings*, em que se pode passar a matriz de *embeddings*, contendo os pesos pré-treinados:

```
weights=[embedding_matrix]
```

Além disso, deve-se marcar o parâmetro que informa se esses valores dos pesos se manterão fixos ou ajustáveis durante o treinamento da rede neural. Por exemplo, para manter os pesos fixos, deve-se marcar:

```
trainable=False
```

Para montar a matriz de *embeddings*, é antes necessário que o modelo com o vocabulário e respectivos *embeddings* sejam carregados:

```
from gensim.models import KeyedVectors
model = KeyedVectors.load_word2vec_format('model.w2v')
```

Então a matriz de *embeddings* é construída, com base no vocabulário construído pelo tokenizador sobre os textos de entrada, inicializada com vetores nulos, e as palavras encontradas também no vocabulário Word2Vec tem seus valores de *embeddings* copiados. As palavras dos textos que não estiverem no modelo vetorial (OOVs) são mantidas com vetor nulo:

```
embedding_matrix = np.zeros((vocabulario, dim_vetor))
ok = 0
for word, i in tokenizer.word_index.items():
    if word in model:
        embedding_matrix[i] = model[word]
        ok += 1
print('Vocabulario:', i)
print('Encontrados no modelo:', ok, '=', ok * 100. / i)
```

As áreas são pré-processadas pelo LabelBinarizer¹⁷, que transforma as dez áreas em uma codificação binária one-hot, da biblioteca Scikit-learn¹⁸.

¹⁷ <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelBinarizer.html>

¹⁸ Scikit-learn - machine learning library: <https://scikit-learn.org/stable/index.html>

3.2.3 Redução dos dados

Textos longos, como excertos e acórdãos, podem criar dificuldades em tarefas de classificação. Foram testados fluxos de treinamento em que a camada de entrada da rede foi reduzida por meio de filtragem. Para facilitar o trabalho de filtragem, por meio de redução da variabilidade dos tokens, foram empregados também técnicas de tratamento de caracteres e lematização dos tokens.

O tratamento preliminar ocorreu por meio de expressões regulares para ajustes e trocas de palavras. Em seguida, procedeu-se à lematização dessas palavras empregando o analisador gramatical CoGrOO4¹⁹, por meio da interface em Python CoGrOO4Py²⁰. Para os *tokens* lematizados, foi calculado o TD-IDF entre todos os registros de texto. Foram filtrados, inicialmente, as *stopwords* e *tokens* com até três caracteres. Finalmente, para cada texto, com originalmente x tokens, dada uma meta de restringi-lo em até y tokens, calculou-se um limite mínimo para filtrar os $x-y$ tokens cujo TF-IDF é menor que esse limite, produzindo assim sequências de até y palavras, que entravam para as redes neurais. Ou seja, empregou-se o TF-IDF não como técnica de codificação, mas como input para uma rotina de filtragem de tokens.

A saída desta etapa de pré-processamento é armazenada em arquivos CSV com os textos pré-processados, que servirão como entrada para a rotina principal de pré-processamento. As reduções planejadas foram, sobre excertos, para 1000 palavras; sobre acórdãos, para 6000 e para 500 palavras.

3.2.4 Geração de embeddings pré-treinados sobre acórdãos

Conforme citado anteriormente, entre as cinco variações de uso de *embeddings*, foram empregadas 2 fontes diferentes de *embeddings* pré-treinados. A primeira foi do NILC-USP, com *embeddings* disponibilizados para download, em formatos de vetores de 50 e 100 valores, e gerados a partir de textos literários e jornalísticos em português brasileiro e europeu. A segunda foi dos acórdãos, criada no contexto deste trabalho, cuja estratégia será descrita aqui.

Os *embeddings* foram gerados a partir da classe Word2Vec²¹, da biblioteca Gensim, que recebe como entrada as sentenças de textos a serem empregadas como corpus. Pela grande quantidade de informação, ao invés de fornecer uma estrutura em memória, forneceu-se um

¹⁹ CoGrOO - Corretor Gramatical acoplável ao LibreOffice: <http://cogroo.sourceforge.net/index.html>

²⁰ Interface para o analisador morfológico do CoGrOO em Python: <https://github.com/gpassero/cogroo4py>

²¹ gensim Word2vec embeddings: <https://radimrehurek.com/gensim/models/word2vec.html>

iterador que vai abrindo os arquivos em disco e processando como próxima sentença o texto de cada novo arquivo (em minúsculas, quebrado em palavras com mais até 100 caracteres):

```
import os
from nltk.tokenize import word_tokenize

class MinhasSentencas(object):
    def __init__(self, pasta):
        self.pasta = pasta

    def __iter__(self):
        for arq in os.listdir(self.pasta):
            texto = ''
            for linha in open(os.path.join(self.pasta, arq), 'r'):
                texto += linha.lower()
            tokens = word_tokenize(texto)
            tokens_ok = []
            for token in tokens:
                if len(token) <= 100:
                    tokens_ok.append(token)
            yield tokens_ok
```

Construída a classe de iteração pelos textos do corpus, basta disparar o Word2Vec, no caso a seguir, com vetores de 50 valores:

```
import gensim
import multiprocessing
from gensim.models import Word2Vec

sentences = MinhasSentencas('../externos/acordaos-todos/')
model = Word2Vec(sentences,
                  max_vocab_size=1000000,
                  min_count=20,
                  window=2,
                  size=100,
                  sample=6e-5,
                  alpha=0.03,
                  min_alpha=0.0007,
                  negative=20,
                  workers=multiprocessing.cpu_count()-1,
                  iter=10)
```

A construção dos *embeddings* é formada por duas etapas:

- A primeira consiste na construção do vocabulário, coletando todas as palavras e sua frequência no corpus. Palavras com frequência baixa são removidas.
- A segunda consiste na efetiva construção dos *embeddings*, por meio do treinamento de uma rede neural com uma única camada escondida, que resolve um problema definido pelo algoritmo CBOW, e cujos pesos são justamente os valores dos *embeddings* que se quer extrair.

3.3 MODELOS DE CLASSIFICAÇÃO

Diversas arquiteturas de redes neurais foram empregadas para as tarefas de classificação de enunciados, excertos de acórdãos e inteiros teores de acórdãos.

Todos os modelos foram treinados usando equipamento com CPU Core i5-7400 3GHz de 4 núcleos; 16 GB de RAM; GPU GeForce GTX 1070 com 1920 núcleos e 8 GB de VRAM; HD 1 TB; sistema operacional 64 bits Ubuntu 18.04.3 LTS, GNOME 3.28.2.

Os treinamentos de classificação com redes neurais foram feitos empregando framework Keras²² sobre TensorFlow²³ 1.0, empregando GPU para treinamento e avaliação dos modelos. A função de perda empregada foi a entropia cruzada categórica, e a métrica de validação acurácia categórica.

Primeiramente, eram testadas diversas combinações de configurações de redes e hiperparâmetros, e se anotava a melhor acurácia obtida e o tempo de treinamento da rede. Algumas configurações foram selecionadas para comparações empregando validação cruzada com 10 folds e scores F1, por meio da biblioteca scikit-learn, e calculando os intervalos de confiança $\alpha=99\%$ considerando distribuição T-student ($t = 3,250$).

Mais comum foram os treinamentos em 20 épocas, exceto alguns (*) com 50 épocas, e em mini-batch de 32 registros, exceto alguns com (**) 64 ou (***) 128. *Embeddings* com valores ajustáveis durante a classificação, exceto alguns, mantidos fixos (****).

3.3.1 Modelos não neurais

Com o propósito de estabelecer uma linha básica de comparação para as redes neurais, foram treinados modelos empregando algoritmos *logistic regression*, *naive bayes*, *random forests* e *linear discriminant analysis*. Utilizou-se a biblioteca scikit-learn e, em geral, empregou-se valores padrões de parâmetros. Foi efetuada validação cruzada com 10 folds e obtenção de scores F1, com alfa de 99% e considerando distribuição t-student.

3.3.2 Redes neurais densas

Segue, abaixo, um exemplo de utilização de rede neural densa. No caso, emprega *word embeddings* sem pré-treino. No caso das redes densas, é necessária uma conversão (*Flatten*) do formato tridimensional da saída dos embeddings (por exemplo, 13312 enunciados x 200 *tokens* x 50 valores por *token*) para o formato unidimensional.

²² Keras high-level neural networks API in Python: <https://keras.io>

²³ TensorFlow platform for machine learning: <https://www.tensorflow.org>

No exemplo abaixo, foi utilizada uma divisão de 80% dos dados para treinamento e 20% para validação. Treinamento com batch de 32 sequências por vez, otimização *adadelta*. Por se tratar de classificação multi-classe, empregou-se como função de perda entropia cruzada categórica. A rede é formada por uma camada de 2048 neurônios, seguida por outra de 1024. Funções de ativação ReLU nas camadas escondidas e softmax na saída. Para regularização, *dropout* de 60% dos neurônios das duas camadas. Treinamento em 50 épocas.

Uma grande desvantagem das redes densas para processamento de textos é o tamanho da entrada da rede, o que exige muita memória da GPU. Para classificação de acórdãos, onde as sequências são maiores e, o número de sequências também (no caso do inteiro teor), houve a necessidade de empregar redes maiores ou realizar algum pré-processamento extra para reduzir o tamanho da entrada.

```
from keras.models import Sequential
from keras.layers import Flatten, Dense, Embedding
from keras.layers.core import Dropout

model = Sequential()
model.add(Embedding(vocabulario, dim_vetor, input_length=x.shape[1]))
model.add(Flatten())
model.add(Dense(2048, activation='relu'))
model.add(Dropout(0.6))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.6))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adadelta',
              metrics=["categorical_accuracy"])
model.fit(x, y, epochs=50, batch_size=32, validation_split=0.2,
        verbose=1, shuffle=True)
```

3.3.3 Redes neurais recorrentes

Foram utilizadas redes recorrentes simples ou duplas de tipo RNN, LSTM e GRU. Eventualmente, uma camada extra de unidades recorrentes ou densa.

No exemplo abaixo, foram utilizados *embeddings* sem pré-treino, divisão de 80% dos dados para treinamento e 20% para validação. Treinamento com *batch* de 32 sequências por vez, otimização *adadelta*, função de perda entropia cruzada categórica. A rede é formada por uma camada simples contendo 256 neurônios recorrentes, saída *softmax*, e sem *dropout*. Treinamento em apenas 20 épocas.

Como a rede recorrente processa os dados de forma sequencial, entrada fica menor, impactando menos a GPU. Em compensação, o tempo de treinamento cresce bastante.

```

from keras.models import Sequential
from keras.layers import SimpleRNN, Dense, Embedding

model = Sequential()
model.add(Embedding(vocabulario, dim_vetor, input_length=x.shape[1]))
model.add(SimpleRNN(256))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adadelta',
              metrics=["categorical_accuracy"])
model.fit(x, y, epochs=20, batch_size=32, validation_split=0.2,
        verbose=1, shuffle=True)

```

3.3.4 Redes neurais convolucionais

Também foi testado o uso de redes convolucionais para as tarefas de classificação. Essas redes foram montadas com camadas convolucionais, seguidas de camadas de *pooling*.

No exemplo abaixo, foram utilizados *embeddings* sem pré-treino, divisão de 80% dos dados para treinamento e 20% para validação. Treinamento com batch de 32 sequências por vez, otimização RMSprop, função de perda entropia cruzada categórica. A rede é formada por uma camada convolucional, seguida por uma camada de MaxPooling, seguida por outra camada de convolucional e outra de MaxPooling. Funções de ativação ReLU nas camadas escondidas e softmax na saída, sem dropout. Treinamento em apenas 20 épocas.

Como as redes convolucionais possuem menos ligações entre neurônios que as redes densas comuns, conseguem processar uma quantidade maior de dados de entrada com menos impacto sobre a GPU. Além disso, como os dados são processados ao mesmo tempo, o tempo de treinamento é bem menor que em redes recorrentes.

```

from keras.models import Sequential
from keras.layers import Embedding, Dense
from keras.layers import Conv1D, MaxPooling1D, GlobalMaxPooling1D
from keras.optimizers import RMSprop

model = Sequential()
model.add(Embedding(vocabulario, dim_vetor, input_length=x.shape[1]))
model.add(Conv1D(32, 7, activation='relu'))
model.add(MaxPooling1D(5))
model.add(Conv1D(32, 7, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=RMSprop(),
              metrics=["categorical_accuracy"])
model.fit(x, y, epochs=20, batch_size=32, validation_split=0.2,
        verbose=1, shuffle=True)

```


4 RESULTADOS

4.1 DESCRIÇÃO DOS DADOS

4.1.1 Enunciados de jurisprudência

Os registros obtidos de enunciados de jurisprudência foram armazenados em um arquivo CSV contendo os seguintes campos:

- COD: chave artificial da tabela de enunciado de jurisprudência;
- NUM_ENUNCIADO: número sequencial do enunciado, gerido pela Seses;
- COD_AREA: código de área, primeiro nível de classificação;
- DESCR_AREA: descrição da área de jurisprudência;
- COD_TEMA: código de tema, segundo nível de classificação;
- DESCR_TEMA: descrição do tema de jurisprudência;
- COD_SUBTEMA: código de tema, terceiro nível de classificação;
- DESCR_SUBTEMA: descrição do subtema de jurisprudência;
- COD_DOC_TRAMITAVEL_ENUNCIADO: código do documento contendo o texto do enunciado, no sistema de Gestão Eletrônica de Documentos (GED);
- TEXTO_ENUNCIADO: texto do enunciado, extraído do GED
- ACORDAO: número do acórdão associado ao enunciado
- TIPO_PROCESSO: tipo do processo associado ao acórdão

Foram obtidos 13.312 registros, alguns deles mostrados na tabela 4. Apenas alguns campos foram selecionados, e seus nomes de coluna foram ajustados para reduzir a largura.

Tabela 4 - Exemplos de registros de jurisprudência.

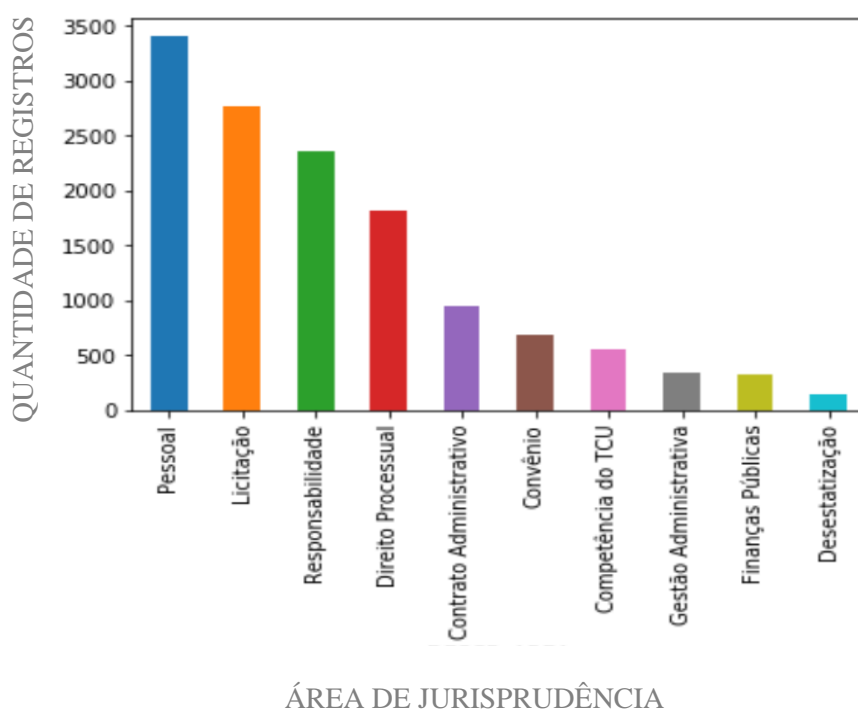
AREA	TEMA	SUBTEMA	TEXTO_ENUNCIADO	ACORDAO
Responsabilidade	Solidariedade	Benefício previdenciário	Não comprovada a participação do beneficiário ...	Acórdão 297/2016 - PL
Finanças Públicas	Exportação	Petróleo	A operação ficta de exportação de plataformas ...	Acórdão 366/2016 - PL
Responsabilidade	Multa	Dosimetria	No âmbito do TCU, a dosimetria da pena tem com...	Acórdão 944/2016 - PL
Direito Processual	Princípio da independência das instâncias	Decisão judicial	O princípio da independência das instâncias pe...	Acórdão 30/2016 - PL
Pessoal	Sistema S	Nepotismo	É vedado aos dirigentes das entidades do Siste...	Acórdão 55/2016 - PL

Atualmente, os enunciados de jurisprudência são distribuídos em dez áreas, distribuídos conforme demonstrado no gráfico da Figura 9. Podemos observar grande desbalanceamento entre as classes, com as quatro classes mais representativas apresentando mais de 1000 ocorrências, três classes medianas, contendo entre 500 e 1000 registros, e três classes mais raras, com poucas centenas de enunciados. São 86 temas, dos quais os 10 mais representativos são:

• Convênio	614
• Obras e serviços de engenharia	390
• Aposentadoria	379
• Débito	376
• Pensão civil	335
• Qualificação técnica	324
• Remuneração	320
• Tempo de serviço	307
• Multa	304
• Ato sujeito a registro	285

Os textos de enunciado possuem entre 6 e 633 palavras, com uma média de 45. Apenas 15 registros, ou seja, pouco mais de 0,11% do total, possuem mais de 200 palavras, por isso este foi o limite definido para as sequências fixas de textos de enunciados para entrada nas redes neurais.

Figura 9 - Distribuição de enunciados por área de jurisprudência



4.1.2 Excertos de acórdãos

Os registros obtidos de excertos de acórdãos para jurisprudência foram armazenados em um CSV similar ao de enunciados, trocando apenas os campos relacionados ao texto:

- **COD_DOC_TRAMITAVEL_EXCERTO**: código do documento contendo o texto do excerto, no sistema de Gestão Eletrônica de Documentos (GED);
- **TEXTO_EXCERTO**: texto do excerto, extraído do GED

Foram obtidos 13.285 registros de enunciados com excerto cadastrado. A distribuição por área, tema e subtema foi muito similar à anterior. Na tabela a seguir, dados de exemplo.

Os textos dos excertos enunciado possuem entre 34 e 5.324 palavras, com uma média de 506. Foram 297 registros, ou seja, pouco mais de 2% do total, com mais de 2000 palavras, por isso este foi o limite definido para as sequências fixas de textos de excertos para entrada nas redes neurais.

Tabela 5 - Exemplos de excertos de acórdãos

ENUNCIADO	COD_DOC_EXCERTO	TEXTO_EXCERTO
1236	54995438	Voto:Cuidam os autos de tomada de contas espec...
1534	55025603	Voto:Cuidam os autos de Solicitação do Congres...
5314	55455375	Relatório:Trata-se de embargos de declaração o...
40	54773747	Voto:8. Em relação a outros processos judiciais...
54	54773403	Voto:11. Relativamente ao ato envolvendo a Sra...

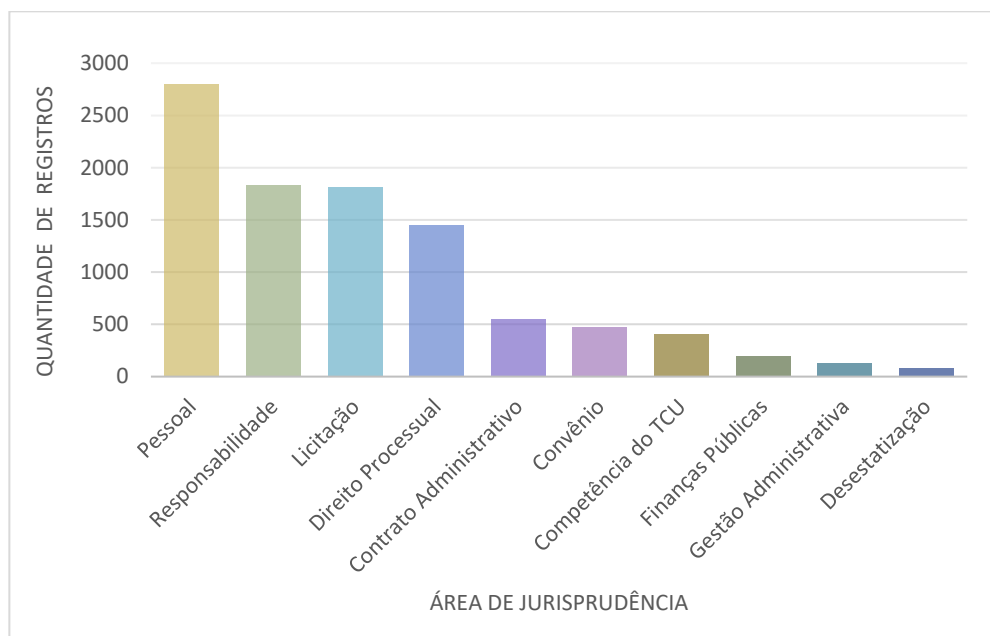
4.1.3 Inteiros teores de acórdãos

Os registros obtidos de inteiros teores de acórdãos para classificação foram armazenados em dois arquivos CSV, um contendo apenas os acórdãos relacionados a uma única área de jurisprudência, outro contendo também os acórdãos relacionados a mais de uma área de jurisprudência. Ambos com a mesma estrutura:

- **ACORDAO**: chave condensada do acórdão, no formato número/ano-colegiado; o sistema de Jurisprudência referencia, em geral, os acórdãos mais recentes, para estes a chave identifica univocamente;
- **ARQUIVO**: nome do arquivo tal como baixado do GED; útil apenas para auditoria e debug do fluxo de dados;
- **AREAS**: área de jurisprudência relacionada ao acórdão, ou múltiplas áreas separadas por vírgulas
- **TEXTO**: texto do inteiro teor do acórdão

Foram obtidos 9.739 registros de acórdãos referenciados pela jurisprudência, com área única. A distribuição por área foi similar às anteriores, com algumas trocas de posição, conforme se pode observar no gráfico da Figura 10.

Figura 10 - Distribuição de acórdãos referenciados por área única de jurisprudência

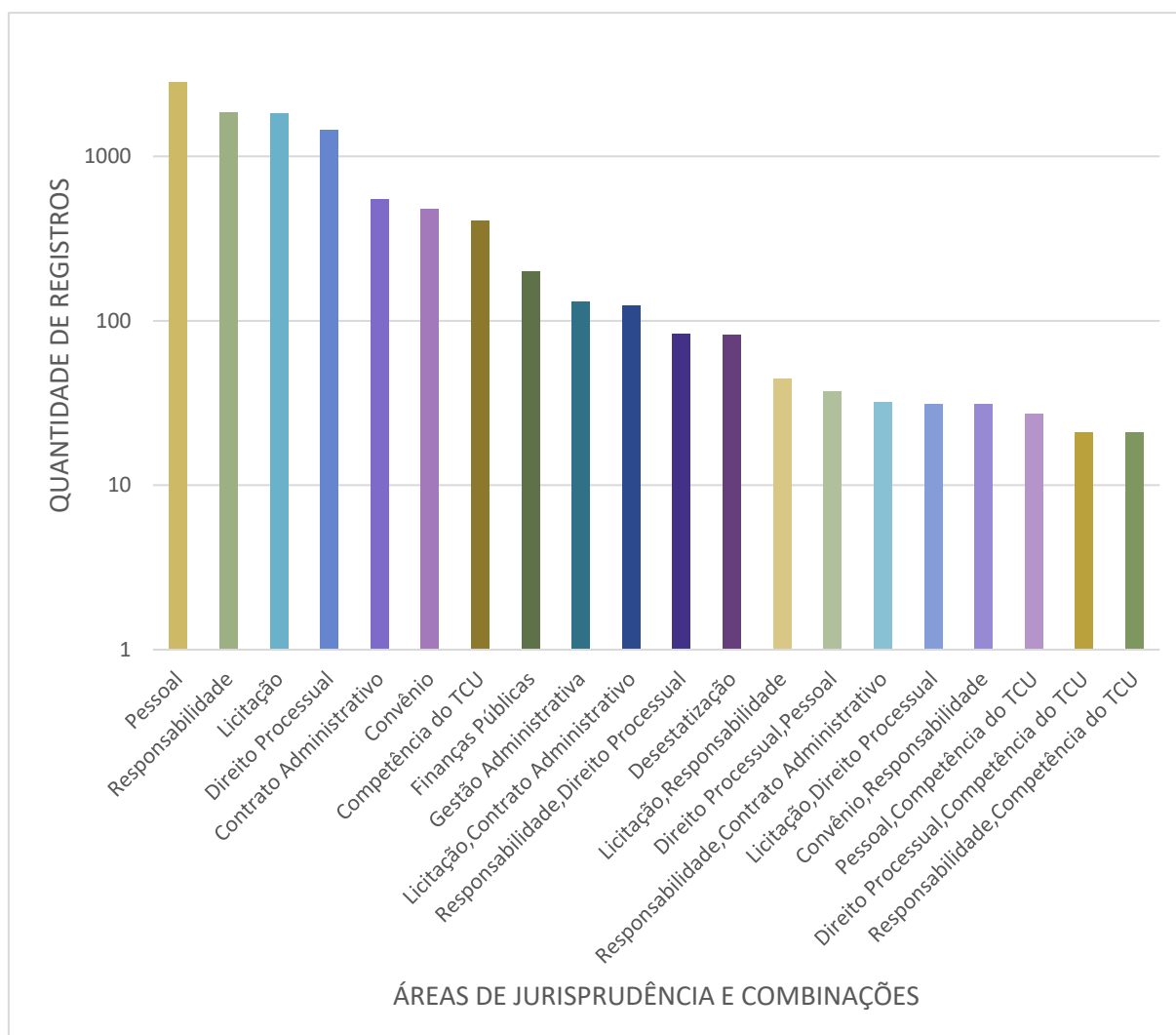


Se adicionarmos os acórdãos relacionados a mais de uma área de jurisprudência, chegamos a 10.524 acórdãos referenciados. Chega-se a 98 combinações diferentes de áreas de jurisprudência, convertidas em classes para classificação. No gráfico da Figura 11, podemos visualizar as 20 combinações mais representativas, em escala logarítmica. Entre elas, as 10 áreas originais, além de 10 duplas. Entre as combinações menos representativas, temos trios de áreas.

Os textos dos acórdãos referenciados possuem entre 520 e 1.754.001 palavras, com uma média de 10.118, ou 9.417 entre acórdãos com única área. Foram 267 registros, ou seja, pouco mais de 2,5% do total, com mais de 40.000 palavras, por isso este foi o limite definido para as sequências fixas de textos de acórdãos para entrada nas redes neurais. Entre acórdãos com área única, foram 201, ou seja, pouco mais de 2% do total, com mais de 40.000 palavras.

Na formação do corpus para geração dos *embeddings*, foram baixados cerca de 314 mil acórdãos, e extraídos seus textos, ocupando cerca de 5,5 GB em arquivos do tipo txt.

Figura 11 - Distribuição de acórdãos por áreas combinadas de jurisprudência.
Mostrando as 20 combinações mais representativas, em escala logarítmica.



4.2 PRÉ-PROCESSAMENTO

4.2.1 Pré-processamento para redes neurais

Na tabela 6, visualizam-se informações sobre o pré-processamento de enunciados e excertos. O tamanho do vocabulário é a quantidade de palavras distintas nos textos de enunciados, excertos e acórdãos. O tamanho da sequência é um valor máximo a partir do qual os textos são recortados. Textos menores são completados (padded). A matriz de *embeddings* terá quantidade de linhas igual ao tamanho do vetor dos *embeddings*, e de colunas igual ao tamanho do vocabulário, mais um (posição zero é reservada).

No caso da filtragem de excertos, a redução da sequência dos textos permitiu o aumento da dimensionalidade dos *embeddings* e do número de neurônios na primeira camada sem estourar o uso da memória da GPU (VRAM).

Tabela 6 - Informações sobre o pré-processamento de enunciados e excertos

Atividade	Métrica	Enunciados	Excertos	
			Originais	Filtrados
Obtenção	Quantidade de textos	13.312	13.285	13.285
Tokenização	Vocabulário gerado	15.387	63.925	22.972
	Tamanho da sequência	200	2.000	500
Embeddings	Tamanho do vetor	100	50	100
	Posições na matriz	1.538.800	3.196.300	2.297.300
NILC-USP	Palavras encontradas	13.758	42.309	19.689
	Taxa	89,4%	66,2%	85,7%
Acórdãos	Palavras encontradas	14.572	43.317	18.860
	Taxa	94,7%	67,8%	82,1%

Na Tabela 7 estão valores referentes pré-processamento de acórdãos, em todas suas variações: áreas únicas x áreas múltiplas, sem filtragem x com filtragem, em 6000 e em 500 palavras.

Tabela 7 - Informações sobre o pré-processamento de acórdãos

Atividade	Métrica	Área única			Áreas múltiplas		
		Originais	Filtro 6k	Filtro 500	Originais	Filtro 6k	Filtro 500
Obtenção	Quantidade de textos	9.739	9.739	9.739	10.524	10.524	10.524
Tokenização	Vocabulário gerado	318.318	95.423	32.293	344.296	102.108	33.339
	Tamanho da sequência	40.000	6000	500	40.000	6000	500
Embeddings	Tamanho do vetor	50	50	100	50	50	100
	Posições na matriz	15.915.950	4.771.200	3.229.400	17.214.850	4.771.200	3.229.400
NILC-USP	Palavras encontradas	110.315	55.676	24.014	115.064	57.562	24.576
	Taxa	34,7%	58,3%	74,4%	33,4%	56,4%	73,7%
Acórdãos	Palavras encontradas	88.887	48.140	25.160	90.722	49.463	25.799
	Taxa	28,0%	50,4%	77,9%	26,4%	48,4%	77,4%

4.2.2 Geração de embeddings a partir de acórdãos

No caso da geração de *embeddings* de 50 valores, a etapa de construção do vocabulário demorou pouco mais de 5 horas, encontrando 853 mil palavras, sendo 640 mil palavras distintas. Foram descartadas as palavras com frequência menor que 20, chegando a 317 mil palavras.

A etapa de treinamento da rede neural durou pouco mais de 50 horas, em 10 épocas, processando pouco mais de 18 mil palavras por segundo.

Para geração dos *embeddings* com 100 valores, a etapa de construção do vocabulário foi muito similar, tanto em tempo gasto como em estatísticas de palavras. A etapa de treinamento da rede neural demorou cerca de três horas a mais.²⁴

4.3 CLASSIFICAÇÃO – TREINAMENTOS EXPLORATÓRIOS

4.3.1 Classificação de enunciados

A Tabela 8 traz registros de alguns treinamentos²⁵ realizados para classificação de enunciados por área de jurisprudência empregando *embeddings* sem pré-treino. No total, os treinamentos levaram cerca de 7 horas e meia, e o melhor valor de acurácia conseguido foi de 82,99%, com uma rede GRU de 256 unidades, e otimização RMSprop. Treinos com redes recorrentes em sua maioria bateram os 80%, mas demoravam 20 minutos ou mais cada. Redes LSTM trouxeram valores similares às GRU, e RNN piores. Redes bidirecionais não se saíram melhores que as redes simples, e foram os mais lentos. Redes densas FFNN atingiram cerca de 75% com otimização adadelta, mas foram bem mais rápidas. Neste caso, redes convolucionais atingiram níveis de acurácia um pouco menores que as RNNs, mas com um tempo de processamento bem menor, com menos de um minuto.

Estes resultados servem como uma noção geral, mas como não foram repetidos, trazem um grau de aleatoriedade que impede de serem estatisticamente relevantes. Por isso, alguns treinamentos foram repetidos depois utilizando técnicas de avaliação mais robustas.

²⁴ Cadernos disponíveis em: <https://github.com/leonardo3108/clacjur/tree/master/vocabularios>.

²⁵ Treinamentos de enunciados disponíveis em: <https://github.com/leonardo3108/clacjur/tree/master/enunciados>.

Tabela 8 - Treinamentos sobre enunciados com embeddings sem pré-treino

Arquitetura	Otimização	Melhor val cat acc	Duração
Densa 2048 + 1024 + dropout .6 *	adadelta	0,7544	00:15:00
Densa 2048 + 1024 + dropout .4	adadelta	0,7341	00:06:00
RNN 256	adadelta	0,6695	00:08:00
GRU 256	adam	0,8179	00:19:00
LSTM 256	adam	0,7998	00:23:20
Bi-RNN 256	adam	0,5543	00:11:40
Bi-GRU 256	adam	<u>0,8059</u>	00:33:20
Bi-LSTM 256	adam	0,8160	00:42:00
GRU 128	adam	<u>0,8040</u>	00:18:40
GRU 64	adam	<u>0,8002</u>	00:18:40
GRU 512	adam	<u>0,8077</u>	00:18:40
GRU 256 + densa 128	adam	0,8171	00:18:20
GRU 256 + densa 64	adam	0,7983	00:18:40
GRU 256 + densa 32	adam	<u>0,8081</u>	00:19:00
GRU 256 + dropout .2 + recorrente .2	adam	0,8126	00:21:40
GRU 256 + dropout .1 + recorrente .5	adam	<u>0,8066</u>	00:22:20
GRU 256	RMSprop	0,8299	00:19:00
GRU 256	adadelta	<u>0,8002</u>	00:18:40
GRU 256 + GRU 32	adam	0,8134	00:37:00
GRU 256 + dropout .1 + recorrente .5 + densa 128 + dropout .4	adam	<u>0,8040</u>	00:22:20
GRU 256 **	adam	0,8122	00:19:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,8258	00:00:40
Conv1D 32x7 + MaxPool 5 + Conv 32x7 + GRU 32 + drop .1 + recor .5	RMSprop	0,7875	00:04:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPool1D **	RMSprop	0,8183	00:00:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPool1D ***	RMSprop	<u>0,8006</u>	00:00:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPooling1D *	RMSp 2e-4	0,8198	00:01:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPool1D *	RMSp 5e-4	<u>0,8092</u>	00:01:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,8224	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 1e-2	0,8130	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	adam	<u>0,8029</u>	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	adadelta	0,7905	00:00:40
Conv1D 32x7 + MaxPool 5 + Conv 32x7 + GRU 32 + drop .2 + recor .2	RMSprop	0,7698	00:04:20
Conv1D 32x7 + drop .2 + MaxPool 5 + Conv 32x7 + drop .2 + GMaxPool	RMSprop	0,8216	00:00:40
Conv1D 32x7 + MaxPooling1D 10 + Conv1D 32x7 + GMaxPooling1D	RMSprop	<u>0,8032</u>	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,8186	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,8107	00:00:40

Observação: 20 ou 50* épocas; batch_size de 32, 64** ou 128***. Embeddings variáveis.

A Tabela 9 mostra alguns treinamentos realizados para classificação de enunciados por área de jurisprudência empregando *embeddings* pré-treinados do NILC-USP. No total, os treinamentos levaram pouco mais de oito horas, e o melhor valor de acurácia conseguido foi de 86,29%, com uma rede GRU de 256 unidades, otimização RMSprop, e *dropout* de 10% e recorrente de 50%. Treinos com redes recorrentes cujos *embeddings* foram deixados ajustáveis durante o treinamento da classificação em sua maioria foram melhores que com *embeddings* não pré-treinados. Redes FFNN apresentaram desempenhos piores, e redes convolucionais pouco melhores.

Tabela 9 - Treinamentos sobre enunciados com embeddings pré-treinados do NILC-USP

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 2048 + 1024 + dropout .6 ****	adadelta	0,6091	00:06:00
Densa 2048 + 1024 + dropout .4 ****	adadelta	0,6095	00:06:00
Densa 2048 + 256 + dropout .4 ****	adadelta	0,6140	00:08:00
Densa 2048 + 1024 + dropout .6	adadelta	0,6884	00:19:00
Densa 2048 + 1024 + dropout .4	adadelta	0,6748	00:23:20
Densa 2048 + 256 + dropout .4	adadelta	0,6973	00:11:40
GRU 256 ****	RMSprop	0,8269	00:18:00
GRU 256	RMSprop	<u>0,8487</u>	00:19:00
RNN 256	RMSprop	0,6568	00:07:40
LSTM 256	RMSprop	0,8573	00:23:00
Bi-RNN 256	RMSprop	0,6632	00:11:40
Bi-GRU 256	RMSprop	0,8532	00:33:00
Bi-LSTM 256	RMSprop	0,8547	00:42:00
GRU 128	RMSprop	<u>0,8483</u>	00:19:00
GRU 64	RMSprop	<u>0,8457</u>	00:19:00
GRU 512	RMSprop	0,8554	00:19:00
GRU 256 + densa 128	RMSprop	<u>0,8453</u>	00:19:20
GRU 256 + densa 64	RMSprop	<u>0,8445</u>	00:21:00
GRU 256 + densa 32	RMSprop	0,8573	00:19:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,8603	00:21:20
GRU 256 + dropout .1 + recorrente .5	RMSprop	0,8629	00:21:00
GRU 256	adam	<u>0,8389</u>	00:18:20
GRU 256	adadelta	<u>0,8393</u>	00:18:40
GRU 256 + GRU 32	RMSprop	<u>0,8490</u>	00:36:20
GRU 256 + dropout .1 + recorrente .5 + densa 128 + dropout .4	RMSprop	0,8584	00:21:00
GRU 256 **	RMSprop	<u>0,8494</u>	00:09:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPooling1D ****	RMSprop	0,7300	00:00:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,8235	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,8179	00:00:40
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 32 + drop.2 + recor.2	RMSprop	0,8077	00:04:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,8284	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7807	00:00:40
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 32 + drop.1 + recor.5	RMSprop	0,8183	00:04:20
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 64 + drop.2 + recor.2	RMSprop	0,8006	00:04:20

Observação: 20 ou 50* épocas; batch_size de 32, 64** ou 128***. Embeddings variáveis ou fixos****

A Tabela 10 mostra alguns treinamentos realizados para classificação de enunciados por área de jurisprudência empregando *embeddings* pré-treinados sobre os textos de acórdãos. No total, os treinamentos levaram pouco mais de 8 horas, e o melhor valor de acurácia conseguido foi de 87,27%, com uma rede GRU de 256 unidades, otimização RMSprop, e dropout de 10% e recorrente de 50%. Treinos com redes recorrentes cujos *embeddings* foram deixados ajustáveis durante o treinamento da classificação em sua maioria foram melhores que com *embeddings* não pré-treinados ou *embeddings* do NILC-USP. Redes FFNN e redes convolucionais apresentaram desempenho similar ao conseguido com *embeddings* sem pré-treino.

Tabela 10 - Treinamentos sobre enunciados com embeddings pré-treinados sobre acórdãos

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 2048 + 1024 + dropout .6 ****	adadelta	0,7075	00:05:40
Densa 2048 + 1024 + dropout .4 ****	adadelta	0,7078	00:05:40
Densa 2048 + 256 + dropout .4 ****	adadelta	0,7157	00:05:20
Densa 2048 + 1024 + dropout .6	adadelta	0,7214	00:06:00
Densa 2048 + 1024 + dropout .4	adadelta	0,7146	00:06:00
Densa 2048 + 256 + dropout .4	adadelta	0,7210	00:06:00
GRU 256 ****	RMSprop	0,8558	00:17:40
GRU 256	RMSprop	0,8674	00:19:00
RNN 256	RMSprop	0,6831	00:07:40
LSTM 256	RMSprop	0,8637	00:24:20
Bi-RNN 256	RMSprop	0,6620	00:12:00
Bi-GRU 256	RMSprop	0,8689	00:33:40
Bi-LSTM 256	RMSprop	0,8633	00:41:20
GRU 128	RMSprop	0,8487	00:18:20
GRU 64	RMSprop	0,8385	00:18:20
GRU 512	RMSprop	0,8659	00:18:40
GRU 256 + densa 128	RMSprop	0,8569	00:18:40
GRU 256 + densa 64	RMSprop	0,8551	00:19:00
GRU 256 + densa 32	RMSprop	0,8558	00:19:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,8716	00:21:20
GRU 256 + dropout .1 + recorrente .5	RMSprop	0,8727	00:21:20
GRU 256	adam	0,8588	00:18:40
GRU 256	adadelta	0,8592	00:18:40
GRU 256 + GRU 32	RMSprop	0,8573	00:36:00
GRU 256 + dropout .1 + recorrente .5 + densa 128 + dropout .4	RMSprop	0,8603	00:21:20
GRU 256 **	RMSprop	0,8607	00:09:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPooling1D ****	RMSprop	0,7841	00:00:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,8002	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,8014	00:00:40
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 32 + drop.2 + recor.2	RMSprop	0,8141	00:04:20
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 64 + drop.2 + recor.2	RMSprop	0,8111	00:04:20

Observação: 20 ou 50* épocas; batch_size de 32, 64** ou 128***. Embeddings variáveis ou fixos****

4.3.2 Classificação de excertos

A Tabela 11 mostra alguns treinamentos²⁶ realizados para classificação de excertos de acórdãos por área de jurisprudência empregando *embeddings* sem pré-treino. No total, os treinamentos levaram quase 94 horas, e o melhor valor de acurácia conseguido foi de 73,96%, com uma rede GRU de 256 unidades, otimização RMSprop, dropout de 10% e recorrente de 40%.

Alguns treinos com redes recorrentes GRU bateram os 70%, mas demoravam 3 horas ou mais cada. Redes LSTM trouxeram valores mais baixos, e RNN piores ainda. As redes bidirecionais testadas se saíram bem pior que as redes simples, e foram lentíssimas. Redes densas FFNN atingiram entre 65% e 70% com otimização adadelta, mas foram bem mais rápidas. As redes convolucionais foram testadas com *embeddings* de 100 valores, ao contrário das demais, que usaram 50, atingindo assim níveis de acurácia próximas às melhores RNNs, com um tempo de processamento menor. Foram testadas redes mistas, com camadas convolucionais seguidas de camadas recorrentes ou densas. As com camadas densas tiveram o melhor desempenho.

A Tabela 12 mostra alguns treinamentos realizados para classificação de excertos de acórdãos por área de jurisprudência empregando *embeddings* pré-treinados do NILC-USP. No total, os treinamentos levaram quase 90 horas, e o melhor valor de acurácia conseguido foi de 78,58%, com uma rede GRU de 1024 unidades, otimização RMSprop, *dropout* de 20% e recorrente de 20%. Treinos com redes recorrentes cujos *embeddings* foram deixados ajustáveis durante o treinamento da classificação em sua maioria foram melhores que com *embeddings* não pré-treinados. Redes FFNN apresentaram desempenhos bem piores, e redes convolucionais pouco piores que as redes recorrentes, apesar de ser utilizado vetor de 100 posições.

A Tabela 13 mostra alguns treinamentos realizados para classificação de excertos de acórdãos empregando *embeddings* pré-treinados sobre os textos de acórdãos. No total, os treinamentos levaram pouco mais de 87 horas, e o melhor valor de acurácia foi de 79,11%, com uma rede GRU de 256 unidades, otimização RMSprop, *dropout* de 20% e recorrente de 20%. A possibilidade de ajuste dos *embeddings* durante o treinamento não foi tão determinante nestes casos. Redes FFNN e redes convolucionais apresentaram desempenho inferior ao conseguido com redes recorrentes, embora as redes CNN tenham empregado vetor de 100 posições.

²⁶ Treinamentos de excertos disponíveis em: <https://github.com/leonardo3108/clacjur/tree/master/excertos>.

Tabela 11 - Treinamentos sobre excertos com embeddings sem pré-treino

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 512 + 256 + dropout .6 *	adadelta	0,6948	00:20:50
Densa 512 + 128 + dropout .4	adadelta	0,6504	00:06:00
RNN 256	adadelta	0,4253	01:12:00
GRU 256	adam	0,7000	02:59:00
LSTM 256	adam	0,6180	03:32:40
Bi-RNN 256	adam	0,4697	01:50:20
Bi-GRU 256	adam	0,6816	06:56:40
Bi-LSTM 256	adam	0,5431	07:03:00
GRU 128	adam	0,6643	03:36:40
GRU 64	adam	0,6417	03:02:00
GRU 512	adam	<u>0,7079</u>	03:00:40
GRU 256 + densa 128	adam	0,6654	03:01:40
GRU 256 + densa 64	adam	0,6549	02:59:20
GRU 256 + densa 32	adam	0,6507	03:03:20
GRU 256 + dropout .2 + recorrente .2	adam	0,6955	03:41:40
GRU 256 + dropout .1 + recorrente .5	adam	<u>0,7049</u>	03:36:20
GRU 256	RMSprop	0,7309	03:00:20
GRU 256	adadelta	0,6556	03:04:40
GRU 256 + GRU 32	adam	0,6763	06:14:40
GRU 256 + dropout .1 + recorrente .5 + densa 128 + dropout .4	adam	0,6782	03:37:00
GRU 256 **	adam	0,6733	01:30:40
GRU 256 + dropout .1 + recorrente .4	RMSprop	0,7396	03:37:40
GRU 512 + dropout .1 + recorrente .4	RMSprop	0,6037	01:58:20
GRU 1024 + dropout .1 + recorrente .4	RMSprop	0,4953	03:56:40
Bi-GRU 256 + dropout .1 + recorrente .4	RMSprop	<u>0,7012</u>	07:05:20
Bi-GRU 512 + dropout .1 + recorrente .4	RMSprop	0,5420	07:46:40
Bi-GRU 1024 + dropout .1 + recorrente .4	RMSprop	0,4321	00:45:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7140	00:02:00
Conv1D 32x7 + MaxPool 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7015	00:59:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,7162	00:02:00
Conv1D 32x7 + MaxPool 5 + Conv 32x7 + GRU 128 + drop .2 + recor.2	RMSprop	0,6839	01:03:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7008	00:02:20
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7369	00:02:20
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7317	00:03:00
Conv64x7 + MaxPool 5 + densa 2048 + drop.2 + densa 128 + drop.2	RMSprop	0,7196	00:07:00

Observação: 20 ou 50* épocas; batch_size de 32 ou 64**. Embeddings variáveis.

Tabela 12 - Treinamentos sobre excertos com embeddings pré-treinados do NILC-USP

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 512 + 256 + dropout .6 * ****	adadelta	0,5088	00:17:30
Densa 512 + 256 + dropout .4 * ****	adadelta	0,4595	00:16:40
Densa 512 + 128 + dropout .4 * ****	adadelta	0,4863	00:17:30
Densa 512 + 256 + dropout .6 *	adadelta	0,5868	00:18:20
Densa 256 + 128 + dropout .4 *	adadelta	0,5958	00:10:00
GRU 256 ****	RMSprop	0,7433	03:15:00
GRU 256	RMSprop	0,7655	03:21:40
RNN 256	RMSprop	0,4204	01:09:20
LSTM 256	RMSprop	0,7482	03:50:00
Bi-RNN 256	RMSprop	0,4539	01:46:00
Bi-GRU 256	RMSprop	0,7689	06:06:40
Bi-LSTM 256	RMSprop	0,7520	07:03:20
GRU 128	RMSprop	0,7633	03:05:00
GRU 64	RMSprop	0,7478	03:03:40
GRU 512	RMSprop	0,7723	03:03:00
GRU 256 + densa 128	RMSprop	0,7689	03:02:20
GRU 256 + densa 64	RMSprop	0,7595	03:01:40
GRU 256 + densa 32	RMSprop	0,7742	03:03:40
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7753	03:34:20
GRU 256 + dropout .1 + recorrente .5	RMSprop	0,7727	03:32:20
GRU 256	adam	0,7580	03:06:40
GRU 256	adadelta	0,7524	03:05:00
GRU 256 + GRU 32	RMSprop	0,7636	06:23:20
GRU 256 + dropout .1 + recorrente .5 + densa 128 + dropout .4	RMSprop	0,7746	03:38:00
GRU 256 + dropout .1 + recorrente .4	RMSprop	0,7749	03:36:40
Bi-GRU 256 + dropout .1 + recorrente .4	RMSprop	0,7753	07:10:00
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,7821	03:49:40
GRU 1024 + dropout .2 + recorrente .2	RMSprop	0,7858	03:58:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,6688	00:01:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,7087	00:02:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7204	01:00:00
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7061	00:02:00
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,6251	00:02:00

Observação: 20 ou 50* épocas; batch_size de 32. Embeddings variáveis ou fixos****

Tabela 13 - Treinamentos sobre excertos com embeddings pré-treinados sobre acórdãos

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 1024 + 512 + dropout .6 ****	adadelta	0,5368	00:12:40
Densa 1024 + 256 + dropout .4 ****	adadelta	0,6014	00:12:20
Densa 512 + 256 + dropout .4 ****	adadelta	0,5886	00:06:40
Densa 1024 + 512 + dropout .6	adadelta	0,5679	00:14:00
Densa 1024 + 256 + dropout .4	adadelta	0,5962	00:13:40
Densa 512 + 256 + dropout .4	adadelta	0,5819	00:08:00
GRU 256 ****	RMSprop	0,7731	03:03:20
GRU 256	RMSprop	<u>0,7644</u>	03:00:00
LSTM 256	RMSprop	0,7478	03:41:20
Bi-GRU 256	RMSprop	0,7753	06:16:40
GRU 128	RMSprop	0,7580	03:01:40
GRU 64	RMSprop	0,7237	03:04:20
GRU 512	RMSprop	0,7817	02:59:40
GRU 256 + densa 128	RMSprop	<u>0,7648</u>	03:05:40
GRU 256 + densa 32	RMSprop	0,7734	03:13:20
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7881	03:34:20
GRU 256 + dropout .1 + recorrente .5	RMSprop	0,7802	03:38:20
GRU 256	adam	<u>0,7682</u>	03:06:40
GRU 256	adadelta	0,7723	03:05:00
GRU 256 + GRU 32	RMSprop	0,7772	06:19:20
GRU 256 + dropout .2 + recorrente .2 + densa 32 + dropout .2	RMSprop	0,7874	03:39:20
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7911	03:41:40
Bi-GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7825	07:04:40
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,7094	03:40:00
GRU 1024 + dropout .2 + recorrente .2	RMSprop	0,7783	03:55:00
LSTM 512 + dropout .2 + recorrente .2	RMSprop	0,7738	04:33:40
LSTM 1024 + dropout .2 + recorrente .2	RMSprop	0,7731	05:14:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPooling1D ****	RMSprop	0,6744	00:01:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7076	00:01:00
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7132	00:01:40
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7317	01:19:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 128 + drop.2 + recor.2	RMSprop	0,7640	00:45:00
Conv1D 64x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7580	00:44:20
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7324	00:01:40
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7317	00:02:20
Conv1D 64x7 + MaxPool1D 5 + densa 2048 + drop.2 + densa 128 + drop.2	RMSprop	0,7226	00:05:40

Observação: 20 épocas; batch_size de 32. Embeddings variáveis ou fixos****

4.3.3 Classificação de excertos filtrados

A Tabela 14 mostra alguns treinamentos realizados para classificação de excertos de acórdãos filtrados, por área de jurisprudência, empregando *embeddings* sem pré-treino. No total, os treinamentos levaram quase 22 horas e meia, e o melhor valor de acurácia conseguido foi de 76,18%, com uma rede neural com camada convolucional 32x7 e MaxPooling 5, camada densa de 256 neurônios, dropout de 20%, saída softmax, e otimização RMSprop.

Tabela 14 - Treinamentos sobre excertos filtrados com embeddings sem pré-treino

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 2048 + 1024 + dropout .6	adadelta	0,7151	00:14:00
Densa 2048 + 1024 + dropout .4	adadelta	0,7072	00:13:20
Densa 2048 + 512 + dropout .4	adadelta	0,7162	00:14:00
Densa 1024 + 512 + dropout .4	adadelta	0,7219	00:07:40
Densa 1024 + 256 + dropout .4	adadelta	0,7279	00:07:40
Densa 512 + 256 + dropout .4	adadelta	0,7170	00:04:00
Densa 512 + 128 + dropout .4	adadelta	0,7113	00:04:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7478	00:52:40
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,7490	00:54:00
LSTM 256 + dropout .2 + recorrente .2	RMSprop	<u>0,7381</u>	01:07:00
LSTM 512 + dropout .2 + recorrente .2	RMSprop	0,7478	01:07:20
Bi-GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7433	01:38:20
Bi-LSTM 256 + dropout .2 + recorrente .2	RMSprop	0,7437	04:10:00
GRU 128 + dropout .2 + recorrente .2	RMSprop	<u>0,7350</u>	00:52:20
GRU 64 + dropout .2 + recorrente .2	RMSprop	0,7256	00:52:40
GRU 512 + dropout .2 + recorrente .2 + densa 128 + dropout .3	RMSprop	<u>0,7365</u>	00:53:40
GRU 256 + dropout .2 + recorrente .2 + densa 128 + dropout .3	RMSprop	0,7482	00:52:40
GRU 512 + dropout .2 + recorrente .2 + densa 64 + dropout .3	RMSprop	0,7456	00:53:00
GRU 256 + dropout .2 + recorrente .2 + densa 32 + dropout .3	RMSprop	<u>0,7328</u>	00:53:00
GRU 256 + dropout .2 + recorrente .2 + densa 32 + dropout .3	adam	0,7241	00:53:20
GRU 256 + dropout .2 + recorrente .2 + densa 32 + dropout .3	adadelta	0,7091	00:52:40
GRU 256 + dropout .2 + recorrente .2 + GRU 32 + dropout .2 + recor .2	RMSprop	0,7260	01:42:40
GRU 512 + dropout .2 + recorrente .2 + GRU 64 + dropout .2 + recor .2	RMSprop	<u>0,7595</u>	01:44:00
GRU 256 + dropout .2 + recorrente .2 **	RMSprop	0,7264	00:26:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7482	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,7460	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	<u>0,7369</u>	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7618	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	<u>0,7565</u>	00:01:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7286	00:13:20
Conv1D 64x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	<u>0,7350</u>	00:12:40
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 128 + drop.2 + recor.2	RMSprop	0,7158	00:11:40

Observação: 20 épocas; batch_size de 32 ou 64**; embeddings variáveis de 100 valores.

A Tabela 15 mostra alguns treinamentos para classificação de excertos filtrados, por área de jurisprudência, empregando *embeddings* pré-treinados do NILC-USP. Totalizaram quase 26 horas e meia, e a melhor acurácia de 79,11%, com uma rede de duas camadas GRU, de 512 e de 128 unidades, otimização RMSprop, dropout de 20% e recorrente de 20%.

Na Tabela 16 estão listados treinamentos realizados para classificação de excertos de acórdãos empregando *embeddings* pré-treinados sobre os textos de acórdãos. No total, os treinamentos levaram pouco mais de 27 horas, e o melhor valor de acurácia foi de 79,45%, com uma rede GRU com duas camadas, de 256 e de 64 unidades, otimização RMSprop, *dropout* de

20% e recorrente de 20%. Redes FFNN e redes convolucionais apresentaram desempenho inferior ao conseguido com redes recorrentes.

Tabela 15 - Treinamentos sobre excertos filtrados com embeddings do NILC-USP

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 2048 + 1024 + dropout .6 ****	adadelta	0,5909	00:13:20
Densa 2048 + 1024 + dropout .6	adadelta	0,6590	00:14:00
Densa 1024 + 512 + dropout .4 ****	adadelta	0,6248	00:06:40
Densa 1024 + 512 + dropout .4	adadelta	0,6805	00:07:20
Densa 1024 + 256 + dropout .4 ****	adadelta	0,6214	00:06:20
Densa 1024 + 256 + dropout .4	adadelta	0,6823	00:07:00
Densa 512 + 256 + dropout .4 ****	adadelta	0,6214	00:03:20
Densa 512 + 256 + dropout .4	adadelta	0,6662	00:03:40
Densa 512 + 128 + dropout .4 ****	adadelta	0,6018	00:03:20
Densa 512 + 128 + dropout .4	adadelta	0,6658	00:03:40
Densa 256 + 128 + dropout .4	adadelta	0,6801	00:02:20
Densa 256 + 64 + dropout .4	adadelta	0,6680	00:02:20
GRU 256 ****	RMSprop	0,7482	00:44:00
GRU 256	RMSprop	0,7779	00:45:40
LSTM 256	RMSprop	0,7659	00:56:00
GRU 512	RMSprop	0,7734	00:46:00
GRU 256 + dropout .2 + recorrente .2 + densa 128 + dropout .2	RMSprop	0,7783	00:52:40
GRU 256 + dropout .2 + recorrente .2 + densa 64 + dropout .2	RMSprop	0,7874	00:52:20
GRU 256 + dropout .2 + recorrente .2 + densa 32 + dropout .2	RMSprop	0,7840	00:52:40
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7810	00:52:00
GRU 256 + dropout .2 + recorrente .2 + GRU 64 + dropout .2 + recor.2	RMSprop	0,7900	01:45:00
GRU 256 + dropout .2 + recorrente .2 + GRU 32 + dropout .2 + recor.2	RMSprop	0,7727	03:38:00
Bi-LSTM 256 + dropout .2 + recorrente .2	RMSprop	0,7670	02:46:40
GRU 512 + dropout .2 + recorrente .2 + densa 256 + dropout .2	RMSprop	0,7907	00:55:00
GRU 512 + dropout .2 + recorrente .2 + densa 128 + dropout .2	RMSprop	0,7881	00:52:20
GRU 512 + dropout .2 + recorrente .2 + densa 64 + dropout .2	RMSprop	0,7757	00:54:20
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,7795	00:53:20
GRU 512 + dropout .2 + recorrente .2 + GRU 128 + dropout .2 + recor.2	RMSprop	0,7911	01:44:40
GRU 512 + dropout .2 + recorrente .2 + GRU 64 + dropout .2 + recor.2	RMSprop	0,7821	01:45:40
LSTM 512 + dropout .2 + recorrente .2	RMSprop	0,7881	01:06:40
Bi-LSTM 512 + dropout .2 + recorrente .2	RMSprop	0,7787	02:09:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPooling1D ****	RMSprop	0,6872	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7448	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,7460	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7399	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7091	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7046	00:01:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7595	00:12:20
Conv1D 64x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7460	00:13:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 128 + drop.2 + recor.2	RMSprop	0,7516	00:13:20

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos**** de 100 valores.

Tabela 16 - Treinamentos sobre excertos filtrados com embeddings de acórdãos

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 2048 + 1024 + dropout .2 ****	adadelta	0,6526	00:13:40
Densa 2048 + 1024 + dropout .2	adadelta	0,6571	00:14:00
Densa 1024 + 512 + dropout .2	adadelta	0,6684	00:07:20
Densa 1024 + 256 + dropout .2	adadelta	0,6699	00:07:00
Densa 512 + 256 + dropout .2	adadelta	0,6673	00:03:40
Densa 256 + 128 + dropout .2	adadelta	0,6616	00:02:20
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,7482	00:44:00
GRU 256 + dropout .2 + recorrente .2 + densa 64 + dropout .2	RMSprop	0,7810	00:52:40
GRU 256 + dropout .2 + recorrente .2 + densa 32 + dropout .2	RMSprop	0,7840	00:52:40
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,7821	00:53:20
GRU 256 + dropout .2 + recorrente .2 + GRU 64 + dropout .2 + recor.2	RMSprop	0,7945	01:45:00
GRU 512 + dropout .2 + recorrente .2 + densa 256 + dropout .2	RMSprop	0,7772	00:53:20
GRU 512 + dropout .2 + recorrente .2 + densa 128 + dropout .2	RMSprop	0,7787	00:54:40
GRU 512 + dropout .2 + recorrente .2 + densa 64 + dropout .2	RMSprop	0,7825	00:53:40
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,7795	00:53:20
GRU 512 + dropout .2 + recorrente .2 + GRU 128 + dropout .2 + recor.2	RMSprop	0,7832	01:44:40
LSTM 512 + dropout .2 + recorrente .2	RMSprop	0,7693	01:06:40
Bi-LSTM 512 + dropout .2 + recorrente .2	RMSprop	0,7648	02:09:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPooling1D ****	RMSprop	0,7015	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7136	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,7335	00:00:40
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,7305	00:01:00
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7079	00:01:00
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,7004	00:01:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7497	00:12:20
Conv1D 64x7 + MaxPool1D 5 + Conv 32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,7840	00:13:00
Conv1D 32x7 + MaxPool1D 5 + Conv 32x7 + GRU 128 + drop.2 + recor.2	RMSprop	0,7595	00:13:20

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos**** de 100 valores.

De modo geral, os excertos filtrados permitiram desempenho um pouco melhor com mesmo modelo sobre o texto original, com um tempo de treinamento bem mais reduzido.

4.3.4 Classificação de acórdãos

A Tabela 17 mostra alguns treinamentos²⁷ realizados para classificação de inteiros teores de acórdãos por área de jurisprudência. Utilizados apenas acórdãos associados a uma única área de jurisprudência. A primeira seção da tabela lista treinamentos realizados empregando *embeddings* sem pré-treino, a seção do meio lista treinamentos realizados empregando *embeddings* pré-treinados do NILC-USP, e a última seção lista treinamento com *embeddings* gerados

²⁷ Treinamentos de acórdãos disponíveis em: <https://github.com/leonardo3108/clacjur/tree/master/acordaos>.

a partir de textos de acórdãos. As redes convolucionais receberam *embeddings* de 100 posições, as demais, *embeddings* de 50 valores.

Tabela 17 - Treinamentos sobre acórdãos associados a única área de jurisprudência

Arquitetura	Otimizador	Melhor val cat acc	Duração
Densa 64 + 256 + dropout .4	adadelta	0,6227	00:17:00
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,6453	00:16:40
Densa 64 + 1024 + 128 + dropout .4	adadelta	0,5970	00:17:20
Densa 64 + 256 + 1024 + 128 + dropout .4	adadelta	<u>0,6165</u>	00:17:20
RNN 256	adadelta	0,2012	17:30:00
GRU 256	RMSprop	Travou na primeira época	
LSTM 256	RMSprop	Travou na primeira época	
GRU 128	RMSprop	0,2028	51:40:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,6201	00:14:00
Conv32x7 + MaxPool5 + Conv32x7 + GRU 256 + drop.2 + recor.2	RMSprop	0,5739	11:18:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSp 5e-3	0,6227	00:13:20
Conv32x7 + MaxPool5 + Conv32x7 + GRU 128 + drop.2 + recor.2	RMSprop	0,5919	11:00:00
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,6222	00:19:20
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,6099	00:17:00
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5+ flat+ densa 256+ drop.2	RMSprop	0,6525	00:20:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	<u>0,6176</u>	00:22:00
Conv32x7 + MaxPool5 + Conv32x7 + MaxPool5+ flat+ densa 256+ drop.2	RMSprop	0,1910	00:15:20
Densa 64 + 256 + dropout .4 ****	adadelta	0,2741	00:32:30
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,2433	00:13:00
Densa 64 + 256 + dropout .4	adadelta	0,2028	00:17:20
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,1961	00:17:20
RNN 128 + dropout .2 + recorrente .2	RMSprop	0,2017	57:56:00
GRU 128 + dropout .2 + recorrente .2 ****	RMSprop	Não executado	
GRU 128 + dropout .2 + recorrente .2	RMSprop	0,2023	68:00:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D ****	RMSprop	0,6001	00:07:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,6289	00:20:00
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5+ flat+ densa 256+ drop.2	RMSprop	0,4938	00:20:20
Conv 64x7 + MaxPool 5 + Conv 32x7 + MaxPool 5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,4713	00:23:00
Densa 64 + 256 + dropout .4 ****	adadelta	0,2793	00:13:00
Densa 64 + 256 + 1024 + dropout .4 ****	adadelta	0,2864	00:13:20
Densa 64 + 256 + dropout .4	adadelta	0,2608	00:17:40
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,2161	00:17:20
RNN 256	RMSprop	0,2464	55:00:00
GRU 256 ****	RMSprop	Parado	280:00:00
GRU 256	RMSprop	Não executado	
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D ****	RMSprop	0,6386	03:50:00
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GlobalMaxPooling1D	RMSprop	0,6407	00:19:20
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5+ flat+ densa 256+ drop.2	RMSprop	0,1910	00:02:40
Conv 64x7 + MaxPool 5 + Conv 32x7 + MaxPool 5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,1910	00:22:20

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos****.

Boa parte dos treinamentos de redes neurais empregando acórdãos não teve sucesso.

Devido ao tamanho enorme dos textos, as estruturas em memória eram grandes e a havia muita dificuldade de convergência das redes. Consequentemente, boa parte dos treinamentos levou a travamento da rotina, tempo excessivo de duração das épocas e níveis baixos de acurácia. Esses casos estão marcados em vermelho na tabela.

De um modo geral, as redes recorrentes não funcionaram porque o tempo de duração era proibitivo. As redes densas tiveram que ser reduzidas enormemente na primeira camada (64 neurônios) para não causar estouro de memória da placa GPU, e boa parte dos treinamentos não convergiu para valores minimamente razoáveis de acurácia. As redes convolucionais foram as que deram melhor resultado, mesmo assim, algumas delas não retornaram valores bons de acurácia. Redes mistas, com camadas convolucionais e recorrentes não funcionaram. Apenas redes puras convolucionais ou mistas com camadas convolucionais e densas.

Os maiores valores de acurácia encontrados foram por volta de 64 a 65%, em convolucionais, densas, ou mistura de ambas.

A Tabela 18 mostra outros treinamentos realizados para classificação de inteiros teores de acórdãos por área de jurisprudência, porém, utilizados todos os acórdãos associados a áreas de jurisprudência (multi-label). Mesmas três seções e mesmos tamanhos de *embeddings* empregados nos treinamentos com acórdãos de área única. Os resultados foram similares, com uma perda da acurácia na maior parte dos casos. Em alguns casos, porém, as redes retornaram uma acurácia entre 64 e 65%, como antes.

Tabela 18 - Treinamentos sobre todos os acórdãos associados a áreas de jurisprudência

Arquitetura	Otimizador	Melhor val cat acc	Total
Densa 64 + 256 + dropout .4	adadelta	0,5995	00:18:20
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,6209	00:19:00
Densa 64 + 1024 + 128 + dropout .4	adadelta	0,6005	00:18:20
Densa 64 + 256 + 1024 + 128 + dropout .4	adadelta	0,5824	00:46:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D	RMSprop	0,5829	00:16:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D	RMSp5e-3	0,5587	00:15:40
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D	RMSprop	0,5933	00:21:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,1767	00:19:20
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,5583	00:28:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	<u>0,5705</u>	00:26:40
Conv64x7+MaxPool5+Conv32x7+MaxPool5+flat+densa 256+drop.2	RMSprop	0,3040	00:17:00
Densa 64 + 256 + dropout .4 ****	adadelta	0,2570	00:13:00
Densa 64 + 256 + 1024 + dropout .4 ****	adadelta	0,1767	00:13:40
Densa 64 + 256 + dropout .4	adadelta	0,2024	00:18:20
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,1948	00:17:20
RNN 128 + dropout .2 + recorrente .2	RMSprop	Parado	60:00:00
GRU 128 + dropout .2 + recorrente .2	RMSprop	Parado	59:56:25
GRU 128 + dropout .2 + recorrente .2	RMSprop	Não executado	
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D ****	RMSprop	0,5653	00:08:00
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D	RMSprop	0,5876	00:21:40
Conv64x7+MaxPool5+Conv32x7+MaxPool5+flat+densa 256+drop.2	RMSprop	0,4903	00:22:00
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,4556	00:25:00
Densa 64 + 256 + dropout .4 ****	adadelta	0,1767	00:13:00
Densa 64 + 256 + 1024 + dropout .4 ****	adadelta	0,2646	00:13:20
Densa 64 + 256 + dropout .4	adadelta	0,2575	00:02:20
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,1853	00:18:40
RNN 128 + dropout .2 + recorrente .2	RMSprop	Parado	80:00:00
GRU 128 + dropout .2 + recorrente .2 ****	RMSprop	Não executado	
GRU 128 + dropout .2 + recorrente .2	RMSprop	Parado	61:31:01
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D ****	RMSprop	0,5620	00:07:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool1D	RMSprop	0,5838	00:20:40
Conv64x7+MaxPool5+Conv32x7+MaxPool5+flat+densa 256+drop.2	RMSprop	0,1767	00:21:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,4622	00:23:40

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos****.

4.3.5 Acórdãos filtrados até 6000 palavras

A Tabela 19 mostra treinamentos realizados para classificação de inteiros teores de acórdãos, com palavras lematizadas e filtradas até obter-se uma sequência de no máximo 6000 palavras, por área de jurisprudência. Utilizados apenas acórdãos associados a uma única área de jurisprudência. A primeira seção da tabela lista treinamentos realizados empregando *embeddings* sem pré-treino, a seção do meio lista treinamentos realizados empregando *embeddings*

pré-treinados do NILC-USP, e a última seção lista treinamento com *embeddings* gerados a partir de textos de acórdãos. As redes convolucionais receberam *embeddings* de 100 posições, as demais, *embeddings* de 50 valores.

Tabela 19 - Treinamentos sobre acórdãos filtrados (6000), única área de jurisprudência

Arquitetura	Otimizador	Melhor val cat acc	Total
Densa 512 + 128 + dropout .4 *	adadelta	0,6319	00:39:10
Densa 512 + 256 + dropout .4	adadelta	0,6232	00:15:40
GRU 256	RMSprop	0,6104	06:33:40
GRU 256 + dropout .1 + recorrente .4	RMSprop	0,6273	08:06:20
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,0359	08:24:20
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,6258	08:16:40
Bi-GRU 256 + dropout .2 + recorrente .2	RMSprop	0,6073	20:00:00
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,6622	00:03:00
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 512 + dropout .2	RMSprop	0,6350	00:03:20
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSp5e-3	0,6591	00:04:20
Conv32x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,6509	00:04:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5888	00:02:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,6247	00:03:40
Densa 64 + 256 + dropout .4 ****	adadelta	0,3039	00:02:20
Densa 64 + 256 + 1024 + dropout .4 ****	adadelta	0,3732	00:02:00
Densa 64 + 256 + dropout .4	adadelta	0,3450	00:02:40
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,3291	00:03:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,6504	08:30:00
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,6417	07:08:20
RNN 256 + dropout .2 + recorrente .2	RMSprop	0,2556	03:39:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,6407	00:01:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,6263	00:03:20
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5749	00:03:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,6191	00:03:40
Densa 64 + 256 + dropout .4 ****	adadelta	0,5252	00:02:00
Densa 64 + 256 + 1024 + dropout .4 ****	adadelta	0,3024	00:03:20
Densa 64 + 256 + dropout .4	adadelta	0,2864	00:03:20
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,3008	00:03:20
RNN 256	RMSprop	0,2623	03:31:00
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,6910	08:30:20
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,6838	07:05:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,6535	00:02:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,6597	00:03:20
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5395	00:03:20
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,5010	00:03:40

Observação: 20 ou 50* épocas; batch_size de 32; embeddings variáveis ou fixos****.

Todos os treinamentos realizados foram concluídos, porém alguns não convergiram para valores de acurácia razoáveis. A maioria, porém, trouxe resultados muito bons, melhores que com os textos originais. Chegou-se até valores de cerca de 69%, como redes recorrentes sobre *embeddings* pré-treinados com textos de acórdãos.

Tabela 20 - Treinamentos sobre acórdãos filtrados (6000), múltiplas áreas de jurisprudência

Arquitetura	Otimizador	Melhor val cat acc	Total
Densa 512 + 128 + dropout .4	adadelta	0,5976	00:16:40
Densa 512 + 256 + dropout .4	adadelta	0,5919	00:17:00
GRU 256 + dropout .1 + recorrente .4	RMSprop	0,5824	08:55:40
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,4152	09:10:00
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,5995	00:03:00
Conv1D 64x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,6119	00:04:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,6147	00:05:00
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,6147	00:04:00
Densa 64 + 256 + dropout .4 ****	adadelta	0,2979	00:02:00
Densa 64 + 256 + 1024 + dropout .4 ****	adadelta	0,2945	00:02:00
Densa 64 + 256 + dropout .4	adadelta	0,3116	00:02:40
Densa 64 + 256 + 1024 + dropout .4	adadelta	0,3287	00:03:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	Não executado	
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop		
RNN 256 + dropout .2 + recorrente .2	RMSprop		
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,5686	00:01:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,5601	00:03:20
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5458	00:03:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,4831	00:04:00
Densa 64 + 128 + dropout .4 ****	adadelta	0,2964	00:04:20
Densa 64 + 256 + dropout .4 ****	adadelta	0,3121	00:02:20
Densa 64 + 128 + dropout .4	adadelta	0,2922	00:03:00
Densa 64 + 256 + dropout .4	adadelta	0,2983	00:03:00
RNN 256	RMSprop	Não executado	
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop		
GRU 256 + dropout .2 + recorrente .2	RMSprop		
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,5715	00:02:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,6128	00:03:20
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5601	00:03:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,5534	00:04:00

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos****.

A Tabela 20 mostra treinamentos realizados para classificação de inteiros teores de acórdãos, com palavras lematizadas e filtradas até obter-se uma sequência de no máximo 6000 palavras, porém, utilizados todos os acórdãos associados a áreas de jurisprudência (multi-label). Mesmas três seções e mesmos tamanhos de *embeddings* empregados nos treinamentos com

acórdãos de área única, ou seja, 100 valores para redes convolucionais, e 50 para redes densas e recorrentes. Houve uma perda de acurácia significativa, e algumas redes recorrentes demoraram um tempo proibitivo, por isso não foram executadas completamente. Os melhores valores de acurácia encontrados foram próximos a 62%.

4.3.6 Acórdãos filtrados até 500 palavras

A Tabela 21 mostra alguns treinamentos realizados para classificação de inteiros teores de acórdãos, por área de jurisprudência, com palavras lematizadas e filtradas até obter-se uma sequência de no máximo 500 palavras. Utilizados apenas acórdãos associados a uma única área de jurisprudência. A primeira seção da tabela lista treinamentos realizados empregando *embeddings* sem pré-treino, a seção do meio lista treinamentos realizados empregando *embeddings* pré-treinados do NILC-USP, e a última seção lista treinamento com *embeddings* gerados a partir de textos de acórdãos. Os *embeddings* sem pré-treino foram criados todos com 100 posições. Já para *embeddings* pré-treinados, as redes convolucionais receberam *embeddings* de 100 valores, as demais, *embeddings* de 50 valores.

Todos os treinamentos realizados foram concluídos, em geral após durações relativamente curtas, porém os valores de acurácia encontrados foram geralmente mais baixos. Chegou-se até valores de cerca de 63%. De fato, observando os textos resultantes após a filtragem, nota-se muita repetição de poucas palavras, e uma perda da informação que podia ser apreendida do texto original. A redução de textos dos inteiros teores de acórdãos, com 40.000 palavras, para 500, que é aproximadamente o tamanho de um enunciado de jurisprudência, empregando apenas como critério os scores TF/IDF dos termos, resultou em perda de informação, não compensada pela possibilidade de uma melhor otimização em redes menores.

Na sequência, a Tabela 22 mostra treinamentos realizados para classificação de inteiros teores de acórdãos, com palavras lematizadas e filtradas até obter-se uma sequência de no máximo 500 palavras, porém, utilizados todos os acórdãos associados a áreas de jurisprudência (multi-label). A primeira seção da tabela lista treinamentos realizados empregando *embeddings* sem pré-treino, a seção do meio lista treinamentos realizados empregando *embeddings* pré-treinados do NILC-USP, e a última seção lista treinamento com *embeddings* gerados a partir de textos de acórdãos. Os *embeddings* sem pré-treino foram criados todos com 100 posições. Já para *embeddings* pré-treinados, as redes convolucionais receberam *embeddings* de 100 valores, as demais, *embeddings* de 50 valores.

Aí também os treinamentos realizados foram concluídos, em geral após durações relativamente curtas, porém os valores de acurácia encontrados foram geralmente baixos. O melhor valor encontrado foi exatamente 60%.

Tabela 21 - Treinamentos sobre acórdãos filtrados (500), única área de jurisprudência

Arquitetura	Otimizador	Melhor val cat acc	Total
Densa 2048 + 256 + dropout .4	adadelta	0,5739	00:10:00
Densa 2048 + 512 + dropout .4	adadelta	0,5672	00:10:00
GRU 256	adadelta	0,5606	00:34:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	<u>0,5986</u>	00:39:20
GRU 256 + dropout .1 + recorrente .4	RMSprop	0,5785	00:39:20
GRU 512 + dropout .2 + recorrente .2	RMSprop	0,4697	00:41:40
Bi-GRU 256 + dropout .2 + recorrente .2	RMSprop	0,5960	01:16:20
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,6099	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 512 + dropout .2	RMSprop	0,5714	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 128 + dropout .2	RMSprop	0,5873	00:00:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	<u>0,5934</u>	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,5667	00:00:40
Densa 2048 + 256 + dropout .4 ****	adadelta	0,5503	00:09:40
Densa 2048 + 512 + 256 + dropout .4 ****	adadelta	0,5488	00:09:40
Densa 2048 + 256 + dropout .4	adadelta	0,5421	00:09:40
Densa 2048 + 512 + 256 + dropout .4	adadelta	0,5518	00:09:40
RNN 128 + dropout .2 + recorrente .2	RMSprop	0,5185	00:16:20
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,6278	00:35:40
GRU 128 + dropout .2 + recorrente .2	RMSprop	0,6360	00:39:20
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,5678	00:00:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,6027	00:00:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	<u>0,5909</u>	00:00:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	<u>0,5945</u>	00:00:40
Densa 2048 + 256 + dropout .4 ****	adadelta	0,6037	00:10:00
Densa 2048 + 512 + 256 + dropout .4 ****	adadelta	0,5888	00:10:00
Densa 2048 + 256 + dropout .4	adadelta	0,6057	00:10:00
Densa 2048 + 512 + 256 + dropout .4	adadelta	0,5595	00:10:00
RNN 256	RMSprop	0,5621	00:16:40
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,6335	00:36:00
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,6278	00:40:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	<u>0,5919</u>	00:00:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	<u>0,5955</u>	00:00:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,6109	00:00:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,6109	00:00:40

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos****.

Tabela 22 - Treinamentos sobre acórdãos filtrados (500), múltiplas áreas de jurisprudência

Arquitetura	Otimizador	Melhor val cat acc	Total
Densa 2048 + 256 + dropout .4	adadelata	0,5406	00:11:00
Densa 2048 + 512 + dropout .4	adadelata	0,5468	00:10:00
GRU 256	adadelata	0,5325	00:36:40
GRU 256 + dropout .2 + recorrente .2	RMSprop	<u>0,5620</u>	00:43:20
GRU 256 + dropout .1 + recorrente .4	RMSprop	<u>0,5663</u>	00:43:20
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,5715	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 128 + dropout .2	RMSprop	0,5577	00:00:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	<u>0,5653</u>	00:00:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + MaxPooling1D 5 + flatten + densa 256 + dropout .2	RMSprop	0,5321	00:00:40
Densa 2048 + 256 + dropout .4 ****	adadelata	0,5064	00:10:00
Densa 2048 + 512 + 256 + dropout .4 ****	adadelata	0,5240	00:09:40
Densa 2048 + 256 + dropout .4	adadelata	0,5178	00:10:00
Densa 2048 + 512 + 256 + dropout .4	adadelata	0,5216	00:10:00
RNN 128 + dropout .2 + recorrente .2	RMSprop	0,4803	00:18:20
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,6000	00:44:00
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,4912	00:40:00
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,5345	00:00:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,5781	00:00:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5553	00:00:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,5501	00:00:40
Densa 2048 + 256 + dropout .4 ****	adadelata	0,5477	00:10:00
Densa 2048 + 512 + 256 + dropout .4 ****	adadelata	0,5468	00:10:40
Densa 2048 + 256 + dropout .4	adadelata	0,5444	00:10:20
Densa 2048 + 512 + 256 + dropout .4	adadelata	0,3748	00:10:40
RNN 256 + dropout .2 + recorrente .2	RMSprop	0,5311	00:18:20
GRU 256 + dropout .2 + recorrente .2 ****	RMSprop	0,5976	00:39:20
GRU 256 + dropout .2 + recorrente .2	RMSprop	0,5872	00:43:40
Conv1D 32x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool ****	RMSprop	0,5914	00:00:20
Conv1D 64x7 + MaxPooling1D 5 + Conv1D 32x7 + GMaxPool	RMSprop	0,5477	00:00:40
Conv64x7+ MaxPool5+ Conv32x7+ MaxPool5+flat+densa 256+drop.2	RMSprop	0,5525	00:00:40
Conv64x7 + MaxPool5 + Conv32x7 + MaxPool5 + flat + densa 1024 + drop.2 + densa 128 + drop.2	RMSprop	0,5591	00:00:40

Observação: 20 épocas; batch_size de 32; embeddings variáveis ou fixos****.

4.4 CLASSIFICAÇÃO – COMPARAÇÕES COM VALIDAÇÃO CRUZADA

4.4.1 Estratégias de embeddings para enunciados

As 5 variações de uso de *embeddings* descritas na seção 3.2.2 foram comparadas por meio de validação cruzada com 10 folds, sobre todos os enunciados. A rede neural empregada foi a rede recorrente GRU de 256 unidades. Na Tabela 23, pode-se verificar os valores F1 globais para cada uma das estratégias.

Tabela 23 - Comparação das estratégias de embeddings – Scores F1 Globais
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

Estratégia de embeddings	Sem pré-treino	NILC-USP		Acórdãos	
	ajustáveis	fixos	ajustáveis	fixos	ajustáveis
micro avg	$0,866 \pm 0,006$	$0,849 \pm 0,010$	$0,892 \pm 0,003$	$0,896 \pm 0,008$	$0,900 \pm 0,007$
weighted avg	$0,864 \pm 0,006$	$0,847 \pm 0,010$	$0,891 \pm 0,003$	$0,895 \pm 0,009$	$0,899 \pm 0,007$
macro avg	$0,766 \pm 0,027$	$0,759 \pm 0,030$	$0,826 \pm 0,022$	$0,830 \pm 0,021$	$0,842 \pm 0,018$

As melhores médias de F1 foram obtidas com *embeddings* pré-treinados a partir de textos de acórdãos, ajustados durante o treinamento da classificação, atingindo 90,0% de F1 micro. Em seguida, *embeddings* de acórdãos mantidos fixos durante a classificação, com 89,6%, e *embeddings* do NILC-USP, ajustados durante o treinamento, com 89,2%. As piores médias ocorreram com os *embeddings* do NILC-USP mantidos fixos, com 84,9%, e *embeddings* sem pré-treino, com 86,6%.

Considerando esses intervalos de 99% de confiança ($\alpha = 0,99$) para uma distribuição T-student com $N = 10$, as medições para as opções 3, 4 e 5 obtiveram médias F1 maiores que as medições para as opções 1 e 2. Não se pode afirmar o mesmo entre as opções 3, 4 e 5, pois há interseções entre os intervalos de confiança de 99%, assim como entre as opções 1 e 2.

Para médias F1 micro e F1 ponderado, as estratégias com *embeddings* ajustáveis apresentaram menores desvios e, consequentemente, intervalos de confiança mais curtos, enquanto as estratégias em que os *embeddings* foram mantidos congelados corresponderam a maiores desvios e intervalos mais amplos.

Para todas as estratégias adotadas, as médias de F1 micro e F1 ponderada foram bem próximas entre si, e com intervalos de confiança menores. Já as médias de F1 macro foram muito menores e os intervalos de confiança menores, devido a diferenças de resultados entre classes mais representativas e aquelas com menos ocorrências.

A Tabela 24 mostra os valores F1 de cada classe, ou seja, área de jurisprudência, para cada uma das estratégias. A média F1 para cada classe variou consideravelmente. Em geral, as classes mais representativas apresentaram valores mais elevados, próximos a 90% ou mais, e em menos representativas os scores F1 obtidos foram próximos de 80% ou 70%. As melhores médias sempre foram obtidas com *embeddings* pré-treinados a partir de acórdãos. Para a maioria das classes, *embeddings* treinados em conjunto com o modelo apresentaram melhores medições, mas para duas classes a melhor média foi obtida com *embeddings* de palavras mantidas fixas durante o treinamento.

Os desvios padrão das medidas de F1 também variaram consideravelmente entre diferentes áreas da jurisprudência. Em geral, as classes mais representativas apresentaram desvios

mais baixos, próximos a 1%, consequentemente intervalos de confiança mais curtos, e em classes menos representativas houve desvios maiores nas medidas de F1, portanto intervalos de confiança mais amplos.

Tabela 24 - Comparação das estratégias de embeddings – Scores F1 por classe
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

Estratégia de embeddings	Sem pré-treino	NILC-USP		Acórdãos		Frequência
	ajustáveis	fixos	ajustáveis	fixos	ajustáveis	
Pessoal	$0,959 \pm 0,008$	$0,942 \pm 0,010$	$0,965 \pm 0,009$	$0,971 \pm 0,007$	$0,971 \pm 0,007$	25,5%
Licitação	$0,898 \pm 0,017$	$0,886 \pm 0,012$	$0,922 \pm 0,015$	$0,922 \pm 0,014$	$0,923 \pm 0,019$	20,8%
Responsabilidade	$0,852 \pm 0,013$	$0,828 \pm 0,018$	$0,881 \pm 0,013$	$0,884 \pm 0,019$	$0,890 \pm 0,017$	17,6%
Direito Processual	$0,880 \pm 0,009$	$0,849 \pm 0,011$	$0,895 \pm 0,015$	$0,898 \pm 0,017$	$0,902 \pm 0,014$	13,6%
Contrato Administrativo	$0,759 \pm 0,031$	$0,738 \pm 0,036$	$0,813 \pm 0,030$	$0,816 \pm 0,034$	$0,828 \pm 0,030$	7,1%
Convênio	$0,720 \pm 0,025$	$0,708 \pm 0,041$	$0,765 \pm 0,026$	$0,787 \pm 0,036$	$0,799 \pm 0,038$	5,1%
Competência do TCU	$0,798 \pm 0,043$	$0,819 \pm 0,028$	$0,831 \pm 0,052$	$0,844 \pm 0,047$	$0,824 \pm 0,044$	4,2%
Gestão Administrativa	$0,600 \pm 0,102$	$0,606 \pm 0,093$	$0,704 \pm 0,077$	$0,680 \pm 0,074$	$0,712 \pm 0,056$	2,5%
Finanças Públicas	$0,709 \pm 0,093$	$0,648 \pm 0,068$	$0,758 \pm 0,035$	$0,747 \pm 0,035$	$0,791 \pm 0,054$	2,5%
Desestatização	$0,482 \pm 0,124$	$0,563 \pm 0,175$	$0,723 \pm 0,148$	$0,747 \pm 0,112$	$0,784 \pm 0,077$	1,1%

4.4.2 Modelos de classificação de excertos

Alguns modelos de classificação de excertos foram comparados por meio de validação cruzada com 10 folds. As duas primeiras redes neurais classificavam com base no texto filtrado, restringido até 500 palavras, a terceira sobre o texto original. A primeira rede era uma rede recorrente com duas camadas de unidades GRU, a primeira com 256 e a segunda com 64 unidades, dropout e dropout recorrente de 20%, saída SoftMax. A segunda possuía uma camada convolucional de 64x5, seguida por uma operação de max pooling de 5, depois uma convolução 32x5, uma camada recorrente de 256 unidades GRU e uma camada de saída SoftMax. A terceira possuía uma camada convolucional de 32x5, uma operação de max pooling de 5, outra camada 32x5, uma camada recorrente de 128 unidades GRU e uma camada de saída SoftMax. Na Tabela 25, pode-se verificar os valores F1 globais obtidos para cada uma das estratégias.

Tabela 25 - Modelos neurais de classificação de excertos – Scores F1 Globais
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

Modelo de classificação	rede recorrente sobre texto filtrado	rede convolucional sobre texto filtrado	rede convolucional sobre texto original
micro avg	$0,839 \pm 0,014$	$0,776 \pm 0,018$	$0,778 \pm 0,016$
weighted avg	$0,838 \pm 0,014$	$0,772 \pm 0,018$	$0,773 \pm 0,017$
macro avg	$0,771 \pm 0,025$	$0,657 \pm 0,030$	$0,632 \pm 0,026$

Os resultados mostram superioridade da rede recorrente sobre as demais, com 84% de acurácia, enquanto as redes convolucionais ficaram abaixo de 78%, fora do intervalo de 99% de confiança ($\alpha = 0,99$). Nota-se uma queda maior da média macro para redes convolucionais, sugerindo maior diferença de desempenho entre as classes.

Isso pode ser confirmado na Tabela 26, com os valores F1 para cada área de jurisprudência: em redes convolucionais há classe com F1 abaixo de 50%, sendo que a rede recorrente se mantém pelo menos próxima a 70%.

Tabela 26 - Modelos neurais de classificação de excertos – Scores F1 por classe
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

Modelo de classificação	rede recorrente sobre texto filtrado	rede convolucional sobre texto filtrado	rede convolucional sobre texto original	Frequência
Pessoal	$0,966 \pm 0,008$	$0,946 \pm 0,012$	$0,951 \pm 0,010$	25,5%
Licitação	$0,878 \pm 0,016$	$0,837 \pm 0,022$	$0,836 \pm 0,020$	20,8%
Responsabilidade	$0,791 \pm 0,017$	$0,727 \pm 0,025$	$0,739 \pm 0,029$	17,6%
Direito Processual	$0,822 \pm 0,025$	$0,760 \pm 0,026$	$0,751 \pm 0,020$	13,6%
Contrato Administrativo	$0,718 \pm 0,040$	$0,570 \pm 0,051$	$0,629 \pm 0,043$	7,1%
Convênio	$0,664 \pm 0,072$	$0,532 \pm 0,091$	$0,518 \pm 0,056$	5,1%
Competência do TCU	$0,698 \pm 0,041$	$0,552 \pm 0,040$	$0,500 \pm 0,110$	4,2%
Finanças Públicas	$0,736 \pm 0,040$	$0,611 \pm 0,058$	$0,560 \pm 0,055$	2,5%
Gestão Administrativa	$0,687 \pm 0,071$	$0,547 \pm 0,106$	$0,584 \pm 0,104$	2,5%
Desestatização	$0,752 \pm 0,157$	$0,489 \pm 0,105$	$0,252 \pm 0,165$	1,1%

4.4.3 Modelos de classificação de acórdãos

Alguns modelos de classificação de acórdãos foram comparados por meio de validação cruzada com 10 folds, sobre os acórdãos com única área de jurisprudência associada. As duas primeiras redes neurais empregadas continham camada convolucional de 64×5 , seguida por operação de MaxPooling entre 5, seguida de outra camada convolucional, de 32×5 , seguida de uma operação de GlobalMaxPooling e uma camada de saída SoftMax. A primeira classificação sobre o texto original do acórdão, limitado a 40.000 palavras, e a segunda sobre o texto filtrado, restringido a 4.000 palavras. A terceira rede era uma rede recorrente simples com 256 unidades GRU sobre texto filtrado. Na Tabela 27, pode-se verificar os valores F1 globais obtido para cada uma das estratégias.

Tabela 27 - Modelos neurais de classificação de acórdãos – Scores F1 Globais
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

Modelo de classificação	Rede convolucional sobre texto original	Rede convolucional sobre texto filtrado	Rede recorrente sobre texto filtrado
micro avg	$0,661 \pm 0,010$	$0,667 \pm 0,013$	$0,667 \pm 0,014$
weighted avg	$0,641 \pm 0,015$	$0,652 \pm 0,014$	$0,634 \pm 0,014$
macro avg	$0,414 \pm 0,028$	$0,452 \pm 0,030$	$0,398 \pm 0,031$

As médias de F1 micro obtidas foram similares, com leve vantagem para redes sobre texto filtrado, ambas com 66,7%, porém dentro do intervalo de 99% de confiança ($\alpha = 0,99$) estava a média de F1 para rede convolucional sobre texto não filtrado, com 66,1%. Nota-se uma média macro bem abaixo da média micro, sugerindo uma enorme diferença de desempenho entre as classes.

Isso pode ser confirmado na Tabela 28, com os valores F1 para cada área de jurisprudência: valores melhores para as 4 áreas dominantes, e valores muito baixo para as demais áreas. A rede recorrente apresentou a maior disparidade.

Tabela 28 - Modelos neurais de classificação de acórdãos – Scores F1 por classe
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

Modelo de classificação	rede convolucional sobre texto original	rede convolucional sobre texto filtrado	Rede recorrente sobre texto filtrado	Frequência
Pessoal	$0,929 \pm 0,011$	$0,930 \pm 0,007$	$0,938 \pm 0,007$	25,5%
Licitação	$0,750 \pm 0,026$	$0,747 \pm 0,026$	$0,758 \pm 0,018$	20,8%
Responsabilidade	$0,587 \pm 0,025$	$0,576 \pm 0,052$	$0,576 \pm 0,048$	17,6%
Direito Processual	$0,527 \pm 0,033$	$0,546 \pm 0,040$	$0,549 \pm 0,036$	13,6%
Contrato Administrativo	$0,265 \pm 0,124$	$0,314 \pm 0,078$	$0,232 \pm 0,099$	7,1%
Convênio	$0,174 \pm 0,079$	$0,228 \pm 0,090$	$0,099 \pm 0,057$	5,1%
Competência do TCU	$0,278 \pm 0,110$	$0,319 \pm 0,109$	$0,036 \pm 0,053$	4,2%
Finanças Públicas	$0,299 \pm 0,078$	$0,384 \pm 0,090$	$0,380 \pm 0,055$	2,5%
Gestão Administrativa	$0,037 \pm 0,067$	$0,138 \pm 0,130$	$0,120 \pm 0,096$	2,5%
Desestatização	$0,293 \pm 0,098$	$0,334 \pm 0,161$	$0,290 \pm 0,166$	1,1%

Os desvios padrão das medidas de F1 também variaram consideravelmente entre diferentes áreas da jurisprudência, geralmente com maior disparidade para o modelo que usa texto original.

4.4.4 Modelos não neurais

As tabelas 29, 30 e 31 mostram as médias globais de F1 para os modelos não neurais testados²⁸, para a classificação de enunciados, excertos de acórdãos e inteiros teores de acórdãos, respectivamente.

Tabela 29 – Classificação de enunciados por modelos não neurais – Scores F1 Globais
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

	Logistic Regression	Naive Bayes	Random Forest	LDA
micro avg	$0,867 \pm 0,009$	$0,826 \pm 0,011$	$0,743 \pm 0,014$	$0,702 \pm 0,013$
weighted avg	$0,860 \pm 0,009$	$0,814 \pm 0,013$	$0,695 \pm 0,017$	$0,707 \pm 0,013$
macro avg	$0,743 \pm 0,023$	$0,652 \pm 0,028$	$0,474 \pm 0,014$	$0,592 \pm 0,028$

Tabela 30 – Classificação de excertos por modelos não neurais – Scores F1 Globais
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

	Logistic Regression	Naive Bayes	Random Forest	LDA
micro avg	$0,832 \pm 0,007$	$0,724 \pm 0,015$	$0,729 \pm 0,010$	$0,808 \pm 0,007$
weighted avg	$0,827 \pm 0,008$	$0,732 \pm 0,014$	$0,694 \pm 0,014$	$0,807 \pm 0,008$
macro avg	$0,746 \pm 0,017$	$0,588 \pm 0,018$	$0,513 \pm 0,026$	$0,735 \pm 0,019$

Tabela 31 – Classificação de acórdãos por modelos não neurais – Scores F1 Globais
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

	Logistic Regression	Naive Bayes	Random Forest	LDA
micro avg	$0,672 \pm 0,010$	$0,598 \pm 0,016$	$0,638 \pm 0,011$	$0,645 \pm 0,010$
weighted avg	$0,654 \pm 0,011$	$0,609 \pm 0,016$	$0,605 \pm 0,012$	$0,647 \pm 0,012$
macro avg	$0,558 \pm 0,027$	$0,525 \pm 0,029$	$0,480 \pm 0,027$	$0,562 \pm 0,027$

As medidas tiradas da Logistic Regression, em geral, foram as melhores. As medidas sobre enunciados e excertos foram muito próximas. As medidas tiradas de outras técnicas de mineração foram piores, especialmente no caso da Random Forest. Vale um destaque para o LDA, que apesar de muito ruim na predição a partir dos enunciados, teve um desempenho relativamente bom para excertos e acórdãos, em comparação com a regressão logística.

Foram também obtidos scores F1 para cada classe. A Tabela 32 mostra um resumo apenas das medições sobre a regressão logística. Notadamente, algumas classes tiveram scores

²⁸ Cadernos disponíveis em: <https://github.com/leonardo3108/clacjur/tree/master/shallow>

muito baixos com esse modelo, piores que em outros modelos. Isso indica que o uso dos comitês de classificadores poderia trazer melhorias de resultado.

Tabela 32 - Classificação de acórdãos por Regressão Logística – Scores F1 por Classe
Intervalos de confiança $\alpha=99\%$ considerando distribuição T-student

	Enunciados	Excertos	Acórdãos
Pessoal	$0,946 \pm 0,013$	$0,959 \pm 0,011$	$0,905 \pm 0,011$
Licitação	$0,897 \pm 0,016$	$0,875 \pm 0,018$	$0,746 \pm 0,035$
Responsabilidade	$0,861 \pm 0,015$	$0,794 \pm 0,019$	$0,608 \pm 0,021$
Direito Processual	$0,881 \pm 0,022$	$0,816 \pm 0,018$	$0,548 \pm 0,037$
Contrato Administrativo	$0,768 \pm 0,033$	$0,726 \pm 0,012$	$0,382 \pm 0,048$
Convênio	$0,730 \pm 0,084$	$0,644 \pm 0,073$	$0,365 \pm 0,057$
Competência do TCU	$0,798 \pm 0,058$	$0,631 \pm 0,068$	$0,348 \pm 0,051$
Gestão Administrativa	$0,666 \pm 0,092$	$0,692 \pm 0,061$	$0,604 \pm 0,076$
Finanças Públicas	$0,639 \pm 0,083$	$0,701 \pm 0,088$	$0,438 \pm 0,093$
Desestatização	$0,244 \pm 0,114$	$0,622 \pm 0,132$	$0,635 \pm 0,161$

4.5 RESUMO

A Tabela 33 mostra algumas das melhores acurácias alcançadas em todas as classificações realizadas, empregando particionamento dos dados de 80%/20% desenvolvimento/validação. Algumas classificações foram objeto também de validação cruzada de 10 partições (particionamento 90%/10%, 10 passagens), e descritas na seção 2.3.4.

Para classificação dos enunciados, as redes recorrentes contendo *embeddings* pré-treinados sobre textos de acórdãos apresentaram os melhores resultados, chegando a 90% de acurácia. Na sequência, redes convolucionais, e redes. Entre modelos não neurais, destacaram-se Logistic Regression, com 86,7%, e Naive Bayes, com 82,6%. Dentro das redes recorrentes, GRU e LSTM tiveram desempenho próximo, e RNN foram pior. O uso de camadas bidirecionais e composições com outras camadas densas ou recorrentes não ajudou.

Para classificação dos excertos de acórdãos, observa-se também a liderança das redes recorrentes, com 83,9% de acurácia. Seguidas pelas redes convolucionais e redes densas. No uso de *embeddings*, resultados semelhantes aos dos enunciados. O uso da técnica de filtragem produziu alguns resultados melhores, porém muito próximos, a grande vantagem sendo o tempo de treinamento bem mais reduzido. Entre modelos não neurais, os melhores resultados ocorreram com Logistic Regression, com 83,2%, e LDA, com 80,8%.

Tabela 33 - Alguns dos melhores resultados com classificações exploratórias

	Tipo de arquitetura	val cat acc
Enunciados	redes recorrentes GRU e LSTM com embeddings de acórdãos ajustáveis	0,8727
	redes recorrentes GRU e LSTM com embeddings NILC-USP ajustáveis	0,8629
	redes recorrentes GRU e LSTM com embeddings de acórdãos fixos	0,8558
	redes recorrentes GRU e LSTM com embeddings sem pré-treino	0,8299
	redes convolucionais puras com embeddings NILC-USP ajustáveis	0,8284
	redes recorrentes GRU e LSTM com embeddings NILC-USP fixos	0,8269
	redes convolucionais puras com embeddings sem pré-treino	0,8258
	redes convolucionais e GRU com embeddings NILC-USP ajustáveis	0,8183
	redes convolucionais e GRU com embeddings de acórdãos ajustáveis	0,8141
	redes densas com embeddings sem pré-treino	0,7544
Excertos	redes recorrentes com embeddings de acórdãos ajustáveis sobre excertos filtrados	0,7945
	redes recorrentes com embeddings NILC-USP ajustáveis sobre excertos filtrados	0,7911
	redes recorrentes com embeddings de acórdãos ajustáveis	0,7911
	redes recorrentes com embeddings NILC-USP ajustáveis	0,7858
	redes convolucionais com embeddings de acórdãos ajustáveis sobre excertos filtrados	0,7840
	redes recorrentes com embeddings de acórdãos fixos	0,7731
	redes convolucionais com embeddings de acórdãos ajustáveis	0,7640
	redes convolucionais com embeddings sem pré-treino sobre excertos filtrados	0,7618
	redes convolucionais com embeddings NILC-USP ajustáveis sobre excertos filtrados	0,7595
	redes recorrentes com embeddings sem pré-treino sobre excertos filtrados	0,7595
	redes densas com embeddings sem pré-treino sobre excertos filtrados	0,7219
Acórdãos single	redes recorrentes GRU com embeddings de acórdãos sobre texto filtrado 6000	0,6910
	redes convolucionais sobre texto filtrado 6000	0,6622
	redes convolucionais sobre texto original	0,6525
	redes recorrentes GRU com embeddings NILC-USP sobre texto filtrado 6000	0,6504
	redes densas com embeddings sem pré-treino sobre texto original	0,6453
	redes recorrentes GRU sobre texto filtrado 500	0,6360
	redes recorrentes GRU com embeddings sem pré-treino sobre texto filtrado 6000	0,6273
	redes recorrentes GRU com embeddings de acórdãos sobre texto filtrado 500	0,6109
	redes densas sobre texto filtrado 500	0,6057
	redes convolucionais sobre texto filtrado 500	0,6027
Acórdãos multi	redes recorrentes GRU com embeddings de acórdãos sobre texto filtrado 6000	0,6447
	redes densas com embeddings sem pré-treino sobre texto original	0,6209
	redes convolucionais sobre texto filtrado 6000	0,6147
	redes recorrentes GRU sobre texto filtrado 500	0,6000
	redes densas com embeddings sem pré-treino sobre texto filtrado 6000	0,5976
	redes convolucionais sobre texto original	0,5933
	redes convolucionais sobre texto filtrado 500	0,5914

Para classificação single-label dos inteiros teores de acórdãos, os melhores resultados ocorreram com Logistic Regression, com 67,2%. Sobre textos filtrados com até 6000 palavras, tanto as redes convolucionais quanto as recorrentes chegaram a valores entre 66 e 67%. As redes convolucionais apresentaram a vantagem da extrema rapidez do treinamento, e funcionaram quase tão bem mesmo sobre o texto original, atingindo 65%. A maior vantagem do treinamento com Logistic Regression foi uma disparidade bem menor entre as classes. O ideal provavelmente seria usar um comitê de classificadores contendo redes convolucionais e rede recorrente sobre texto filtrado, bem como modelos com Logistic Regression e LDA.

Redes recorrentes sobre texto original do acórdão apresentaram péssimo desempenho, e o resultados com textos filtrados para até 500 palavras ficaram abaixo dos demais. Aparentemente, em classificações sobre o texto original, o uso de *embeddings* pré-treinados não ajudou.

Para classificação multi-label dos inteiros teores de acórdãos, observou-se a liderança também das redes recorrentes sobre textos filtrados até 6000 palavras, com 64,5%. As redes densas e as convolucionais ficaram logo abaixo, e não foram realizados testes multi-label com redes não neurais.

5 CONCLUSÃO

O projeto visava verificar a viabilidade de classificação automática por área de jurisprudência de enunciados de jurisprudência e acórdãos, a partir dos textos dos enunciados, dos excertos ou dos inteiros teores dos acórdãos.

Empregando textos mais curtos de entrada, como enunciados, excertos, e acórdãos reduzidos até o máximo de 6000 palavras por acórdão, os resultados apontaram melhor desempenho em redes recorrentes com *embeddings* pré-treinados a partir de textos do mesmo domínio do conhecimento. No caso de enunciados, a acurácia máxima obtida foi 90,0%, excertos, 83,4%, acórdãos reduzidos, 66,7%. A redução de excertos produziu acurácia similar à com textos originais, e a redução dos acórdãos, para o máximo de 500 palavras, piorou a maior parte dos resultados.

Para textos mais longos, dos inteiros teores dos acórdãos, com pouco pré-processamento, as redes convolucionais e modelos não neurais, como Logistic Regression e LDA, apresentaram melhor desempenho. O pré-treinamento de *embeddings* foi menos relevante nesses casos. Redes recorrentes tiveram treinamento excessivamente demorado, e com resultados inexpressivos.

As classificações multi-label com acórdãos apresentaram uma acurácia menor cerca de 3% em relação a classificações single-label em redes neurais. Não foram realizados testes similares com modelos não neurais.

O excelente destaque para enunciados era esperado, uma vez que o texto é, comumente, curto, significativo, e de alta qualidade, por ser produzido por especialistas em jurisprudência no Tribunal. Em compensação, é onde deve haver o menor ganho com automatização. Apesar da grande queda de acurácia entre classificações sobre textos de enunciados e acórdãos, os valores atingidos foram bons, considerando a dificuldade de lidar com textos grandes e a dificuldade de se extrair informação jurisprudencial de um acórdão. Além disso, a classificação de acórdãos serve tanto para possíveis enriquecimentos da plataforma de pesquisa de acórdãos e sistemas de apoio ao pós-julgamento em geral, mas particularmente pode ajudar a orientar a geração de jurisprudência dentro do Tribunal.

Como trabalho futuro, as mesmas tarefas de classificação podem ser aprimoradas pela introdução de arquiteturas de última geração, como redes de atenção, e modelos BERT, ELMO e ULMfit. Outro aprimoramento possível é o maior investimento em tarefas de pré-processamento, por exemplo, promovendo uma redução de dados de entrada mais criteriosa, removendo

elementos do acórdão que não ajudam para fins de jurisprudência, como anexos, tabelas, repetições, entre outros.

Melhorias também podem ser conseguidas com a incorporação de técnicas como Grid Search de hiperparâmetros, busca ortogonal, repeated cross-validation, bem como a incorporação de penalizações L1 e L2 na função de custo. Entre os possíveis benefícios, melhores scores, menos overfitting, intervalos de confiança mais restritos. Além disso, a formação de comitês de classificadores, incorporando algumas redes neurais e não neurais pode trazer melhorias especialmente na classificação para áreas menos representativas, seja por votação, combinação ou stacking de modelos.

A produção do vocabulário a partir dos textos de acórdãos também pode ser aprimorada, pelo investimento em tarefas de pré-processamento, inclusão de peças de processos e outros documentos do GED, bem como a utilização de outros algoritmos como Skip-row, Glove, FastText, etc. Permite inclusive uma análise comparativa desses métodos empregando a classificação de enunciados como método de benchmark.

No apoio ao processo de seleção da jurisprudência, tarefas de geração de texto provavelmente são mais úteis, embora bem mais complexas. A elaboração de excerto a partir do inteiro teor do acórdão e a produção do enunciado a partir do excerto ou do inteiro teor podem trazer melhorias mais significativas ao processo. A extensão da classificação de área para tema, subtema e outros indexadores também podem ajudar.

REFERÊNCIAS

- ALLAHYARI, Mehdi et al. **A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques**. In: KDD BIGDAS. Halifax, Canada: 2017.
- CHOLLET, François. **Deep Learning with Python**. Manning, 1st ed., USA: 2017.
- BENGIO, Yoshua, GLOROT, Xavier. **Understanding the difficulty of training deep feed-forward neural networks**. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS-2010), Italy: 2010.
- ELMAN, Jeffrey. **Finding Structure in Time**. In: Cognitive Science, v. 14, n. 2, p. 179-211, 1990.
- HARTMANN, Nathan, FONSECA, Erick, SHULBY, Christopher, TREVISO, Marcos, RODRIGUES, Jéssica, ALUÍSIO, Sandra. **Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks**. In: Symposium in Information and Human Language Technology (STIL 2017).
- HOCHREITER, Sepp, SCHMIDHUBER, Jürgen. **Long Short-Term Memory**. Neural Computation 9(8):1735-1780, 1997.
- LAI, Siwei, LIU, Kang, HE, Shizhu, ZHAO, Jun. **How to Generate a Good Word Embedding**. In: IEEE Intelligent Systems, vol. 31, no. 6, pp. 5-14, 2016.
- MIKOLOV, Tomas et al. **Subword Language Modeling with Neural Networks**. 2011.
- MIKOLOV, Tomas, CHEN, K., CORRADO, G., DEAN, J. **Efficient Estimation of Word Representations in Vector Space**. In: Proceedings of International Conference on Learning Representations Workshop (ICLR-2013).
- MINER, Gary et al. **Practical text mining and statistical analysis for non-structured text data applications**. Elsevier, 1st ed., USA: 2012.
- NGUYEN, Michael. **Illustrated Guide to LSTM's and GRU's: A step by step explanation**. Disponível em <<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>>. Acesso em 05/10/2019.
- PISSANETZKY, Sergio. **Sparse Matrix Technology – Electronic Edition**. Scientific Controls: 2008.
- SEBASTIANI, Fabrizio. **Machine Learning in Automated Text Categorization**. In: ACM Computing Surveys, Vol. 34, No. 1. New York, USA: 2002.
- SOKOLOVA, Marina, LAPALME, Guy. **A systematic analysis of performance measures for classification tasks**. In: Information processing & management 45 (2009): 427-437
- ZAKI, Mohammed, MEIRA, Wagner. **Data Mining and Analysis. Fundamental Concepts and Algorithms**. Cambridge University Press, 1st ed., USA: 2014.