

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação Lato Sensu em Ciência de Dados e Big Data

Sandro Luiz de Aguiar

**CLASSIFICAÇÃO E SUMARIZAÇÃO DE ACÓRDÃOS DE JULGAMENTO DO
CARF UTILIZANDO ALGORÍTMOS DE PROCESSAMENTO DE LINGUAGEM
NATURAL**

Belo Horizonte
2021

Sandro Luiz de Aguiar

**CLASSIFICAÇÃO E SUMARIZAÇÃO DE ACÓRDÃOS DE JULGAMENTO DO
CARF UTILIZANDO ALGORÍTMOS DE PROCESSAMENTO DE LINGUAGEM
NATURAL**

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte

2021

SUMÁRIO

1. Introdução.....	5
1.1. Contextualização.....	5
1.2. O problema proposto.....	5
2. Coleta de Dados	10
2.1 Baixar dados do Carf	10
2.2 Ler pdf acordo e gravar dataframes Carf_atributos e Carf_Voto_Dividido_Em_Frases	12
3. Processamento/Tratamento de Dados	18
3.1 Preparação dos dataframes X e y (treinamento e teste).....	18
3.2 Preprocessamento do voto	20
3.3 Vetorização do voto	23
3.4 Sumarização do voto	24
3.5 Tratamento da variável alvo	25
4. Análise e Exploração dos Dados	26
4.1 Dataframe Carf - atributos	26
4.2 Dataframe Carf_Voto_Dividido_Em_Frases.....	28
4.3 Dataframe Carf_Ementas.....	29
4.4 Palavras mais frequentes das conclusões	30
4.5 Nomes dos relatores	31
5. Criação de Modelos de Machine Learning	32
5.1 Classificação dos votos com a SpaCy	33
5.2 Classificação dos votos usando TensorFlow	36
5.3 Classificação dos votos sumarizados.....	40
5.3.1 Usando a codificação em python das técnicas	41
5.3.2 Usando os sumarizadores da biblioteca Sumy	42
5.4 Pesquisa.....	45
6. Apresentação dos Resultados	48
6.1 Modelo de Canvas por Vasandani	48
6.2 Modelos SpaCy.....	49
6.3 Modelos Rede Neural Convolutacional Profunda TensorFlow.....	49

6.4 Modelo de melhor acurácia aplicado aos sumarizadores	50
6.5 Avaliação de sumarizadores por humanos.....	52
Conclusão	59
7. Links.....	60
REFERÊNCIAS.....	61
APÊNDICE.....	62
Estrutura do Github	62
arquivos	62
pastas	62
notebooks	62
dataframes web scraping	64
dataframes gerados nos notebooks.....	64
modelos.....	65
pesquisa.....	66
scripts	66

1. Introdução

1.1. Contextualização

O Contencioso Administrativo Federal é constituído pelas Delegacias de Julgamento (DRJ) e pelo Conselho Administrativo de Recursos Fiscais (CARF).

As DRJs, órgãos de deliberação interna e natureza colegiada, têm por finalidade julgar processos que versem sobre a aplicação da legislação referente aos tributos administrados pela Secretaria Especial da Receita Federal do Brasil do Ministério da Economia, conforme estabelecido em seu Regimento Interno.

Compete às DRJs apreciar, por decisão colegiada:

I - em primeira instância, a impugnação ou manifestação de inconformidade apresentada pelo sujeito passivo; e

II - em última instância, os recursos contra as decisões de que trata o inciso I do caput, em relação ao contencioso administrativo fiscal de pequeno valor, assim considerado aquele cujo lançamento fiscal ou controvérsia não supere sessenta salários mínimos. (Art. 2º da Portaria ME 340, de 2020)

O CARF, órgão colegiado, paritário, integrante da estrutura do Ministério da Fazenda, tem por finalidade julgar recursos de ofício e voluntário de decisão de 1ª (primeira) instância, exceto os recursos relativos ao contencioso administrativo fiscal de pequeno valor, bem como os recursos de natureza especial, que versem sobre a aplicação da legislação referente a tributos administrados pela Secretaria da Receita Federal do Brasil (RFB). (Art. 1º do Regimento Interno aprovado pela Portaria MF 343, de 2015)

Atualmente, há um estoque de processos gigantesco em apreciação nas DRJs e no Carf. Em torno de 250 mil nas DRJs e 120 mil no Carf.

1.2. O problema proposto

O julgamento de um processo é realizado a partir da análise dos seguintes documentos:

1 Termo de Verificação Fiscal e Auto de Infração: contêm o lançamento fiscal com os motivos, com a descrição dos fatos e com o enquadramento legal que embasaram a autuação.

2 Impugnação: contém as alegações do contribuinte feitas com o objetivo de afastar ou diminuir o lançamento fiscal.

O resultado do julgamento é escrito em um documento chamado Acórdão que contém dados descritivos do processo (número, interessado, assunto, nome dos julgadores, etc), um relatório com o resumo do lançamento e da impugnação e o voto que descreve a análise do julgador e o seu resultado.

O relatório de um acórdão requer um trabalho de sumarização pelo julgador dos documentos citados.

A sumarização automática com a utilização de processamento de linguagem natural é um importante recurso, nesse contexto, para acelerar o trabalho de julgamento de processos e, em consequência, auxiliar na diminuição do passivo existente.

Como a proposta da PUC para este TCC é ser um trabalho prático e a Receita Federal do Brasil espera a utilização na sua atividade, o ideal seria uma proposta de trabalho aplicada diretamente nos dados das Delegacias de Julgamento, pois é esta a minha unidade de trabalho e a minha atividade.

Entretanto, há uma impossibilidade legal de utilização dos documentos dos processos em discussão nas DRJ, em especial, os de interesse deste trabalho: os Termos de Verificação Fiscal, as impugnações e os acórdãos, por conterem informações de natureza econômico-fiscal protegidas pela regra de sigilo fiscal veiculada pelo art. 198 do Código Tributário Nacional, Lei 5.172, de 1966, acolhido pela Constituição Federal com força de Lei Complementar.

Diante disso, a alternativa encontrada foi a utilização dos acórdãos do Carf, pois esses são públicos, uma vez que o Carf não faz parte da Fazenda Pública que é a destinatária do comando legal supra.

Os acórdãos do Carf são formados por textos de conteúdo jurídico similares aos utilizados pelas DRJ, pois se tratam de uma segunda análise sobre os mesmos dados e sob um mesmo ordenamento jurídico, e são, portanto adequados ao objetivo deste trabalho.

Todos os processos julgados tem os seus dados descritivos e o inteiro teor do acórdão publicados no sítio do Carf, <http://idg.carf.fazenda.gov.br/>.

Os dados dos acórdãos serão analisados com o objetivo de serem classificados e sumarizados.

O objetivo principal do trabalho é avaliar as técnicas de sumarização automática de textos aplicadas a um corpus de textos jurídicos.

A estratégia adotada para isso será criar um modelo de classificação para ser testado com o conteúdo integral do voto e, também, com o conteúdo resumido.

Inicialmente, serão treinados diferentes modelos de classificação com base nos textos completos e tratados de diferentes maneiras. O modelo que apresentar os melhores resultados será escolhido para a avaliação dos resumos.

Um bom resumo permite que o destinatário avalie a compreensão do texto lido, incluindo a compreensão global, o desenvolvimento das idéias e a articulação entre elas (Machado, 2004).

No caso do resumo de um voto, é essencial que o leitor consiga distinguir o assunto, o resultado do voto e as controvérsias: motivos do lançamento, alegações da defesa e fundamentos da decisão.

A seleção de atributos (*feature selection*) é uma etapa fundamental em um projeto de Ciência de Dados e se utiliza de medidas estatísticas para estabelecer a correlação e dependência das variáveis preditoras com a variável alvo.

O desempenho de um modelo é fortemente influenciado pela qualidade e eficiência dos atributos.

Fazendo uma analogia com a classificação de textos, os atributos mais eficientes são as palavras que melhor se correlacionam com o assunto que se quer classificar.

O resumo de um texto irá diminuir a quantidade de palavras do texto original.

É premissa do trabalho que os melhores resumos são aqueles que conseguem extrair do texto original as frases que melhor representam os seus principais aspectos, as frases que melhor identificam o que é relevante.

Se assim o for, os melhores resumos conseguirão manter as palavras que melhor se correlacionam com a variável alvo permitindo que o modelo não tenha uma perda significativa de acurácia ao ser testado com o texto reduzido.

Para isso, os textos dos votos da base de teste serão sumarizados por meio de diversas técnicas de sumarização e, após a sumarização, serão submetidos à classificação pelo melhor modelo para obtermos a acurácia do modelo em classificar os textos sumarizados.

Pelo exposto, espera-se que os melhores sumarizadores sejam aqueles com a melhor acurácia na classificação, pois serão os que preservarão as características mais importantes do texto original.

Com essa estratégia, espera-se aferir automaticamente a qualidade dos resumos.

Entretanto, essa premissa pode não refletir a realidade.

Considerando que o trabalho tem a expectativa de subsidiar eventual criação de ferramenta para a criação de relatórios automáticos, é importante verificar se a classificação automática dos resumos está em consonância com a visão dos futuros usuários desses relatórios, os julgadores.

Para isso, elaborei uma pesquisa para conhecer e comparar a classificação dos resumos feita pelos julgadores.

A pesquisa e seus resultados serão mostrados ao final do trabalho.

A tabela abaixo, baseada na abordagem 5W-1H, resume a proposta do trabalho.

Classificação e sumarização de acórdãos de julgamento do Carf utilizando algoritmos de processamento de linguagem natural	
Why? Por que esse problema é importante?	<p>O estoque de processos em apreciação nas DRJs e no Carf é gigantesco. Em torno de 250 mil nas DRJs e 120 mil no Carf.</p> <p>Há necessidade de criar instrumentos para facilitar e acelerar o julgamento dos processos como enfrentamento do estoque crescente e diante da insuficiência de pessoas.</p>
Who? De quem são os dados analisados?	Dados públicos relativos ao julgamento, em 2ª instância, de processos administrativos fiscais no âmbito federal.
What? Quais os objetivos com essa análise?	Avaliar a qualidade dos sumarizadores automáticos com a utilização de processamento de linguagem natural e a aplicabilidade na atividade de julgamento de processos administrativos.
Where? Aspectos geográficos e logísticos.	Não se aplica
When? Qual o período está sendo analisado?	Não se aplica
How? Como será feita a análise?	<ol style="list-style-type: none"> 1. Classificação dos textos. 2. Sumarização dos textos. 3. Seleção do melhor modelo de classificação 4. Nova classificação, agora, dos resumos. 5. Avaliação da qualidade dos resumos pela acurácia da segunda classificação. 6. Pesquisa sobre a qualidade dos resumos feita com julgadores. 7. Comparação dos resultados.

2. Coleta de Dados

Todo o trabalho de Web Scraping feito utilizando-se a linguagem de programação Python com a criação de scripts do ContÁgil¹ da Receita Federal.

2.1 Baixar dados do Carf2

De acordo com o Manual de redação e elaboração de atos administrativos da Secretaria da Receita Federal do Brasil, 2014, “*Ementa é a síntese da decisão, a que evidencia a regra aplicada ao caso concreto. Tem grande relevância para a compreensão da matéria objeto do processo e é citada como base na argumentação em outros julgamentos. Uma ementa bem elaborada facilita as pesquisas à jurisprudência administrativa*”.

Os dados das ementas foram obtidos nas páginas do sítio do Carf³. Confira-se a seguir.

VOCÊ ESTÁ AQUI: [PÁGINA INICIAL](#) > [ACÓRDÃOS](#)

Sistema Push

Carta de Serviços

Agenda de Audiências

JURISPRUDÊNCIA

Nova Pesquisa de Acórdãos - VER

Acórdãos CARF

Súmulas CARF

Jurisprudência/Acórdãos

Pesquisa : Processo (10880.910301/2008-02)
Acórdãos Encontrados: 1

Acórdão: 1003-000.776

Número do Processo: 10880.910301/2008-02

Data de Publicação: 05/07/2019

Contribuinte: JORGE S IMOVEIS E ADMINISTRACAO LTDA

Relatoria: WILSON KAZUMI NAKAYAMA

Ementa: Assunto: Imposto sobre a Renda de Pessoa Jurídica - IRPJ Exercício: 1999 PER/DCOMP. DIREITO CREDITÓRIO. PRESCRIÇÃO. Ao pedido de restituição pleiteado administrativamente antes de 09.06.2005, no caso de tributo sujeito a lançamento por homologação, aplica-se o prazo prescricional de 10 (dez) anos, contado do fato gerador. RECONHECIMENTO DO DIREITO CREDITÓRIO. ANÁLISE INTERROMPIDA. Inexiste reconh

Decisão: Vistos, relatados e discutidos os presentes autos. Acordam os membros do colegiado, por unanimidade de votos, em dar provimento parcial ao recurso, para aplicação da Súmula Vinculante CARF nº 91 e reconhecimento da possibilidade de análise do indébito, determinando o retorno dos autos à DRF que jurisdiciona a Recorrente, de modo a evitar a supressão de instância e o cerceamento de defesa Carmen F

Foi gerado o dataset *Carf - ementas* com a seguinte estrutura.

¹ O ContÁgil é um aplicativo desenvolvido na linguagem de programação JAVA que pode ser executado localmente, sem necessidade de estar conectado em rede, de uso interno da Receita Federal.

² Script TCC Big Data - Baixar acórdãos Carf, código disponível no github.

³ <https://carf.fazenda.gov.br/sincon/public/pages/ConsultarJurisprudencia/consultarJurisprudenciaCarf.jsf> e <https://carf.fazenda.gov.br/sincon/public/pages/ConsultarJurisprudencia/listaJurisprudenciaCarf.jsf>

Nome da coluna/campo	Descrição	Tipo
Processo	Número do processo	Texto
Ementa	Síntese de uma decisão colegiada (acórdão)	Texto

Veja a seguir recorte do dataframe em processamento no Colab.

Processo	Ementa
10070000461200797	Ementa(s) Assunto: Imposto sobre a Renda de Pessoa Física - IRPF Ano-calendário: 2003 IRPF. OMISSÃO DE RENDIMENTOS DO TRABALHO. AUXÍLIO MATERNIDADE. A fonte pagadora dispõe de convênio com o INSS...
10070000654200748	Ementa(s) Assunto: Imposto sobre a Renda de Pessoa Física - IRPF Exercício: 2004 OMISSÃO DE RENDIMENTOS Os rendimentos tributáveis recebidos pelo contribuinte devem ser integralmente informados em...
10070000947200644	Ementa(s) Assunto: Imposto sobre a Renda de Pessoa Física - IRPF Exercício: 2003 RENDIMENTOS DECORRENTES DE AÇÃO JUDICIAL. DESPESAS COM ADVOGADO. Poderá ser deduzido, para fins de determinação da...

Confira-se recorte do código da extração citada a seguir.

```
try:
    web.abrirPagina("https://carf.fazenda.gov.br/sincon/public/pages/ConsultarJurisprudencia/consultarJurisprudenciaCarf.jsf")
except Exception as erro:
    print(f'Processo {nrProcesso} com erro ao acessar página consultarJurisprudenciaCarf. \nErro: {erro}.')
    continue

try:
    # formulario para pesquisa
    form = web.getPaginaAtual().getFormulario("consultaJurisprudenciaForm")

    form.setCampo("valor_pesquisal", nrProcesso)
    form.setCampo("AJAXREQUEST", "_viewRoot")
    form.setCampo("consultaJurisprudenciaForm", "consultaJurisprudenciaForm")
    form.setCampo("j_id51", "j_id51")

    web.submeterFormulario(form)

    # formulario para detalhamento
    form = web.getPaginaAtual().getFormulario("formAcordaos")

    form.setCampo("tblJurisprudencia:0:numDecisao", "tblJurisprudencia:0:numDecisao")
    form.removeCampo("j_id89")
    form.removeCampo("botaoImprimir")

    web.submeterFormulario(form)

    tabela = web.getPaginaAtual().getTabela(0)

    for linha in range(0, tabela.getNumLinhas()):
        if str(tabela.getCelula(linha, "COLUNA-00"))[:6] == "Ementa":
            tab_A.setCelula(linhaCarf, "Processo", nrProcesso)
            tab_A.setCelula(linhaCarf, "Ementa", tabela.getCelula(linha, "COLUNA-0"))
            linhaCarf += 1

except Exception as erro:
    print(f'Processo {nrProcesso} com erro ao acessar o formulário consultaJurisprudenciaForm. \nErro: {erro}.')
    continue
```

Este dataset será utilizado para a formação da lista de palavras bônus do sumarizador Edmundson. O procedimento será descrito no tópico relativo aos sumarizadores.

Em seguida, em outra página⁴, o pdf com o inteiro teor do acórdão foi baixado por meio do método `exportaConteudoBinario`⁵ da classe `WebExtrator`⁶ do `ContÁgil`.



Confira-se recorte do código da extração citada a seguir⁷.

```
try:
    # pegando o anexo
    form = web.getPaginaAtual().getFormulario("formAcordaos")
    form.removeCampo("formAcordaos:j_id64")
    form.removeCampo("botaoImprimir")
    form.setCampo("formAcordaos:_idcl", "formAcordaos:j_id60:0:j_id61")
    web.submeterFormulario(form)
    web.exportaConteudoBinario("D:\\IACarf\\AcordaoBaixadoCarf\\" + nrProcesso + "_AcordaoCarf.pdf")
except Exception as erro:
    print(f'Processo {nrProcesso} com erro no download do acórdão. \nErro: {erro}.')
    continue
```

Foram baixados 18.594 arquivos pdf sendo 3.932 no dia 24/12/2020, de 11:34h às 20:21h, 2.386 no dia 17/01/2021, de 13:20h às 23:59h, 534 no dia 18/01/2021, de 00:00h às 01:54h, 4.947 no dia 18/01/2021, de 06:02h às 23:59h, no dia 19/01/2021, de 00:00h às 08:48h, 3.761 no dia 19/01/2021, de 11:19h às 15:49h.

2.2 Ler pdf acordao e gravar dataframes `Carf_atributos` e `Carf_Voto_Dividido_Em_Frases`

Para a leitura do pdf e obtenção dos textos do voto foi necessário excluir os textos com rotação e também foi preciso limitar a área de cobertura dos dados para excluir textos em segundo plano, marcas de cópia, número de folhas, conforme se pode observar na figura seguinte:

⁴ <https://carf.fazenda.gov.br/sincon/public/pages/ConsultarJurisprudencia/listaJurisprudenciaCarf.jsf>

⁵ Exporta o conteúdo binário que foi respondido na última requisição de página. Este é um método alternativo ao convencional `getPaginaAtual`. É recomendável para situações em que o resultado de uma consulta não é uma página HTML, mas sim o conteúdo de algum arquivo (como no caso de um arquivo PDF, por exemplo).


⁶ Objeto que pode ser utilizado por uma linguagem de script para extrair dados de um serviço Web (HTTP). Este objeto está disponível para uma linguagem de script através do nome "web". Isto é, já está definido para qualquer linguagem de script um objeto do tipo `WebExtrator` cujo nome é "web", bastando executar seus métodos.

⁷ Código integral do script disponível no github, link https://github.com/sla01/TCC_PUC_Minas

0	2	4	6	8	10	12	14	16	18	20
---	---	---	---	---	----	----	----	----	----	----

DF CARF MF

FL 151
SI-TER3
FL 151



MINISTÉRIO DA FAZENDA
CONSELHO ADMINISTRATIVO DE RECURSOS FISCAIS
 PRIMEIRA SEÇÃO DE JULGAMENTO

Processo nº	10805.908224/2011-11
Recurso nº	Voluntário
Acórdão nº	1803-002.610 - 3ª Turma Especial
Sessão de	24 de março de 2015
Matéria	PERD/COMP
Recorrente	LAB HORN - Laboratório Especializado em dosagens hormonais Ltda
Recorrida	FAZENDA NACIONAL

ASSUNTO: IMPOSTO SOBRE A RENDA DE PESSOA JURÍDICA - IRPJ
 Exercício: 2004

LUCRO PRESUMIDO, PERCENTUAIS, REQUISITOS ESPECÍFICOS, PROVA, INTERPRETAÇÃO DA LEGISLAÇÃO TRIBUTÁRIA, POSICIONAMENTO JUDICIAL SUJEITO À SISTEMÁTICA DOS RECURSOS REPETITIVOS, VINCULAÇÃO DA ESFERA ADMINISTRATIVA.

1. Os percentuais de lucro presumido, no imposto sobre a renda e na contribuição social sobre o lucro líquido, definidos para serviços equiparados à hospitalares, para exercícios anteriores à 2009, independem de comprovação de requisitos específicos, limitado a exigência do objeto próprio da atividade.
2. Possibilidade de reconhecimento de crédito pleiteado, se o conjunto probatório e as condições especiais da demanda justificarem a relativização do formalismo processual, com base no princípio da verdade real.

Vistos, relatados e discutidos os presentes autos.

Acordam os membros do Colegiado, por unanimidade de votos, pelo provimento do recurso voluntário, com reconhecimento do direito creditório.

(assinado digitalmente)

Carmen Ferreira Saraiva – Redatora Designada Ad Hoc e Presidente

Composição do colegiado. Participaram do presente julgamento os Conselheiros: Sérgio Rodrigues Mendes, Roberto Armond Ferreira da Silva, Meigan Sack Rodrigues, Ricardo Diefenthaler, Fernando Ferreira Castellani e Carmen Ferreira Saraiva.

Documento assinado digitalmente conforme MP nº 2.200-2 de 24/08/2001
 Autenticado digitalmente em 02/07/2015 por CARMEN FERRERA SARAIVA, Assinado digitalmente em 02/07/2015 por CARMEN FERRERA SARAIVA
 Impresso em 13/07/2015 por RECEITA FEDERAL - PARA USO DO SISTEMA

Os textos delimitados na figura ou sublinhados em vermelho são textos que prejudicam a legibilidade do documento por meios automáticos e não agregam nenhum valor ao sentido do texto.

A definição da exclusão foi feita desprezando-se os textos com rotação superior a 5 graus.

Quanto aos textos em segundo plano, foi executada uma leitura preliminar do texto para verificar a existência dos textos "Impresso em" "Assinado digitalmente em" na área superior a 782.30 no eixo y (função `parametro_PdfClasse_leitura_preliminar`). Existindo os textos, foi obtido o tamanho

da fonte para posterior marcação como tamanho de fonte a desprezar na leitura definitiva.

Na leitura definitiva (função *parametro_PdfClasse_leitura_definitiva*), a área do pdf a ser lida foi delimitada entre 64.32 e 782.30 no eixo y.

Dessa forma, obteve-se o texto limpo dos pdf. Ressalte-se que essa estratégia de leitura foi definida após diversos testes de leitura em amostras dos arquivos. Confira-se o código a seguir.

```
def parametro_PdfClasse_leitura_preliminar(pdfClasse, pastaPesquisarPdf, Pdf):
    pdfClasse.setRotacaoMaxima(5)
    pdfClasse.setPosicaoYMinima(782.30)
    tamanhoFonteDesprezar = 0
    try:
        pdf = pdfClasse.abrirPDFComoTexto(pastaPesquisarPdf, Pdf)
        tab = pdfClasse.getDadosUltimaExtracao().getTamanhosAmostras(True)
        for linhaFonte in range(0, tab.getNumLinhas()):
            if 'Impresso em ' in tab.getCelula(linhaFonte, 'TEXT0') or 'Assinado digitalmente em' in tab.getCelula(linhaFonte, 'TEXT0'):
                tamanhoFonteDesprezar = tab.getValorNumerico(linhaFonte, 'TAMANHO')
                break
    except:
        pass
    return tamanhoFonteDesprezar

def parametro_PdfClasse_leitura_definitiva(pdfClasse, tamanhoFonteDesprezar):
    pdfClasse.setRotacaoMaxima(5)
    pdfClasse.setPosicaoYMaxima(782.30)
    pdfClasse.setPosicaoYMinima(64.32)
    pdfClasse.addTamanhoDesprezar(tamanhoFonteDesprezar)
```

O resultado foi altamente satisfatório, pois em todos os pdf observados, o conteúdo do acórdão estava completo e sem sujeiras.

Após a recuperação do texto do acórdão, foi feita a localização dentro do pdf da parte relativa ao voto do julgador, pois o trabalho de classificação e sumarização seria feito com base no texto do voto.

Esse texto recebeu uma limpeza para prepará-lo para a tokenização em sentenças.

A limpeza do texto (função *limpar_texto*) excluiu o ponto separador de milhar, pois estava influenciando a tokenização; alterou a codificação de leis de 9.240/96 para 9240 de 96; substituiu o ponto em ementas/citações dentro do texto por vírgula, pois estavam sendo geradas sentenças com uma/duas palavras quando tratavam-se de títulos de ementas das citações; foram alteradas siglas/abreviaturas comuns para extenso, além de excluídos caracteres e palavras típicas utilizadas nos acórdãos, mas que caracterizavam verdadeiras *stop words* específicas destes textos (exemplo: dele conheço, assinado digitalmente, como voto, etc).

Por exemplo, citações no voto como abaixo geravam várias frases em virtude dos pontos finais após os títulos. Então, utilizando expressões regulares, transformei os pontos em vírgulas para que os títulos das citações fossem interpretados como parte da próxima frase da citação.

A propósito é da jurisprudência deste CARF:

"IMPOSTO PAGO NO EXTERIOR. COMPENSAÇÃO. REVISÃO DE HOMOLOGAÇÃO DE COMPENSAÇÃO PUBLICADA. IMPOSSIBILIDADE. PRECLUSÃO. NULIDADE DO SEGUNDO DESPACHO. Uma vez publicado um Despacho Decisório que homologa totalmente a compensação, extinguindo, portanto, o crédito tributário por inteiro, não é possível voltar atrás para publicar um novo Despacho Decisório que homologa apenas parte da compensação e extingue, então, apenas parcialmente o crédito tributário. **Havendo publicação de homologação de compensação, extinto está o crédito e preclusa qualquer nova tentativa de examiná-lo, salvo no caso de nulidade do primeiro Despacho Decisório. É, portanto, nulo o segundo Despacho Decisório.**" (Processo 13839.720127/2010-18, Acórdão 1401- 001.487). **Negrito do Relator.**

A tokenização desta citação ficou assim:

a propósito é da jurisprudência deste carf: imposto pago no exterior, compensação, revisão de homologação de compensação publicada, impossibilidade, preclusão, nulidade do segundo despacho, uma vez publicado um despacho decisório que homologa totalmente a compensação, extinguindo, portanto, o crédito tributário por inteiro, não é possível voltar atrás para publicar um novo despacho decisório que homologa apenas parte da compensação e extingue, então, apenas parcialmente o crédito tributário.;

havendo publicação de homologação de compensação, extinto está o crédito e preclusa qualquer nova tentativa de examiná-lo, salvo no caso de nulidade do primeiro despacho decisório.;

O código utilizou expressões regulares. Confira-se parte do código da função `limpar_texto`.

```
def limpar_texto(texto_recebido):
    texto_recebido = texto_recebido.replace('CPC.', 'CPC')
    texto_recebido = texto_recebido.replace('CC.', 'CC')
    texto_recebido = texto_recebido.replace('ART.', 'ARTIGO')
    texto_recebido = texto_recebido.replace('ARTIS.', 'ARTIGOS')
    #tratamento valores numéricos em real. Exclui o ponto, pois estava influenciando a tokenização.
    texto_recebido = re.sub('([0-9]*)[.]\s*([0-9]+, [0-9]+)', r'\1\2', texto_recebido)
    #altera a codificação de leis. De 9.240/96 para 9240 de 96 e nas ementas troca o ponto final para vírgula.
    texto_recebido = re.sub('([0-9]+)\.([0-9]+\s*\s*[0-9]+\s*\s*)', r'\1\2 de \3', texto_recebido)
    #altera a codificação de leis. De 9.240/96 para 9240 de 96
    texto_recebido = re.sub('([0-9]+)\.([0-9]+\s*\s*[0-9]+\s*\s*)', r'\1\2 de \3', texto_recebido)
    #tratamento para ementas. substitui o ponto após duas palavras maiúsculas por vírgula.
    texto_recebido = re.sub('([A-Z0-9A-Ü]+[ ]+[A-Z0-9A-Ü]+)\.', r'\1,', texto_recebido)
    #tratamento para ementas. substitui o ponto após palavra maiúscula por vírgula.
    texto_recebido = re.sub('([A-ZA-Ü]+[0-9]*[A-ZA-Ü]*)\.', r'\1,', texto_recebido)
    #tratamento para ementas. substitui o ponto final por vírgula após palavras maiúsculas com número dentro de parênteses (ANO DE 2003).
    texto_recebido = re.sub('([A-ZA-Ü ]+[0-9]*)\.', r'\1,', texto_recebido)
    #tratamento para ementas. substitui o ponto final por vírgula em datas. DE 1995.
    texto_recebido = re.sub('([A-ZA-Ü]+[ ]+[0-9]+)\.', r'\1,', texto_recebido)
```

Algumas limpezas foram determinadas após a observação do prejuízo causado pelos caracteres/*stop words* em amostras e pela minha experiência como julgador.

O texto do voto foi dividido em sentenças e foi gravado o dataset *Carf_Voto_Dividido_Em_Frases*⁸ com a seguinte estrutura.

Nome da coluna/campo	Descrição	Tipo
Processo	Número do processo	Texto
Frase001 a 510	Voto dividido em frases	Texto

Veja a seguir recorte do dataframe em processamento no Colab.

processo	frase001	frase002	frase003	frase004	frase005	frase006	frase007	frase008	frase009	frase010	frase011	frase012	frase013	frase014	frase015	frase016
10070000654200748	o recurso voluntário é tempestivo e reúne os r...	no que concerne à omissão de rendimentos, veri...	cabe mencionar nesse ponto que o valor de prev...	a folha 10, anexa comprovante de rendimentos p...	238,50 e desconto de R\$ 1181,94 relativo à p...	não consta do referido comprovante desconto ...	assim o lançamento será alterado de ofício par...	do exame dos documentos fornecidos por esta f...	quanto à compensação indevida de lrrf, não mer...	o sujeito passivo de fato não contestou o valo...	cumprir esclarecer que os recolhimentos efetua...	por todo o exposto, voto por conhecer do recur...	()	NaN	NaN	NaN
10070000947200644	por ser tempestivo e por preencher as demais c...	conforme já descrito no relatório supra, o ple...	parágrafo único.	poderá ser deduzido, para fins de determinac...	juntamente com o recurso voluntário, o contrib...	multo embora o alegado equívoco possa ter ocor...	nos termos do inciso xiii do artigo 224 do reg...	por outro lado, a leitura integrada dos artigo...	tal conclusão é corroborada pelo artigo 1º do a...	não obstante, em relação ao este valor sobre o...	por outro lado, igualmente comum é o fato de ...	contudo, o comprovante de rendimentos fornecid...	carlos alberto do amaral azereado	NaN	NaN	NaN

O número de frases/sentenças foi limitado em 510. Processos que ultrapassaram esse limite foram desprezados.

Também foram desprezados 402 processos cujo pdf estava com erro de leitura por problema na ocerização ou qualquer outro problema.

Da leitura do pdf do acórdão foram extraídos os dados descritivos que formaram o dataset *Carf - atributos*⁹ com a seguinte estrutura.

Nome da coluna/campo	Descrição	Tipo
Processo	Número do processo	Texto

⁸ Os dataframes obtidos por web scraping foram transformados em arquivos csv e gravados no Dive para utilização no Google Colab. O arquivo *Carf_Voto_Dividido_Em_Frases* foi gravado como `/content/drive/MyDrive/df_voto.csv`

⁹ Os dataframes obtidos por web scraping foram transformados em arquivos csv e gravados no Dive para utilização no Google Colab. O arquivo *Carf - atributos* foi gravado como `/content/drive/MyDrive/df_atributos.csv`

Recurso	Tipo de recurso	Texto
Materia	Tipo de matéria	Texto
Assunto	Tipo de assunto	Texto
Relator	Nome do relator	Texto
Presidente	Nome do presidente da turma	Texto
Conselheiros	Nome dos conselheiros que participaram do julgamento	Texto
Decisao	Quorum de votação da decisão	Texto
Voto vencedor	Se houve voto vencedor	Texto

Veja a seguir recorte do dataframe em processamento no Colab.

processo	recurso	materia	assunto	relator	presidente	conselheiros	decisao	voto vencedor
10070000654200748	voluntário	irpf - omissão de rendimentos	irpf	mônica renata mello fereira stoll	cláudia cristina noira passos da costa devely...	claudia cristina noira passos da costa devely...	unanimidade	NaN
10070000947200644	voluntário	irpf	irpf	determinando o recalculo do tributo devido nos...	carlos henrique de oliveira	carlos henrique de oliveira, ana cecilia lust...	unanimidade	NaN

Para localização dos atributos no pdf foi utilizada pesquisa aos títulos dos atributos com expressões regulares, pois havia um padrão nos pdf. Confira-se a seguir a localização da matéria.

```
def atributo_materia(acordao):
    try:
        limites = "mat.ria(.*)\n"
        materia = re.search(limites, acordao).group(1)
        materia = materia.strip().lower()
    except:
        materia = ""
    return materia
```

O script utilizou as bibliotecas `os` (para percorrer a pasta com os arquivos pdf baixados), a biblioteca `shutil` (para mover de pasta os arquivos desprezados), a biblioteca `re` (para utilização na limpeza dos dados e na localização de textos no

pdf), as classes *PDFExtractor*¹⁰ e *ArquivoPDF*¹¹ (para leitura e tratamento dos pdf) e a biblioteca *NLTK* (para a tokenização das sentenças).

3. Processamento/Tratamento de Dados

3.1 Preparação dos dataframes X e y (treinamento e teste)

Após a coleta dos dados pelo webscraping, todo o trabalho de análise, exploração dos dados, criação de modelos de Machine Learning, sumarização, geração de dados para pesquisa e apuração foi executado por meio de notebooks no Google Colab.

Os notebooks foram documentados no Colab em seções de texto utilizando a linguagem de marcação Markdown e anexados a este trabalho permitindo que a parte principal do texto não precise descrever tantos códigos, pois eles estarão descritos com riqueza de detalhes e disponíveis no link https://github.com/sla01/TCC_PUC_Minas.

O primeiro dos notebooks, *TCC_p01_preparar_X_treinamento_e_y_teste*, tem por objetivo preparar e salvar os dataframes X e y (treinamento e teste) que serão usados para treinar e testar os diferentes modelos (SpaCy e DCNN com stemer, lema e palavras originais) e, também, para a geração e classificação dos resumos. Como a base para as experiências precisa ser a mesma, após a preparação dos dados e a divisão em X e y, os dataframes serão salvos no drive. Assim, os dataframes poderão ser utilizados nos demais notebooks evitando, na medida do possível, procedimentos duplicados.

Foram importados os os datasets *Carf_Voto_Dividido_Em_Frases* (df_v) e *Carf – atributos* (df_a) utilizando a biblioteca *Pandas* e gerando 2 dataframes.

¹⁰ Objeto de script que disponibiliza diversos recursos avançados para extrair textos a partir de arquivos PDF. Existem diversos métodos mais simples de converter arquivo PDF em texto, implementados em GerenciadorArquivos. Por exemplo, o método *abrirPDFComoTexto*. Porém, existem situações em que se quer utilizar alguns critérios mais avançados. Por exemplo, para considerar somente textos que apresentam uma determinada fonte, dado o nome da fonte. Esta classe se destina a realizar esses critérios avançados.

¹¹ Classe que disponibiliza diversas operações relativas a um arquivo PDF.

```
url = '/content/drive/MyDrive/df_atributos.csv'
df_a = pd.read_csv(url, sep=';', engine='python', encoding='windows-1252', low_memory=True)
df_a.columns = df_a.columns.str.lower()
url = '/content/drive/MyDrive/df_voto.csv'
df_v = pd.read_csv(url, sep=';', engine='python', encoding='windows-1252', low_memory=True, quotechar='\"', error_bad_lines=False)
df_v.columns = df_v.columns.str.lower()
print(f'{df_a.shape}, {df_v.shape}')
```

Criei o atributo *qtdfrase* no dataset *Carf_Voto_Dividido_Em_Frases* com a quantidade de frases/sentenças de cada processo. O quantitativo foi calculado por meio da função *soma_frases* e processado utilizando o método *apply* no eixo das linhas sendo computadas apenas as frases com pelo menos 4 palavras (*regex*).

```
df_v['qtdfrase'] = df_v.apply(soma_frases, axis=1)
```

```
def soma_frases(linha):
    """
    Calcula a quantidade de frases/sentenças do voto. Não considera as frases com menos de 4 palavras e
    os conteúdos NaN.

    :parâmetro linha: recebe as linhas do df que contêm as frases do voto.
    :retorno qtd: int contendo a quantidade de frases da linha.
    """
    qtd = 0
    primeira_coluna = True
    for conteudo_coluna in linha:
        if primeira_coluna == True:
            primeira_coluna = False
            continue
        if isinstance(conteudo_coluna, str):
            if re.search('[a-zA-Z0-9]+ [a-zA-Z0-9]+ [a-zA-Z0-9]+ [a-zA-Z0-9]+', str(conteudo_coluna)): #desp
                qtd += 1
    return qtd
```

Criei o atributo *voto* no dataset *Carf_Voto_Dividido_Em_Frases* com a concatenação de todas as frases/sentenças. A concatenação foi feita por meio da função *concatena_frases* também utilizando o método *apply* no eixo das linhas.

A quantidade de frases era variável e as frases sem conteúdos foram geradas como NaN pelo *Pandas*. As frases Nan foram desprezadas.

```
df_v['voto'] = df_v.apply(concatena_frases, axis=1)
```

```
def concatena_frases(linha):
    """
    Concatena as frases do voto agrupando-as em uma única string. Despreza as frases com menos de 4 palavras e
    os conteúdos NaN. Despreza a primeira coluna que contém o número do processo.

    :parâmetro linha: recebe as linhas do df que contém as frases do voto.
    :retorno texto_original: string contendo o voto completo após a concatenação das frases.
    """
    texto_original = ''
    primeira_coluna = True
    for conteudo_coluna in linha:
        if primeira_coluna == True:
            primeira_coluna = False
            continue
        if isinstance(conteudo_coluna, str):
            if re.search('[a-zA-Z0-9]+ [a-zA-Z0-9]+ [a-zA-Z0-9]+ [a-zA-Z0-9]+', str(conteudo_coluna)):
                texto_original += conteudo_coluna + ' '
    return texto_original
```

O *dataframe* de treinamento e validação foi criado a partir dos *dataframes* *df_v* e *df_a* com os atributos voto e assunto e o atributo processo foi a chave de ligação entre os *dataframes*.

O *dataframe* foi dividido em treinamento e validação (teste) na proporção de 80 e 20%. Para isso, utilizei o método *train_test_split* da biblioteca *sklearn*.

```
i_Seed = 20111974 # semente por questões de reproducibilidade
f_Test_Size = 0.2
X_treinamento, X_teste, y_treinamento, y_teste = train_test_split(df_X, df_y, test_size = f_Test_Size, random_state = i_Seed)
```

3.2 Preprocessamento do voto

O voto é a feature que será utilizada como preditora para a classificação. Por ser uma coleção de frases e palavras com pontuações, valores numéricos, stop words¹², precisa receber tratamento para limpeza do conteúdo inútil.

Além disso, três formas de tratamento de palavras foram testadas:

- 1) Original: nenhum tratamento foi realizado nas palavras.
- 2) Lematização¹³: No contexto puramente lexicográfico a palavra dicionarizada recebe a denominação de lema ou forma canônica. A lematização é o

¹² Na computação, uma palavra vazia é uma palavra que é removida antes ou após o processamento de um texto em linguagem natural. São palavras que podem ser consideradas irrelevantes para o conjunto de resultados a ser exibido. Não existe uma lista universal de palavras vazias usadas por todas as ferramentas de processamento de linguagem natural e nem todas ferramentas fazem uso de uma lista dessas palavras. Qualquer grupo de palavras pode ser escolhido como grupo de "palavras vazias" de acordo com o objetivo do processamento. https://pt.wikipedia.org/wiki/Palavra_vazia

ato de representar as palavras através do infinitivo dos verbos e masculino singular dos substantivos e adjetivos.

3) Stemização¹⁴: é o processo de reduzir palavras flexionadas (ou às vezes derivadas) ao seu tronco (*stem*), base ou raiz, geralmente uma forma da palavra escrita.

A limpeza e o tratamento das palavras foram realizados pela função `preprocessamento`.

A função recebe o texto do voto e faz a limpeza excluindo pontuações, stop words, números em moedas, leis, letras soltas, palavras com duas letras, hífen iniciais, r\$, espaços duplos e símbolos. O objetivo é deixar apenas as palavras que serão a base para as classificações.

A função foi preparada para ser aplicada a coluna voto pelo método `apply()` do pandas e recebe o parâmetro adicional tipo com a seleção entre os tratamentos de palavras (lema ou stemer') ou sem tratamento (original).

```
X_treinamento_lema = X_treinamento['voto'].apply(preprocessamento, args=('lema',))
```

Após o tratamento, o dataframe `X_treinamento` foi gravado no drive para posterior utilização para treinamento dos modelos.

```
X_treinamento_lema.to_csv('/content/drive/MyDrive/X_treinamento_lema.csv', sep = ';', index = False, encoding = 'windows-1252')
X_treinamento_stemer.to_csv('/content/drive/MyDrive/X_treinamento_stemer.csv', sep = ';', index = False, encoding = 'windows-1252')
X_treinamento_original.to_csv('/content/drive/MyDrive/X_treinamento_original.csv', sep = ';', index = False, encoding = 'windows-1252')
```

A seguir, recortes do dataframe `X_treinamento` nos três formatos descritos¹⁵, além de uma nuvem de palavras mostrando as 100 palavras mais freqüentes.

¹³ http://www.nilc.icmc.usp.br/nilc/download/lematizacao_versus_steming.pdf

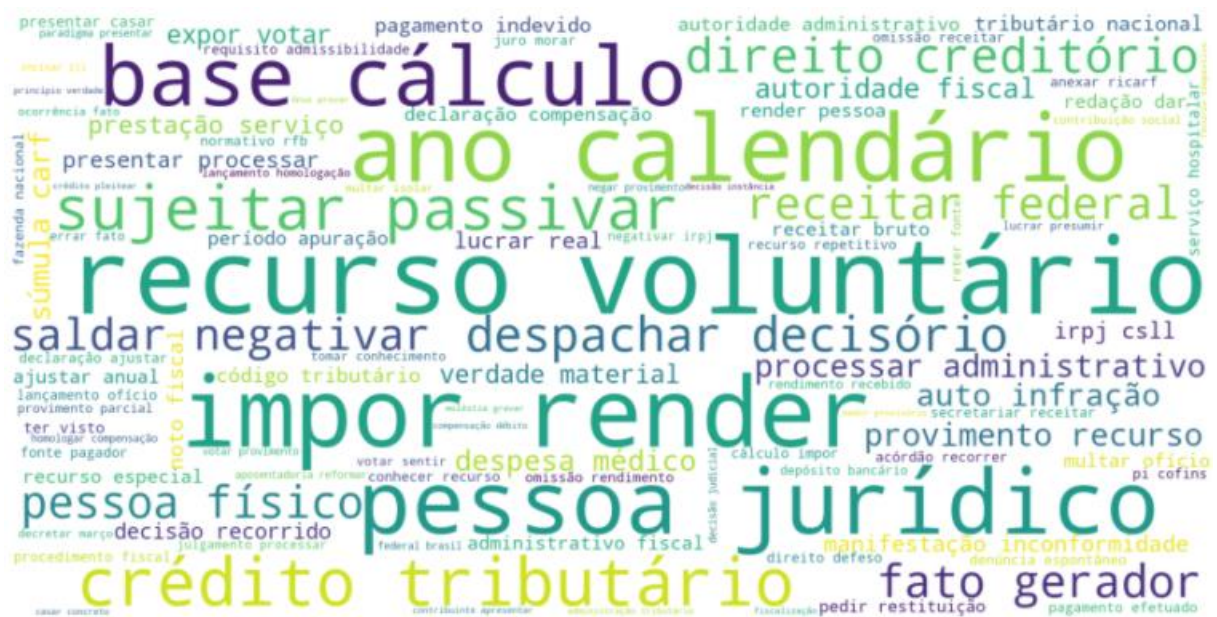
¹⁴ <https://pt.wikipedia.org/wiki/Stemização>

¹⁵ Dataframes gravados no drive pelo notebook TCC_p01_preparar_X_treinamento_e_y_teste descrito em anexo



Exemplos: tomo=tomar, cumpre=cumprir, expressa=expresso, respeito=respeitar

	Exemplos: tomar=tomar, cumprir=cumprir, expressa=expresso, respeito=respeitar	voto
0	recurso tempestivo cumprir requisito legar admissibilidade razão tomar conhecimento passar apreciar recurso tempestivo cumprir requisito legar admissibilidade razão tomar conhecimento passar aprec...	
1	registrar oportunidade casar vir julgamento recurso voluntário tempestivo recorrente devidamente representar preliminar nulidade recorrente alegar despachar decisório homologar dcomp nulo fundamen...	
2	recurso voluntário apresentar ater pressuposto admissibilidade ser dignar conhecimento surgir presentar lidar respeitar necessidade expresso declaração homologação compensação tela compulsar auto ...	
3	julgamento processar seguir sistemático recurso repetitivo regulamentar artigo anexar ricaf aprovar portar junho presentar litigio aplica-se decidir acórdão proferir julgamento processar paradigm...	
4	julgamento processar seguir sistemático recurso repetitivo regulamentar artigo anexar ricaf aprovar portar junho presentar litigio aplica-se decidir acórdão proferir julgamento processar paradigm...	



Inicialmente, criei o objeto *Tokenizer* fornecendo o número máximo de palavras para manter no vocabulário e um *token* fora do vocabulário para codificação de palavras dos dados de teste não encontradas no treinamento.

Em seguida, executei o método *fit_on_texts* para atualizar o vocabulário interno com base na lista de textos. Esse método é pré-requisito para o método *texts_to_sequences* que, efetivamente, é o que converte as palavras em uma sequência de números utilizando o índice de palavras criado na tokenização para as palavras do corpus recebido (no caso os dados de treinamento).

Cada seqüência tem uma quantidade de palavras convertidas em números, mas é preciso que todas as seqüências tenham o mesmo formato. Então, gerei todas as seqüências com o tamanho igual ao tamanho da maior seqüência. Os vázios para as seqüências menores foram preenchidos com zeros à direita. Esse preenchimento é feito pelo método *pad_sequences*.

Confira-se a seguir o código¹⁷.

```
1 # Tokenizar os dados de treinamento
2 tokenizer = Tokenizer(num_words=num_words, oov_token=oov_token)
3 tokenizer.fit_on_texts(X_treinamento_preprocessado)
4
5 # Obter total de palavras de dados de treinamento
6 vocab_size = len(tokenizer.word_index) + 1
7
8 # Transforma cada texto em uma sequência de inteiros.
9 X_treinamento_vetor_numerico = tokenizer.texts_to_sequences(X_treinamento_preprocessado)
10
11 # Obter o comprimento máximo da sequência de treinamento para uso no padding (preenchimento)
12 max_len = max([len(x) for x in X_treinamento_vetor_numerico])
13
14 # Transformar a lista de seqüências de tamanho variável em uma matriz 2D numpy de forma (número de linhas, comprimento da maior seqüência).
15 X_treinamento_vetor_numerico = pad_sequences(X_treinamento_vetor_numerico, padding=pad_type, truncating=trunc_type, maxlen=max_len)
```

3.4 Sumarização do voto

A sumarização dos votos também é um processo de transformação dos dados.

A sumarização recebe um voto como entrada e calcula o parâmetro quantidade de melhores frases igual a 40% do total de frases do voto.

¹⁷ Código disponível no notebook TCC_p03_Classificação_DCNN.

Em seguida, utilizando a técnica escolhida, calcula as melhores frases do voto recebido e devolve o resumo do voto composto por essas frases.

Confira-se a função que faz a sumarização utilizando a biblioteca Sumy. O objeto sumarizador que recebe o *parser*¹⁸ do voto e a quantidade de frases alvo é o sumarizador instanciado. A função é a mesma para todos os sumarizadores da biblioteca.

```

1 def sumarizar_sumy(texto):
2     """
3     Recebe uma coleção de frases e escolhe as melhores na proporção 40%.
4     :parâmetro texto: recebe o texto integral de um voto.
5     : retorno melhores_sentencas: string com as 40% melhores sentenças.
6     """
7     parser = PlaintextParser.from_string(texto, TokenizerSumy('portuguese'))
8     quantidade_sentencas = int(0.4 * len(parser.document.sentences))
9     if quantidade_sentencas < 4:
10         quantidade_sentencas = 4
11
12     resumo = sumarizador(parser.document, quantidade_sentencas)
13     melhores_sentencas = []
14     for sentenca in resumo:
15         melhores_sentencas.append(str(sentenca).replace("Sentence: ", ""))
16     melhores_sentencas = str(melhores_sentencas).replace("'", "").replace('"', '').replace('[', '').replace(']', '')
17
18     return melhores_sentencas

```

3.5 Tratamento da variável alvo

A variável alvo contém os assuntos dos votos, IRPJ, IRPF, NGDT e PAF.

O modelo da biblioteca SpaCy utiliza a variável alvo no formato de um dicionário com as chaves iguais às 4 categorias e o valor True para a categoria correta e False para as demais categorias.

Por isso, houve a necessidade de percorrer todos os registros para converter a variável alvo em dicionário. Confira-se o código a seguir¹⁹.

¹⁸ Parser é um tokenizador da biblioteca Sumy. Retorna um documento dividido em sentenças, palavras, etc.

¹⁹ Código disponível no notebook TCC_p02_Classificacao_Spacy.

```

1 base_dados_treinamento_final = []
2 for voto, assunto in zip(df_treinamento['voto'], df_treinamento['assunto']):
3     if assunto == 'irpf':
4         dic = ({'irpf': True, 'ngdt': False, 'irpj': False, 'paf': False})
5     elif assunto == 'ngdt':
6         dic = ({'irpf': False, 'ngdt': True, 'irpj': False, 'paf': False})
7     elif assunto == 'irpj':
8         dic = ({'irpf': False, 'ngdt': False, 'irpj': True, 'paf': False})
9     elif assunto == 'paf':
10        dic = ({'irpf': False, 'ngdt': False, 'irpj': False, 'paf': True})
11    base_dados_treinamento_final.append([voto, dic.copy()])

```

No caso da rede neural convolucional, a variável alvo precisa ser convertida em um valor numérico. Confira-se o código a seguir²⁰.

```

1 y_treinamento_rotulos[y_treinamento_rotulos == 'irpj'] = 0
2 y_treinamento_rotulos[y_treinamento_rotulos == 'irpf'] = 1
3 y_treinamento_rotulos[y_treinamento_rotulos == 'ngdt'] = 2
4 y_treinamento_rotulos[y_treinamento_rotulos == 'paf'] = 3
5 y_treinamento_rotulos = y_treinamento_rotulos.astype(np.int32)
6 y_treinamento_rotulos

```

4. Análise e Exploração dos Dados

4.1 Dataframe Carf - atributos

O dataframe Carf – atributos tem 18.586 linhas. A variável assunto é a variável alvo para a classificação.

Nos processos extraídos, foram encontradas 30 categorias de assunto, sendo irpf a categoria com maior frequência, 3.734.

```
1 df_a.assunto.describe()
```

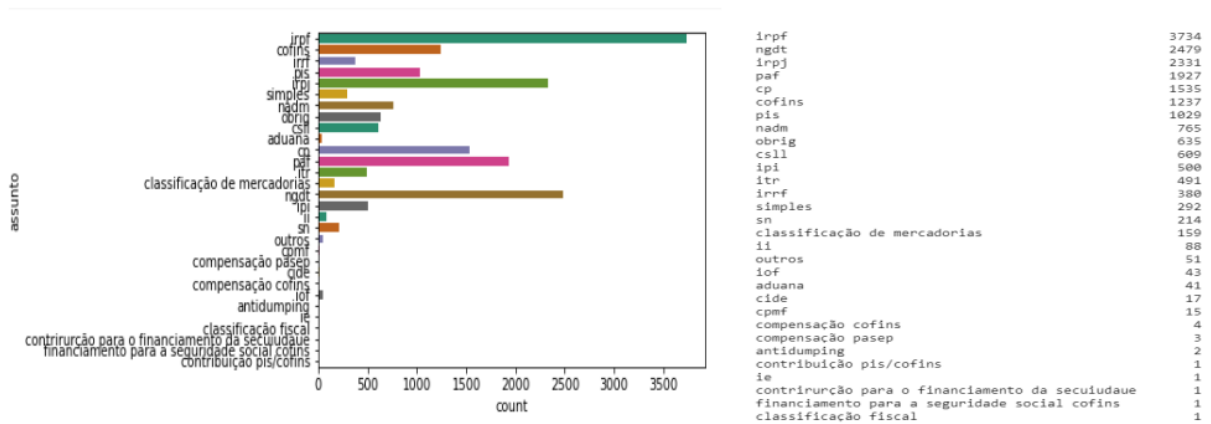
```

count      18586
unique         30
top         irpf
freq        3734
Name: assunto, dtype: object

```

²⁰ Código disponível no notebook TCC_p03_Classificação_DCNN.

O gráfico de contagem ilustra a distribuição da variável categórica assunto.

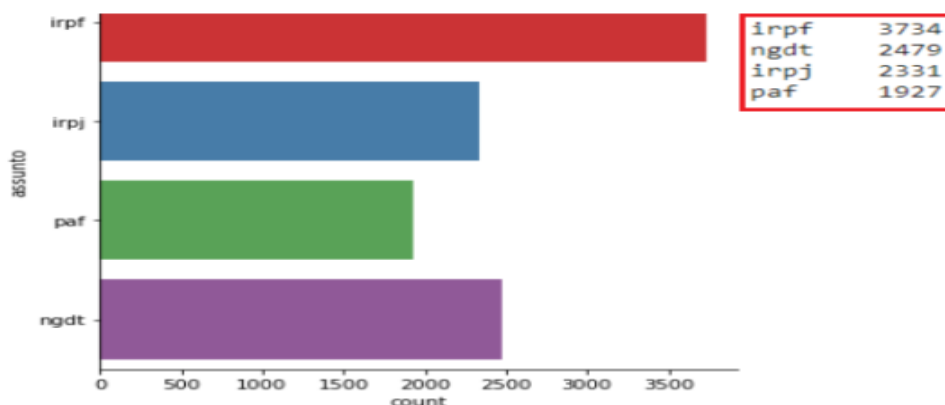


Selecionei os 4 assuntos com maior quantidade de processos. Eram eles o Imposto de Renda da Pessoa Jurídica (IRPJ), o Imposto de Renda da Pessoa Física (IRPF), as Normas gerais de direito tributário (NGDT) e o Processo administrativo fiscal (PAF).

Foram selecionados: IRPF 3734; NGDT 2479; IRPJ 2331 e PAF 1927.

Os assuntos são definidos e cadastrados pelo relator do voto. IRPJ são os acórdãos relativos à controvérsia do tributo Imposto de Renda da Pessoa Jurídica. IRPF são os relativos à Imposto de Renda de Pessoa Física. NGDT não são relativos a um tipo de tributo, mas às normas gerais que permeiam todo o sistema tributário como, por exemplo, o instituto da Decadência. Por fim, PAF, também é matéria que permeia o sistema sendo as normas processuais relativas ao processo administrativo fiscal.

A nova distribuição ficou assim.

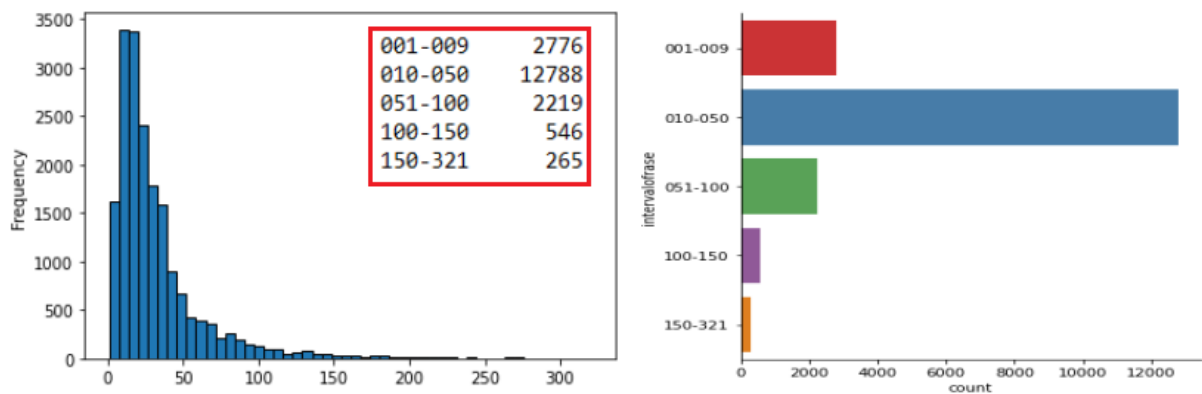


4.2 Dataframe Carf_Voto_Dividido_Em_Frases

O foco do trabalho é a verificação da qualidade dos sumarizadores.

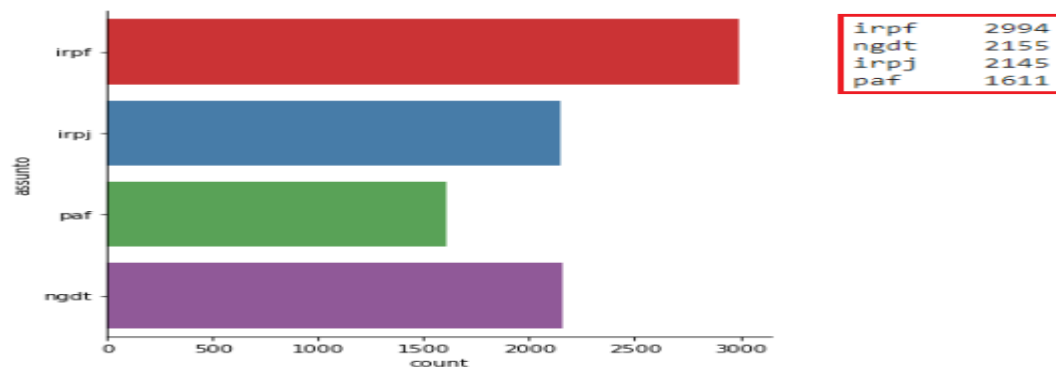
A quantidade de frases do voto é um parâmetro fundamental para a sumarização, pois, sendo do tipo extrativo, os sumarizadores precisam receber a quantidade de frases que serão mantidas no resumo.

A distribuição dos processos por quantidade de frase demonstra que a maioria dos processos estão concentrados no intervalo entre 10 e 50 frases. A quantidade média de frases é 32,36.

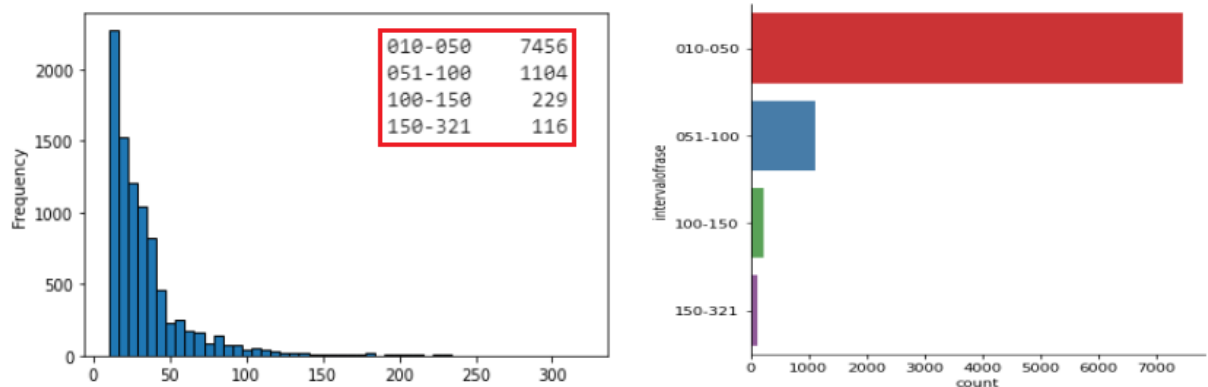


O objetivo de calcular o número de sentenças foi selecionar apenas os textos com pelo menos 10 sentenças, pois a sumarização de textos pequenos não é muito útil e poderia prejudicar a análise da qualidade dos sumarizadores.

Após a seleção por assunto e por quantidade de frases, ficou assim a nova distribuição por assunto.



Nova distribuição por quantidade de frases. A nova média de frases ficou 34,34.



4.3 Dataframe Carf_Ementas

O dataframe Carf_Ementas foi utilizado para a formação da lista de palavras bônus para o sumariador Edmundson.

	Processo	Ementa
0	10070000461200797	Ementa(s) Assunto: Imposto sobre a Renda de Pessoa Física - IRPF Ano-calendário: 2003 IRPF. OMISSÃO DE RENDIMENTOS DO TRABALHO. AUXÍLIO MATERNIDADE. A fonte pagadora dispõe de convênio com o INSS...
1	10070000654200748	Ementa(s) Assunto: Imposto sobre a Renda de Pessoa Física - IRPF Exercício: 2004 OMISSÃO DE RENDIMENTOS Os rendimentos tributáveis recebidos pelo contribuinte devem ser integralmente informados em...
2	10070000947200644	Ementa(s) Assunto: Imposto sobre a Renda de Pessoa Física - IRPF Exercício: 2003 RENDIMENTOS DECORRENTES DE AÇÃO JUDICIAL. DESPESAS COM ADVOGADO. Poderá ser deduzido, para fins de determinação da...

As palavras da ementa receberam o mesmo tratamento descrito no item 3.2 *Preprocessamento do voto* para limpeza dos dados deixando apenas as palavras. No caso, sem o tratamento de lematização ou stemização, pois as palavras bônus são usadas como apoio à sumarização que trabalha o texto em seu formato original.

A visualização abaixo mostra a ementa após o tratamento.

	Processo	Ementa
0	10070000461200797	imposto renda pessoa irpf omissão rendimentos trabalho maternidade dispõe convênio inss pagamento maternidade comprovado autos diligência equivoco duplicidade informações fontes afastada omissão r...
1	10070000654200748	imposto renda pessoa exercicio omissão rendimentos rendimentos recebidos contribuinte informados declaração cabendo lançamento parcela omitida contribuição previdência dedução determinação cálculo...
2	10070000947200644	imposto renda pessoa exercicio rendimentos ação despesas advogado deduzido fins determinação cálculo sujeita incidência despesas ação recebimento rendimentos

Em seguida, foi calculada a frequência das palavras e, após visualização dos dados, definido o corte em 200 ocorrências. Assim, a lista de palavras das ementas com 200 ou mais ocorrências foi gravada no drive para utilização pelo sumariador Edmundson.

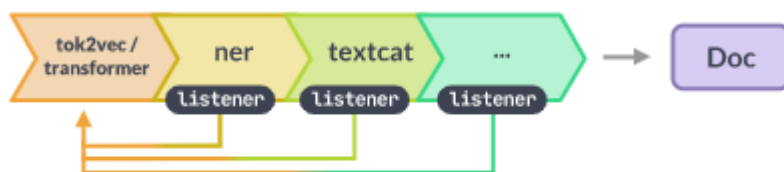
A seguir, nuvem das palavras mais frequentes das ementas.

Os modelos utilizaram os três tipos de formatos de palavras. Portanto, combinados, são seis tipos de modelos treinados para escolha do modelo de *machine learning* que será usado para a classificação dos resumos que é a segunda parte do trabalho de classificação.

Todos os treinamentos foram feitos com base no mesmo conjunto de documentos (X e y treinamento) e foram validados no mesmo conjunto de teste (X e y teste), conforme descrito no item 3.1 *Preparação dos dataframes X e y (treinamento e teste)*.

5.1 Classificação dos votos com a SpaCy

Para a categorização pela SpaCy foi usada a arquitetura padrão que é o conjunto empilhado de um modelo linear de saco de palavras e um modelo de rede neural. A rede neural é construída sobre uma camada Tok2Vec e usa atenção²³.



Um pipeline com uma camada de incorporação compartilhada à qual os componentes podem se "conectar" por meio de um ouvinte

Fonte: <https://explosion.ai/blog/spacy-v3>

A SpaCy apresenta modelos de aprendizado profundo para reconhecimento de entidade nomeada, análise de dependência, classificação de texto e previsão de similaridade com base em arquitetura que pode ser resumida como uma fórmula simples de quatro etapas: incorporar , codificar , assistir , prever.

O artigo “Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models”²⁴, de Matthew Honnibal, desenvolvedor da SpaCy,

²² O TensorFlow é uma plataforma completa de código aberto para machine learning. Ele tem um ecossistema abrangente e flexível de ferramentas, bibliotecas e recursos da comunidade que permite aos pesquisadores levar adiante machine learning de última geração e aos desenvolvedores criar e implantar aplicativos com tecnologia de machine learning. <https://www.tensorflow.org/?hl=pt-br>

²³ <https://spacy.io/api/textcategorizer> e <https://spacy.io/api/architectures#TextCatEnsemble>

descreve a arquitetura utilizada pela SpaCy. Confira o resumo extraído do artigo: "*Uma estratégia de quatro etapas para aprendizado profundo com texto - As representações de palavras incorporadas, também conhecidas como “vetores de palavras”, são agora uma das tecnologias de processamento de linguagem natural mais amplamente utilizadas. Os embeddings de palavras permitem que você trate palavras individuais como unidades de significado relacionadas, em vez de IDs totalmente distintos. No entanto, a maioria dos problemas de PNL requer a compreensão de extensões mais longas de texto, não apenas de palavras individuais. Agora existe uma solução simples e flexível que está obtendo excelente desempenho em uma ampla gama de problemas. Depois de embutir o texto em uma sequência de vetores, RNNs bidirecionais são usados para codificar os vetores em uma matriz de frase. As linhas desta matriz podem ser entendidas como vetores de token- eles são sensíveis ao contexto sentencial do token. A peça final do quebra-cabeça é chamada de mecanismo de atenção. Isso permite que você reduza a matriz da frase a um vetor de frase, pronto para a previsão.*"

O categorizador de texto utilizado *textcat* prevê categorias em um documento inteiro não sendo necessário transformar as palavras em vetores, pois isso é feito internamente pela biblioteca. O modelo pode aprender um ou mais rótulos, e os rótulos são mutuamente exclusivos - há exatamente um rótulo verdadeiro por documento. Confira o código a seguir.

```
1 modelo = spacy.blank('pt')
2 categorias = modelo.create_pipe("textcat")
3 categorias.add_label("irpf")
4 categorias.add_label("ngdt")
5 categorias.add_label("irpj")
6 categorias.add_label("paf")
7 modelo.add_pipe(categorias)
8 historico = []
```

A variável alvo é passada como um dicionário com as categorias existentes conforme explicado no item 3.5 *Tratamento da variável alvo*.

Esse procedimento foi realizado no notebook 02. Lá, estão os códigos python utilizados com descrição da sequência de eventos e com comentários nos códigos. A seguir, o código do treinamento do modelo.

²⁴ <https://explosion.ai/blog/deep-learning-formula-nlp>

```

1 #Rodou o treinamento em 9547s com 30 épocas passando de 512 em 512 registros, ou 14 batches por época.
2 #Treinamento com lematização
3 modelo.begin_training()
4 for epoca in range(30): #número de épocas. 30
5     random.shuffle(base_dados_treinamento_final)
6     losses = {}
7     for batch in spacy.util.minibatch(base_dados_treinamento_final, 512): #número de registros passados.
8         textos = [modelo(texto) for texto, entities in batch]
9         annotations = [{'cats': entities} for texto, entities in batch]
10        modelo.update(textos, annotations, losses=losses)
11        historico.append(losses)
12    if epoca % 5 == 0: #mostrar de 5 em 5 épocas.
13        print(losses)

```

```

{'textcat': 3.8230729160204646e-05}
{'textcat': 5.282458772626342e-06}
{'textcat': 1.6528499102719252e-06}
{'textcat': 8.78107712765086e-07}
{'textcat': 6.063314117454865e-07}
{'textcat': 5.040392290212026e-07}

```

O modelo da SpaCy foi treinado recebendo o conjunto de palavras do voto lematizadas, stemizadas e sem tratamento, ou seja, palavras originais.

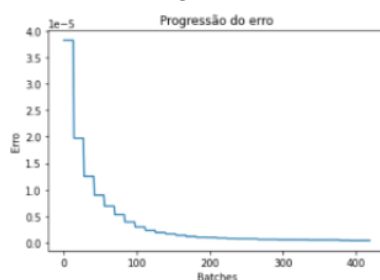
Os únicos parâmetros passados para o modelo foram o número de épocas e a quantidade de registros no treinamento para a atualização dos pesos (minibatch).

Após a execução dos treinamentos, foram plotados gráficos de progressão dos erros em relação às épocas (o número de execução por época foi de 14 – razão entre a quantidade de registros 7.124 e o número de lotes).

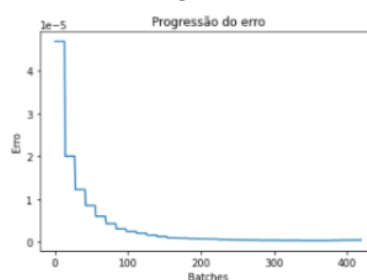
Foram feitos testes utilizando 30 e 50 épocas. Com 50 épocas, o tempo de processamento foi excessivo e sem melhora nos resultados.

Conforme demonstram os gráficos seguintes, em todos os casos, há uma estabilização dos erros após 200 batches (aproximadamente 14 épocas) indicando que o número de épocas usado no treinamento, 30, estava satisfatório.

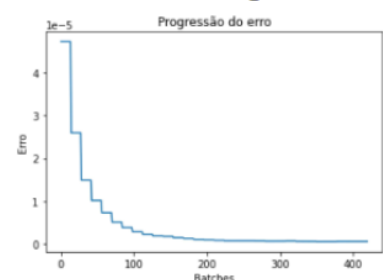
Lematização



Stemização



Palavras originais



5.2 Classificação dos votos usando TensorFlow

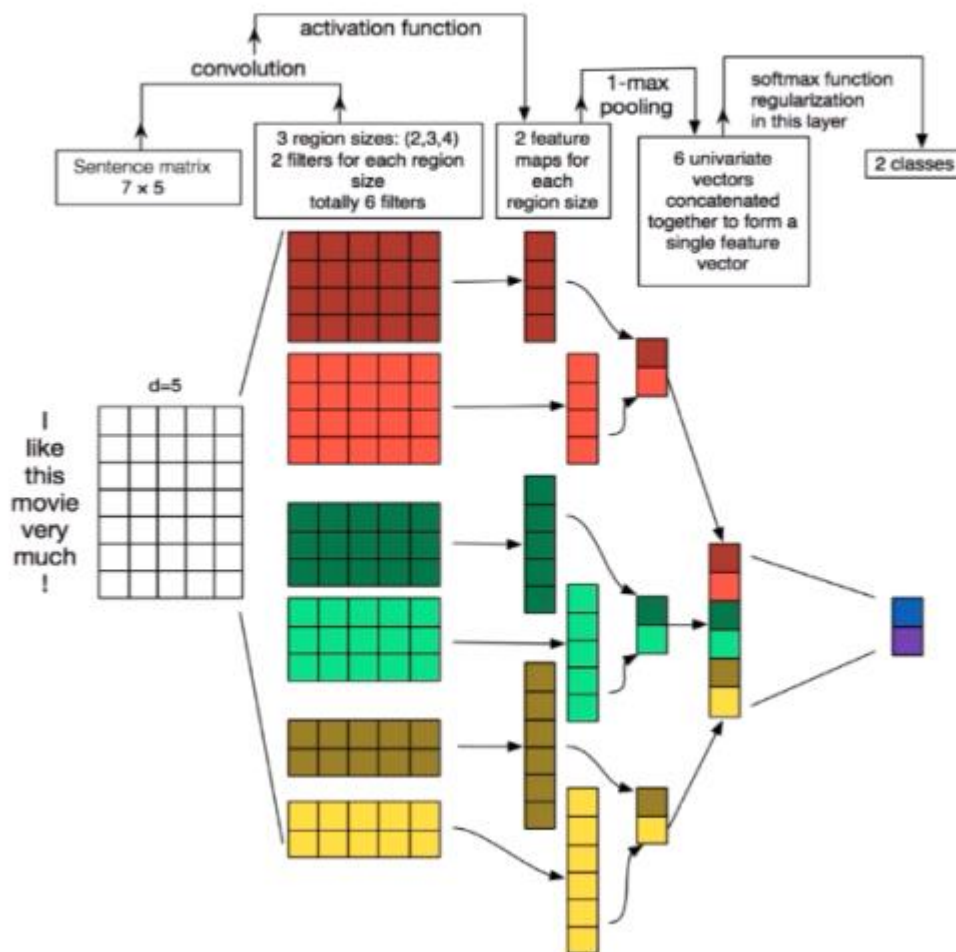
Para a categorização com o TensorFlow, optei por utilizar uma rede neural convolucional.

Uma rede neural convolucional profunda (DCNN) consiste em muitas camadas de redes neurais. Dois tipos diferentes de camadas, convolucionais e pooling, são normalmente alternados.

Nas camadas de convolução, a informação passa por vários filtros (que na prática são matrizes numéricas) com a função de acentuar padrões regulares locais, ao mesmo tempo em que vão reduzindo a dimensão dos dados originais. Os resultados de vários filtros são sumarizados por operações de pooling. Na parte mais profunda das convoluções, espera-se que os dados num espaço dimensional reduzido contenham informação suficiente sobre esses padrões locais para atribuir um valor semântico ao dado original. Esses dados passam então por uma estrutura de FFN clássica para a tarefa de classificação²⁵. Confira o modelo a seguir.

²⁵ <https://iaexpert.academy/2020/06/08/os-tipos-de-redes-neurais/>

CNN para NLP – arquitetura



Fonte: <https://iaexpert.academy/courses/processamento-linguagem-natural-deep-learning-transformer/>

No caso da DCNN, é necessário transformar o documento em um saco de palavras e as palavras em vetores numéricos. Os rótulos também são transformados em categorias numéricas. A preparação dos dados está descrita no item 3.3 *Vetorização do voto*.

Foi utilizada uma classe DCNN herdando de `tf.keras.model` (classe base do keras do tensorflow para a construção de modelos de redes neurais). O modelo agrupa camadas em um objeto com recursos de treinamento e inferência. São duas as maneiras de instanciar o modelo : 1 - Com a API Funcional ou 2 - Subclassificando a classe Model. Foi utilizada a subclassificação da classe, conforme descrito no curso Processamento de Linguagem Natural com Deep

Learning, IA Expert Academy, tópico Redes neurais convolucionais para PLN – implementação.

Esse procedimento foi realizado no notebook 03. Lá, estão os códigos python utilizados com descrição da seqüência de eventos e com comentários nos códigos. A seguir, o código da classe DCNN e do treinamento.

```

1 class DCNN(tf.keras.Model):
2     '''
3     Construção do modelo:
4     Para isso, utilizei uma classe DCNN herdando de tf.keras.model (classe base do keras do tensorflow para a construção de modelos de redes neurais).
5     O modelo agrupa camadas em um objeto com recursos de treinamento e inferência.
6     São duas duas maneiras de instanciar o modelo : 1 - Com a API Funcional ou 2 - Subclassificando a classe Model.
7     Optei pela subclassificação da classe, conforme descrito no curso Processamento de Linguagem Natural com Deep Learning, IA Expert Academy,
8     tópico Redes neurais convolucionais para PLN – implementação.
9     Os parâmetros do construtor da classe utilizados foram Self (indica que ao criarmos uma instância desta classe,
10    ela vai se referir a um objeto e não à própria classe), vocab_size (tamanho do vocabulário - varia conforme tratamento das palavras por lema,
11    stemer ou palavras originais), emb_dim (200 - tamanho da matriz de embeddings escolhido após alguns testes.
12    Não fiz tuning, pois a classificação não é o foco do TCC), nb_filters (100 - número de filtros para cada dimensão),
13    ffn_units (256 - número de neurônios da rede neural densa) e nb_classes (4 - número de classes ou categorias), dropout_rate=0.2
14    (técnica utilizada para evitar overfitting . Serão zerados 20% dos neurônios).
15    O método construtor da classe é chamado para criar um novo modelo.
16    A camada de embedding gera uma matriz linha igual ao tamanho do vocabulário linhas e 200 colunas conforme definido em emb_dim.
17    O objetivo é encontrar redução de dimensionalidade na representação dos vetores e ao mesmo tempo adicionar relação semântica entre as palavras.
18    Definição das camadas de convolução:
19    São 3 camadas de convolução com 100 filtros de 2 linhas na primeira camada, 3 linhas na segunda e 4 linhas na terceira.
20    O preenchimento padding= same retorna o mesmo tipo de dados no mesmo formato. Como função de ativação utilizei a padrão, relu.
21    Pooling (Agregação) e concatenação:
22    Em seguida, está a definição da camada de pooling para reduzir escala (downsampling) que irá gerar o vetor final
23    para entrada na rede neural densa. Utilizada a função GlobalMaxPool1D que reduz a resolução da representação de entrada,
24    assumindo o valor máximo após o produto escalar entre os vetores que representam as palavras e os vetores que representam os filtros.
25    Estrutura da rede neural densa
26    A primeira camada oculta tem 256 neurônios e não precisamos definir a quantidade de neurônios da camada de entrada,
27    pois depois iremos juntar o final do pooling com a entrada e o próprio Tensorflow fará automaticamente.
28    A função de ativação SoftMax é usada (normaliza entre 0 e 1 as saídas, em um problema multi-classes, obtidas por um classificador linear - ex. neurônios -
29    e é aplicada na camada de saída da rede - output layer), pois temos 4 categorias e a função irá retornar uma probabilidade para cada uma das classes.
30    Foi definida função para ligar a matriz de embedding com os filtros da camada bigram (duas palavras) e desta para a camada de pooling,
31    com os filtros da camada trigram (três palavras) e desta para a camada de pooling, e com filtros da camada fourgram (quatro palavras) e
32    desta para a camada de pooling. O processamento é em paralelo. Ao final, as saídas são concatenadas e tornam-se a entrada para a rede neural em seqüência,
33    camada densa, camada de dropout e, finalmente, a saída (output layer).
34    Treinamento:
35    Os únicos parâmetros passados foram o número de épocas (5) e a quantidade de registros no treinamento para a atualização dos pesos (minibatch = 64).
36    '''

```

```

def __init__(self,
              vocab_size,
              emb_dim=200,
              nb_filters=100,
              ffn_units=256,
              nb_classes=4,
              dropout_rate=0.2,
              training=True,
              name="dcnn"):
    super(DCNN, self).__init__(name=name)

    self.embedding = layers.Embedding(vocab_size, emb_dim)
    self.bigram = layers.Conv1D(filters=nb_filters, kernel_size=2, padding='same', activation='relu')
    self.trigram = layers.Conv1D(filters=nb_filters, kernel_size=3, padding='same', activation='relu')
    self.fourgram = layers.Conv1D(filters=nb_filters, kernel_size=4, padding='same', activation='relu')
    self.pool = layers.GlobalMaxPool1D()

    self.dense_1 = layers.Dense(units = ffn_units, activation = 'relu')
    self.dropout = layers.Dropout(rate = dropout_rate)
    self.last_dense = layers.Dense(units = nb_classes, activation = 'softmax')

def call(self, inputs, training):
    x = self.embedding(inputs)
    x_1 = self.bigram(x)
    x_1 = self.pool(x_1)
    x_2 = self.trigram(x)
    x_2 = self.pool(x_2)
    x_3 = self.fourgram(x)
    x_3 = self.pool(x_3)

    merged = tf.concat([x_1, x_2, x_3], axis = -1) # (batch_size, 3 * nb_filters)
    merged = self.dense_1(merged)
    merged = self.dropout(merged, training)
    output = self.last_dense(merged)

    return output

```

```

1 Dcnn = DCNN(vocab_size=vocab_size, emb_dim=emb_dim, nb_filters=nb_filters,
2             ffn_units=ffn_units, nb_classes=nb_classes, dropout_rate=dropout_rate)
3 Dcnn.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

```

1 history = Dcnn.fit(X_treinamento_vetor_numerico, y_treinamento_rotulos,
2                   batch_size = batch_size,
3                   epochs = nb_epochs,
4                   verbose = 1,
5                   validation_split = 0.10)

```

```

Epoch 1/5
101/101 [=====] - 1335s 13s/step - loss: 1.0086 - accuracy: 0.5870 - val_loss: 0.4031 - val_accuracy: 0.8668
Epoch 2/5
101/101 [=====] - 1338s 13s/step - loss: 0.2919 - accuracy: 0.9082 - val_loss: 0.3237 - val_accuracy: 0.8822
Epoch 3/5
101/101 [=====] - 1331s 13s/step - loss: 0.0883 - accuracy: 0.9804 - val_loss: 0.3398 - val_accuracy: 0.8738
Epoch 4/5
101/101 [=====] - 1333s 13s/step - loss: 0.0361 - accuracy: 0.9926 - val_loss: 0.3331 - val_accuracy: 0.8864
Epoch 5/5
101/101 [=====] - 1336s 13s/step - loss: 0.0146 - accuracy: 0.9977 - val_loss: 0.3583 - val_accuracy: 0.8864

```

Para o treinamento da DCNN, os parâmetros passados foram: dimensões de embedding = 200, número de filtros por região da camada convolucional = 100, número de neurônios na camada escondida = 256, número de épocas 5, taxa de dropout = 0,2 e número de classes = 4.

Como o foco do trabalho é a comparação entre os sumarizadores e não a qualidade dos classificadores, não foi feita otimização dos parâmetros.

5.3 Classificação dos votos sumarizados

De acordo com o artigo Sumarização de Textos com Processamento de Linguagem Natural – IA Expert, “a área de *Processamento de Linguagem Natural – PLN (Natural Language Processing – NLP)* é uma subárea da *Inteligência Artificial* que tem como objetivo tornar os computadores capazes de entender a linguagem humana, tanto escrita quanto falada. Alguns exemplo de aplicações práticas são: tradutores entre idiomas, tradução de texto para fala ou fala para texto, chatbots, sistemas automáticos de perguntas e respostas, geração automática de descrições para imagens, adição de legendas em vídeos, classificação de sentimentos em frases, dentre várias outras! Outro exemplo importante de aplicação é a sumarização automática de documentos, que consiste em gerar resumos de textos. Vamos supor que você precise ler um artigo com 50 páginas, porém, não possui tempo suficiente para ler o texto integral. Nesse caso, você pode utilizar um algoritmo de sumarização para gerar um resumo deste artigo. O tamanho deste resumo pode ser configurável, ou seja, você pode transformar 50 páginas em um texto com somente 20 páginas que contenha somente os pontos mais importantes do texto!”²⁶.

De acordo com Pardo (2008, p. 4, apud Jones, 1993a), “em termos de formação, sumários podem ser classificados como extratos ou abstracts”. Pardo acrescenta que “Extratos são sumários compostos por trechos inalterados do texto-fonte. Eles são construídos por operações de cópia e cola de trechos do texto-fonte, literalmente. Abstracts, por sua vez, apresentam partes (ou mesmo tudo) reescritas, ou seja, há algum nível de modificação na estrutura e/ou significado dos trechos extraídos do texto-fonte. O termo abstract também foi importado da língua inglesa e assim é utilizado em português. Não é incomum encontrar o termo traduzido como “abstrato” em português, apesar de pouco aceito na área. Em geral, o termo sumário é utilizado para se referir tanto a extrato quanto a abstract”.

²⁶ <https://iaexpert.academy/courses/sumarizacao-de-textos-com-processamento-de-linguagem-natural/>

No presente trabalho, os sumarizadores testados são do tipo extrato, ou seja, resumos são formados pelas frases do texto original sem alterações.

Dos sumarizadores escolhidos, sete são objetos da biblioteca Sumy e dois são códigos em python.

Em todos os casos, o resumo será formado por 40% das frases do texto original.

5.3.1 Usando a codificação em python das técnicas

Algoritmo de Luhn: Luhn propôs a criação automática de resumos de texto selecionando-se as frases mais importantes do texto de acordo com a sua frequência. As palavras significativas são aquelas que aparecem com mais frequência no texto, mas, ao mesmo tempo, não fazem parte das *stop words* do idioma. As palavras com frequências muito altas ou muito baixas seriam desconsideradas²⁷.

Para a sumarização pelo algoritmo de Luhn foi utilizado o código disponibilizado no curso IA Expert Academy - Sumarização de Textos com Processamento de Linguagem Natural. O procedimento foi realizado no notebook 05.

Há uma função para calcular a nota da sentença com base na frequência das palavras e outra que seleciona as melhores sentenças com base na nota da função anterior. 40% das frases do documento original é a quantidade de sentenças selecionadas.

Similaridade do cosseno: Neste método, a importância das frases é dada pela nota de semelhança de cada frase com as demais calculada pela similaridade do cosseno²⁸ e classificada com a utilização do algoritmo PageRank²⁹.

²⁷ Fabrício Shiguero Catae – Classificação Automática de Texto por meio de similaridade de palavras: um algoritmo mais eficiente, 2012.

²⁸ Similaridade de cosseno é uma medida de similaridade entre dois vetores diferentes de zero de um espaço de produto interno. É definida como igual ao cosseno do ângulo entre eles, que também é o mesmo que o produto interno dos mesmos vetores normalizados para ambos terem comprimento 1.
https://en.wikipedia.org/wiki/Cosine_similarity

Para a sumarização pela similaridade do cosseno foi utilizado o código disponibilizado no curso IA Expert Academy - Sumarização de Textos com Processamento de Linguagem Natural. O procedimento foi realizado no notebook 06.

Há uma função para calcular a similaridade entre as sentenças. A função recebe duas sentenças/frases e cria uma lista sem repetição das palavras das duas sentenças. Em seguida, cria um vetor para cada uma das sentenças com o número de posições igual ao número de palavras. Depois, para cada sentença, conta a quantidade de vezes que cada palavra ocorre e registra a quantidade na posição correspondente do vetor. Ao final, calcula a similaridade entre os vetores (sentenças) utilizando a função *cosine_distance* da biblioteca NLTK. Quanto menor a distância, maior a similaridade. A distância varia entre 0 e 1.

Há uma função para sumarizar o texto. Ela recebe o texto/documento e obtém a matriz com a similaridade entre as sentenças do documento utilizando a função anterior. Transforma a matriz em um grafo. O grafo mostra as ligações entre as sentenças. Por meio do algoritmo *PageRank* calcula a classificação das frases (nodos dos grafos) com base na estrutura dos links de entrada (a ligação entre as sentenças). Com a classificação das frases, calcula a quantidade de sentenças que representa 40% do documento e obtém as melhores sentenças com base nessa quantidade.

5.3.2 Usando os sumarizadores da biblioteca Sumy³⁰

Os procedimentos de sumarização utilizando a biblioteca Sumy foram realizados no notebook 07.

Todos os sumarizadores recebem o parâmetro quantidade de sentenças igual a 40% das frases do documento a ser sumarizado.

²⁹ é um algoritmo utilizado pela ferramenta de busca Google para posicionar websites entre os resultados de suas buscas. O PageRank mede a importância de uma página contabilizando a quantidade e qualidade de links apontando para ela. Não é o único algoritmo utilizado pelo Google para classificar páginas da internet, mas é o primeiro utilizado pela companhia e o mais conhecido.

³⁰ A descrição das técnicas de sumarização foi obtida em <https://miso-belica.github.io/sumy/summarizators.html>

A seguir, breve descrição sobre as técnicas de sumarização.

LuhnSummarizer: Utiliza o algoritmo de Luhn cuja técnica foi descrita no item anterior, mas o procedimento é feito pela biblioteca Sumy.

EdmundsonSummarizer: Método heurístico com pesquisa estatística anterior - o aprimoramento do método Luhn. Edmundson adicionou mais 3 heurísticas ao método para medir a importância das sentenças. Ele encontra as chamadas palavras pragmáticas, as palavras que estão nos cabeçalhos e a posição dos termos extraídos. Portanto, este método possui 4 sub-métodos e a combinação adequada deles resulta no método de Edmundson. Importante ressaltar que este método é o mais dependente do idioma porque precisa da lista de palavras bônus, palavras estigmatizadas e palavras irrelevantes (chamadas de palavras nulas).

Por outro lado, a construção da lista de palavras permite uma adequação maior ao seu conjunto de dados.

Neste trabalho, a criação da lista de palavras bônus foi elaborada a partir do dataset *Carf – ementas* e pelas palavras mais freqüentes das duas últimas frases do voto.

Conforme já descrito, as ementas são uma síntese da decisão e trazem uma sequência de palavras chaves que indicam o assunto, a tese jurídica e o conteúdo da decisão. Logo, são bastante adequadas à formação de um corpus que influencie o sumador na escolha das melhores sentenças.

A lista das palavras irrelevantes foi formada pelas *stop words* da SpaCy e da NLTK adicionadas dos nomes dos relatores.

A geração das listas de palavras bônus e palavras nulas foi feita no notebook 04 conforme explicado no item 4. *Análise e Exploração dos Dados*.

Não houve formação de lista de palavras estigmatizadas.

LsaSummarizer: Método algébrico - o método mais avançado é independente do idioma, mas também o mais complicado (computacionalmente e

mentalmente). O método é capaz de identificar sinônimos no texto e os tópicos que não estão explicitamente escritos no documento. Usa semântica latente e avaliação resumida.

LexRankSummarizer: Abordagem não supervisionada inspirada nos algoritmos *PageRank* e *HITS* - algoritmos criados para a rede mundial de computadores. Eles tentam encontrar conexões entre as frases e identificar aquelas relacionadas com as palavras / tópicos mais significativos.

TextRankSummarizer³¹: é um modelo de classificação baseado em gráfico. A maneira de decidir a importância de um vértice dentro do gráfico é pela “votação” ou “recomendação”. Quando um vértice se vincula a outro, ele basicamente está lançando um voto para aquele outro vértice. A conexão entre duas sentenças é determinada se houver uma relação de “similaridade” entre elas, onde a “similaridade” é medida em função de sua sobreposição de conteúdo. Quanto maior o número de votos que são lançados para um vértice, maior a importância do vértice. Abordagem não supervisionada inspirada nos algoritmos *PageRank*.

SumBasicSummarizer: Método frequentemente usado como linha de base na literatura para comparar a pontuação dos algoritmos. Não tem nenhuma vantagem especial sobre o LSA ou o TextRank.

RandomSummarizer: Método de teste que não deve ser usado para aplicações do mundo real. É usado apenas durante a avaliação dos resumos para comparação com os outros algoritmos. A ideia por trás disso é que se algum sumariador tiver uma pontuação pior do que este, provavelmente é um algoritmo muito ruim ou há alguma falha/bug sério na implementação.

Após a sumarização dos dados de testes, eles foram submetidos ao modelo de classificação selecionado, DCNN utilizando palavras lematizadas, e as acurácias de cada sumariador foram comparadas elegendo-se os melhores sumariadores com base nelas.

³¹ <https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf>

5.4 Pesquisa

Na perspectiva da Receita Federal, o objetivo dos trabalhos de TCC no curso de pós-graduação oferecido aos seus servidores é a possibilidade de aplicação às suas atividades. Sob esta ótica, é importante validar a qualidade dos sumarizadores sob o olhar de servidores que poderiam beneficiar-se das suas conclusões.

Embora para o TCC, isso não seja pré-requisito, decidi elaborar uma pesquisa sobre os sumarizadores com a participação de julgadores.

Como a participação na pesquisa é voluntária e para evitar a falta de colaboradores, optei por submeter ao crivo dos colegas apenas 3 sumarizadores.

Selecionei o sumariador Edmundson que obteve os melhores resultados na avaliação automática proposta neste trabalho.

Selecionei o sumariador Lsa que obteve o segundo melhor resultado, é uma outra técnica, e é considerado um método avançado capaz de identificar sinônimos no texto e tópicos que não estão explicitamente escritos no documento, conforme descrição supra.

Não selecionei o Luhn – Sumy considerando que o Edmundson é o aprimoramento do Luhn com o acréscimo das palavras bônus, estigmatizadas e irrelevantes. Sendo assim, preferi selecionar outra técnica.

Por fim, selecionei o sumariador TextRank que é baseado em grafos e é uma abordagem não supervisionada inspirada nos algoritmos PageRank. O TextRank obteve a terceira melhor nota de avaliação,

Em seguida, selecionei 48 processos aleatoriamente. Todos foram sumarizados pelos 3 sumarizadores. Cada julgador recebeu os 48 processos sendo 16 sumarizados por cada um dos 3 sumarizadores. Assim, todo julgador poderia avaliar resumos elaborados pelos 3 sumarizadores.

Para melhor distribuir a pesquisa, os julgadores nascidos nos dias 01 a 10 receberam um pdf com os 16 primeiros processos sumarizados por Edmundson, os 16 seguintes por TextRank e os 16 finais por Lsa. Para os nascidos entre os dias 11

e 20, a ordem foi Lsa, Edmundson e TextRank. Para os demais, a ordem foi TextRank, Lsa e Edmundson.

O arquivo continha o resumo do voto e o voto completo, sendo que, no voto completo, as frases que não foram usadas no resumo estavam em vermelho. Confira a seguir.

Processo: 10380902416200856

Resumo do voto

o que se julga é questão preliminar relacionada à extinção do direito de utilização do crédito informado em dcomp, sobre a matéria, o carf já se pronunciou definitivamente através da súmula carf 91: súmula carf 91 ao pedido de restituição pleiteado administrativamente antes de 9 de junho de 2005, no caso de tributo sujeito a lançamento por homologação, aplica-se o prazo prescricional de 10 (dez) anos, contado do fato gerador.

o fato gerador deu-se em 31/12/1998, possibilitando apresentação de dcomp até o final do ano de 2008, sendo a dcomp de 2004, não prevalece a afirmação do despacho decisório (folha 07) de que, na data de transmissão da dcomp, já estava extinto o direito de utilização do saldo negativo.

a fim de evitar supressão de instância no julgamento da lide, entendo que o processo deve retornar à origem para análise da existência, suficiência e disponibilidade do crédito pela drf que originalmente proferiu despacho decisório.

diante do exposto, voto por dar provimento parcial ao recurso, concluindo que não havia ocorrido a prescrição do direito de utilização do crédito, determinando o retorno dos autos à autoridade preparadora competente, para apreciação do mérito da compensação efetuada.

Processo: 10380902416200856

Voto completo

o recurso apresentado atende aos requisitos de admissibilidade previstos no decreto 70235 de 1972 e decreto 7574 de 2011, que regulam o processo administrativo-fiscal dele conhecido.

o que se julga é questão preliminar relacionada à extinção do direito de utilização do crédito informado em dcomp, sobre a matéria, o carf já se pronunciou definitivamente através da súmula carf 91: súmula carf 91 ao pedido de restituição pleiteado administrativamente antes de 9 de junho de 2005, no caso de tributo sujeito a lançamento por homologação, aplica-se o prazo prescricional de 10 (dez) anos, contado do fato gerador.

(vinculante, conforme portaria mf 277, de 07/06/2018, dou de 08/06/2018).

a regra, então, aplica-se a pleito administrativo anterior a 09/06/2005.

o prazo prescricional é de dez anos contados do fato gerador.

no caso concreto, a dcomp (folhas 02 a 06) foi transmitida em 14/05/2004.

aplica-se, portanto, o prazo prescricional de dez anos.

o fato gerador deu-se em 31/12/1998, possibilitando apresentação de dcomp até o final do ano de 2008, sendo a dcomp de 2004, não prevalece a afirmação do despacho decisório (folha 07) de que, na data de transmissão da dcomp, já estava extinto o direito de utilização do saldo negativo.

assim, restou prejudicada a análise do mérito da compensação desde sua origem, já que foi equivocadamente interrompida por questão preliminar.

a fim de evitar supressão de instância no julgamento da lide, entendo que o processo deve retornar à origem para análise da existência, suficiência e disponibilidade do crédito pela drf que originalmente proferiu despacho decisório.

diante do exposto, voto por dar provimento parcial ao recurso, concluindo que não havia ocorrido a prescrição do direito de utilização do crédito, determinando o retorno dos autos à autoridade preparadora competente, para apreciação do mérito da compensação efetuada.

A avaliação proposta foi a seguinte:

1. Responder qual é o assunto do processo dentre os 4 possíveis: IRPJ, IRPF, NGDT e PAF.
2. Responder o resultado do julgamento (improcedente ou procedente/procedente em parte).
3. Dar uma nota para a compreensão do texto. Se considera que conseguiu compreender o voto resumido (lide, argumentos, decisão). Nota de 1 a 10.

4. Dar uma nota para a coesão do texto, para o encadeamento das frases. Como o resumo exclui frases, se aparentemente as frases mais importantes foram mantidas. Se não parece haver uma lacuna entre as frases. Nota de 1 a 10.

Com o objetivo de não onerar os colaboradores e incentivar a participação, selecionei processos com número de frases entre 11 e 12. Foram selecionados 12 processos de cada uma das classes de classificação.

Esse procedimento foi realizado no notebook 08.

Por fim, considerando a participação voluntária, os julgadores foram orientados a contribuir com a quantidade de avaliações possíveis. No caso de não avaliarem todos os resumos, que o fizessem em ordem aleatória para permitir a avaliação de diferentes modelos.

6. Apresentação dos Resultados

6.1 Modelo de Canvas por Vasandani

Classificação e sumarização de acórdãos de julgamento do Carf utilizando algoritmos de processamento de linguagem natural		
Definição do problema	Resultados e previsões	Aquisição de dados
Avaliar a qualidade de sumarizadores para a utilização no julgamento de processos administrativos com o objetivo de facilitar e acelerar o julgamento como enfrentamento do estoque crescente e diante da insuficiência de pessoas.	<p>Pretende-se classificar um texto com conteúdo jurídico tributário por assunto. O preditor são as palavras contidas em um voto do Carf e a variável alvo é o assunto.</p> <p>Pretende-se classificar resumos de textos com conteúdo jurídico tributário por assunto. O preditor são as palavras contidas nos resumos e a variável alvo é o assunto.</p>	Os dados são públicos e estão disponíveis no sítio do Carf. Há necessidade de construir Web Scraping para a raspagem dos dados necessários ao trabalho.
Modelagem	Avaliação do modelo	Preparação de dados
<p>Para a fase de classificação, os modelos apropriados para o trabalho são os que permitem a classificação utilizando processamento de linguagem natural. Serão testados modelos da biblioteca SpaCy e uma rede neural convolucional com TensorFlow.</p> <p>Para a fase de sumarização, principal foco do trabalho, é necessária a aplicação de técnicas de sumarização de textos.</p>	<p>Os modelos serão avaliados pela quantidade de classificações corretas, ou seja, pela acurácia.</p> <p>O melhor modelo será aplicado aos textos sumarizados.</p> <p>O melhor sumariador será aquele que permitir ao modelo a melhor acurácia.</p>	<p>A preparação dos dados consiste, sinteticamente, em transformar os votos em um conjunto de palavras. Nesse processo, pontuações, palavras sem conteúdo, números, etc, precisam ser excluídos.</p> <p>Outra preparação dos dados consiste na transformação das palavras em vetores numéricos para a rede neural convolucional.</p>

6.2 Modelos SpaCy

Após o treinamento, os modelos foram aplicados aos dados de teste e os resultados obtidos mostrados em gráficos de matriz de confusão com mapa de calor.

Lematização

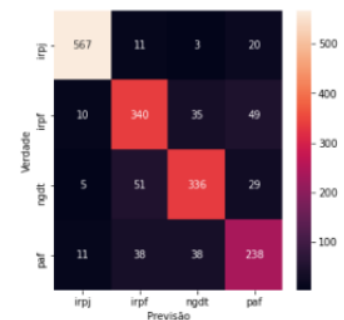
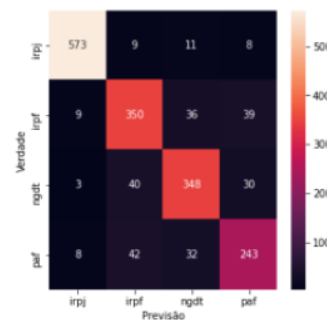
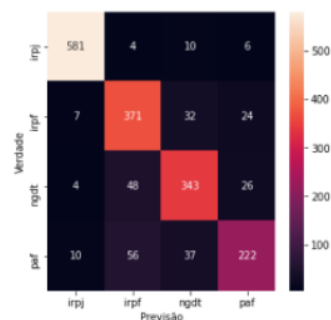
Stemização

Palavras originais

Acurácia 85,17%

Acurácia 85,00%

Acurácia 83,15%



Em relação à acurácia, o modelo que usou as palavras lematizadas apresentou o melhor resultado com 85,17% de acertos. O segundo melhor modelo foi o que usou palavras stemizadas. Esse modelo teve índice de acerto de 85,00%, ou seja, apresentou uma acurácia muito próxima ao modelo lematizado.

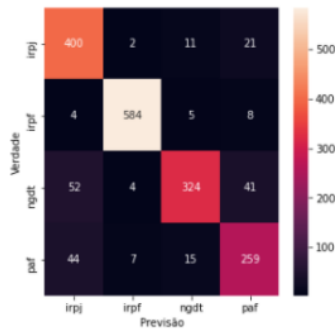
Já o modelo que não fez nenhuma transformação nas palavras usando-as em seu formato original apresentou a menor acurácia com 2% de perda em relação aos demais modelos.

6.3 Modelos Rede Neural Convolucional Profunda TensorFlow

Após o treinamento, os modelos foram aplicados aos dados de teste e os resultados obtidos mostrados em gráficos de matriz de confusão com mapa de calor.

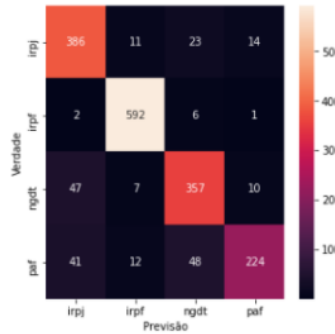
Lematização

Acurácia 87,98%



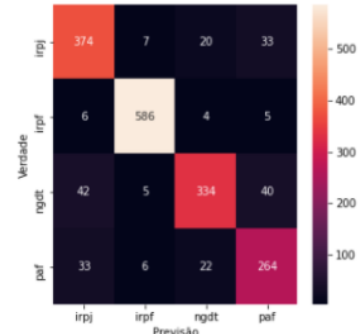
Stemização

Acurácia 87,53%



Palavras originais

Acurácia 87,47%



Percebe-se que a classificação por acurácia manteve a mesma ordem da SpaCy com palavras lematizadas, stemizadas e sem tratamento. Entretanto, a diferença entre os modelos foi pequena não apresentando a mesma queda no caso de palavras sem tratamento.

6.4 Modelo de melhor acurácia aplicado aos sumarizadores

A acurácia dos modelos Spacy situou-se na casa de 85% e os da DCNN, na casa de 87% de acurácia.

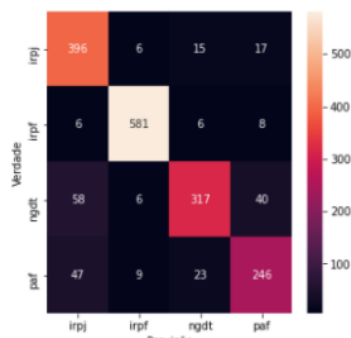
Observa-se em ambos os casos que, embora não destacadamente, os modelos trabalhados com o corpus de palavras lematizadas obtiveram os melhores resultados e os piores resultados foram para classificações sem tratamento das palavras.

O melhor resultado foi obtido pela DCNN com palavras lematizadas.

Portanto, este foi o modelo selecionado para a classificação dos sumarizadores. Os resultados foram os seguintes:

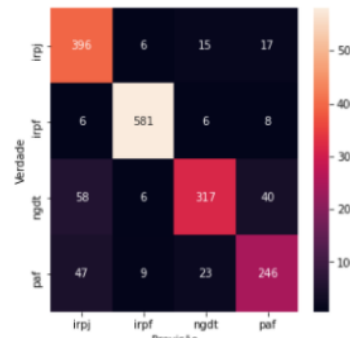
Luhn - Sumy

Acurácia 86,46%



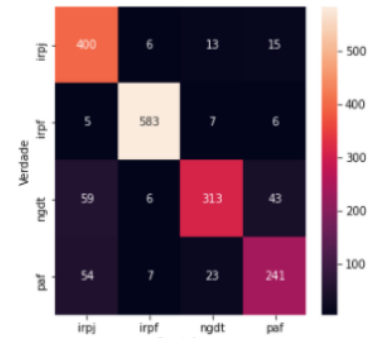
Lsa

Acurácia 86,46%



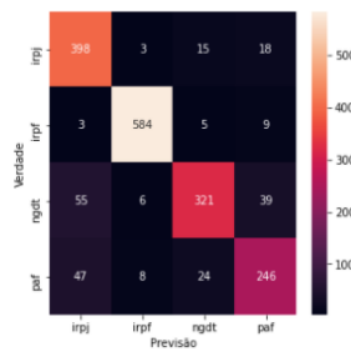
TextRank

Acurácia 86,29%



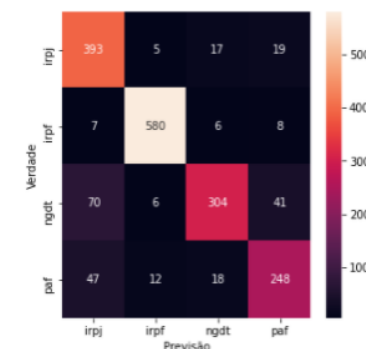
Edmundson

Acurácia 86,97%



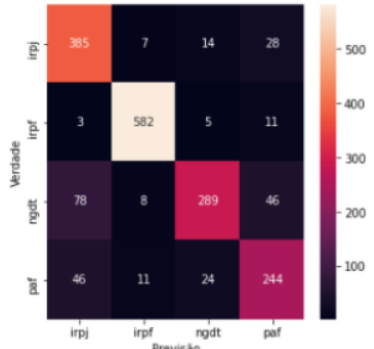
LexRank

Acurácia 85,62%



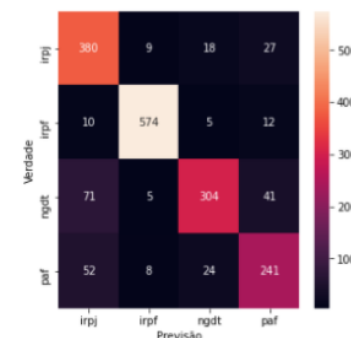
SumBasic

Acurácia 84,22%



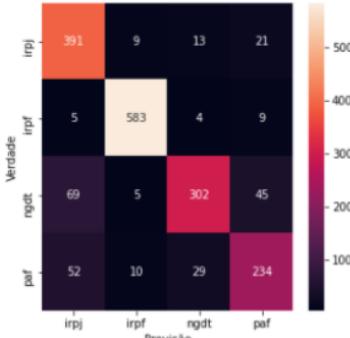
Random

Acurácia 84,16%



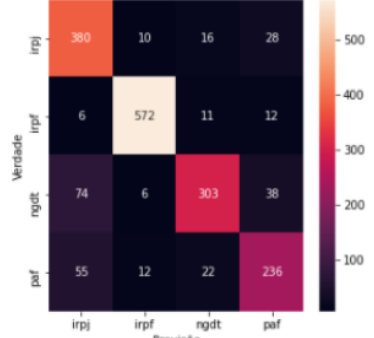
Similaridade Cosseno

Acurácia 84,78%

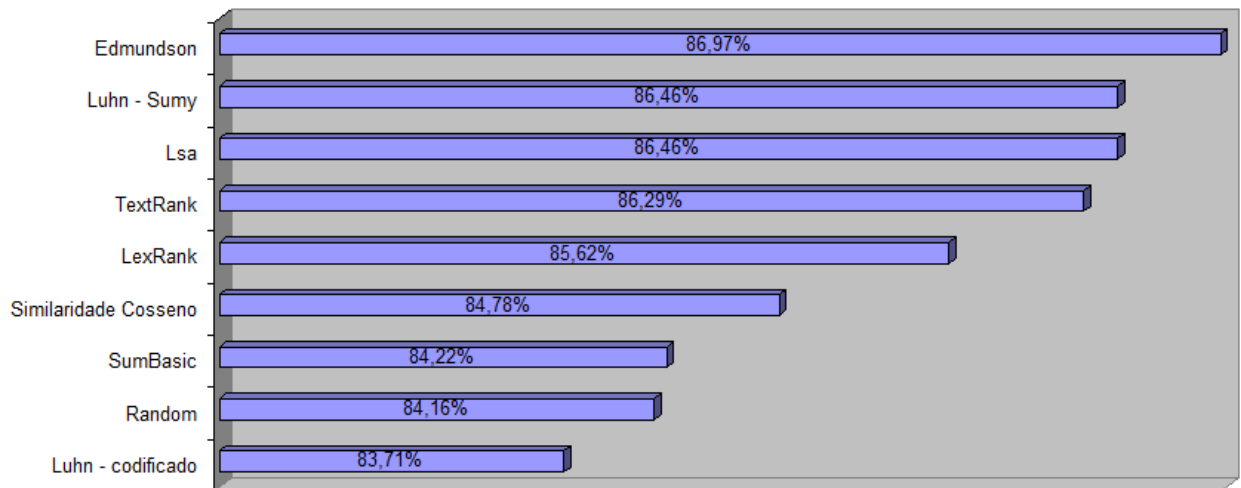


Luhn - codificado

Acurácia 83,71%



A seguir, a relação dos sumarizadores classificados pela acurácia.



Como se pode perceber, houve uma queda na acurácia comparando-se à classificação com o texto completo. Entretanto, considerando que os resumos foram elaborados com 40% do texto, a perda de acurácia foi pequena.

Se a premissa do trabalho está correta, a qualidade dos sumarizadores é satisfatória, pois conseguiram manter as palavras relevantes para a classificação.

6.5 Avaliação de sumarizadores por humanos

A pesquisa foi realizada por meio de formulário na plataforma colaborativa Microsoft Teams da Receita Federal e ficou disponível para os julgadores no período de 23/03/2021 até 17/04/2021 quando foi feita a apuração das respostas.

Pesquisa qualidade dos resumos automáticos.

Pessoal, o número do processo está sendo alterado pelo Teams. Há relatos que está sendo incrementado em 1. Não se preocupem. Eu tenho como tratar isso na apuração. Então, copiem e coletem o número do processo que não haverá problema. Obrigado a todos que estão participando.

Ola Sandro, quando enviar este formulário, o seu nome e endereço de email serão exibidos para o proprietário do formulário.

* Obrigatória

1. Primeiro Grupo *

- ☐ Primeiro grupo
- ☐ Segundo grupo
- ☐ Terceiro grupo

2. Informar o número do processo analisado. *

O valor deve ser um número

3. Qual é o assunto do processo? *

- ☐ IRPJ/CSLL
- ☐ IRPF - Imposto de Renda Pessoa Física
- ☐ NGDT - Normas Gerais de Direito Tributário
- ☐ PAF - Processo Administrativo Fiscal

4. Qual é o resultado do recurso do contribuinte ao Carf? *

- ☐ Procedente ou procedente em parte
- ☐ Improcedente
- ☐ Não sei

5. Dar uma nota para a sua compreensão do texto. Se você considera que conseguiu compreender o voto resumido (lide, argumentos, decisão). *

- ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10

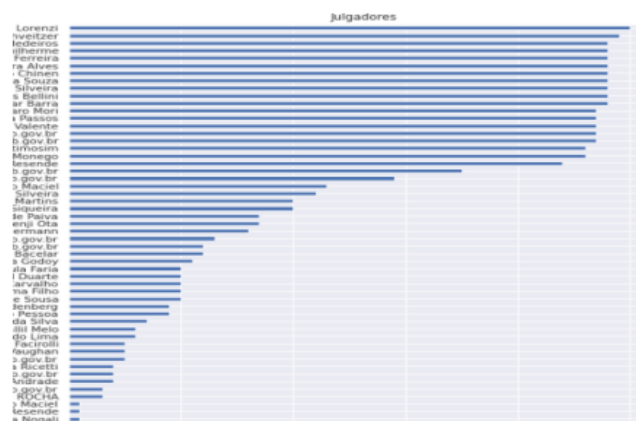
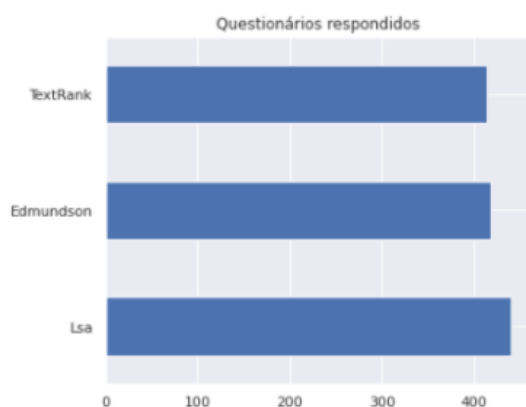
6. Dar uma nota para a coesão do texto, para o encadeamento das frases. Como o resumo exclui frases, se aparentemente as frases mais importantes foram mantidas. Se não parece haver uma lacuna entre as frases. *

- ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10

Enviar

Os julgadores deveriam responder qual era o assunto e o resultado do recurso, dar nota de compreensão e de coesão, para, no conceito de Machado (2004) *supra*, indicar a qualidade do resumo sob a ótica dos julgadores.

Foram respondidos 1.272 questionários por 53 julgadores das Delegacias de Julgamento da Receita Federal, conforme mostram os gráficos a seguir:



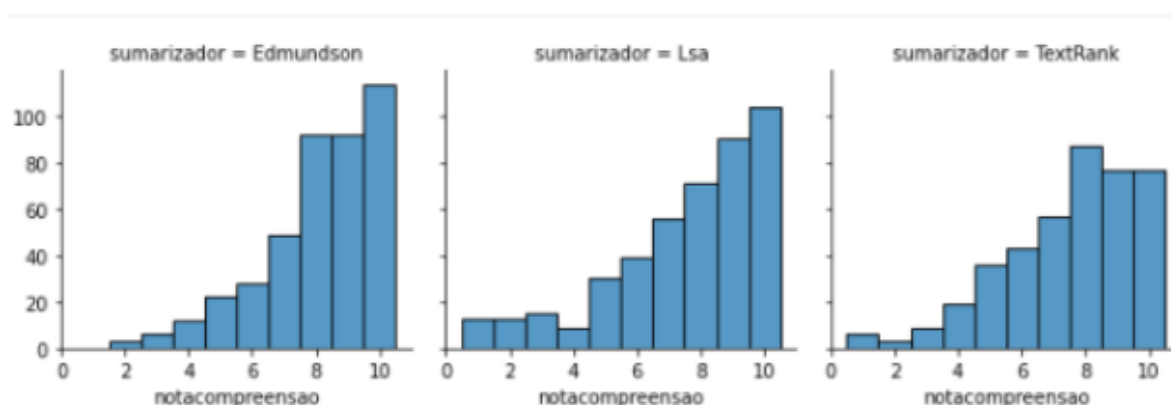
O número de questionários respondidos por tipo de sumariizador ficou bastante equilibrado.

Os julgadores responderam os questionários de acordo com a disponibilidade, não havendo a obrigatoriedade de responder todos, por isso a diferença quantitativa entre eles.

A geração dos pdf com os resumos para serem trabalhados na pesquisa e a apuração do resultado foram executadas no notebook 08.

A seguir, gráficos relativos às notas de compreensão e coesão.

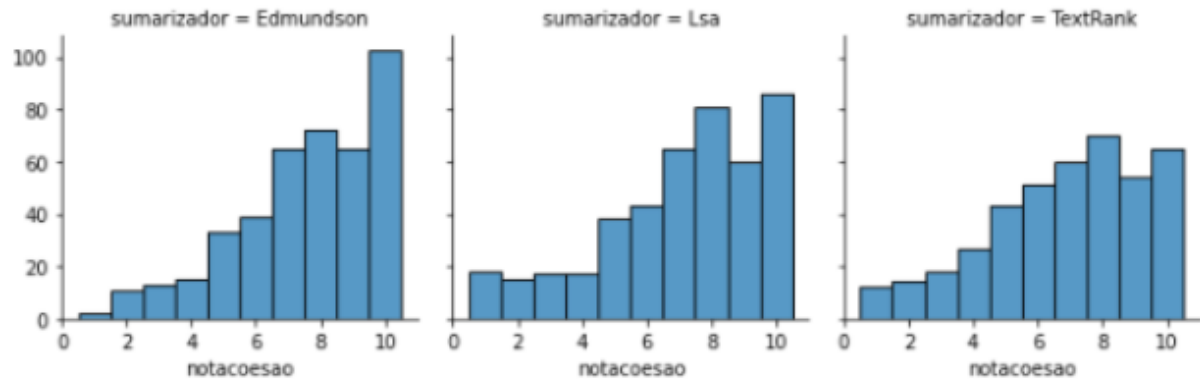
Notas de compreensão por sumariizador - histograma



O resultado da avaliação pelos julgadores da compreensão dos textos ficou coerente com a avaliação automática. Aqui, também, a ordem foi Edmundson, Lsa e TextRank.

A visualização dos gráficos mostra que embora a nota média tenha sido alta, existem algumas notas muito baixas indicando que o sumariizador não conseguiu entregar um resumo satisfatório para aquele processo. Entretanto, a maioria dos resumos obteve notas satisfatórias.

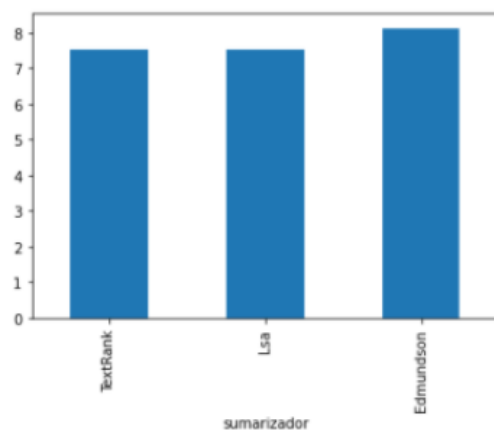
Notas de coesão por sumarizador - histograma



As notas de coesão mantêm a distribuição majoritariamente positiva, mas há uma queda previsível. É que os sumarizadores são do tipo extrativo e é natural que frases de conexão com a utilização de preposições, conjunções e advérbios, palavras vazias de conteúdo na lógica do processamento de linguagem natural sejam as mais fortes candidatas à exclusão. Tal situação, embora não prejudique a compreensão do texto, prejudica a concatenação das sentenças.

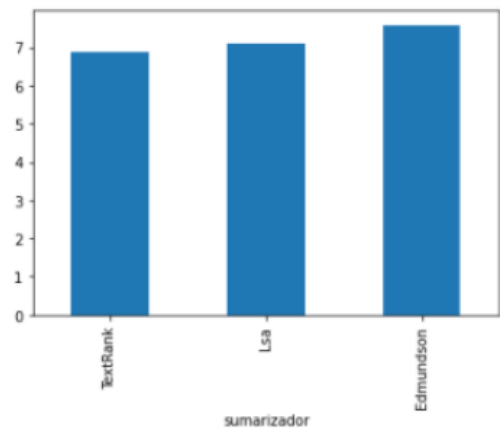
Nota de compreensão:

Edmundson 8,12, Lsa 7,53 e TextRank 7,51



Nota de coesão:

Edmundson 7,57, Lsa 7,08 e TextRank 6,85

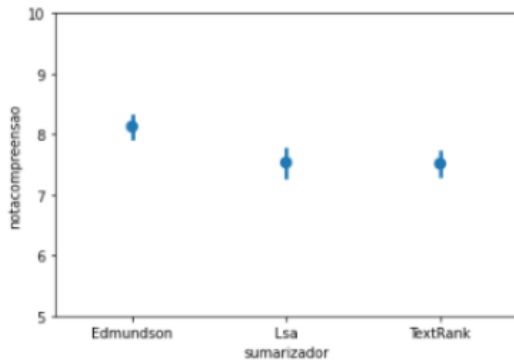


As notas médias de compreensão e de coesão indicam que os sumarizadores geraram resumos com bom nível e que podem ser úteis na atividade de julgamento.

A seguir, os gráficos de pontos mostram estimativas da tendência central para as notas de compreensão e coesão pela posição dos pontos do gráfico de dispersão e fornecem indicação da incerteza em torno dessa estimativa usando barras de erro.

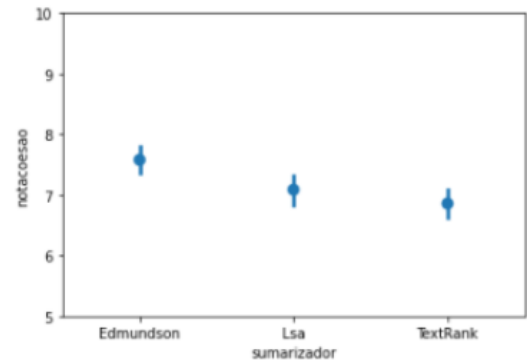
Nota de compreensão:

Edmundson 8,12, Lsa 7,53 e TextRank 7,51

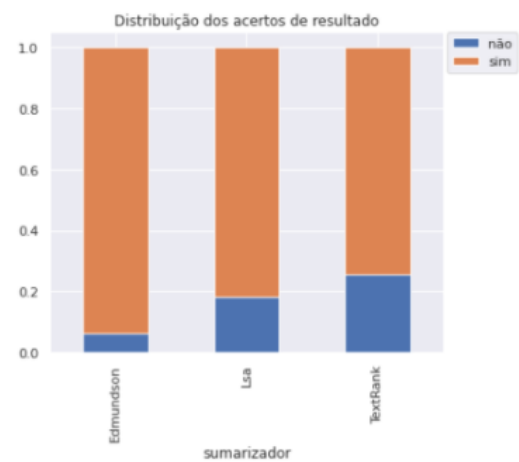
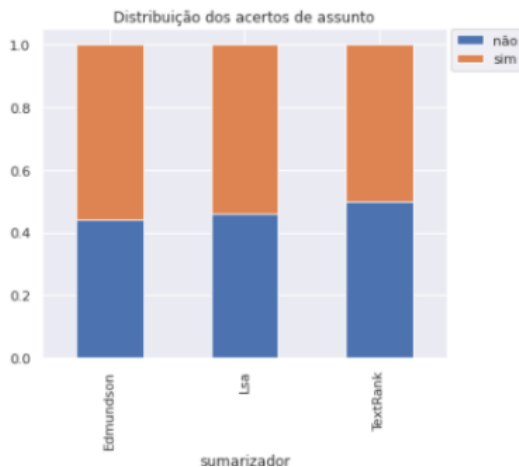


Nota de coesão:

Edmundson 7,57, Lsa 7,08 e TextRank 6,85



A seguir, o resultado da classificação pelos julgadores dos resumos por assunto e por resultado de julgamento.



Percebe-se que o índice de acerto na classificação dos textos por assunto caiu muito na avaliação por humanos. A classificação feita pela rede neural convolucional acertou 86,97% com Edmundson, 86,46% com Lsa e 86,29% com TextRank. Já os julgadores acertaram 55,98% com Edmundson, 54,09% com Lsa e 50,00% com TextRank.

Acertos na classificação por assunto

Edmundson

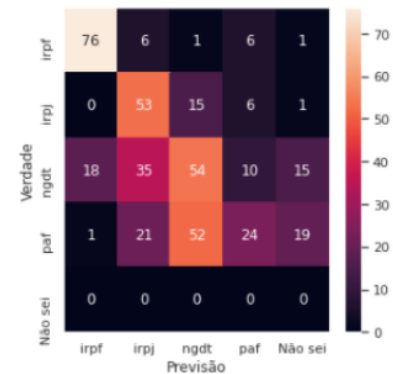
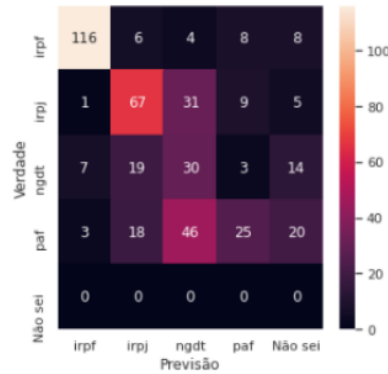
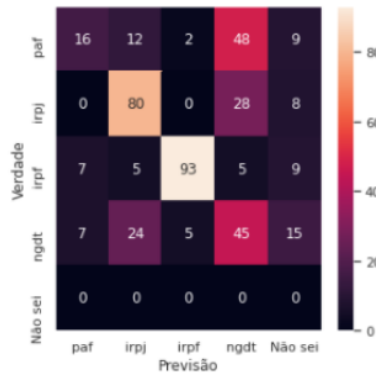
Lsa

TextRank

Acurácia 55,98%

Acurácia 54,09%

Acurácia 50,00%



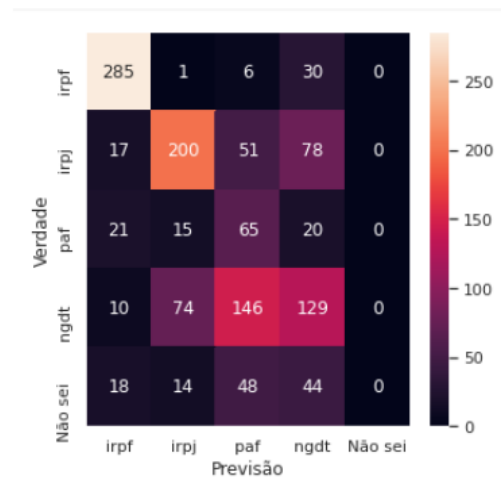
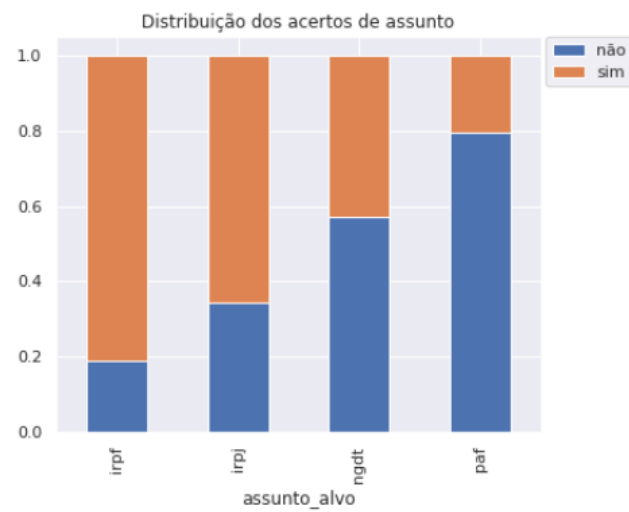
Percentual de acertos de assunto para os julgadores que leram o resumo Edmundson: em 418 tentativas, acerto de 234, ou 55.98%
 Percentual de acertos de assunto para os julgadores que leram o resumo Lsa: em 440 tentativas, acerto de 238, ou 54.09%
 Percentual de acertos de assunto para os julgadores que leram o resumo TextRank: em 414 tentativas, acerto de 207, ou 50.00%

Esses resultados foram surpreendentes, em princípio, porque a nota de compreensão foi alta e, ao mesmo tempo, o percentual de acerto em assunto foi baixo o que é contraditório. Como não sabe o assunto se compreendeu o conteúdo?

O grau de especialização dos julgadores pode ser a resposta. Há julgadores que julgam apenas processos de Pessoa Física, outros de IRPJ/CSLL, e outros que julgam matérias que sequer estavam presentes neste trabalho como PIS/Cofins, IPI, Comércio Exterior. Os processos de Normas Gerais de Direito Tributário e Processo Administrativo Fiscal permeiam todo tipo de processo.

Nesta lógica, seria justificável que um modelo treinado em todos os tipos de matérias tivesse um desempenho melhor que um modelo especialista. É possível que os julgadores tenham melhores resultados se testados nas suas especialidades como um modelo que se adaptou à base de treinamento. Isso é um bom exemplo de “*overfitting* em humanos”.

Verificando a incidência dos erros em assunto, constata-se que o maior índice deu-se na inversão dos assuntos NGDT e Paf. Nesse caso, o que poderia explicar é a similaridade conceitual entre os assuntos e a presença simultânea em um mesmo processo. Os gráficos seguintes, mostram a distribuição dos erros por assunto.



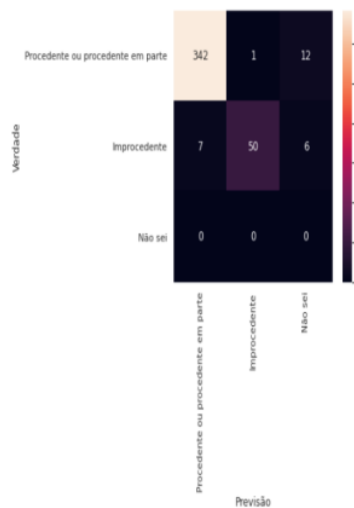
Quando se trata de indicar o resultado de julgamento, temos uma pergunta neutra em relação à especialização. O desconhecimento da matéria não prejudica a percepção sobre o resultado.

Neste quesito, os resultados foram muito bons indicando que os sumarizadores mantiveram as frases responsáveis por este importante conteúdo para a compreensão global deste tipo de texto.

Acertos na classificação por resultado de julgamento

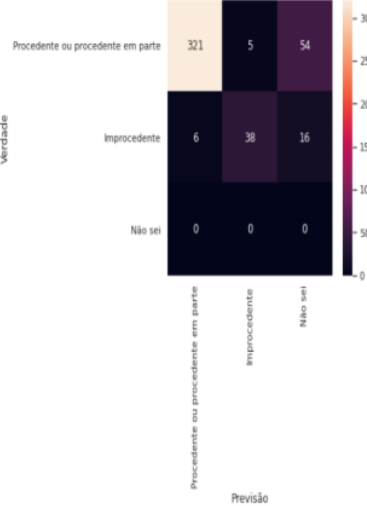
Edmundson

Acurácia 93,78%



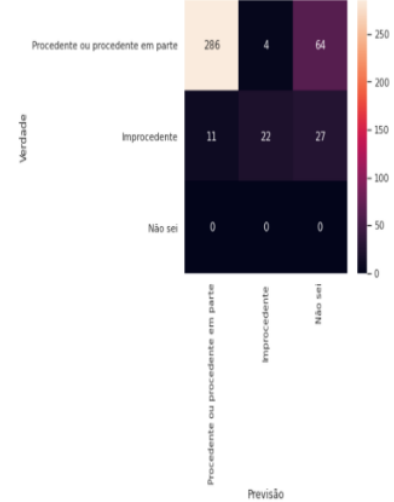
Lsa

Acurácia 81,59%



TextRank

Acurácia 74,40%



Percentual de acertos de resultado para os julgadores que leram o resumo Edmundson: em 418 tentativas, acerto de 392, ou 93.78%
 Percentual de acertos de resultado para os julgadores que leram o resumo Lsa: em 440 tentativas, acerto de 359, ou 81.59%
 Percentual de acertos de resultado para os julgadores que leram o resumo TextRank: em 414 tentativas, acerto de 308, ou 74.40%

Conclusão

O objetivo do trabalho era avaliar a qualidade de sumarizadores com base na acurácia de classificação utilizando um modelo de Machine Learning.

Para isso, foram comparados os modelos de classificação da biblioteca SpaCy com uma Rede Neural Convolutacional – DCNN utilizando o TensorFlow.

Em ambos os casos, os modelos foram treinados em um corpus de palavras originais, lematizadas e stemizadas.

O modelo que apresentou o melhor resultado foi a Rede Neural Convolutacional utilizando palavras lematizadas.

Foram testados sete sumarizadores da biblioteca Sumy e codificados outros dois sumarizadores. Os textos sumarizados foram utilizados para a classificação pelo modelo DCNN e os sumarizadores com melhor desempenho foram considerados mais qualificados.

Para validar a conclusão do trabalho, foi feita uma pesquisa com julgadores avaliando a qualidade de três dos sumarizadores testados.

O resultado da pesquisa confirmou a avaliação automática mantendo a mesma ordem de classificação dos sumarizadores. As notas de compreensão e coesão dadas pelos julgadores corroboram a possibilidade de aproveitamento deste trabalho para a elaboração de ferramenta para geração de relatórios resumos na atividade de julgamento dos processos administrativos na Receita Federal.

Com base nos resultados, é viável propor a implementação da ferramenta, porém com precaução. Os sumarizadores não estão suficientemente maduros para substituir o trabalho humano.

A nota de coesão menor que a nota de compreensão demonstra a necessidade de tratamento dos resumos pelos julgadores. Apesar do resumo ter sido muito bem avaliado no quesito compreensão do texto, a coesão não apresentou o mesmo nível de resultado.

Nesse contexto, uma ferramenta que proponha o resumo para ser validado e ajustado pelo usuário parece ser a melhor alternativa. Pequenos retoques pelos julgadores podem ser suficientes para assegurar um texto de alta qualidade.

Ainda assim, o realce das frases mais importantes pelo sumariador é uma excelente orientação para o julgador e proporciona possibilidade de efetivo ganho na execução da atividade.

Dessa forma, o julgamento de processos aproveita a tecnologia do processamento de linguagem natural, mas a pondera com a experiência do usuário.

7. Links

Link para o vídeo: https://youtu.be/X0k3JwW_kMY

Link para o repositório: https://github.com/sla01/TCC_PUC_Minas

REFERÊNCIAS

MACHADO, Anna Rachei Machado, Eliane Gouvêa Lousada, Lília Santos Abreu-Tardelli. **Resumo**. São Paulo:Parábola Editorial, 2004.

CATAE, Fabrício Shiguero. **Classificação Automática de Texto Por Meio De Smilaridade De Palavras**. São Paulo: 2012.

PARDO, Thiago Alexandre Salgueiro. **Sumarização Automática: Principais Conceitos e Sistemas para o Português Brasileiro**. São Carlos, 2008

JONES, K. SPARCK **Discourse Modelling for Automatic Summarisation. Tech. Report No. 290**. University of Cambridge. UK, February. 1993a

CABRAL, Luciano de Souza. **Uma Plataforma para Sumarização Automática de Textos Independente de Idioma**. Recife: 2015.

GRANATYR, Jones. **Sumarização de Textos com Processamento de Linguagem Natural**. São Paulo: IA Expert Academy, 2021.

GRANATYR, Jones. **Processamento de Linguagem Natural com Deep Learning**. São Paulo: IA Expert Academy, 2021.

GRANATYR, Jones. **Processamento de Linguagem Natural com spaCy e Python**. São Paulo: IA Expert Academy, 2021.

APÊNDICE

Estrutura do Github

arquivos

- TCC Sandro Luiz de Aguiar.pdf => texto do TCC. Este documento.

pastas

notebooks

01) TCC_p01_preparar_X_treinamento_e_y_teste.ipynb => o objetivo deste notebook é preparar e salvar os dataframes X e y. O X será usado para treinar os diferentes modelos (SpaCy e DCNN com stemer, lema e palavras originais) e o y será utilizado para apuração da acurácia dos modelos de classificação. Posteriormente, o mesmo y será utilizado por todos os sumarizadores para a geração dos resumos e, em seguida, para apuração da acurácia do modelo escolhido em relação aos resumos. Como a base para as experiências precisa ser a mesma, após a preparação dos dados e a divisão em X e y, os dataframes serão salvos no drive. Assim, esse procedimento será executado apenas uma vez. Ver item dataframes gerados nos notebooks.

02) TCC_p02_classificacao_SpaCy.ipynb => O objetivo deste notebook é treinar os modelos utilizando a biblioteca SpaCy. Para isso, lê do drive os dataframe X_treinamento preprocessados e gerados pelo notebook 01 nos três tipos de formato (lema, stemer e original) e também o Y_treinamento. Faz o treinamento dos modelos `mod_TCC_spacy_lema`, `mod_TCC_spacy_stemer` e `mod_TCC_spacy_original` e grava-os no drive para eventual utilização posterior pelo notebook 07. Faz o teste dos modelos e apura a acurácia para a seleção do melhor modelo.

03) TCC_p03_Classificação_DCNN.ipynb => O objetivo deste notebook é treinar os modelos utilizando a biblioteca SpaCy. Para isso, lê do drive os dataframe X_treinamento preprocessados e gerados pelo notebook 01 nos três tipos de formato (lema, stemer e original) e também o Y_treinamento. Faz o treinamento dos

modelos `modelo_Dcnn_lemma`, `modelo_Dcnn_stemer` e `modelo_Dcnn_original` e grava-os no drive para eventual utilização posterior pelo notebook 07. Faz o teste dos modelos e apura a acurácia para a seleção do melhor modelo.

04) `TCC_p04_Prepara_Palavras_Bonus_e_Nulas_para_Edmundson.ipynb` => o objetivo deste notebook é criar palavras bônus e palavras nulas para o sumariador Edmundson. As palavras bônus serão criadas a partir das ementas e das últimas duas frases do voto (normalmente, as conclusões). Já as palavras nulas serão formadas pelas stop words da biblioteca SpaCy, da biblioteca NLTK e pelos nomes dos relatores da base de acórdãos. Grava os arquivos gerados para utilização pelo notebook 07. Ver item dataframes gerados nos notebooks.

05) `TCC_p05_Sumarização_Luhn_DCNN.ipynb` => executa a sumarização utilizando a codificação do algoritmo de Luhn. Em seguida, executa o melhor modelo (`modelo_Dcnn_lemma`, ver item 6.4 Modelo de melhor acurácia aplicado aos sumariadores) para o resumo e faz a apuração e visualização dos resultados.

06) `TCC_p06_Sumarização_Cosseno_DCNN.ipynb` => executa a sumarização utilizando a codificação da similaridade do cosseno. Em seguida, executa o melhor modelo (`modelo_Dcnn_lemma`, ver item 6.4 Modelo de melhor acurácia aplicado aos sumariadores) para o resumo e faz a apuração e visualização dos resultados.

07) `TCC_p07_Sumarização_Sumy_DCNN.ipynb` => executa a sumarização utilizando a biblioteca Sumy. Em seguida, executa o melhor modelo (`modelo_Dcnn_lemma`, ver item 6.4 Modelo de melhor acurácia aplicado aos sumariadores) para os resumos e faz a apuração e visualização dos resultados.

08) `TCC_p08_Gerar_Pdf_Sumarizado_e_Apuração_Pesquisa.ipynb` => o objetivo deste notebook é gerar resumos dos sumariadores Edmundson, Lsa e TextRank em um pdf para que julgadores respondam a uma pesquisa de qualidade dos resumos. O resultado da pesquisa será comparado com o resultado da avaliação automática com base na acurácia de classificação dos resumos por assunto. Os sumariadores foram escolhidos em função da classificação automática. Serão resumidos 48 processos por sumariador. Os processos são os mesmos para todos os sumariadores. Gera os pdf Primeiro grupo - 01E02T03L.pdf, Segundo grupo - 01L02E03T.pdf e Terceiro grupo - 01T02L03E.pdf com os resumos para a pesquisa.

A apuração e visualização dos resultados da pesquisa também é feita neste notebook que recebe o arquivo Pesquisa qualidade dos resumos automáticos.csv com os dados da pesquisa.

dataframes web scraping

- df_ementas.csv => arquivo com as ementas.
- df_atributos.csv => arquivo com os atributos do processo.
- df_voto.csv => arquivo com o conteúdo integral do voto.

dataframes gerados nos notebooks

- X_treinamento_7124.csv => arquivo com 80% dos dados para treinamento dos modelos de machine learning. O arquivo X contém a variável preditora voto. 7124 é o total de registros do arquivo. Gerado no notebook 01.
- y_treinamento_7124.csv => o arquivo y contém a variável alvo assunto relativa aos dados de treinamento. Gerado no notebook 01.
- X_teste_1781.csv => arquivo com 20% dos dados para teste dos modelos de machine learning. O arquivo X contém a variável preditora voto. 1781 é o total de registros do arquivo. Gerado no notebook 01.
- y_teste_1781.csv => o arquivo y contém a variável alvo assunto relativa aos dados de teste. Gerado no notebook 01.
- X_treinamento_lemma.csv => versão do X_treinamento preprocessado com as palavras do voto lematizadas. Gerado no notebook 01.
- X_treinamento_stemmer.csv => versão do X_treinamento preprocessado com as palavras do voto stemizadas. Gerado no notebook 01.
- X_treinamento_original.csv => versão do X_treinamento preprocessado com as palavras do voto no formato original. Gerado no notebook 01.

- `X_teste gerar_pdf_sumarizado_pesquisa_1781.csv` => versão do `X_teste` acrescida dos atributos processo, qtdfrase e assunto. Essa versão foi criada para facilitar a geração dos pdf com os resumos para a pesquisa, pois na geração houve a seleção pelo número de frases, pelo assunto e o número do processo foi mostrado no relatório. A versão original de `X_teste` só tinha o atributo voto. Gerado no notebook 01.
- `df_palavras_ementa.csv` => arquivo com as palavras mais freqüentes das ementas gerado no notebook 04 para formar as palavras bônus do sumarizador Edmundson. Gerado no notebook 04.
- `df_palavras_ultima_frase.csv` => arquivo com as palavras mais freqüentes das duas últimas frases dos votos gerado no notebook 04 para formar as palavras bônus do sumarizador Edmundson. Gerado no notebook 04.
- `df_palavras_relatores.csv` => arquivo com os nomes dos relatores gerado no notebook 04 para formar as palavras nulas do sumarizador Edmundson. Gerado no notebook 04.
- `df_palavras_stop_words.csv` => arquivo com as palavras stop words da biblioteca SpaCy e da biblioteca NLTK gerado no notebook 04 para formar as palavras nulas do sumarizador Edmundson. Gerado no notebook 04.

modelos

- `mod_TCC_spacy_lemma` => modelo da biblioteca SpaCy treinado com as palavras lematizadas no notebook 02.
- `mod_TCC_spacy_stemmer` modelo da biblioteca SpaCy treinado com as palavras stemizadas no notebook 02.
- `mod_TCC_spacy_original` modelo da biblioteca SpaCy treinado com as palavras originais no notebook 02.
- `modelo_Dcnn_lemma` => modelo DCNN treinado com as palavras lematizadas no notebook 03.
- `modelo_Dcnn_stemmer` => modelo DCNN treinado com as palavras stemizadas no notebook 03.

- modelo_Dcnn_original => modelo DCNN treinado com as palavras originais no notebook 03.

pesquisa

- Pesquisa qualidade dos resumos automáticos.csv => resultado da pesquisa.
- Primeiro grupo - 01E02T03L.pdf => pdf com os 16 primeiros processos sumarizados por Edmundson, os 16 seguintes por TextRank e os 16 finais por Lsa para os julgadores do primeiro grupo.
- Segundo grupo - 01L02E03T.pdf => mutatis mutandis, segundo grupo com a ordem Lsa, Edmundson e TextRank.
- Terceiro grupo - 01T02L03E.pdf => mutatis mutandis, segundo grupo com a ordem TextRank, Lsa e Edmundson.

scripts

- TCC Big Data - Baixar acórdãos Carf.py => script ContÁgil para web scraping do sítio do Carf e geração do dataframe Carf_Ementas.
- TCC Big Data - Ler pdf acordao e gravar dataframes com atributos e com voto em frases.py => script ContÁgil para geração dos dataframes Carf_Atributo e Carf_Voto.