Projeto Integrado de Aprendizado de Máquina
Pontifícia Universidade Católica de Minas Gerais
Professor: Geanderson Esteves dos Santos

Classificação textual multi-rótulo de normas jurídicas estaduais

Duílio Campos Sasdelli

1 Introdução

Classificação de documentos textuais consiste na atribuição de um ou mais rótulos (categorização) a documentos de acordo com o conteúdo textual dos mesmos. No âmbito do **Aprendizado de Máquina e do Processamento de Linguagem Natural** a área de Classificação de documentos consiste no conjunto de algoritmos, técnicas e modelos que visam realizar tal tarefa de forma automática, com mínima ou nenhuma intervenção humana. Quando há intervenção humana em prévia categorização de documentos, os quais serão utilizados para "aprender" a categorizar novos documentos pelos algoritmos, diz-se que é uma **técnica supervisionada**. Já quando não há qualquer intervenção humana e o modelo aprender a agrupar documentos por métricas de similaridade de palavras ou documentos, diz-se tratar de uma **técnica não-supervisionada**.

A atribuição de rótulos de forma automática pode ser subdividida em diferentes **níveis de granularidade** a depender do tamanho do texto a ser classificado. Por exemplo, pode-se categorizar o documento como um todo, parágrafo por parágrafo ou, até mesmo, sentença por sentença. Trata-se de uma tarefa que pode ser utilizada com diferentes propósitos, tais como a análise de sentimentos, detecção de spams, triagem de documentos ou identificação de gênero textual.

A área de classificação de documentos possui grande relevância no**ramo jurídico**, dadas as particularidades inerentes este ramo tais como a grande quantidade de informações, a variedade de temáticas (ou rótulos), a estrutura e tamanho particular de leis, normas e decisões judiciais. Ademais, trata-se de uma área de estudo ainda incipiente pois são poucos as

Para o presente trabalho, serão aplicados diferentes algoritmos de classificação multirótulo de documentos no âmbito jurídico. Mais especificamente, serão utilizados algoritmos e técnicas de classificação sobre o *corpus* de normas legislativas estaduais do Estado de Minas Gerais, as quais versam sobre diferentes temáticas. O objetivo é definir, de forma automática e para cada nova norma escrita, a quais categorias pertence sem a necessidade de análise manual. A principal motivação é facilitar tanto o trabalho de triagem de documentos quanto o de definição de comissões ou grupos interessados.

2 Referencial Teórico

A utilização de algoritmos de classificação textual de normas jurídicas não tem sido amplamente abordada na literatura. Especialmente no Brasil, a maioria dos trabalhos versa sobre classificação de jurisprudência e não leis [5]. Portanto foi necesário recorrer a artigos

que realizam clasificação de leis no exterior. Um dos artigos [3] pesquisados prevê arquiteturas com redes neurais complexas baseados em modelos de atenção BIGRU (Bidirectional Gated Recurrent Unit) e redes convolucionais (CNN). Tais modelos apresentaram bons resultados inclusive em casos de aprendizado *zero-shot, one-shot*, XMTC (*Extreme multi-label text classification*).

3 Conceitos preliminares

Nesta seção serão explanados conceitos referentes à classificação multirótulo e sobre o domínio do trabalho, qual seja legislação estadual.

3.1 Classificação multirótulo

Quando a classificação de documentos envolve a atribuição de um mais rótulos a determinado documento, diz-se tratar de um problema de **Classificação Multi-rótulo** [7]. Por envolverem a atribuição de mais de um rótulo, tais problemas envolvem, de modo geral, são mais desafiadores quando comparado ao de classificação binária ou multiclasse unitária. Alguns classificadores podem ser facilmente adaptados para essa tarefa, tais como Árvores de Decisão, Redes Neurais, kNN, etc. Outros devem ser adaptados usando diferentes técnicas, como Relevância Binária, Cadeira, Todos-contra-todos, etc. No presente trabalho optou-se por utilizar classificadores naturalmente multi-rótulo, além de um kNN adaptado.

3.2 Legislação estadual

A legislação estadual corresponde ao conjunto de normas jurídicas, como leis ordinárias, leis complementares, decretos, emendas à Constituição Estadual, a própria constituiçãoe estadual produzidas pela casa legislativa, no Brasil chamada Assembleia Legislativa. A competência legislativa estadual é tida como residual pois são poucos os campos de atuação dos legisladores estaduais. Dentre as temáticas que não se legisla nos Estados destacam-se o Direito Penal e o Direito Civil, de competência exclusivamente federal.

3.3 Corpus de normas da ALMG

Serão utilizados dados de normas estaduais no âmbito do Estado de Minas Gerais disponíveis no Portal de Dados Abertos da Assembléia Legislativa de Minas Gerais [10]. A base contém um total de 28.309 normas que datam de 1835 até 2020 abrangendo um total de 646 assuntos gerais e 36 temas. Cada norma pode ter um ou mais assuntos gerais e um ou mais temas. No entanto, uma norma pode não ter sido categorizada em um assunto geral ou norma específica.

4 Técnicas e algoritmos utilizados

Na presente seção explicar-se-á a sequência de técnicas e algoritmos utilizados ao longo deste trabalho. A arquitetura seguiu o disposto na imagem 1 seguintes etapas: pré-processamento, modelagem vetorial, aplicação de técnicas de classificação, análise e calibragem, aplicação de clustering de rótulos na melhor técnica.

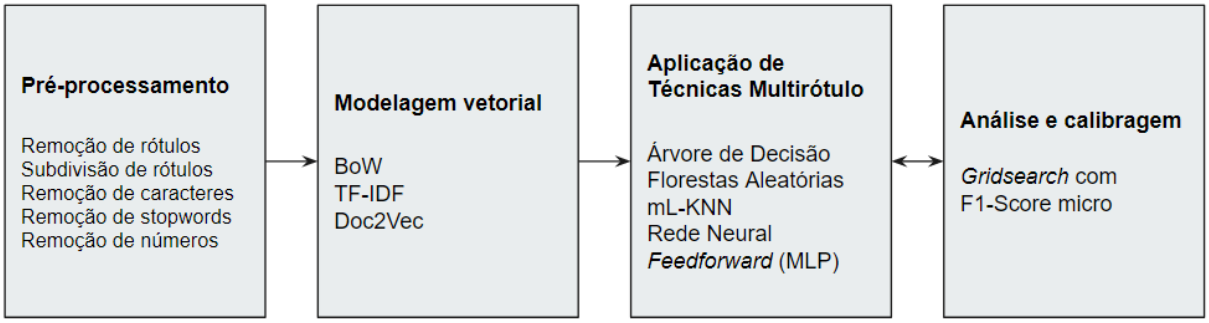

Figure 1: Arquitetura da modelagem proposta

4.1 Pré-processamento

O pré-processamento dos dados foi subdividos em duas etapas: a primeira envolveu a base de dados como um todo e a segunda apenas o conteúdo textual das normas. A primeira etapa consistiu na transformação e seleção de rótulos que seriam utilizados. Foram removidos assuntos muito frequentes, tais como os que lidam Utilidade Pública e Próprio Público. Além disso, foram padronizados os nomes de alguns assuntos parecidos, como "Tributos" e "Tributo", "Trabalho, Emprego e Renda" e "Trabalho Emprego Renda".

A segunda consistiu no pré-processamento do texto propriamente dito com transformação em caixa baixa, remoção de *stopwords*, caracteres especiais, números e numerais romanos. Para tanto foram utilizadas técnicas de Processamento de Linguagem Natural presentes nas bibliotecas NLTK [2] e RE (Expressão Regular).

4.2 Modelagem vetorial

Após o pré-processamento, cada documento será modelado por meio de vetores usando *Bag of Words (BoW)* TF-IDF [11], e Doc2Vec (baseado *word embeddings*) [1]. Tais modelos serão utilizados e comparados entre si nas etapas subsequentes de classificação propriamente dita.

4.2.1 Bag of words (BoW)

O modelo de representação vetorial Bag of words (BoW) consiste no mais simples modelo de representação vetorial de documentos. Ele consiste na simples contagem de termos de um texto e inserção dessas contagens em um vetor de N dimensões em que N é o total palavras de todo o conjunto de textos.

4.2.2 TF-IDF

O modelo TF-IDF [11] consiste em uma melhoria do BoW pois mensura a importância relativa de um termo em determinado texto, conforme equação a seguir:

$$tfidf_{t,d} = tf_{t,d} \cdot \log \frac{N}{df_t} \tag{1}$$

Em que $tf_{t,d}$ consiste na frequência do termo t no documento d, df_t, o número de documentos contendo o termo t e N a quantidade de documentos no corpus.

4.2.3 Doc2Vec

A ideia do Doc2vec é criar uma representação numérica do documnento independente de seu tamanho. Assim como o word2vec faz com palavras, o doc2vec trata cada documento com um vetor qualquer. Na etapa de treinamento os vetores de palavras W são treinados juntamento com os vetores de documentos D. São utilizadas duas abordagens principais, Memória Distribuída (PV-DM) e Bag of Words distribuído (DBOW), análogos, respectivamente, aos modelos Skip-gram e CBOW do Word2Vec padrão. O modelo de memória distribuída usa o contexto atual juntamente com o documento atual para prever a próxima palavra . Já o DBOW utiliza apenas o documento atual para prever todas as palavras de contexto. No presente trabalho optou-se por utilizar essa última abordagem pois apresentou melhores resultados em testes preliminares.

4.3 Algoritmos de classificação

Como se trata de um problema de classificação multi-rótulo, serão utilizados algoritmos específicos para esse tipo de tarefa. A biblioteca Sci-kit Learn [4] prevê várias opções de modelos para tratar com dados com múltiplos rótulos. A modelagem pode envolver, por exemplo, uma classificação um-contra-todos em que é construído um classificador por rótulo. Outra modelagem envolve uma classificação um-contra-um em que é construído um classificador para cada par de classes. Dentre as técnicas que poderão utilizados, destacam-se: Classificação por Árvores de Decisão; Florestas Aleatórias; Máquina de Vetor de Suporte (com SVC). Além dessas, de acordo com a necessidade e dos resultados encontrados, poderão ser aplicadas técnicas baseadas em Redes Neurais Artificiais [6]. Finalmente, o resultado da classificação será analisado com métricas tais como precisão, *recall* e F1 (média harmônica entre ambos).

4.3.1 Árvores de decisão

Uma árvore de decisão [9] é um modelo preditivo que consiste em uma sequência de observações sobre um item, as quais são "ramificações" da árvore até uma determinada conclusão, ou valor, por sua vez representada pelas "terminações". Em cada ramificação é realizada uma decisão responsável por particionar o espaço amostral em diferentes ramos de acordo com alguma métrica pré-determinada. A biblioteca *sklearn* permite duas métricas de particionamento, também chamadas de critério de impureza, quais sejam:

$$I_G = 1 - \sum_{j=1}^{c} p_j^2 \qquad I_H = -\sum_{j=1}^{c} p_j log_2(p_j) \tag{2}$$

Em que I_g consiste no índice Gini e I_H consiste no índice de entropia. Ambas fazem uso da probabilidade de determinada classe na amostram atual (p_j). O algoritmo de classificação por árvore de decisão do *sklearn* permite o uso de ambas.

Dentre os parâmetros utilizados por uma árvore de decisão destacam-se o seu tamanho máximo, o número de instâncias consideradas para particionamento e o número máximo de instâncias nos nós folha.

4.3.2 Florestas aleatórias

Um classificador baseado em florestas aleatórias [**randomforest**] consiste em uma combinação de árvores de decisão e cujo resultado é obtido por meio da moda de cada uma

delas. Na etapa de treinamento são construídas diferentes árvores de decisão para diferentes subconjuntos de amostras de treinamento selecionados aleatoriamente, técnica conhecida como *boostraping* ou *bagging*. Para cada árvore criada são selecionados diferentes conjuntos aleatórios de características (*feature boostraping*) que serão levadas em conta por cada árvore de decisão em seu processo decisório de particionamento.

De modo geral, o número de estimadores de uma árvore de decisão varia entre algumas dezenas até algums centenas enquanto o número de características consideradas geralmente é dado por $sqrt(|\vec{F}|)$, com F sendo o conjunto de características. Além destes, podem ser utilizados ainda os mesmos parâmetros das árvores de decisão listados na seção anterior.

4.3.3 Ml-kNN

O algoritmo Ml-kNN[12] consiste em uma extensão multi-rótulo do kNN (*k-nearest neighboors*, em português **k-vizinhos** mais próximos) tradicional baseada em aprendizado preguiçoso (*lazy learning*). A ideia do algoritmo é, assim como o kNN tradicional, para cada elemento ainda não observado, primeiramente identificar os **k-vizinhos** mais próximos. Em seguida, é extraído a quantitativo de vizinhos pertencentes a cada uma das classes, sobre o qual é utilizado o princípio do máximo a posteriori (MAP) para definir o conjunto de rótulos do elemento ainda não observado.

O princípio do máximo a posteriori (MAP) consiste em uma estimativa de um valor desconhecido baseado na moda da distribuição posterior. Para o kNN, baseia-se na frequência total de rótulos (a priori) quanto na frequência de rótulos vizinhos a outros rótulos de mesmo tipo, conforme equação a seguir:

$$y_t(l) = argmax_{b\in\{0,1\}} P(H_b^l) P(E^l_{C_t(l)}|H_b^l), l \in Y \tag{3}$$

Em que $P(H_b^l)$ consiste na probabilidade a priori, ou seja, o percentual de instâncias com $(b = 1)$ ou sem $(b = 0)$ o rótulo l na base de treino. Já $P(E^l_{C_t(l)}|H_b^l)$ consiste na probabilidade a posteriori, a qual consiste na proporção de instâncias de treino com $(b = 1)$ ou sem $(b = 0)$ rótulo l que contenha exatamente j vizinhos com rótulo l.

O Ml-kNN da biblioteca *multi-sklearn* faz uso de dois parâmetros, quais sejam, o número de vizinhos k e um parâmetro de suavização (ou viés) s a ser utilizado para reduzir as probabilidades de acordo com a ordem de vizinhança.

4.3.4 Perceptron multicamada

Um *perceptron* multicamada (MLP) consiste em uma classe de redes neurais articiais (RNA) *feedforward* (alimentação para frente). Uma MLP possui ao menos três camadas: uma camada de entrada, uma ou mais camadas oculta e uma camada de saída. De modo geral, cada neurônio da camadas oculta e de saída possui uma função de ativação não-linear, ou seja, é capaz de classificar dados não linearmente separados.

Dentre as funções de ativação mais utilizadas e que foram testadas no presente trabalho destacam-se:

$$y(v_i) = tanh(v_i) \qquad y(v_i) = max(0, x) \tag{4}$$

Correspondendo respectivamente à ativação por tangente hiberpolica e por retificador linear (ReLU). Este último tem sido amplamente utilizado por lidar melhor com problemas numéricos das funções sigmóides tais como a tangente hiperbólica.

Assim como outros tipos de redes neurais o aprendizado é realizado por meio de uma técnica chamada propagação retrógada (*backpropagation*). Tal técnica consiste no cálculo do gradiente da função de perda em relação a cada um dos pesos da rede. O gradiente é propagado para trás para frente, ou seja, da última para a primeira camada, de forma eficiente por meio de uma técnica chamada gradiente descendente. Ao final, o resultado é utilizado para atualizar cada peso da rede com vistas a reduzir o erro total da mesma.

Chama-se de resolvedor da rede neural a técnica utilizada para atualização dos pesos durante propagação retrógada. Uma das técnicas mais básicas é o **Gradiente Estocástico Descendente (SGD)**, a qual se trata de uma aproximação do gradiente descendente usando apenas um subconjunto aleatórios de dados para cálculo do gradiente descendente a partir de uma taxa de aprendizado fixa. Uma técnica mais sofisticada é chamada de Adam (Estimação de Momento Adaptativa) [8] e consiste na utilização de taxas de aprendizado adaptativas utilizando o primeiro e segundo momentos estatísticos (média e variância). A ideia do Adam é suavizar as atualizações dos pesos desacelarando-as, ou seja, garantindo que permaneça na direção anterior e impedindo sua inércia ou oscilação demasiada:

$$w_t = w_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{5}$$

Em que $\hat{m}_t$ e $\hat{v}_t$ são a média e variância não centralizada já sem viés, calculadas por meio das seguintes equações:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{6}$$

Em que m_t são v_t são a média e variância descentralizada. β representa hiperparâmetros fixos, geralmente utiliza-se 0.9 e 0.99 e representa o nível de suavização dos momentos.

Dentre os parâmetros utilizados para calibragem do classificador no presente trabalho, foram utilizados: a arquitetura da rede, ou seja, quantidade e dimensões das camadas ocultas; função de ativação das camadas ocultas (ReLU ou tanh); e o resolvedor (SGD ou Adam).

5 Experimentos e resultados

Foi realizada uma validação cruzada com dois particionamentos, teste e treino, além de estratificação iterativa para garantir a distribuição correta de classes por particionamento, inclusive em combinação entre si. A escolha de parâmetros envolveu o uso de uma grade de pesquisa (*grid search*) composta com os parâmetros definidos na tabela 1.

5.1 Métricas de avaliação

Os algoritmos serão comparados utilizando-se a métrica **f1-micro**, a qual consiste na média do score f1 por classificação realizada, ou seja, trata-se de uma média micro, conforme equações 1 e 2.

$$PrecisaoMicro = \frac{\sum_i^n TP_{C_i}}{\sum_i^n TP_{C_i} + \sum_i^n FP_{C_i}}, RecallMicro = \frac{\sum_i^n TP_{C_i}}{\sum_i^n TP_{C_i} + \sum_i^n FN_{C_i}} \tag{7}$$

Em que TP_{C_i}, FP_{C_i} e FN_{C_i} correspondem, respectivamente, ao quantitativo de verdadeiros positivos, falsos positivos e falsos negativos para cada classe C_i.

Table 1: Grade de parâmetros utilizada

Árvores de Decisão	Florestas Aleatórias	Ml-kNN	MLP
min samples split: $[0.005, 0.010, 2]$ **max depth:** $[16, 32, None]$	**min samples split:** $[0.005, 0.010, 2]$ **n estimators:** $[100, 110, 120]$ **min samples leaf:** [5, 3, 1]	**k:** $[6, 8, 10, 12]$ **s:** $[0.5, 1.0, 1.5, 2.0]$	**layer sizes:** $[(150), (100, 100), (50, 50, 50)]$ **activation:** $[tanh, relu]$ **solver:** $[sgd, adam]$
9 combinações	27 combinações	16 combinações	12 combinações
27 com modelos	81 com modelos	48 com modelos	36 com modelos

Total de 384 modelos com validação cruzada de duas partições

$$F1Micro = 2 \times \frac{PrecisaoMicro + RecallMicro}{PrecisaoMicro \times RecallMicro} \tag{8}$$

O **f1-macro**, em contrapartida, consiste na média das métricas f1 para todas as amostras de um rótulo específico, ou seja, dá o mesmo peso para todas as classes na base de testes. O **f1-macro** dá um peso proporcional maior para as classes menos representadas.

No presente trabalho, apesar da grande assimetria na distribuição de rótulos, optou-se por utilizar a métrica f1-micro porque deseja-se avaliar as classificações corretas ou incorretas independentemente da distribuição de classes. Dessa forma, os classificadores testados tenderão a se sobressair nas classes mais presentes.

5.2 Resultados

5.2.1 Árvore de decisão

Os resultados para a aplicação do algoritmo de Árvore de decisão são exibidos nas tabelas 2 a 4.

Table 2: Resultados para árvore de decisão com modelagem BoW

max depth	min samples split	mean fit time	mean train score	mean test score
16	0.0050	4.9716	0.7067	0.6429
16	0.0100	4.6879	0.6835	0.6393
16	2.0000	6.3916	0.7907	0.6368
32	0.0050	6.3834	0.7766	0.6791
32	0.0100	6.1524	0.7416	0.6758
32	2.0000	8.0246	0.9135	0.6639
-	0.0050	7.3649	0.8175	<u>0.6860</u>
-	0.0100	6.8024	0.7706	0.6830
-	2.0000	9.2464	1.0000	0.6542

Pode-se extrair das tabelas 2 a 4 as seguintes conclusões:

Table 3: Resultados para árvore de decisão com modelagem TF-IDF

max depth	min samples split	mean fit time	mean train score	mean test score
16	0.0050	10.6429	0.7138	0.6491
16	0.0100	10.2673	0.6954	0.6480
16	2.0000	11.7186	0.7751	0.6459
32	0.0050	14.4353	0.7917	0.6818
32	0.0100	13.5527	0.7526	0.6850
32	2.0000	15.8470	0.9013	0.6678
	0.0050	18.4709	0.8421	0.6926
	0.0100	16.5481	0.7954	0.6985
	2.0000	19.9294	1.0000	0.6645

Table 4: Resultados para árvore de decisão com modelagem Doc2Bow

max depth	min samples split	mean fit time	mean train score	mean test score
16	0.0050	24.2283	0.6475	0.5767
16	0.0100	24.6089	0.6158	0.5710
16	2.0000	26.1113	0.7137	0.5582
32	0.0050	27.1475	0.6553	0.5785
32	0.0100	29.1056	0.6181	0.5677
32	2.0000	27.6735	0.7329	0.5385
	0.0050	27.4904	0.6533	0.5734
	0.0100	26.6175	0.6127	0.5697
	2.0000	29.7168	0.7272	0.5356

- Árvores de decisão sem limite máximo de profundidade apresentaram resultados melhores que as árvores com limites de 16 e 32. Apesar disso, tendem a causar *overfitting*, especialmente quando combinadas com o parâmetro de número mínimo de amostras para divisão (*min samples split*).
- O *min samples split* auxilia a reduzir o *overfitting* e o valor de 1% apresentou o melhor resultado de todas combinações de parâmetros para árvore de decisão (com TF-IDF e sem limites de profundidade.
- O tempo de execução é mais sensível à profundidade da árvore do que ao parâmetro mínimo de amostras para divisão;
- A modelagem fazendo uso de Doc2Vec não apresentou bons resultados, possivelmente pelo tamanho reduzido das bases. O pior resultado foi identificado para os valores padrões de árvore de decisão (profundidade máxima ilimitada e mínimo de amostras para divisão igual a 2);

5.2.2 Florestas aleatórias

Os resultados para a aplicação do algoritmo de Florestas Aleatórias são exibidos nas tabelas 5 a 7.

Pode-se extrair das tabelas 5 a 7 as seguintes conclusões:

Table 5: Resultados para florestas aleatórias com modelagem BoW

min samples leaf	min samples split	n estimators	mean fit time	mean train score	mean test score
5	0.0050	100	8.6113	0.6355	0.6069
5	0.0050	110	9.0049	0.6365	0.6064
5	0.0050	120	9.9670	0.6358	0.6054
5	0.0100	100	8.2255	0.6152	0.5945
5	0.0100	110	8.8539	0.6142	0.5914
5	0.0100	120	9.3522	0.6152	0.5917
5	2.0000	100	9.1637	0.6475	0.6123
5	2.0000	110	9.8154	0.6466	0.6127
5	2.0000	120	10.5620	0.6461	0.6126
3	0.0050	100	9.4090	0.6628	0.6243
3	0.0050	110	10.6299	0.6607	0.6236
3	0.0050	120	11.1434	0.6623	0.6256
3	0.0100	100	9.7503	0.6347	0.6051
3	0.0100	110	11.0635	0.6313	0.6037
3	0.0100	120	10.2679	0.6336	0.6033
3	2.0000	100	10.5610	0.6935	0.6372
3	2.0000	110	13.0211	0.6957	0.6408
3	2.0000	120	13.2317	0.6959	0.6402
1	0.0050	100	12.1470	0.7048	0.6421
1	0.0050	110	13.0383	0.7016	0.6409
1	0.0050	120	14.0506	0.7037	0.6412
1	0.0100	100	10.3056	0.6549	0.6160
1	0.0100	110	11.5402	0.6552	0.6151
1	0.0100	120	12.1726	0.6545	0.6151
1	2.0000	100	18.3970	0.9998	0.6789
1	2.0000	110	20.0013	0.9999	0.6814
1	2.0000	120	20.1383	0.9999	0.6811

- O número de estimadores (ou árvores de decisão) não apresentou grande impacto no resultado final para nenhum dos modelos e parâmetros testados.
- Assim como para árvores de decisão o parâmetro *min samples split* auxilia a reduzir o *overfitting*;
- O melhor resultado foi identificado para os parâmetros: modelagem TF-IDF, *min samples leaf* igual a 1, m*in samples split* igual a 2 e número de estimadores igual a 110. Infelizmente tal resultado também apresentou *overfitting* na base de treinamento.
- O tempo de execução é mais sensível ao número mínimo de amostras para divisão do que aos outros parâmetros. Isso ocorre porque tal parâmetro efetivamente controle a profundidade da árvore.
- Novamente a modelagem fazendo uso de Doc2Vec não apresentou bons resultados.

Table 6: Resultados para florestas aleatórias com modelagem TF-IDF

min samples leaf	min samples split	n estimators	mean fit time	mean train score	mean test score
5	0.0050	100	13.6353	0.6596	0.6220
5	0.0050	110	14.5407	0.6554	0.6218
5	0.0050	120	16.2538	0.6589	0.6210
5	0.0100	100	13.2579	0.6406	0.6126
5	0.0100	110	14.4841	0.6398	0.6089
5	0.0100	120	15.2923	0.6382	0.6067
5	2.0000	100	14.9008	0.6688	0.6265
5	2.0000	110	15.3830	0.6682	0.6258
5	2.0000	120	16.9312	0.6700	0.6262
3	0.0050	100	15.5227	0.6874	0.6384
3	0.0050	110	18.2123	0.6857	0.6378
3	0.0050	120	18.0357	0.6858	0.6354
3	0.0100	100	15.4880	0.6585	0.6200
3	0.0100	110	16.5554	0.6583	0.6205
3	0.0100	120	17.5967	0.6581	0.6220
3	2.0000	100	16.9978	0.7213	0.6520
3	2.0000	110	18.6453	0.7185	0.6490
3	2.0000	120	19.5587	0.7180	0.6499
1	0.0050	100	19.7984	0.7304	0.6536
1	0.0050	110	21.1146	0.7292	0.6528
1	0.0050	120	23.1994	0.7299	0.6526
1	0.0100	100	17.5468	0.6802	0.6296
1	0.0100	110	19.3696	0.6820	0.6312
1	0.0100	120	20.9992	0.6800	0.6306
1	2.0000	100	25.5599	0.9997	0.6825
1	2.0000	110	27.5232	1.0000	0.6831
1	2.0000	120	28.5935	0.9999	0.6813

O pior resultado ocorreu para os parâmetros: *min samples leaf* igual a 1, *min samples split* igual a 01%, número de estimadores igual a 100.

5.2.3 mL-KNN

Os resultados para a aplicação do algoritmo de mL-kNN aleatórias são exibidos nas tabelas 8 a 10.

Pode-se extrair das tabelas 8 a 10 as seguintes conclusões:

- Quanto maior a quantidade de vizinhos (k) considerados, pior o resultado tanto para a base de teste quanto para a base de treinamento para todos os modelos treinados;
- O parâmetro s não afeta o resultado significativamente. No entanto, o melhor resultado identificado teve parâmetro s igual a 1 e número de vizinhos igual k, além de utilização de modelagem vetorial;

Table 7: Resultados para florestas aleatórias com modelagem Doc2Vec

min samples leaf	min samples split	n estimators	mean fit time	mean train score	mean test score
5	0.0050	100	54.0277	0.6010	0.5688
5	0.0050	110	58.1041	0.5993	0.5674
5	0.0050	120	92.0143	0.5992	0.5701
5	0.0100	100	50.6494	0.5511	0.5347
5	0.0100	110	55.0003	0.5523	0.5387
5	0.0100	120	58.4316	0.5561	0.5362
5	2.0000	100	54.6610	0.6428	0.5924
5	2.0000	110	58.6687	0.6428	0.5944
5	2.0000	120	63.6673	0.6413	0.5932
3	0.0050	100	55.4461	0.6049	0.5727
3	0.0050	110	60.1204	0.6061	0.5728
3	0.0050	120	64.7921	0.6042	0.5738
3	0.0100	100	53.9570	0.5501	0.5356
3	0.0100	110	56.7838	0.5569	0.5377
3	0.0100	120	61.4074	0.5539	0.5349
3	2.0000	100	59.2064	0.6785	0.6110
3	2.0000	110	66.5480	0.6772	0.6099
3	2.0000	120	68.7834	0.6775	0.6098
1	0.0050	100	58.2495	0.6053	0.5707
1	0.0050	110	61.8017	0.6064	0.5693
1	0.0050	120	69.1566	0.6073	0.5701
1	0.0100	100	56.2805	0.5513	0.5338
1	0.0100	110	60.0591	0.5540	0.5361
1	0.0100	120	63.2751	0.5588	0.5387
1	2.0000	100	67.8059	0.7478	0.6214
1	2.0000	110	72.1795	0.7457	0.6231
1	2.0000	120	77.5964	0.7479	0.6219

- O tempo de execução é pouco alterado com a alteração dos parâmetros, mesmo os vizinhos mais próximos;
- Novamente os piores resultados foram identificados para a modelagem vetorial fazendo uso de Doc2Vec;

5.2.4 Perceptron multicamada (MLP)

Os resultados para a aplicação do algoritmo MLP estão disponíveis nas tabelas 11 a 14.

Pode-se extrair das tabelas 11 a 14

- O resolvedor que fez uso de SGD teve problema de convergências para algumas instâncias, o que explica o baixo **f1-score** tanto para a base de teste quanto para a base de treinamento;
- O resolvedor Adam apresentou resultados consistentemente superiores ao SGD em

Table 8: Resultados para Ml-kNN com modelagem BoW

k	s	mean fit time	mean train score	mean test score
6	0.5000	13.2898	0.7950	0.7094
6	1.0000	13.2347	0.7939	0.7099
6	1.5000	13.2336	0.7925	0.7094
6	2.0000	13.3359	0.7902	0.7091
8	0.5000	12.8595	0.7785	0.7074
8	1.0000	12.8516	0.7772	0.7074
8	1.5000	13.1460	0.7750	0.7061
8	2.0000	13.0637	0.7702	0.7024
10	0.5000	14.3474	0.7682	0.7023
10	1.0000	12.8123	0.7658	0.7022
10	1.5000	14.3741	0.7624	0.6993
10	2.0000	14.1043	0.7572	0.6970
12	0.5000	13.3172	0.7577	0.6975
12	1.0000	13.4367	0.7539	0.6958
12	1.5000	13.6212	0.7491	0.6930
12	2.0000	13.8545	0.7448	0.6915

todas as instâncias treinadas, mesmo quando este último convergiu;

- A melhor arquitetura foi a de uma única camada oculta com 150 neurônios. Em seguida vieram as arquiteturas (100, 100) e (50, 50, 50). Disso conclui-se que é melhor a rede não ser profunda, ou seja, deve-se valorizar mais a quantidade de neurônios em uma única camada, horizontalizando o modelo. Deve-se pontuar que isso elimina relações entre palavras, já que não há ligações entre elas na cama oculta;

- A função da ativação ReLU foi consistentemente superior à tanh em todos os modelos treinados. Isso é algo já previsto e conhecido na literatura;

- O tempo de treinamento foi maior para redes mais horizontalizadas do que as mais profundas, dado o maior número de interconexões e multiplicações de matrizes;

5.3 Exemplo de uma instância classificação

Nas tabela 15 e figura 2 é exibida uma instância de classificação utilizando particionamento de metade da base de dados.

Da tabela 15 e figura 2 conclui-se que, conforme já discutido, a utilização da métrica *f1-score* micro valorizou as classes mais frequentes visto que ele é calculado sobre cada classificação realizada. Mesmo que a precisão seja alta, o *recall-micro* foi relativamente baixo já que a grande maioria das classes não é identificada nos testes, apesar da precisão relativamente alta de 92%. De modo geral, as médias macro apresentaram resultado de *f1-score* muito baixo, dada a dificuldade do algoritmo de classificar em rótulos menos frequentes.

A título de exemplo, o rótulo "Incentivo Fiscal" apresentou 100% de precisão apesar de ter identificado uma única instância com esse rótulo, tendo acertado. O recall, no entanto, foi extremamente baixo, deixando de classificar 12 instâncias na base de teste. O resultado foi um f1-score de apenas 0.16.

Table 9: Resultados para Ml-kNN com modelagem TF-IDF

k	s	mean fit time	mean train score	mean test score
6	0.5000	13.7048	0.8330	0.7664
6	1.0000	13.3529	0.8324	0.7665
6	1.5000	13.0727	0.8311	0.7657
6	2.0000	12.7718	0.8292	0.7637
8	0.5000	13.2088	0.8234	0.7636
8	1.0000	13.3803	0.8221	0.7634
8	1.5000	12.9964	0.8194	0.7613
8	2.0000	12.8739	0.8172	0.7597
10	0.5000	13.3095	0.8168	0.7614
10	1.0000	13.0316	0.8150	0.7613
10	1.5000	13.2358	0.8111	0.7588
10	2.0000	13.2762	0.8048	0.7556
12	0.5000	13.2834	0.8093	0.7572
12	1.0000	13.2254	0.8066	0.7566
12	1.5000	13.2278	0.8010	0.7518
12	2.0000	13.0845	0.7956	0.7493

Outro exemplo a ser destacado é o do rótulo "Executivo", o qual apresentou precisão e recall muito baixos, em relação aos demais rótulos frequentes. Isso ocorre porque se trata de um rótulo que co-ocorre com muitos outros e em muitos contextos diferentes, o que gera classificações incorretas ou ausentes.

6 Trabalhos futuros

Para trabalhos futuros pretende-se estudar a aplicação de diferentes abordagens ao lidar com o problema de classificação multirótulo de normas jurídicas, dentre as quais pode-se destacar:

- Utilização de outras modelagens vetoriais, tais como o Alocação latente de Dirichlet (LDA), Word2Vec com médio dos vetores de palavras para cada palavra do texto, dentre outros. Além disso, retreinar o Doc2Vec utilizando maior quantidade de documentos;
- Aplicação de técnicas de redução de dimensionalidade como Análise de Componentes Principais (PCA) com vistas a reduzir o tamanho do espaço vetorial utilizado para alimentar os algoritmos de classificação;
- Uso de estimadores baseados em redes neurais mais complexos (LSTM, CNN, etc) que provavelmente apresentarão resultados superiores;

7 Conclusão

A métrica de avaliação utilizando o *f1-micro* apresentou a grave deficiência de subestimar as classes menos representadas na base de dados por considerar a média de todos as

Table 10: Resultados para Ml-kNN com modelagem Doc2Vec

k	s	mean fit time	mean train score	mean test score
6	0.5000	43.2487	0.8017	0.7313
6	1.0000	38.0475	0.8004	0.7339
6	1.5000	41.9220	0.7953	0.7343
6	2.0000	39.0170	0.7931	0.7304
8	0.5000	36.8002	0.7877	0.7266
8	1.0000	37.2788	0.7868	0.7284
8	1.5000	33.8701	0.7821	0.7312
8	2.0000	30.0602	0.7792	0.7274
10	0.5000	29.8255	0.7755	0.7271
10	1.0000	31.4301	0.7755	0.7285
10	1.5000	32.6740	0.7723	0.7262
10	2.0000	29.5816	0.7673	0.7248
12	0.5000	29.6221	0.7732	0.7271
12	1.0000	29.8728	0.7676	0.7235
12	1.5000	29.7510	0.7621	0.7204
12	2.0000	30.4186	0.7577	0.7135

classificações realizadas. Além disso, o uso de *doc2vec* não trouxe ganhos de desempenho quando comparados com modelos vetoriais mais simples. A causa disso está relacionada à baixa quantidade de textos utilizados para treinar o modelo da linguagem.

Há de ressaltar, contudo, que o resultado mostrou-se satisfatório para classes mais frequentemente presentes na base de dados, especialmente se não co-ocorrem com muitas outras. No entanto, dado o baixo *recall* médio do modelo, não é recomendado utilizá-lo em um ambiente que se penaliza mais a não-classificação do que a classificação incorreta, como pode ser o caso de classificação textual.

References

[1] *Biblioteca Gensim*. `https://pypi.org/project/gensim/`. Acesso em 25/10/2020.

[2] *Biblioteca NLTK: The Natural Language Toolkit*. `https://www.nltk.org/`. Acesso em 25/10/2020.

[3] et. al. Chalkidis. I. "Extreme Multi-Label Legal Text Classification: A case study in EU Legislation." In: *Proceedings of the Natural Legal Language Processing Workshop* (2019), pp. 78–87.

[4] *Classificação multi-rótulo no SKLEarn*. `https://scikit-learn.org/stable/modules/multiclass.html`. Acesso em 25/10/2020.

[5] Luis Otávio de Colla Furquim. "Agrupamento e Categorização de Documentos Jurídicos". In: *PUC-RS, Faculdade de Informática, Programa de Pós Graduação* (2011).

[6] Geoffrey E. Hinton. "Connectionist learning procedures". In: *Artificial intelligence 40.1* (1989), pp. 185–234.

Table 11: Resultados para MLP com modelagem BoW

activation	hidden layer sizes	solver	mean fit time	mean train score	mean test score
tanh	150	sgd	689.3050	0.8448	0.7508
tanh	150	adam	941.5917	0.9996	0.7907
tanh	(100, 100)	sgd	455.7334	0.8176	0.7317
tanh	(100, 100)	adam	610.4199	0.9998	0.7832
tanh	(50, 50, 50)	sgd	236.6059	0.7131	0.6508
tanh	(50, 50, 50)	adam	325.5805	0.9948	0.7543
relu	150	sgd	663.7065	0.8943	0.7620
relu	150	adam	842.4843	0.9998	0.7846
relu	(100, 100)	sgd	457.9912	0.9218	0.7637
relu	(100, 100)	adam	564.0259	1.0000	0.7797
relu	(50, 50, 50)	sgd	233.5775	0.9177	0.7428
relu	(50, 50, 50)	adam	213.0475	0.9999	0.7608

Table 12: Resultados para MLP com modelagem TF-IDF

activation	hidden layer sizes	solver	mean fit time	mean train score	mean test score
tanh	150	sgd	266.2043	0.4163	0.4137
tanh	150	adam	356.9774	0.9998	0.7941
tanh	(100, 100)	sgd	163.7233	0.5072	0.5031
tanh	(100, 100)	adam	252.1229	0.9999	0.7837
tanh	(50, 50, 50)	sgd	76.7089	0.3356	0.3329
tanh	(50, 50, 50)	adam	102.9910	0.9975	0.7463
relu	150	sgd	238.4223	0.4277	0.4252
relu	150	adam	340.3201	0.9998	0.7969
relu	(100, 100)	sgd	157.2616	0.5139	0.5114
relu	(100, 100)	adam	227.8457	1.0000	0.7738
relu	(50, 50, 50)	sgd	74.5037	0.5414	0.5303
relu	(50, 50, 50)	adam	125.8403	1.0000	0.7221

[7] Geoff Holmes Jesse Read Bernhard Pfahringer and Eibe Frank. "Classifier Chains for Multi-label Classification. Machine Learning Journal". In: 85.3 (2011).

[8] Diederik P. Kingma and Jimmy Lei Ba. "Adam : A method for stochastic optimization". In: (2014).

[9] R. Olshen L. Breiman J. Friedman and C. Stone. "Classification and Regression Trees". In: *Wadsworth, Belmont, CA* (1984).

[10] *Portal de Dados Abertos da Assembléia Legislativa de Minas Gerais*. `http://dadosabertos.almg.gov.br/ws/ajuda/sobre`. Acesso em 25/10/2020.

[11] Webb G.I Sammut C. "TF–IDF". In: *Encyclopedia of Machine Learning.* (2011). Acesso em 25/10/2020.

[12] Min-Ling Zhang and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning". In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

Table 13: Resultados para MLP com modelagem Doc2Vec

activation	hidden layer sizes	solver	mean fit time	mean train score	mean test score
tanh	150	sgd	25.0046	0.5887	0.5818
tanh	150	adam	27.1082	0.8475	0.7586
tanh	(100, 100)	sgd	26.2182	0.5995	0.5961
tanh	(100, 100)	adam	28.1114	0.8695	0.7528
tanh	(50, 50, 50)	sgd	24.9678	0.5281	0.5252
tanh	(50, 50, 50)	adam	26.2418	0.8022	0.7287
relu	150	sgd	26.1269	0.6033	0.5957
relu	150	adam	27.0755	0.8567	0.7569
relu	(100, 100)	sgd	27.6873	0.6415	0.6327
relu	(100, 100)	adam	28.9128	0.8905	0.7491
relu	(50, 50, 50)	sgd	25.8912	0.6183	0.6101
relu	(50, 50, 50)	adam	26.7335	0.8379	0.7338

Table 14: Resultados agregados para modelagem MLP

origin	mean fit time	mean train score mean	mean train score max	mean test score	mean test score max
cv	519.5058	0.9253	1.0000	0.7546	0.7907
tfidf	198.5768	0.7283	1.0000	0.6111	0.7969
d2v	26.6733	0.7236	0.8905	0.6685	0.7586

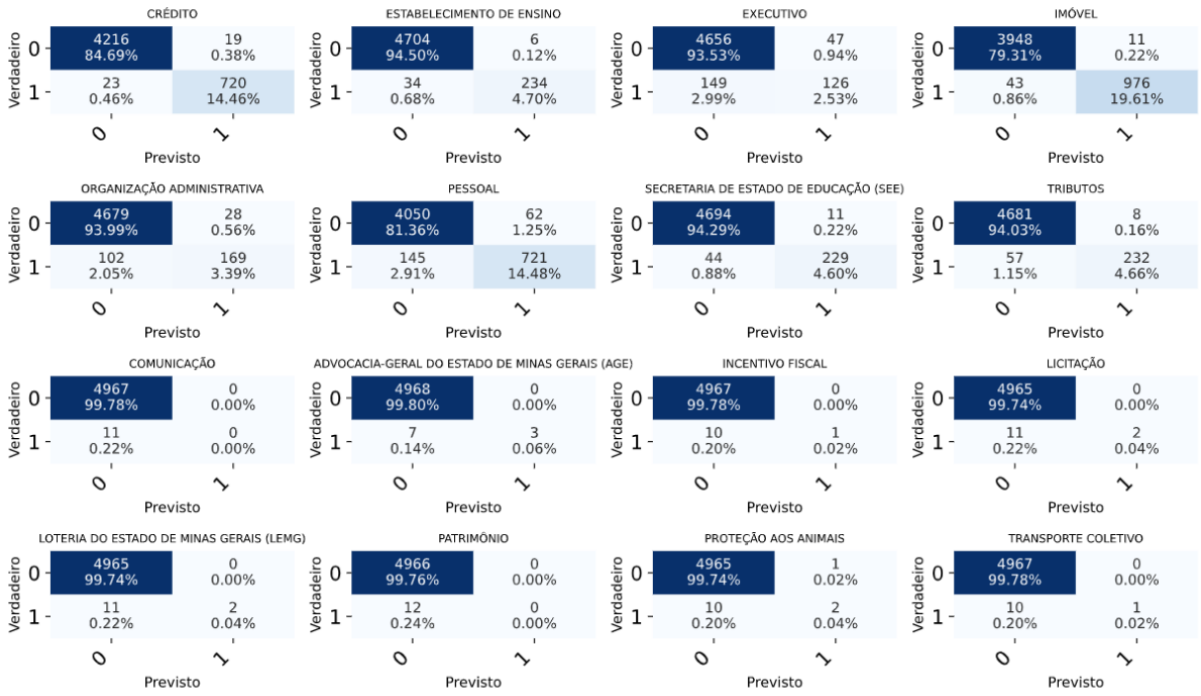

Figure 2: Matrizes de confusão para 8 classes mais e menos frequentes na instância de teste fazendo uso de MLP com TF-IDF

Table 15: Resultados de precisão, recall e F1 para MLP com modelagem TF-IDF

	precision	recall	f1-score	support
samples avg	0.8034	0.7541	0.7650	7552.0000
weighted avg	0.8985	0.6962	0.7652	7552.0000
macro avg	0.8082	0.4361	0.5294	7552.0000
micro avg	0.9229	0.6962	0.7937	7552.0000
IMÓVEL	0.9889	0.9578	0.9731	1019.0000
PESSOAL	0.9208	0.8326	0.8745	866.0000
CRÉDITO	0.9743	0.9690	0.9717	743.0000
TRIBUTOS	0.9667	0.8028	0.8771	289.0000
EXECUTIVO	0.7283	0.4582	0.5625	275.0000
SECRETARIA DE EDUCAÇÃO (SEE)	0.9542	0.8388	0.8928	273.0000
ORGANIZAÇÃO ADMINISTRATIVA	0.8579	0.6236	0.7222	271.0000
ESTABELECIMENTO DE ENSINO	0.9750	0.8731	0.9213	268.0000
ADMINISTRAÇÃO ESTADUAL	0.8532	0.5254	0.6503	177.0000
DIVISÃO ADMINISTRATIVA	0.9573	0.9075	0.9318	173.0000
...	....	...	...	...
DEFENSORIA PÚBLICA DE MG (DPMG)	0.8750	0.5385	0.6667	13.0000
SECRETARIA DE SEGURANÇA (SSPMG)	0.0000	0.0000	0.0000	13.0000
LOTERIA DO ESTADO DE MG (LEMG)	1.0000	0.1538	0.2667	13.0000
LICITAÇÃO	1.0000	0.1538	0.2667	13.0000
PROTEÇÃO AOS ANIMAIS	0.6667	0.1667	0.2667	12.0000
PATRIMÔNIO	0.0000	0.0000	0.0000	12.0000
TRANSPORTE COLETIVO	1.0000	0.0909	0.1667	11.0000
COMUNICAÇÃO	0.0000	0.0000	0.0000	11.0000
INCENTIVO FISCAL	1.0000	0.0909	0.1667	11.0000
ADVOCACIA-GERAL DE MG (AGE)	1.0000	0.3000	0.4615	10.0000