

Inferring the source of official texts: can SVM beat ULMFiT?

Pedro Henrique Luz de Araujo¹ Teófilo Emidio de Campos¹
Marcelo Magalhães Silva de Sousa²

¹Departamento de Ciência da Computação, Universidade de Brasília, Brasília – DF, Brazil

²Tribunal de Contas do Distrito Federal, Zona Cívico-Administrativa, Brasília – DF, Brazil

*pedrohluzaraujo@gmail.com, t.decampos@st-annes.oxon.org,
marcelomsousa@tc.df.gov.br*

23 de julho de 2020

Artigo e código disponíveis em

<https://cic.unb.br/~teodecampos/KnEDLe/propor2020/>



Pontos principais

- 1 Introdução
- 2 Interlúdio
- 3 Os dados
- 4 Métodos
- 5 Resultados
- 6 Conclusão

Introdução

Motivação

- Diários Oficiais são uma rica fonte de informações de interesse público:
 - ▶ Nomeações, contratações, licitações, atos oficiais.
 - ▶ Gastos públicos estão sujeitos a fraudes e irregularidades.
- Dificuldades:
 - ▶ Dados não estruturados.
 - ▶ Linguagem específica: textos oficiais.

Exemplos

Trecho 1

O GOVERNADOR DO DISTRITO FEDERAL, no uso das atribuições que lhe confere o artigo 100, incisos XXVI e XXVII, da Lei Orgânica do Distrito Federal, resolve [...]

Trecho 2

Presidente da COVED, acolhendo os pareceres inseridos nos processos abaixo, declara habilitados para a venda à PRAZO os itens a seguir: [...]

Trecho 3

[...] TORNAR PÚBLICO o resultado das investigações constantes nos processos dos servidores listados abaixo e que se configuraram em acidente de serviço, sem dano, nos termos do artigo 23, § 1º, inciso IV, do Decreto nº 34.023, de 10 de dezembro de 2012, observando-se a seguinte ordem: número do processo, nome e matrícula. [...]

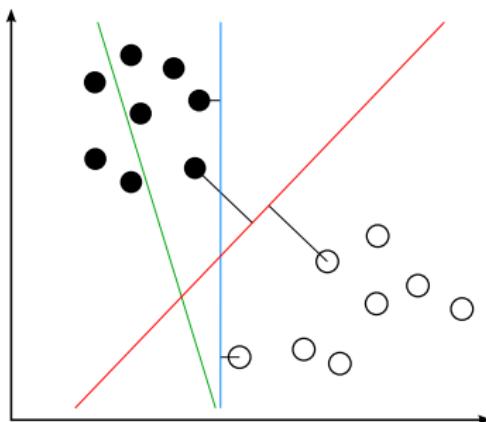
Objetivos

- Usar Aprendizado de Máquina para processar textos do DODF e identificar o órgão público de origem.
 - ▶ Indexação de informação;
 - ▶ Auditoria pública;
 - ▶ Uso de expressões regulares e regras não é tão robusto.
- Lidar com o pequeno conjunto de dados a partir de transferência de aprendizado.

Interlúdio

Aprendizado de máquina

Algoritmos que aprendem a realizar tarefas a partir de experiência (dados).



Como ensinar uma máquina? I



Como ensinar uma máquina? II



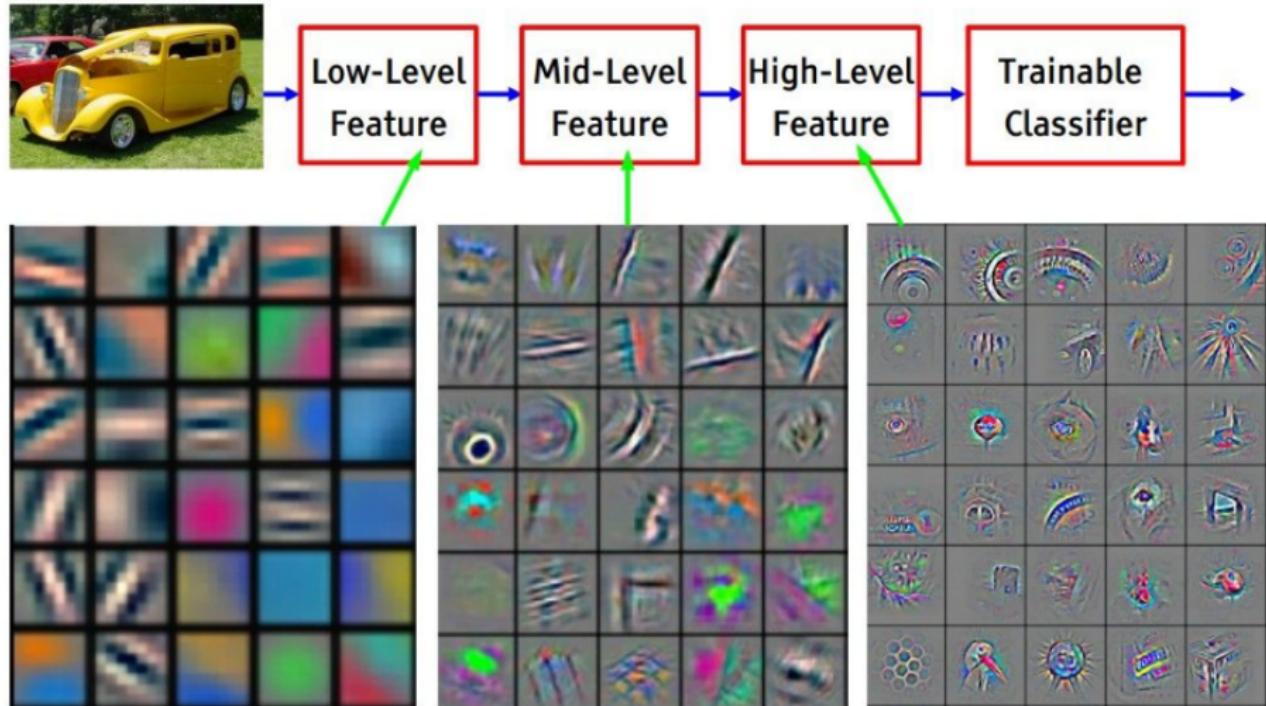
Como ensinar uma máquina? III



Como ensinar uma máquina? IV



Como ensinar uma máquina? V



1

Como ensinar uma máquina? VI

- Método baseados em regras para textos:
- Prós:
 - ▶ Dispensam anotação de dados.
- Contras:
 - ▶ Menos robustos (sensíveis a mudanças, menor sensibilidade.)
 - ▶ Podem se tornar complexos demais e de difícil manutenção (excesso de regras.)

Como ensinar uma máquina? VII

```
if re.compile('pdf').search(str(file_name)):
    if re.compile('Seç|Sec|seç|sec').search(str(file_name)) and ano < 2017:
        if soupLinkDownload2.find_all('b'):
            label_nome_arquivo = soupLinkDownload2.find_all('b')[0]
            label_nome_arquivo = label_nome_arquivo.text, label_nome_arquivo.next_sibling
            label_nome_arquivo = label_nome_arquivo[0]
        if label_nome_arquivo not in DODF_name:
            if label_nome_arquivo.startswith("--"):
                label_nome_arquivo = label_nome_arquivo.replace("--", "DODF ")
                label_nome_arquivo = label_nome_arquivo.replace("EDI", "DODF EDI")
                DODF_name = label_nome_arquivo
        else:
            DODF_name = label_nome_arquivo

    if re.compile('dodf|DODF[ -_]?[ -_]?[ -_]?\\d*').search(str(file_name)):
        subst = re.findall('dodf|DODF[ -_]?[ -_]?[ -_]?\\d*', file_name)[0]
        file_name = file_name.replace(subst, '')
        file_name_path = dir_mes + "/" + DODF_name + " " + file_name
    else:
        if soupLinkDownload2.find_all('b'):
            label_nome_arquivo = soupLinkDownload2.find_all('b')[0]
            label_nome_arquivo = label_nome_arquivo.text, label_nome_arquivo.next_sibling
            label_nome_arquivo = label_nome_arquivo[0]
        if label_nome_arquivo not in file_name:
            if label_nome_arquivo.startswith("--"):
                label_nome_arquivo = label_nome_arquivo.replace("--", "DODF ")
                label_nome_arquivo = label_nome_arquivo.replace("EDI", "DODF EDI")
                file_name = label_nome_arquivo + ".pdf"
        else:
            file_name = label_nome_arquivo + ".pdf"

    if re.compile('SECAO|SEÇÃO|SEÇAO|SECÃO').search(str(file_name)) and ano > 2016:
        file_name = file_name.replace('SECAO1', 'INTEGRA')
```

Os dados

Os dados

- 2.652 de textos extraídos do DODF.²
- Regras de regex para extrair:
 - ▶ data de publicação;
 - ▶ número de seção e título;
 - ▶ órgão de origem;
 - ▶ etc.
- 797 dos textos manualmente examinados: 724 deles livres de erros.

²<https://www.dodf.df.gov.br/>.

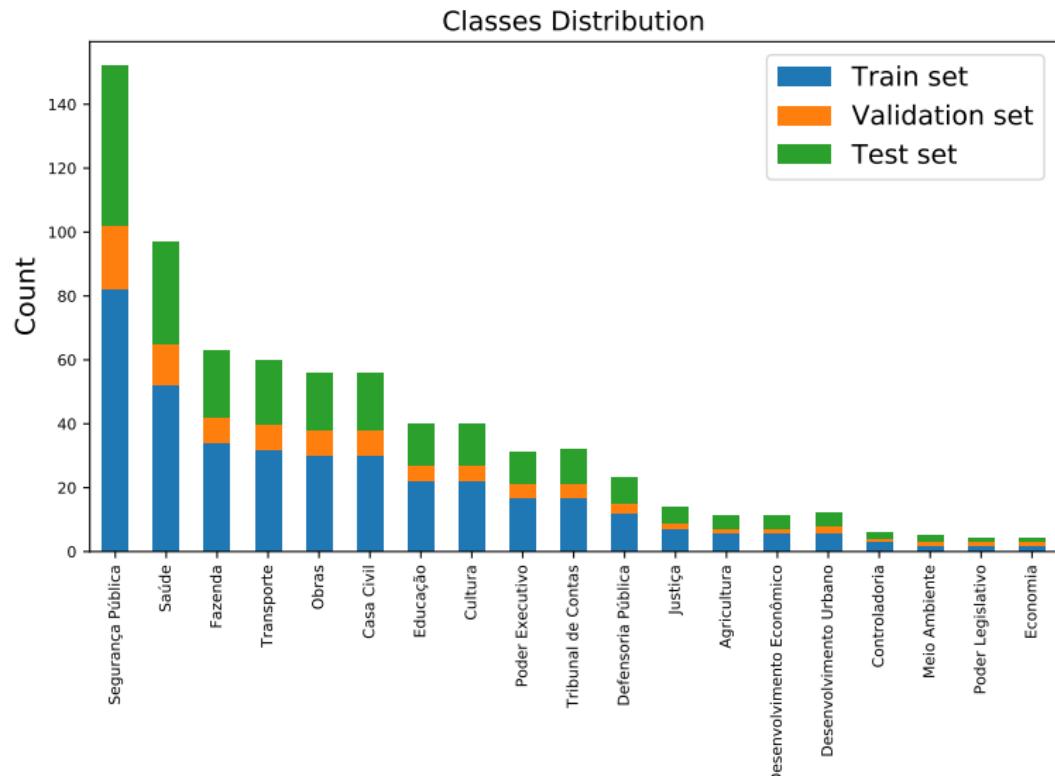
Os dados

- Documentos originados de 25 órgãos.
- Após retirar órgãos com menos de três entidades:
 - ▶ 717 textos rotulados;
 - ▶ 19 classes (órgãos que produziram o texto).
- Dividimos os dados em duas partes:
 - ▶ 717 exemplos rotulados para classificação.
 - ▶ 1.928 textos verificados ou rotulados incorretamente para treinamento de um modelo de linguagem.

Dados de classificação

- Seguintes subdivisões:
 - ▶ 384 (8/15) dos textos para treinamento;
 - ▶ 96 (2/15) dos textos para validação; e
 - ▶ 237 (5/15) dos textos para teste.
- Dados desbalanceados:
 - ▶ classe mais frequente (Segurança Pública) com 140 textos;
 - ▶ classes menos frequentes com menos de 5 documentos.

Distribuição dos dados



Dados do modelo de linguagem

- Tarefa padrão de modelagem de linguagem:
 - ▶ O rótulo de cada token é o próximo token do período.
 - ★ Fui para a padaria comprar ____.
- Dados de modelagem de linguagem:
 - ▶ Deletamos duas entradas com texto vazio, resultando em 1.926 documentos com 984.580 tokens.
 - ▶ Subdivisões:
 - ★ 784.260 tokens para treinamento (80%)
 - ★ 200.320 para validação (20%)

Métodos

Pré-processamento

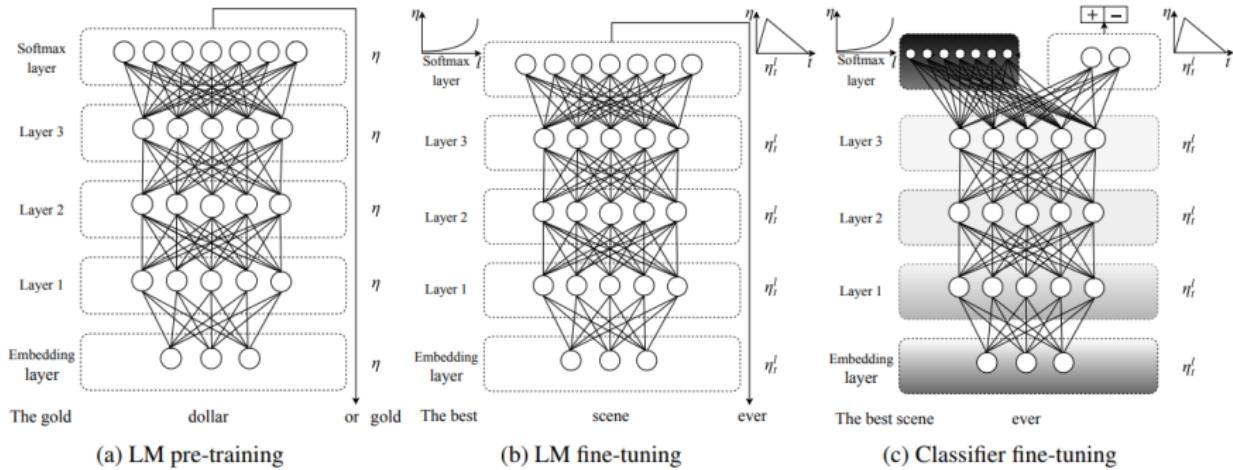
- Tokenização dos textos de entrada.
- Adição de tokens especiais que indicam: padding, capitalização da primeira letra, capitalização da palavra inteira, repetição de palavra ou caractere etc.³
- Vocabulário final de 8.552 tokens, incluindo palavras, partes de palavras, tokens especiais e pontuações.

³<https://docs.fast.ai/text.transform.html>

- Dois tipos de sacos-de-palavras:
 - ▶ valores tf-idf;
 - ▶ contagens de token.
- Dois classificadores:
 - ▶ Naïve Bayes (NB);
 - ▶ Support Vector Machine (SVM).

Transfer learning

Universal Language Model Fine-Tuning (ULMFiT) [Howard and Ruder, 2018]⁴



⁴ Modelo pré-treinado em 166.580 artigos da Wikipedia (100.255.322 tokens)
disponível em <https://github.com/piegu/language-models/tree/master/models> ↗ ↘

Resultados

Métricas

$$\text{Precisão} = \frac{\text{vp}}{\text{vp} + \text{fp}}$$

$$\text{Sensibilidade} = \frac{\text{vp}}{\text{vp} + \text{fn}}$$

$$\text{Escore F}_1 = 2 \times \frac{\text{Precisão} \times \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}}$$

$$\text{Acurácia} = \frac{\text{vp} + \text{vn}}{\text{vp} + \text{vn} + \text{fp} + \text{fn}}$$

Resultados I

Tabela: Escores F_1 (em %) das classes no conjunto de teste.

Classe	NB	SVM	F-ULMFiT	B-ULMFiT	F+B-ULMFiT	Quantidade
Casa Civil	66.67	78.95	80.00	82.35	88.24	18
Controladoria	80.00	80.00	100.00	100.00	100.00	2
Defensoria Pública	100.00	100.00	100.00	100.00	100.00	8
Poder Executivo	80.00	85.71	78.26	90.91	86.96	10
Poder Legislativo	66.67	100.00	66.67	66.67	100.00	1
Agricultura	50.00	66.67	57.14	50.00	57.14	4
Cultura	91.67	91.67	91.67	91.67	91.67	13
Desenv. Econômico	66.67	66.67	66.67	66.67	66.67	4
Desenv. Urbano	75.00	75.00	75.00	85.71	75.00	4
Economia	66.67	100.00	100.00	100.00	100.00	1
Educação	76.19	91.67	81.48	75.00	88.00	13
Fazenda	90.48	90.48	95.00	95.24	97.56	21
Justiça	75.00	66.67	60.00	66.67	66.67	5
Obras	88.24	90.91	88.24	76.92	85.71	18
Saúde	92.75	92.31	92.31	94.12	95.52	32
Segurança Pública	98.99	94.34	94.34	97.09	94.34	50
Transporte	94.74	97.56	92.31	92.31	97.56	20
Meio Ambiente	100.00	100.00	66.67	0.00	0.00	2
Tribunal de Contas	100.00	100.00	100.00	100.00	100.00	11
F_1 Médio	82.09	87.82	83.46	80.6	83.74	237
F_1 Ponderado	88.68	90.49	88.90	88.88	90.88	237
Acurácia	88.61	90.72	89.03	89.45	91.56	237

Resultados II

- Todos os modelos foram melhores do que simplesmente escolher a classe majoritária, que daria escores F_1 de 7.35% e 1.83% e uma acurácia de 21.10%.
- Os modelos SVM e ULMFiT foram os melhores em quase todas as categorias.
- Escores F_1 e acuráncias próximas de 90.00% indicam bons resultados, embora não tenhamos resultados obtidos por humanos para comparação.

Resultados III

- SVM e ULMFiT têm resultados comparáveis: o primeiro é melhor na média simples, enquanto o segundo é melhor na média ponderada e acurácia.
- SVM tem vantagens importantes:
 - ▶ Tempo: SVM treina em menos de dois segundos numa CPU, enquanto o ULMFiT exige mais de meia hora com uma placa de vídeo.
 - ▶ Simplicidade: o treino do SVM é simples, enquanto ULMFiT exige três passos com uma série de fatores a serem ajustados.
- Consequência: ULMFiT tem mais hiper-parâmetros a serem ajustados além de cada iteração ser cara—enquanto se treina um modelo ULMFiT, pode-se treinar mais de 1.000 modelos SVM com diferentes configurações.

Conclusão

Conclusão

- Apresentamos um corpo textual de textos do Diário Oficial do DF com anotação para classificação de órgão de origem.
- SVM foi comparável com um método estado-da-arte, ULMFiT.
 - ▶ Para a tarefa, ordem de palavra não parece ser tão importante quanto a presença de termos com forte correlação com classes específicas.
- Primeiro passo em direção ao uso de aprendizado de máquina para extração de informações no DODF.

Próximos passos

- Extração e rotulação de dados para aprendizado de outras tarefas:
 - ▶ Identificação e ligação de entidades.

Referências I

-  Howard, J. and Ruder, S. (2018).
Fine-tuned language models for text classification.
CoRR, abs/1801.06146.