UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO (EMC)
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

ANTONIO PIRES DE CASTRO JUNIOR

Aplicação da Inteligência Artificial, Ontologia e Mineração de Dados para Classificação de Sentenças Judiciais

GOIÂNIA,
20/12/2021

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[ ] Dissertação [X] Tese

2. Nome completo do autor

Antonio Pires de Castro Junior

3. Título do trabalho

“Aplicação da Inteligência Artificial, Ontologia e Mineração de Dados para Classificação de Sentenças Judiciais”

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Wesley Pacheco Calixto**, **Usuário Externo**, em 31/03/2022, às 15:17, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **ANTONIO PIRES DE CASTRO JUNIOR**, **Discente**, em 04/04/2022, às 15:19, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2799017** e o código CRC **21F6A9CF**.

ANTONIO PIRES DE CASTRO JUNIOR

Aplicação da Inteligência Artificial, Ontologia e Mineração de Dados para Classificação de Sentenças Judiciais

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação, da Escola de Engenharia Elétrica, Mecânica e de Computação, da Universidade Federal de Goiás (UFG), como requisito para obtenção do título de Doutor em Engenharia Elétrica e de Computação.
Área de concentração: Engenharia de Computação
Linha de pesquisa: Inteligência Artificial e Aprendizagem de Máquina

Orientador: Professor Doutor Wesley Pacheco Calixto

GOIÂNIA
20/12/2021

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Castro Junior, Antonio Pires de
Aplicação da Inteligência Artificial, Ontologia e Mineração de Dados para Classificação de Sentenças Judiciais [manuscrito] / Antonio Pires de Castro Junior. - 2021.
CLXV, 165 f.

Orientador: Prof. Wesley Pacheco Calixto.
Tese (Doutorado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2021.
Bibliografia. Apêndice.
Inclui siglas, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. inteligência artificial. 2. aprendizado por similaridade. 3. classificação de texto. 4. aprendizado de máquina. 5. gestão do conhecimento. I. Calixto, Wesley Pacheco, orient. II. Título.

CDU 62+004+005

UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE TESE

Ata Nº **14** da sessão de Defesa de Tese de **Antonio Pires de Castro Junior** que confere o título de Doutor em **Engenharia Elétrica e de Computação**, na área de concentração em **Engenharia de Computação.**

Aos **vinte dias do mês de dezembro de dois mil e vinte e um**, a partir das **16h30min.**, realizou-se a sessão pública de Defesa de Tese intitulada **"Aplicação da Inteligência Artificial, Ontologia e Mineração de Dados para Classificação de Sentenças Judiciais"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor **Wesley Pacheco Calixto (EMC/UFG)** com a participação dos demais membros da Banca Examinadora: Professor Doutor **Igor Santos Peretta - (ENG/UFU)**, membro titular externo; Professor Doutor **Wanderson Rainer Hilário de Araújo - (PUCGoiás/ELE)**, membro titular externo, Professor Doutor **Fabrizzio Alphonsus Alves de Melo Nunes Soares - (INF/UFG)**, membro titular externo, Professora Doutora **Viviane Margarida Gomes (COMP/IFG)** membro titular externo, Professor Doutor **Rodrigo Pinto Lemos - (EMC/UFG)**, membro titular interno: **cujas participações ocorreram através de videoconferência**. Durante a argüição os membros da banca **não fizeram** sugestão de alteração do título do **trabalho**. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor **Wesley Pacheco Calixto**, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos **vinte dias do mês de dezembro de dois mil e vinte e um**.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **WANDERSON RAINER HILÁRIO DE ARAÚJO**, **Usuário Externo**, em 31/03/2022, às 09:06, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **VIVIANE MARGARIDA GOMES**, **Usuário Externo**, em 31/03/2022, às 11:05, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Wesley Pacheco Calixto**, **Usuário Externo**, em 31/03/2022, às 15:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **ANTONIO PIRES DE CASTRO JUNIOR**, **Discente**, em 04/04/2022, às 15:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fabrízzio Alphonsus Alves De Melo Nunes Soares**, **Professor do Magistério Superior**, em 04/04/2022, às 15:24, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rodrigo Pinto Lemos**, **Professor do Magistério Superior**, em 05/04/2022, às 11:27, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2799000** e o código CRC **6633A74A**.

Referência: Processo nº 23070.066449/2021-45 SEI nº 2799000

"Todos os dias quando acordo
Não tenho mais o tempo que passou
Mas tenho muito tempo
Temos todo o tempo do mundo.

Todos os dias antes de dormir
Lembro e esqueço como foi o dia
Sempre em frente
Não temos tempo a perder.

Nosso suor sagrado
É bem mais belo que esse sangue amargo
E tão sério.

...

Não tenho medo do escuro
Mas deixe as luzes
Acesas agora.

..."

Renato Russo
em *Tempo Perdido, "Álbum - Dois", Legião Urbana*, 1986.

A minha amada esposa e meus dois alegres filhos com nomes de santos, Antônio e Pedro, aos meus pais queridos, as minhas bondosas irmãs, dedico este trabalho.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por tudo que me proporcionou e me proporciona nesta vida. Por me abençoar com a família, amigos, casa, comida, estudos, trabalho e várias outras coisas.

Minha amada esposa Layza: meu amor; minha companheira; amiga e confidente. Aos meus queridos filhos: Antônio Neto e Pedro, obrigado pela sua existência. Amo vocês!

Meus queridos pais: Antônio Pires e Leonira, pelo carinho, amor, criação, educação e ajuda em todos os aspectos na minha vida. Vocês são exemplos para mim.

As minhas bondosas e leais irmãs, aos meus cunhados e sobrinhos. Simbolos de união e amizade.

Ao meu orientador e amigo, Wesley Pacheco Calixto, pelos ensinamentos, tanto acadêmicos quanto de caráter. Levo comigo a lembrança de duas principais características: sua humildade e paciência. Obrigado pelos ensinamentos, orientações e dedicação. Suas palavras de incentivo e motivação foram o combustível necessário para chegar até aqui.

Agradeço o nosso grupo de trabalho em ontologia no NEXT, pelas excelentes reuniões e contribuições com este trabalho: Viviane; Lais (*in memoriam*); Ernesto e Pedro Maionne.

Aos meus amigos e profissionais do Tribunal de Justiça do Estado de Goiás, da Faculdade de Tecnologia do SENAI/FATESG, do grupo de pesquisa do NEXT e da UFG/EMC, que de alguma forma ou outra ajudaram no desenvolvimento desta obra.

Agradeço o apoio financeiro da Faculdade de Tecnologia do SENAI/FATESG, da UFG/EMC e do Tribunal de Justiça do Estado de Goiás, pela ajuda nos custos para apresentação dos trabalhos científicos relacionados com esta Tese.

RESUMO

O propósito deste trabalho é aplicar de forma conjunta a ontologia com modelos de saco-de-palavras, aprendizagem por similaridade e classificação de documentos em textos com decisões proferidas. O intuito é melhorar os resultados da mineração de dados em banco de dados de decisões judiciais. Desenvolve-se método automático para buscar sentenças em processos judiciais correlatos ao que está em julgamento, utilizando o modelo de frequência do termo-inverso da frequência nos documentos juntamente com o coeficiente de similaridade de Jaccard, estabelecendo pesos na coocorrência de termos em textos jurídicos da mesma categoria. Utiliza-se conjunto de dados com a vetorização dos documentos para treinamento supervisionado de algoritmos de aprendizado de máquina, objetivando classificar novos processos da justiça. O método apresentado proporciona agilidade no Poder Judiciário, agregando o trabalho dos assessores jurídicos na preparação das decisões judiciais com menor tempo e mais eficiência na busca por padrões de jurisprudência. Os resultados obtidos apresentam que, por meio das métricas de acurácia, o modelo proposto é eficaz e eficiente, podendo ser aplicado no processo de identificação das decisões judiciais. Desta forma, a aplicação da inteligência artificial, ontologia e mineração de dados é indicada para recuperação de informações em decisões judiciais.

Palavras-chave: inteligência artificial, aprendizado por similaridade, classificação de texto, aprendizado de máquina, gestão do conhecimento.

APPLICATION OF ARTIFICIAL INTELLIGENCE, ONTOLOGY AND DATA MINING FOR CLASSIFICATION OF JUDICIAL SENTENCES

ABSTRACT

The objective of this work is to apply together ontology with bag-of-words models, similarity learning, and document classification in texts with uttered decisions. The objective is to improve the results of data mining in a database of court decisions. An automatic method of searching sentences in judicial processes related to the one under judgment is developed using the frequency term-inverse of frequency in documents model together with the Jaccard similarity coefficient, establishing weights on the co-occurrence of terms in legal texts of the same category. A dataset with document vectorization is used for supervised training of machine learning algorithms, aiming to classify new justice processes. The proposed methodology provides flexibility to the Judiciary, simulating the role of legal advisors in preparing court decisions with less time and efficiency in the search for jurisprudential standards. The results obtained show that, through accuracy metrics, the proposed model is effective and efficient, and can be applied in the process of identification of court decisions. Thus, the application of artificial intelligence, ontology, and data mining is indicated for information retrieval in court decisions.

Keywords: artificial intelligence, similarity learning, text classification, machine learning, knowledge management.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SÍMBOLOS

t_f	–	Termo Frequência
i_{df}	–	Inverso Documento Frequência
$tf - idf$	–	Termo Frequência - Inverso Documento Frequência
D	–	Documentos
t_i	–	Determinado termo i no vetor
D_m	–	Cluster de Documentos D, com m documentos no vetor
$w_{(j,i)}$	–	Peso de determinado termo na matriz termo $\times$ termo
p_{t_i}	–	Peso do termo t_i
W_{t_i}	–	Somatório dos p_{t_i}
S_m	–	Similaridade
S_{mc}	–	Similaridade pelo cálculo do cosseno
S_{mj}	–	Similaridade pelo cálculo de Jaccard
S_α	–	Expressão alterada de Jaccard
C_p	–	Corretamente positivas
F_p	–	Falso positivas
T_p	–	Verdadeiramente positivas
C_n	–	Corretamente negativas
F_n	–	Falso negativo
P	–	Precisão
R	–	Revocação
A	–	Acurácia
F	–	Medida-f
ϕ	–	Aplica limite no coeficiente de similaridade na coocorrência dos termos

LISTA DE ABREVIATURAS E SIGLAS

SDM	–	Sistema de Decisões Monocráticas
SPG	–	Sistema de Primeiro Grau
SSG	–	Sistma de Segundo Grau
CNJ	–	Conselho Nacional de Justiça
PROJUDI	–	Sistema de Processo Judicial Eletrônico
e-DOC	–	Sistema Processual Eletrônico da Justiça do Trabalho
PJ-e	–	Sistema de Processo Judicial Eletrônico do CNJ
RC	–	Representação do Conhecimento
IA	–	Inteligência Artificial
RI	–	Recuperação da Informação
CI	–	Ciência da Informação
SMART	–	*System for the Manipulation and Retrieval of Text*
FTP	–	*File Transfer Protocol*
DSI	–	Disseminação Seletiva da Informação
IoT	–	*Internet of Things*
GPS	–	*Global Positioning System*
OWL	–	*Ontology Web Language*
RDF	–	*Resource Description Framework*
RDFS	–	*Resource Description Framework Schema*
XML	–	*Extensible Markup Language*
MVC	–	*Model-View-Controller*
TJGO	–	Tribunal de Justiça do Estado de Goiás
TRT	–	Tribunal Regional do Trabalho
TRE	–	Tribunal Regional Eleitoral
STF	–	Supremo Tribunal Federal
STJ	–	Superior Tribunal de Justiça
SQL	–	*Structured Query Language*
SDLC	–	*Syncronous Data Link Control*
xDSL	–	*Types of Digital Subscriber Line*
UML	–	*Unified Modeling Language*
UNL	–	*Universal Networking Language*
PLN	–	*Processamento de Linguagem Natural*
PULO	–	*Portuguese Unified Lexical Ontology*
BM25	–	*Okapi Best Matching 25*
PLN	–	*Processamento de Linguagem Natural*
WSDL	–	*Web Services Description Languagel*
LDA	–	*Latent Dirichlet Allocation*
MLP	–	*Redes Neurais Perceptron Multicamadas*
FA	–	*Floresta Aleatória*
RG	–	*Reforço de Gradiente*
ImA	–	*Impulso Adaptativo*

PG – *Processo Gaussiano*
MVS – *Máquina de Vetores de Suporte*
NB – *Naive Bayes*
KVP – *k-vizinhos mais próximos*
AD – *Árvores de Decisão*
ABH – *Algoritmo de Busca Harmônica*
AG – *Algoritmo Genético*
PSO – *Enxames de Partículas*
ACO – *Colônia de Formigas*
RK – *Rebanho de Krill*
CS – *Busca do Voo do Cuco*
LC – *Otimizador por Lobo Cinzento*
BA – *Ecolocalização dos Morcegos*
KM – *k-médias*

CAPÍTULO 1

INTRODUÇÃO

O volume de ações protocoladas no Poder Judiciário, frente a força tarefa dos servidores e magistrados, bem como as possibilidades e fluxos possíveis impostos na legislação brasileira tem prejudicado o rápido atendimento aos direitos da sociedade. Assim, visto que os recursos para julgar a quantidade de processsos que ingressam na justiça são insuficientes, acredita-se que técnicas de otimização em conjunto com a tecnologia da informação podem ser aplicadas para melhorar este cenário.

Magalhães (1998) insere na Emenda Constitucional nº 19, de 04 de junho de 1998, a eficiência como princípio da administração pública, objetivando garantir maior qualidade, dinamicidade e celeridade na prestação dos serviços públicos, como os ofertados pelo Judiciário, visando, principalmente, satisfazer os anseios da coletividade (ROVER, 2005).

Ambrosio (2007) informa que uma das principais inovações na área computacional nos últimos anos é a web semântica, com a construção de rede em contexto semântico que possibilite a significação (semântica) e tratamento automático de informações. O desfecho do processo judicial ocorre após a decisão do juiz, informando na sentença se a parte litigante (solicitante) tem razão no pleito ou não, conforme documentos/evidências descritos no processo, bem como a forma que deve ser estabelecida na resolução do problema. Se a lei estabelecida nos códigos fosse suficiente, perfeita, imutável, anespacial e atemporal, bastaria localizar qual norma se encaixaria a cada caso e aplicá-la, o que tornaria o servidor da justiça um simples autômato, programado para tal tarefa. Entretanto, os magistrados são mais que isto, valem-se do raciocínio, do coração, da bagagem cultural e de vida, da moral, dos costumes e de decisões já proferidas em casos correlatos. O juiz, portanto, deve analisar o caso com a ótica do valor, do fato e da norma (NETTO, 2011).

Considerar banco de dados de sentenças já proferidas é importante no sentido de conhecer o entendimento da maioria dos magistrados sobre determinado assunto. Em alguns países, como os Estados Unidos da América, este banco de dados é vital para o sucesso da ação judicial. Nestes países o banco de dados das sentenças forma a base de entendimento pacificado sobre assuntos correlatos nos processos judiciais, atribuindo maior importância com o estudo e aplicação por analogia de

casos passados. Já em outros países, como o Brasil, de tradição romano-germânica, é ensinado que a lei nos códigos é a principal fonte do Direito, deixando às demais fontes como secundárias no caso da ausência de norma decorrente da lei. Entretanto, na prática, percebe-se que cada vez mais são aplicadas as sentenças já proferidas como referências na composição da decisão do processo em julgamento.

Por considerar que a busca em decisões já proferidas é etapa essencial no fluxo processual e que a maioria das ferramentas de busca são substancialmente textuais, localizadas nos sítios dos tribunais de justiça, fica claro que o trabalho humano realizado neste procedimento é moroso, árduo e pouco produtivo (SILVA, 2009). O trabalho de estudar o processo e pesquisar por decisões similares é realizado pelos assessores de magistrados.

Esses assessores Judiciários, responsáveis pela busca na internet, consomem tempo significativo no trabalho de estudar o processo e encontrar, por meio de buscas textuais, decisões semelhantes ao caso em julgamento. A busca textual depende, ainda, da interpretação da pessoa, bem como da estruturação do banco de dados disponibilizado, podendo resultar em problemas na precisão da informação e na qualidade da recuperação, influenciando diretamente na minuta da decisão para estudo pelo juiz (SILVA, 2009).

Segundo Castro (2014), a gestão do conhecimento por meio de software é ferramenta importante para o processo decisório em várias áreas do conhecimento. O autor apresenta solução eletrônica para gestão de informações processuais, implementada na Corregedoria-Geral da Justiça, no Tribunal de Justiça do estado de Goiás. A solução gera alertas dos processos judiciais paralisados para o corregedor geral realizar inspeções eletrônicas remotas nas unidades judiciais.

Com foco na pesquisa e recuperação da informação, Sewald (2011) apresenta a ontologia como método promissor no Judiciário, aplicada na gestão do conhecimento com objetivo de modelar, guardar e apresentar base de dados. Apesar do trabalho não gerar dados conclusivos, pois apresenta-se como ensaio inicial, o autor afirma que a gestão baseada em conhecimento aumenta a eficiência na recuperação e manutenção da informação, facilitando o processo decisório no Judiciário.

Sewald e Rover (2012) demostram que a gestão do conhecimento fornece técnicas e ferramentas para o Judiciário melhorar sua eficiência. Neste sentido, os autores

realizam trabalho junto ao Tribunal de Justiça do Amazonas e apontam a falta de reutilização do conhecimento aplicado no processo decisório dos magistrados, não ocorrendo a recuperação da informações. Os autores desenvolvem o trabalho apresentando diagramas em Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) para representar a forma com a qual as decisões judiciais são pesquisadas pelos magistrados e seus assistentes. Apesar de não trabalharem modelos de recuperação da informação, como modelos quantitativos e/ou modelos dinâmicos, Sewald e Rover informam que a recuperação da informação usando ontologia pode ser a solução para melhoraria da eficiência no Judiciário.

Gomes (2009) discute a contextualização das bases de dados e o futuro das ferramentas de mineração de dados, atribuindo a melhoria na contextualização à ontologia. O trabalho de Gomes faz referência a três exemplos de sistemas de recuperação do conhecimento, que facilitam o processamento automático de informações contidas em documentos e permite que computadores consigam recuperar dados de forma mais eficiente, estas ferramentas são: i) *Universal Networking Language* (UNL), ii) *Semantic web* e iii) *WordNet*. As soluções UNL e *WordNet* focam no relacionamento das palavras contidas nos documentos e são fundamentais a aplicação da semântica nas ferramentas de busca das informações.

Silveira (2003) trabalha com Tesauros, que são ferramentas que realizam o controle e padronização para indexação dos documentos em banco de dados. O autor trata a recuperação da informação em estudo de caso na área jurídica. O trabalho combina os dados de entrada do usuário com a indexação do Tesauros, sendo adotado para o motor de busca as redes Bayesianas. Como resultado tem-se os documentos ordenados por relevância onde o autor avalia o sistema de recuperação construído utilizando a base de dados da jurisprudência dos tribunais federais brasileiros e o Tesauros da justiça federal. Os resultados das consultas indicam ganhos médios em precisão à ordem de 31%.

Ordonez (2015) utiliza a ontologia na área jurídica, sendo construída relação semântica da base de decisões judiciais proferidas. Ordonez apresenta o progresso no desenvolvimento do sistema utilizando tecnologia de processamento de linguagem natural, bem como agrupamento (*clustering*) para otimizar o processo de recuperação e análise das bases de decisões judiciais.

Pu (2015) discute as dificuldades encontradas na construção da ontologia jurídica

chinesa. O autor afirma que os desafios não são apenas na deficiência das técnicas de ontologia, mas também na organização dos dados do sistema jurídico chinês. Para esse sistema, o autor levanta três obstáculos: i) ambiguidade da linguagem jurídica, ii) deficiência na inferência das decisões judiciais e iii) papel limitado dos casos na China. Com base nestas dificuldades, a ontologia jurídica proposta é a mistura entre os documentos normativos e os casos jurídicos.

Gangemi (2016) desenvolve trabalho utilizando a web semântica, sendo usada a biblioteca de conhecimentos jurídicos que baseia-se nos metadados contidos em documentos judiciais, onde os relacionamentos semânticos são construídos pela extração de fragmentos em textos jurídicos. A biblioteca gerada em OWL2, que é a segunda versão da linguagem de ontologia na web (*Ontology Web Language* - OWL), é chamada de JudO e apresenta as interpretações dos juízes na condução do seu raciocínio jurídico. Gangemi é da área de ciência da computação, sendo que sua contribuição está na modelagem do conhecimento jurídico, resultante dos estudos baseado em casos e padrões de projeto na ontologia.

Milosavljevic (2017) desenvolve componente semântico central para sistema de gerenciamento de documentos orientado por ontologia. No estudo de caso é utilizado o domínio judicial com o intuito de melhorar a eficiência por meio da introdução semântica. O autor não realiza recuperação da informação em bases jurisprudenciais e/ou decisões já proferidas.

Gao (2017) dá continuidade ao trabalho de Pu (2015), criando sistema de recuperação da informação para a base de precedentes jurídicos e estatutos. O sistema permite realizar a recuperação dos julgamentos e das normas por relevância. O autor informa que a ontologia suporta o raciocínio lógico e incorpora estrutura hierárquica. Além da contribuição ontológica gerada, Gao afirma que consegue melhorar a métrica de precisão na recuperação da informação utilizando algoritmos genéticos e abordagem do vizinho mais próximo, no motor de busca.

Sulis et al. (2021) trabalham na identificação e classificação de documentos judiciais em texto da União Europeia, utilizando simultaneamente dois métodos de geração de recursos, sendo: i) anotação manual de oito categorias estabelecidas pelos autores e ii) rede de coocorrência entre termos. Várias métricas são aplicadas na rede de coocorrência para estabelecer valores na combinação de termos. Após a geração das características por este método manual e automático, os algoritmos de classificação:

regressão logística, árvores de decisão, máquina de vetores de suporte (*Support Vectors Machine* – SVM) e k-vizinhos mais próximos são aplicados. Nos resultados, o melhor classificador é obtido com a aplicação da SVM, atingindo o valor de 83% na medida-f e 76% na precisão.

Mandal et al. (2021) trabalham com textos judiciais conhecidos como precedentes, tratam-se de casos conhecidos e consolidados no Judiciário. O conjunto de dados usado são os casos da Suprema Corte Indiana. Mandal et al. (2021) investigam o desempenho de 56 metodologias diferentes para calcular similaridade textual em precedentes. São utilizados vários modelos de saco-de-palavras e de saco-de-conceitos, como a frequência do termo-inverso da frequência nos documentos (*term frequency-inverse document frequency* – $tf - idf$). Os autores observaram que métodos tradicionais que dependem de representação de saco-de-palavras, têm melhor desempenho que métodos sensíveis ao contexto (CHALKIDIS, 2021). A precisão média obtida com métodos utilizando saco-de-palavras é de $\approx 75\%$, enquanto a média utilizando métodos sensíveis ao contexto, como saco-de-conceitos é de $\approx 70\%$.

Radygin et al. (2021) desenvolvem software para buscar e analisar documentos de tribunais arbitrais de acordo com a Legislação Federal da Rússia. A proposta é usar inteligência artificial (IA) para detectar violações da Lei Federal russa. No fluxo de preparação do documento, antes da construção dos pesos, os textos são pré-processados. Para vetorização, $tf - idf$ é utilizado e o algoritmo de classificação SVM. Os documentos de treinamento são obtidos de repositório aberto da internet russa, chamado Justiça Eletrônica. O artigo não detalha os documentos ou como os documentos são selecionados para realização dos testes de classificação, mas informa que os resultados obtidos são aproximadamente de 87% na medida-f das simulações.

Hausladen et al. (2020) aplicam aprendizado de máquina em documentos de Tribunais nos Estados Unidos da America (EUA), com o intuito de explorar e avaliar classificadores para prever decisões conservadoras ou liberais. No EUA, os juízes exercem poder significativo devido ao sistema de *common law*, que significa que priorizam decisões em casos já julgados, que normas constituídas na legislação. Na vetorização dos documentos, são utilizados os modelos $tf-idf$ unigrama e n-gramas. Os classificadores usados no trabalho obtiveram aproximadamente o valor de 74,5% na medida-f e 77,1% na acurácia, no conjunto de dados denominado Atividade Econômica.

Waltl et al. (2018) trabalham na classificação manual e automática de normas no campo do direito civil alemão, consistindo em nove categorias/rótulos de documentos de texto. O modelo utilizado na vetorização é o saco-de-palavras em conjunto com o $tf - idf$. A metodologia proposta para classificação manual e automática é comparada usando as métricas medida-f, precisão e revocação. Cinco algoritmos de classificação são usados nas simulações e os melhores valores são encontrados na classificação automática utilizando o classificador SVM, sendo 85% na precisão, 84% em revocação e 83% na medida-f.

Katz et al. (2017) atuam no âmbito da Suprema Corte de Justiça do EUA, com o intuito de prever seu comportamento. É desenvolvida evolução temporal utilizando o classificador floresta randômica, bem como solução baseada em metadados cadastrados no banco de dados do Supremo Tribunal. São utilizados conjunto de dados de 240.000 votos em tribunais e 28.000 resultados de casos ao longo de quase dois séculos. Os resultados deste volume de dados alcançam valores de precisão de 70,2% na tentativa de prever o resultado dos casos e 71,9% na previsão de votos.

Medvedeva et al. (2020) apresentam método para prever decisões de casos futuros com base em casos anteriores, trabalha com julgamentos e documentos de texto do Tribunal Europeu de Direitos Humanos. Os autores informam que com a disponibilidade de julgamentos, grande volume de dados sobre a justiça e a jurisprudência, começam a ser possível realizar estudos na justiça, aplicando soluções de aprendizado de máquina. Para a vetorização de documentos, o modelo $tf - idf$ é aplicado com o método n-gramas. O classificador SVM é usado para prever apenas duas categorias de resultados: violação dos direitos humanos e não violação dos direitos humanos. A precisão alcançada é de 75%.

Observa-se nos trabalhos relacionados a tendência em aplicar modelos de recuperação de informação e inteligência artificial na área do Direito, objetivando a vetorização dos termos, classificação dos textos jurídicos e a predição de julgamentos para processos judiciais em tramitação. Observa-se ainda a tendência em aplicar técnicas relacionadas a: i) ontologia no domínio jurídico para conceituar termos e ii) modelos de saco-de-palavras para estabelecer pesos nos termos dos textos, possibilitando a vetorização dos documentos e aplicação de algoritmos de aprendizado de máquina.

Os estudos e as pesquisas realizadas neste trabalho referem-se à aplicação no *corpus* de documentos reais de ações judiciais, na instituição governamental de justiça do

Brasil, Estado de Goiás. Os dados utilizados nesta pesquisa são documentos de julgamento de juízes, conhecidos como decisões. Estes documentos descrevem o evento ocorrido, abrangendo todos os aspectos do direito inerentes ao caso e as solicitações dos advogados. São documentos com informações, integrando fatos gerados e leis aplicadas. Estas decisões moldam a forma como a lei é interpretada e aplicada pelo Poder Judiciário, bem como as definições tratadas das divergências sociais. (CASTRO et al., 2020)(CASTRO et al., 2017a)(CASTRO et al., 2017b)(JASANOFF, 2018).

Este trabalho aborda três principais temas na tentativa de auxiliar no fechamento das **lacunas**: i) relacionar processos em julgamento com decisões proferidas, por meio de soluções de inteligência artificial no Judiciário Goiano, ii) julgamentos divergentes ocorrem para ações judiciais com histórias e fatos semelhantes, o que pode levar a desigualdades e até mesmo possíveis discriminações e iii) limitações na aplicação dos modelos de saco-de-palavras, em atenção ao $tf - idf$, por não ser capaz de estabelecer valores na coocorrência de termos no *corpus*.

Portanto, tem-se como **hipótese principal** deste trabalho que: se é possível aplicar conjuntamente a ontologia com modelos de saco-de-palavras, aprendizagem por similaridade e classificação de documentos, em textos com decisões proferidas, logo pode-se modelar solução de IA para recuperar decisões proferidas que guardam relação ao processo judicial que está em julgamento, acelerando a etapa de construção da sentença do processo judicial.

O **objetivo principal** desta pesquisa é proporcionar celeridade nos julgamentos dos processos judiciais, bem como reduzir as desigualdades e a discriminação, uma vez que identifica e classifica os processos recebidos, permitindo a conexão com os casos julgados. Como **objetivos específicos** têm-se: i) avaliar as diferentes técnicas de treinamento, binárias e de frequência, bem como nove algoritmos de classificação que apresentam as melhores métricas de acurácia, medida-f, precisão e revocação para os documentos de texto não estruturados do tribunal, ii) comparar e avaliar o método proposto com outros trabalhos de pesquisa, aplicando outros conjuntos de dados, iii) comparar e avaliar a aplicação do modelo *tf-idf* e *BM25*, juntamente com a similaridade de Jaccard, iv) construir e disponibilizar o código-fonte no Github para realizar o preprocessamento em documentos em texto da justiça (em Português) e v) implementar nove algoritmos de classificação que serão disponibilizados no Github, para a evolução das pesquisas atuais.

O número de processos no Judiciário brasileiro tem aumentado a cada ano e superlotado as unidades judiciais, prejudicando o rápido atendimento aos direitos da sociedade (SEWALD, 2011). Com o intuito de transformar esta realidade, faz-se necessário inserir ferramentas tecnológicas que apoiem os trabalhos realizados pelos serviços forenses, promovendo celeridade, efetividade, eficiência e eficácia dos trabalhos desenvolvidos (CASTRO, 2014). Estas ferramentas tecnológicas de apoio em consonância com a gestão estratégica, aperfeiçoam a gestão do conhecimento no Judiciário. O Conselho Nacional de Justiça estabelece que a Justiça desenvolva o Mapa Estratégico para ciclo de cinco anos. Este Mapa encontra-se disponível para acesso no portal do Tribunal de Justiça do estado de Goiás. No Mapa, a meta que se relaciona com este trabalho é a Número 9, sendo abordada em processo administrativo no sistema Proad, com número 127942 (MARQUES FILHO, 2017).

As **originalidades** inerentes a esta pesquisa aplicada e não encontradas em outros trabalhos são: i) no contexto acadêmico, o modelo proposto aperfeiçoa o tradicional saco-de-palavras com a aplicação conjunta de técnica de aprendizagem por similaridade para encontrar, de forma automática, a coocorrência dos termos, possibilitando vetorizar os documentos em texto em apenas uma dimensão, ii) este trabalho desenvolve e disponibiliza, em repositório aberto Github, solução de preprocessamento de texto jurídico em língua Portuguesa e iii) no contexto social no Brasil, país com sérios problemas como a pobreza e a discriminação, este trabalho propõe ferramenta que ajuda na padronização de julgamentos, pois o modelo proposto consegue relacionar casos novos com processos semelhantes julgados.

Os resultados obtidos neste trabalho atendem duas metas sustentáveis no mundo, de acordo com as dezessete metas estabelecidas pelas Nações Unidas, sendo os indicadores: i) 10.3 que garante a igualdade de oportunidades e redução das desigualdades de resultados, incluindo eliminação de leis, políticas e práticas e promoção de legislação, políticas e ações adequadas a este respeito e ii) 16.6 promoção do desenvolvimento de instituições eficazes, responsáveis e transparentes em todos os níveis.

Este trabalho está estruturado de forma a abordar a teoria envolvida no desenvolvimento do método proposto. O Capítulo 2 descreve os dados do Judiciário brasileiro e a história do Judiciário em Goiás, expondo, ainda, a sua estrutura. O Capítulo 3 explica os conceitos e aplicações da ontologia, principalmente em ciência da informação, inclui, ainda, o sentido filosófico da ontologia. No Capítulo 4 são descritos os conceitos, aplicações e os modelos utilizados pela inteligência artificial, principalmente

na ciência da informação e na ciência da computação. Após estes levantamentos literários, no Capítulo 5 é descrita toda a metodologia aplicada no desenvolvimento do método proposto e após, no Capítulo 6 são apresentados os resultados obtidos, no Capítulo 7 as conclusões e sugestões para trabalhos futuros.

CAPÍTULO 2

HISTÓRICO E DADOS DO JUDICIÁRIO GOIANO

Neste capítulo são apresentados de forma sucinta os dados do Judiciário brasileiro, a história e a estrutura do Judiciário no Estado de Goiás. São informações necessárias para apresentar o Judiciário ao leitor e contextualizar sua ligação com a sociedade. Ainda neste capítulo, são abordados de forma resumida o planejamento estratégico institucional, a gestão da informação para a representação do conhecimento e a classificação desta informação.

2.1 Judiciário

Guimarães (1988), diz que o Legislativo, Executivo e Judiciário são poderes da união, independentes e harmônicos. Assim, o autor confere autonomia institucional, administrativa e financeira ao Poder Judiciário, assegurando independência funcional aos magistrados. O pioneiro nesta corrente tripartite (governo em três) foi Aristóteles, pois contemplava a existência de três órgãos separados a quem cabiam as decisões de Estado: i) Poder Deliberativo, ii) Poder Executivo e iii) Poder Judiciário (ARISTÓTELES, 2001).

De acordo com Guimarães (1988) cabe ao Poder Judiciário a função jurisdicional, que consiste na aplicação da lei no caso concreto, que lhe é apresentado como resultado do conflito de interesses. O autor assegura que nenhum poder irá sobrepor-se a outro, trazendo independência harmônica nas relações de governança.

No Judiciário, o número de processos que ingressam no Tribunal de Justiça vem aumentando a cada ano, sendo que este aumento não é possível de ser acompanhado pela realização do trabalho executado dos novos magistrados e servidores (VIEIRA, 2006). O Judiciário tem sido alvo de análises e críticas devido à incapacidade que tem demostrado em atender às crescentes demandas sociais à justiça. A falta de efetividade na distribuição da justiça tem provocado o que constitui o principal aspecto do que se convencionou denominar de Crise da Justiça ou Crise do Judiciário (TAVARES, 2003). Do ponto de vista dos integrantes do sistema, o número de contratações de servidores não acompanha a demanda gerada pelo aumento do número de processos em tramitação.

2.2 Dados do Judiciário Goiano

De acordo com a justiça em números, que é o projeto do Conselho Nacional de Justiça (CNJ), o Poder Judiciário nacional finalizou o ano de 2020 com mais de 75 milhões de processos em tramitação, Figura 2.1, adaptada de Fux (2021). Mesmo tendo arquivado mais processos que o quantitativo ingressado (índice de atendimento à demanda de 108,15%), nos quais tais resultados são basicamente o reflexo direto da Justiça Estadual, que abarca 77,4% dos processos pendentes, Figura 2.2, adaptada de Fux (2021). Os casos pendentes, por sua vez, são todos aqueles que nunca receberam movimento de baixa, em cada uma das fases analisadas. (FUX, 2021)

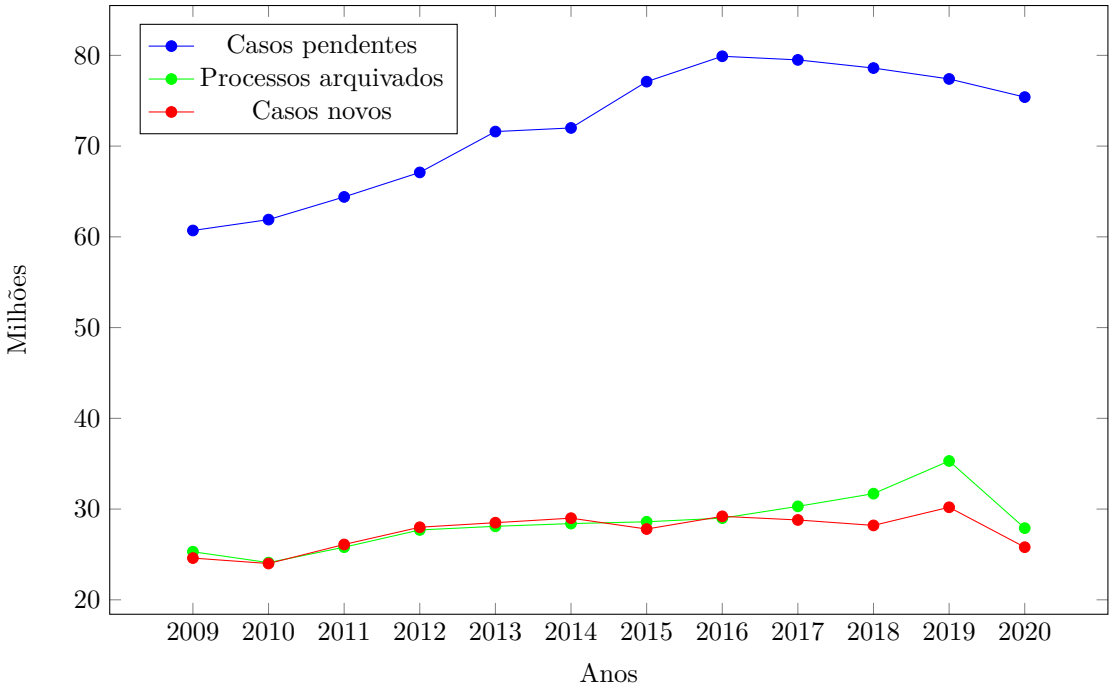

Figura 2.1 - Série histórica da movimentação processual do Poder Judiciário.

Observa-se na Figura 2.1 que o Judiciário está conseguindo arquivar aproximadamente a quantidade de casos que chegam, ou seja, os casos novos, porém o passivo (pendentes) continua com a mesma dimensão. No contexto do estado de Goiás, atualmente têm-se quase dois milhões de processos em tramitação no primeiro grau de jurisdição e, em torno de 370 magistrados (no primeiro grau). Como o Judiciário está conseguindo julgar e arquivar a quantidade de casos novos e não houve incremento

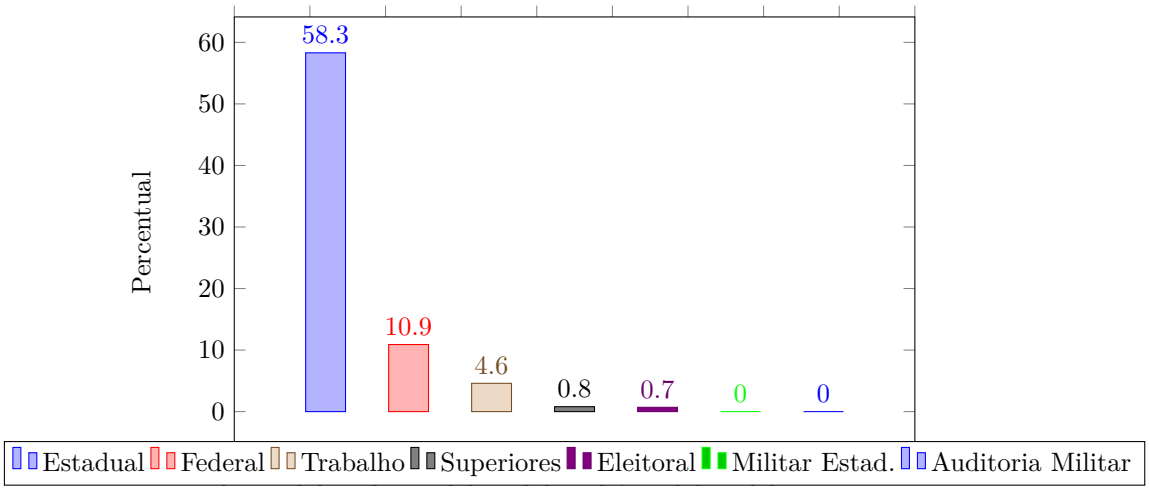

Figura 2.2 - Casos pendentes do Poder Judiciário, por ramos da justiça.

do número de juizes, têm-se média anual de 5.880 ações por magistrados, sendo que existem apenas três assessores para cada magistrado.

2.3 História do Judiciário em Goiás

Desde a época da colonização do Brasil, a organização da justiça foi uma das principais preocupações da Coroa. Os ouvidores receberam a competência para conhecer problemas de ordens cíveis e criminais nas capitanias. Houve também, a criação dos cargos de ouvidor-geral, no governo central da colônia, sendo responsáveis pelos julgamentos dos recursos dos ouvidores.

Já na capitania de *Goyaz*, em 1728, Bartolomeu Bueno da Silva Filho foi nomeado superintendente-geral, sendo o primeiro a exercer atividades judicantes (exerce as funções de juiz) neste estado. Houve ainda outra personalidade, João Leite da Silva Ortiz, que na mesma data foi nomeado o guarda-mor. Assim, constituem-se as primeiras pessoas com capacidade de decisão e julgamento de primeira instância em Goiás. Os casos julgados por eles, em grau de recursos foram analisados no único Tribunal de segunda instância no Brasil, situado na Bahia, chamado Tribunal da Relação. Em 1751, na cidade de São Sebastião do Rio de Janeiro, foi constituído o Tribunal da segunda Relação, visto o aumento dos recursos e o tamanho do Brasil. A representação dos casos de parte do centro e todo o sul do Brasil, ficava a cargo deste segundo Tribunal, incluindo a comarca de Goiás (ARAUJO E. FARIA, 2000).

Em 1809 foi criada a segunda comarca em Goiás, sendo que foi o rei que paga os salários dos juízes, homens de carreira, atuando na primeira instância, conhecidos como juízes ordinários. Na época, a segunda instância começava a ser exercida na capital da comarca, a cargo dos ouvidores. Poucas pessoas tentavam recursos nos processos, pois além de caros, os resultados eram duvidosos (ARAUJO E. FARIA, 2000).

Em 1824, com a independência, entrava em vigor a Constituição Imperial, o Supremo Tribunal de Justiça no Brasil foi criado, objetivando julgar as causas em segunda e terceira instâncias. Mas, o Tribunal da Relação foi instituído em Goiás somente após o decreto nº 2.342, em 1873. O Tribunal da Relação foi o atual Tribunal de Justiça do estado de Goiás. A primeira sessão foi realizada em 5 de maio em 1874, mesma data em que foi convocado magistrados para substituírem desembargadores. Nesta época, já existiam dezesseis comarcas (VAZ, 2014).

Em 1891 foi promulgada a Primeira Constituição da república do Brasil, com ela a justiça foi reorganizada, gerando autonomia administrativa e a obediência a singularidade da nova Constituição. O Superior Tribunal, instalado em 1893, passou a ser o órgão máximo de segunda instância, sendo que os seus cinco integrantes receberam o título de ministros em 1898.

Em 1903 firmou-se o curso de Direito em Goiás, sendo que em 1923 houve o primeiro bacharel formado neste curso, a ocupar cargo de desembargador. Em 1913, Goiás possuia vinte comarcas e 42 municípios (VAZ, 2014). Seguiram, então, dois importantes momentos na história da estruturação do Judiciário em Goiás: i) o código de organização judiciária, em 1929 e ii) o regimento interno do Poder Judiciário Goiano, em 1930. Com o regimento interno, estruturou-se o Tribunal em duas câmaras: i) cível e ii) criminal (ARAUJO E. FARIA, 2000).

Em 1935 com o estabelecimento da Constituição estadual, entrou em cena a Corte de Apelação, orgão colegiado máximo do Judiciário Goiano, atualmente chamado de Corte Especial. Na época, todas as propostas de leis do Poder Judiciário local deviam ser primeiramente aprovadas por este orgão colegiado, antes de serem encaminhadas para o Poder Legislativo.

Nas décadas de 1940 e 1950 o Judiciário permaneceu vigilante no cumprimento das leis e na busca da garantia dos direitos do cidadão. Em Goiás houve aumento

do número de comarcas, magistrados e servidores. Já nas décadas de 1960, 1970 e 1980, época da ditadura militar, o Judiciário se manteve constante e vigilante, não submetendo-se à nova ordem, respeitando a constituição. Porém, os atos institucionais (AI) alteraram a constituição, suspenderam direitos, como: garantias individuais e apreciação de atos judiciais pelos magistrados.

A Constituição de 1988 representou para o Poder Judiciário brasileiro a mudança, dando-lhe a necessária independência. Os Tribunais de Justiça do Brasil assimilaram gradativamente o novo ordenamento jurídico e sua filosofia político-social. Em Goiás, as gestões seguiram semeando a modernidade indispensável à melhoria dos serviços jurisdicionais, principalmente com a modernização dos procedimentos (CASTRO, 2014).

A década 1980 iniciou o contato do Poder Judiciário Goiano com a informática, quando foi adquirido o primeiro computador, o Cobra 400 I, com quatro terminais de entrada de dados, época de implantação do sistema de acompanhamento de processo no primeiro grau, chamado de TJG. Nesta mesma década foram implantados sistema eletrônico de processos, chamado SEP e o sistema de acompanhamento de processos no segundo grau, conhecido como PDJ. Na década de 1990, deu-se início a informatização do Judiciário no estado, sendo em 1992 instituída a Comissão Permanente de Informatização pela Emenda Regimental n. 07/RITJGO (23/06/1992).

Em 1996 os sistemas TJG e PDJ foram reformulados, sendo desenvolvido o novo Sistema Primeiro Grau (SPG). No mesmo ano foi realizado o primeiro concurso público para cargos na área de informática: i) digitador; ii) operador; iii) programador e iv) analista de sistemas, tomando posse nos cargos em 1997 e 1998. Em seguida, em 1999, o Sistema de Segundo Grau (SSG) foi implantado, software que faz a gestão dos processos no segundo grau. Os dois sistemas foram escritos em linguagem de programação Natural®, com banco de dados ADABAS®. Todo o processamento do SPG e SSG era centralizado em mainframe da IBM®, em sistema operacional s/390®.

Depois de oito anos do SPG em funcionamento, em 2004, existiam 20 comarcas no estado de Goiás interligadas em redes de longa distância, utilizando protocolo de transmissão de dados *Synchronous Data Link Control* (SDLC). De 2005 a 2008 houve significativo avanço na interligação em redes das comarcas, sendo todas as 119 comarcas interligadas a intranet e internet, agora em redes de dados em protocolo de

internet (IP), com enlaces de dados com tecnologia Frame Relay e xDSL. Em 2007 e 2008 iniciou-se a era do processo judicial eletrônico em Goiás, sistema chamado PROJUDI, advindo da promulgação da Lei do Processo Eletrônico de 2006, sendo passo relevante para aplicação da tecnologia da informação no Judiciário.

2.4 Lei do Processo Judicial Eletrônico

Anterior à promulgação da Lei Federal 11.419/2006, os processos judiciais eram formados de pilhas de papeis, não podendo ter sua tramitação totalmente eletrônica, pois não era autorizado legalmente a utilização da assinatura eletrônica nas sentenças, decisões, despachos, citações e outros documentos de ordem jurídica. Necessita-se do processo em papel para armazenar todos os documentos produzidos e assinados manualmente. Na época, só era permitido aos sistemas realizarem o acompanhamento das tramitações das pastas em papel, sem armazenar o inteiro teor de todos os documentos. Mesmo assim, alguns softwares armazenam o inteiro teor com o objetivo de apenas informar, sem valor jurídico.

Em Goiás, o sistema de processo eletrônico, conhecido como PROJUDI, é implantado em 2007. Com o PROJUDI foi possível tornar a tramitação dos processos judiciais na íntegra em meio eletrônico, utilizando certificado digital para assinar eletronicamente os documentos. Desta forma, todo o processo judicial pode ter seus andamentos, encaminhamentos, inserção de peças de forma eletrônica.

Outros softwares foram desenvolvidos no País para hospedar os processos judiciais, em atendimento a Lei 11.419, como: e-DOC e o PJ-e. Atualmente, o Conselho Nacional de Justiça, pela Resolução 185/2013, determina que todas as esferas do Judiciário nacional (Justiças Estaduais, Justiças Federais, Justiças do Trabalho, Justiça Militar, Justiças Eleitorais e outras) implantem de forma única o Processo Judicial Eletrônico, conhecido como PJ-e. O planejamento estratégico instituído no Judiciário permite estabelecer metas para aplicação dos recursos tecnológicos, financeiros e humanos.

2.5 Planejamento Estratégico Institucional

O Planejamento Estratégico é o mecanismo que permite analisar a organização sob vários ângulos, definindo seus rumos por meio de direcionamento que possa ser monitorado nas suas ações concretas, utilizando-se, para tanto, de instrumento denominado plano estratégico (TRT, 2013).

Em 2009, o Conselho Nacional de Justiça aprovou a Resolução nº 70/2009 que dispõe sobre o Planejamento e a Gestão Estratégica no âmbito do Poder Judiciário. A partir desta Resolução os órgãos da Justiça no Brasil começaram a fazer seus planos estratégicos institucionais. Esta nova orientação é aplicada em todo o Judiciário nacional, permitindo que o futuro seja planejado e que as ações necessárias sejam empreendidas.

No Poder Judiciário Goiano, o primeiro planejamento estratégico acontece em 2007, deflagrando nova mudança voltada para a elaboração de metas e indicadores, bem como pela possibilidade de medi-los de acordo com o tempo necessários para cumpri-las. O planejamento estratégico aliado ao processo judicial eletrônico são, atualmente, a mola propulsora do Judiciário nacional, mas conhecer a estrutura do Judiciário é fundamental para conseguir aplicar o planejamento estratégico.

2.6 A Estrutura do Judiciário em Goiás

A estrutura do Judiciário Goiano está definida na Lei 9.129, de 22 de Dezembro de 1981, que dispõe sobre o Código de Organização Judiciária do Estado de Goiás. O território do Estado de Goiás, para a administração das justiças, divide-se em comarcas e distritos, e constitui um todo para efeito da jurisdição do Tribunal de Justiça e da Justiça Militar. Cada comarca é formada por um ou mais municípios contíguos. Existem requisitos mínimos para criação de comarcas, como: população mínima de 20.000 habitantes; mínimo de 3.000 eleitores; arrecadação mínima de R$ 72.727,27; média de serviço forense mínimo de 150 feitos ajuizados no triênio anterior e extensão territorial mínima de $50km^2$.

As comarcas se classificam em três entrâncias: entrância inicial; entrância intermediária e entrância final. O Poder Judiciário é exercido pelos seguintes órgãos: Tribunal de Justiça; Juízes de Direito, Juízes Substitutos e Juízes Militares. O Tribunal, com sede em Goiânia (capital do estado), órgão máximo do Poder Judiciário do Estado de Goiás e Jurisdição no território estadual, compõe-se, atualmente, de trinta e seis desembargadores. São órgãos integrantes do Tribunal de Justiça: Tribunal Pleno; Câmaras Cíveis Reunidas; Câmaras Criminais Reunidas; Câmaras Cíveis Isoladas; Câmaras Criminais Isoladas; Presidência; Vice-Presidência; Conselho da Magistratura; Corregedoria-Geral da Justiça e as Comissões Permanentes.

O Tribunal é composto pelo Presidente e o Vice-Presidente, eleitos dentre os mem-

bros de maior antiguidade, para o período de dois anos, proibida a reeleição. O Corregedor da Justiça, os Presidentes das Câmaras, os membros das Comissões Permanentes e quatro membros do Conselho da Magistratura são eleitos, também para o período de dois anos, na forma do que dispuser o Regimento Interno. As Câmaras Isoladas, Cíveis e Criminais, numeradas ordinalmente, são compostas de quatro desembargadores e divididas em turmas de três Juízes, para efeito de julgamento.

O Poder Judiciário Goiano conta hoje com 127 (cento e vinte e sete) comarcas em todo o Estado, sendo a comarca final em Goiânia, várias intermediárias e iniciais. Possui também vários Juizados Cíveis e Criminais, tendo ainda o Juizado da Infância e Juventude, bem como o da Violência Doméstica Contra a Mulher. Presta serviço primordial à comunidade Goiana, mantendo a ordem e o equilíbrio entre os polos passivo e ativo nos processos.

Em cada comarca há o juiz titular ou substituto de 1º grau, tendo, ainda, o diretor do Fórum. O Fórum ou Foro é o prédio onde ocorrem os trabalhos forenses da comarca. As informações referentes ao expediente forense em sistema de informática permitem que os gestores adquiram conhecimento necessário para realizar a administração/gestão do Judiciário.

2.7 Gestão da Informação para Representação do Conhecimento

Novos métodos de gestão, novas ferramentas de apoio, novos sistemas de informação, tudo isto representa esforço por aperfeiçoar a gestão. Neste contexto, além dos métodos, ferramentas, sistemas, informações e aperfeiçoamento de gestão, os softwares que conseguem gerar alertas automatizados baseado em indicadores são mecanismos cada vez mais aplicados.

Nos bancos de dados dos órgãos da justiça (inclusive via internet) podem ser encontrados informações como: i) andamento e/ou fases de processos, ii) conteúdo (integral ou ementado) de decisões judiciais, iii) legislação, iv) doutrina, v) informações gerais jurídicas e vi) assuntos administrativos de interesse da justiça são fontes de informações preciosas e bastante pesquisadas.

Apesar da informação estar disponível para pesquisa de qualquer pessoa na internet, a responsabilidade em encontrá-la é do ser humano. Assim, a capacidade de extrair conhecimento é do cliente, fazendo com que a qualidade da informação disponibilizada dependa da forma como é consultada pelo operador. Os softwares de

buscas/pesquisas disponibilizados pelo Judiciário trazem campos abertos para coleta de dados do inteiro teor dos documentos, porém não há nenhuma inteligência artificial aplicada.

Os operadores do direito, embora possuam hoje imensos repositórios de dados, encontram-se desprovidos de informação e, consequentemente, sem conhecimento adequado para tomadas de decisões que poderiam ser baseadas nas informações do banco de dados existente.

Apesar da inclusão destes tipos de representação da informação em sistemas atuais, ainda está aquém da capacidade de transformar os grandes volumes de dados em gestão do conhecimento para tomada de decisão. Contudo, este progresso não acelera por inteiro, o serviço relacionado à prestação jurisdicional. Está faltando o principal, o sistema inteligente para os serviços de cartório e gabinete do juiz, notadamente no primeiro grau de jurisdição (OLIVEIRA, 2008).

Fica latente o elevado volume de dados existente nas bases de dados do Judiciário, pois além dos metadados existentes, tem-se ainda dados indexados dos documentos digitalizados, ou seja, existe elevada quantidade de informação disponível. Assim, percebe-se a necessidade de implementar algoritmos refinados para classificação das informações, objetivando melhorar a identificação dos dados desejados pelos usuários. Estes mecanismos de refinamento das informações passam a ser um diferencial, visto o volume de dados existente.

2.8 Considerações Finais

O Judiciário é o poder da união, bem estruturado, com autonomia institucional, administrativa e financeira. Apesar da sua história secular, o Judiciário vem aplicando nos últimos anos técnicas de gestão, planejamento estratégico e de tecnologia da informação para conseguir aumentar sua produtividade, objetivando atender a tempo e hora os anseios da sociedade em geral. No próximo capítulo será abordado a ontologia como recurso para representação do conhecimento e como estrutura de conceitos representados pelo vocabulário, no trabalho em tela a classificação da base de conhecimento gerada a partir das decisões proferidas no Judiciário.

CAPÍTULO 3

ONTOLOGIA

Objetiva-se neste capítulo apresentar os conceitos relacionados à ontologia e sua relação com o trabalho a ser desenvolvido. Serão parcialmente abordados os registros históricos da ontologia: i) no sentido filosófico, ii) na metafísica aristotélica, iii) na teoria kantiana, iv) na fenomenologia de Hurssel, v) na ciência da computação e vi) na ciência da informação. Serão analisados os principais usos, benefícios e problemas relacionados às ontologias.

3.1 Conhecimento Gerado pelo Registro Histórico

Na Antiguidade, os escribas eram os profissionais que tinham a função de escrever textos, registrar dados numéricos, redigir leis, copiar e arquivar informações. Como poucas pessoas dominavam a arte da escrita, eles possuíam destaque social. Os escribas eram, geralmente, funcionários reais, pois eram comandados pelo governante e deviam registrar tudo o que seu superior ordenasse. É possível identificar os escribas na época de Cristo (região da Palestina) e também no Egito Antigo.

Da impressão da Bíblia às pesquisas científicas, fontes preciosas de conhecimento potencial foram sendo armazenadas por séculos. O registro das informações podem acontecer de diversas formas: i) manuscrito, ii) desenhos, iii) esculturas, iv) pinturas, v) músicas, vi) histórias e estórias contadas, vii) versos e prosas e viii) outros. Absorver tais informações fazem com que o pesquisador adquira conhecimentos e, assim, possam reproduzí-los, aplicá-los e aprimorá-los, deixando novos conhecimentos para a geração futura.

O registro histórico do passado, sejam pelos fatos ocorridos ou pelas observações dos fenômenos naturais, são ações importantes para que a humanidade consiga permanecer em sua linha de evolução. A filosofia tem relevância significativa na evolução do conhecimento humano, bem como a ontologia, que é parte da filosofia.

3.2 Sentido Filosófico da Ontologia

Em seu sentido filosófico, a ontologia possui diversas definições. De acordo com Bax (2003), é o ramo da metafísica que estuda os tipos de coisas que existem no mundo. A palavra é derivada do grego *ontos*, ser, e *logos*, palavra. Entretanto, seu termo

de origem é a palavra aristotélica **categoria**, termo utilizado no sentido de classificação. Neste sentido, Aristóteles apresenta categorias que servem de base para classificar entidades, e introduz o termo **diferença**. A Figura 3.1, adaptada de Ambrosio (2007), ilustra a ideia de **categoria** e **diferença** desenvolvida por Aristóteles.

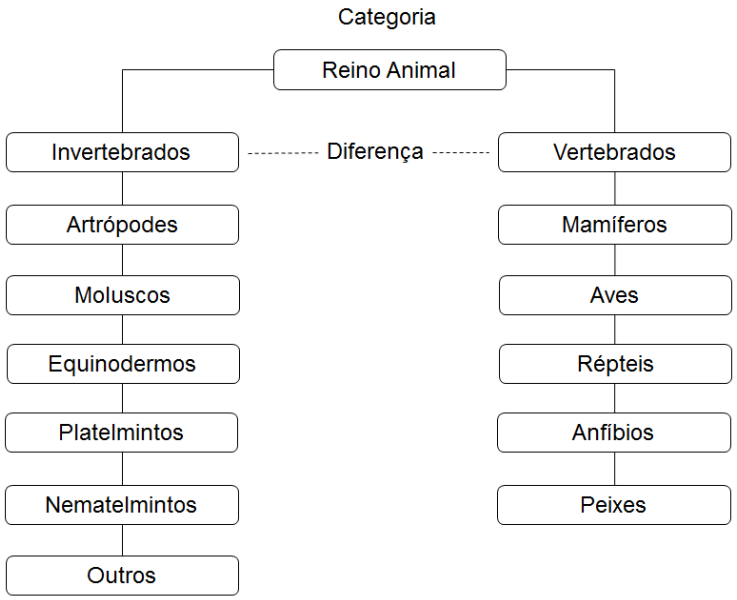

Figura 3.1 - Ideia de categoria e diferença apresentada por Aristóteles.

Para Marcondes (1997), ontologia é a parte da filosofia que trata da natureza do ser, ou seja, da realidade, da existência dos entes e das questões metafísicas em geral. Smith (2006), refere-se ao termo como a ciência do **que é...** ou o **estudo das coisas que existem**, da descrição de tipos e estruturas de objetos, suas propriedades, eventos, processos e relacionamentos em cada área do mundo real.

3.2.1 Ontologia em Aristóteles

Almeida (2014) afirma que Aristóteles provavelmente foi o primeiro filósofo a usar a palavra grega *kategoria* como termo técnico para predicação. Acredita-se que o sistema de categorias é capaz de dar o inventário das coisas que existem. Em toda a história percebe-se a característica do homem em querer classificar e dar nomes as coisas, servindo como princípio de entendimento, objetivando a absorção de conhecimento. O principio da classificação, *kategoria* aparentemente iniciado por Aristóteles serve como crescimento evolutivo das gerações. Até hoje, quando é descoberto coi-

sas novas, faz-se a sua classificação, primeiro tentando enquandrá-la em conjuntos conhecidos, caso não seja possível, é aberto novo conjunto, ou nova classificação.

Na época de Aristóteles, utiliza-se apenas o termo *kategoria*. Como exemplo do trabalho de Aristóteles tem-se a Tabela 3.1, extraída de Almeida (2014). O próprio Aristótele entendia na época a complexidade crescente na classificação das coisas, não parecia acreditar em único nível, mas em vários. A classificação se dá por níveis dos tipos das coisas.

Tabela 3.1 - Categorias genéricas idealizadas por Aristóteles.

Termo Aristotélico	Significado Moderno	Exemplo
Ti esti, ousia	substância	homem
Poson	quantidade	cinco metros
Poion	qualidade	branco
Pros ti	relação	metade
Pou	local	no mercado
Pote	data	ontem
Keisthein	postura	sentado
Echein	estado	vestido
Poitein	ação	queimar
Paschein	sentimento	ser queimado

Para Aristóteles, a identificação das classes e tipos se baseiam nas características fundamentais, nas essências das coisas. Se for de certo tipo, é porque foi encontrada sua essência. Se encontrar outros com a mesma essência, a entidade irá compartilhar as propriedades necessárias e suficientes com os outros membros daquele tipo (ACKRILL, 1963). Como exemplo, tem-se a espécie humana, que pertence ao gênero *Homo* e sua *differentia* é a racionalidade. Com esta noção nasce o método aristotélico para distinguir essências, usando a distinção genero-espécie e a divisão dicotômica (LENNOX, 2000).

Assim, para distinguir determinada espécie de outra do mesmo gênero, usando a distinção gênero-espécie, é necessário obter a essência real de dada espécie e combiná-la com seu gênero e sua *differentia*. Com a divisão dicotômica, Aristóteles consegue separar cada gênero em entidades que tem certa *differentia* daquelas que não tem, permitindo ainda, ordenar categorias de coisas de acordo com suas características essenciais. Como exemplo, coisas vivas são divididas em animais e plantas pela *differentia* de auto-movimento (ALMEIDA, 2014). Outros filósofos questionam a forma

não sistemática que as categorias de Aristóteles teriam sido escolhidas e propõem novos sistemas (JANSEN, 2008).

3.2.2 Ontologia em Kant

O filósofo Immanuel Kant busca o equilíbrio entre duas formas de gerar conhecimento: i) os limites da razão e ii) os limites da experiência. Desta busca surge a teoria dos juízos, ou seja, a capacidade de julgar os fenômenos relacionados aos objetos. A preocupação é conseguir analisar os objetos, baseados nos fenômenos que acontecem neles a cada momento, incluindo os seus relacionamentos. A maioria dos filósofos da época moderna colocam o objeto no centro e o sujeito ao redor, objetivando analisar as características que se apresentam e as que não se apresentam do objeto. Kant muda este raciocínio, tira o objeto do centro e coloca o sujeito, agora o sujeito tem a responsabilidade de analisar os predicados que se apresentam dos objetos, permitindo fazer o julgamento utilizando os limites da razão e da experiência do sujeito.

Assim, visto ser necessário haver mecanismo que facilite a classificação dos objetos, gerando conhecimento relacionados aos próprios objetos, Kant propõe quatro aspectos de julgamentos: i) quantidade, ii) qualidade, iii) relação e iv) modalidade (WOOD, 2004). Nestes aspectos de julgamentos, cria-se as subdivisões: 1) relacionado a quantidade, têm-se: i) universal, ii) particular ou iii) singular; 2) relacionado a qualidade, têm-se: i) afirmativo, ii) negativo e iii) infinito; 3) relacionado a modalidade, têm-se: i) problemática, ii) assertória e iii) apodítica e 4) no aspecto relação, têm-se: i) categórico, ii) hipotético e iii) disjuntivo.

Trabalhando a dicotomia entre a razão e a experiência, Kant consegue estabelecer a classificação ontologica dos objetos, lidando com categorias do entendimento humano e não com objetos do mundo, como disposto na Tabela 3.2, adaptada de Almeida (2014). A teoria de Kant mantém importância ontológica, visto que as categorias se aplicam *a priori* a objetos da cognição. Outros filósofos trabalham a ontologia mediante estudos fenomenológicos.

3.2.3 Ontologia de Husserl

Edmund Gustav Albrecht Husserl desenvolve, no início do século 20, estudo de como as coisas do mundo se apresentam na mente, trazendo a capacidade do sujeito de compreender os acontecimentos dos fatos, isto se dá o nome de fenomenologia.

Tabela 3.2 - Aspectos propostos por Kant para classificar julgamentos.

Aspecto quantidade de julgamentos		
Universal	Particular	Singular
Todos S	Algum S	Este S
Todos os cães tem pelos	Alguns cães tem pelos	Rex tem pelos
Aspecto qualidade		
Afirmativo	Negativo	Infinito
P	não-P	não-P (outros P são possíveis)
O livro é vermelho	O livro não é vermelho	O livro não é vermelho, mas ele é de outra cor.
Aspecto modalidade		
Problemática	Assertória	Apodítica
S é possívelmente P	S é efetivamente P	S é necessariamente P
A mesa é possivelmente marrom	A mesa é efetivamente marrom	A mesa é necessariamente marrom
Aspecto relação		
Categórico	Hipotético	Disjuntivo
todo S é P	Se S é P, então S é R	S é P ou R
Todos os animais de estimação são animais	Se um animal de estimação é um cão, então ele late	Um cão está latindo ou um gato está miando

A identificação pelos sentidos humanos dos fenômenos que acontecem ao redor do sujeito depende da percepção e da conciência, traz a luz aquilo que está implícito no senso comum, permitindo assim, realizar a classificação das coisas.

A fenomenologia é aquilo que se apresenta pelos sentidos, ou seja, estuda as essências das coisas e como são percebidas no mundo, sendo que a essência é a intencionalidade da mente. Husserl ensina como se abster da percepção do mundo natural e assumir posição transcendental objetivando descrever somente a consciência pura. Ele afirma que toda conciência é conciência de alguma coisa, neste sentido, pode ser classificada e relacionada de forma ontologica. **Categoria** para Husserl é a entidade do reino das essências formais (BEYER, 2011).

Husserl cria mecanismo para identificar as distinções da conciência pura, estabelecendo três reinos ontologicos: i) reino dos fatos, ii) reino das essências e iii) reino dos significados ou sentidos. O reino dos fatos tem domínio nas ciências empíricas, o reino das essências tem domínio nas ciências relativo as formas e o reino dos significados capta o conjunto de fenôminos e como se manifestam na mente, sendo o conteúdo da experiência através do tempo e espaço. A Tabela 3.3, adaptada de Almeida (2014)

e Smith (2005), dispõe a ontologia de Husserl.

Tabela 3.3 - Ontologia básica de Hurssel.

I	Fatos (entidades concretas)
I-1	Indivíduos
I-1.1	indivíduos independentes (substratos)
I-1.2	indivíduos dependentes (momentos)
I-2	Estados das coisas
I-3	Eventos
I-3.1	experiências
II	Essências (entidades ideais)
II-1	Essência formal
II-1.1	Indivíduo
II-1.2	Espécie, Qualidade, Relação
II-1.3	Estado de coisas
II-2	Essência material
II-3	Região (Espécie material de mais alto nível)
II-3.1	Natureza
II-3.2	Consciência
II-3.3	Espírito (humanidade)
III	Significado ou Sentido oriundas da nossa atitude natural(conteúdo da experiência)
III-1	Sentidos dos indivíduos
III-2	Sentidos predicativos de espécies, qualidades e relações
III-3	Sentidos manifestados, proposições

Na Tabela 3.3 é disposta a classificação em níveis, separados por reinos como indicados por Husserl, que são: i) I reino dos fatos, ii) II reino das essências e iii) III reino dos sentidos. Dentro destes reinos ocorre as subdivisões, sendo: i) as entidades concretas relacionadas: ao I-1 indivíduos e ao I-2 estados das coisas no tempo e espaço; ii) entidades que guardam as essências do ser, relacionadas: ao II-1 são essência formal das coisas ao II-2 são a essência formal da matéria e ao II-3 são essência interna da consciência e espírito da maneira como se apresentam; iii) guardam os sentidos relacionados a atitude natural do ser: ao III-1 sentidos do indivíduo, ao III-2 sentidos predicativos da espécie como suas relações e qualidades inerentes e ao III-3 sentidos manifestados, sentidos evidentes que não podem ser negados.

Vários são os pensadores/filósofos que se embrearam nos estudos da ontologia. A ontologia atual não é meramente instrumento de estudo e sim de aplicações em várias áreas do conhecimento. Seu entendimento auxilia na categorização de dados e na representatividade e recuperação de informação entre outras aplicações.

3.2.4 Ontologia em Ciência da Computação

Desde 1960, o termo ontologia é usado para se referir a estrutura de conceitos representados por vocabulário lógico, já indicando o seu relacionamento com representação do conhecimento (RC), sendo subárea da inteligência artificial (IA). A utilização da ontologia no desenvolvimento de sistemas de informação, na área de engenharia de software, surge em 1980. Nos anos de 1990, surge o termo **web semântica**, que representa conjunto de tecnologias aplicadas a internet.

Gruber (2005) afirma que ontologia é a descrição de conceitos e relacionamentos que existem entre estes conceitos. Borst (1997) informa que ontologia é a especificação explícita e formal da conceitualização compartilhada, sendo que: i) explícita são os conceitos, relações, propriedades, funções, axiomas e restrições definidas; ii) especificação formal significa algo legível aos computadores; iii) conceitualização representa modelo abstrato de algum fenômeno do mundo real e iv) compartilhada significa conhecimento consensual.

Considerando as atividades e os agentes envolvidos na tarefa de representar conhecimento, é possível entender o papel da ontologia em RC. Sistemas declarativos contém afirmações que representam fatos governados por regras. Exemplo de fato é **Goiânia é uma cidade em Goiás** e regra é **todas as pessoas que vivem em Goiânia vivem em Goiás**. Esta combinação de fatos e regras compõem a base de conhecimento do sistema. A base de conhecimento é construída e mantida por engenheiro do conhecimento, que tem como tarefa formalizar o conhecimento do grupo de especialistas (ALMEIDA, 2014).

Os engenheiros do conhecimento realizam abstrações e generalizações relacionados aos fatos e regras que governam parte da realidade. Para isto acontecer, é necessário fazer uso das suas intuições, que transcendem a natureza física das coisas. Assim, infere-se que para aplicar a ontologia faz-se necessário utilizar lógica, algumas vezes carregada de conhecimento, cultura e experiência de vida. Esta lógica, em ciência da computação, deve ser apresentada por linguagens expressivas, objetivando gerar o conjunto de modelos representativos da realidade. Neste cenário, o papel da ontologia é tornar explícitos axiomas que restringem modelos, com o intuito de igualar, o que for possível, modelos que contém o significado pretendido (GIARETTA, 1995) (GUARINO, 1998).

Observa-se dois significados principais para o termo ontologia em ciência da computação. O primeiro diz respeito ao uso de princípios ontológicos para entender e modelar a realidade (WEBER, 1990) e (WAND, 1999). O uso do termo neste caso está alinhado com seu papel original na filosofia, ou seja, fornecer descrição do que existe e caracterizar entidades nas atividades de modelagem. O segundo significado diz respeito a representação do domínio em linguagem de representação computacional (STUDER, 2004). A ontologia, nesse caso, consiste de conjunto de declarações expressas em linguagem de representação, o qual podem ser processados por mecanismos de inferência automatizados (ALMEIDA, 2014). Para Ambrosio (2007), ontologia define as regras de combinação entre os termos e seus relacionamentos, estes relacionamentos são criados por especialistas, e os usuários formulam consultas usando os conceitos especificados.

Cabe citar, entretanto, que mesmo considerando o uso de princípios ontológicos no caso da modelagem, nem sempre são tomadas decisões ontológicas para representar a realidade. Decisões são tomadas em função de limitações de desempenho dos sistemas, resultando em representações incoerentes com a realidade (WELTY, 2001).

3.2.5 Ontologia em Ciência da Informação

Projetos de engenharia de software envolvendo ontologias exibem paralelos com teorias da ciência da informação, tal como classificação facetada, vocabulários controlados e lexicografia (VICKERY, 1997). Existem diferenças e similaridades entre ontologias em ciência da computação e outros dois termos comuns: i) taxonomia e ii) tesauros. A possibilidade de restringir a linguagem natural parece ser o ponto de contato entre estes três tipos de estruturas (GILCHRIST, 2003). Além disto, mesmo que o termo ontologia as vezes apareça como sinônimo para vocabulários controlados, as três estruturas tem objetivos diferentes (CURRAS, 2003). Existem ainda outros usos para princípios ontológicos em ciência da informação.

Fundamentos filosóficos são utilizados para explicar a hipótese de que a informação tem *status* ontológico similar às proposições subjacentes ao texto: ambos são atemporais, não-espaciais e são objetos abstratos (FOX, 1983). Proposições não coincidem com sentenças do texto, na medida em que várias sentenças podem expressar a mesma proposição. Por exemplo, as sentenças **Marte tem duas luas**, ***Two moons circle Mars*** e ***Mars a deux lunes*** carregam a mesma proposição, ou seja, o fato de que Marte possui duas luas.

A informação contida no documento não é identificada com o texto do documento, mas sim como seu conteúdo proposicional. Para alcançar o conteúdo proposicional, é necessário conduzir análise proposicional da informação seguindo o princípio: a informação transportada pela sentença S é uma proposição apropriadamente associada a S (FOX, 1983).

A abordagem proposicional se originou no estudo da lógica e filosofia da linguagem e está relacionada a distinção proposta por Friedrich Ludwig Gottlob Frege, que utiliza como exemplos as expressões **Estrela da manhã** e **Estrela do entardecer**, as quais parecem ter diferentes significados, mas se referem a mesma entidade, o planeta Vênus. Frege atribuía as entidades de sua teoria aos três vértices do triângulo, nomeando-os por símbolo, sentido e referência (ALMEIDA, 2014).

No contexto dos sistemas de recuperação da informação, Blair (2006) discorre sobre o seu comprometimento com a filosofia segunda de Ludwig Joseph Johann Wittgenstein, a qual contém críticas aos princípios ontológicos de Aristóteles. A abordagem de Wittgenstein à categorização é baseada na noção de semelhança de família, a qual tem duas suposições básicas: os membros de um grupo taxonômico compartilham o conjunto de traços similares; tais traços não necessariamente ocorrem em todos os membros do grupo e apenas naquele grupo, como é o caso da abordagem de Aristóteles.

A significância da visão de Wittgeinstein para o projeto e o uso de sistemas de recuperação da informação reside no fato que indeterminação será a característica essencial de esforços de recuperação de conteúdo. Portanto, deve-se assumir a existência da indeterminação e planejar estratégias para mitigar cada caso, como por exemplo, a sobrecarga de categorias. Ao que parece, os autores de ciência da informação nem sempre mencionam explicitamente o termo **ontologia**, mesmo que princípios ontológicos apareçam na literatura do campo de pesquisa.

3.3 Diferença de ontologia e base de conhecimento

Os sistemas de informática para as empresas sempre tiveram o propósito de registrar dados, na maioria das vezes em banco de dados. Dados como registros clássicos, por exemplo: i) nomes, ii) números de documentos, iii) filiação, v) endereço e vi) outros. Com a necessidade de mais informações, vieram os registros de dados clássicos combinados, por exemplo: i) financeiro relacionado as pessoas, ii) procedimentos

relacionados as pessoas, iii) documentos relacionados as pessoas, iv) relacionamento entre as pessoas com características específicas e v) outros. Após, há a necessidade de guardar não só os metadados, mas arquivos relacionados as pessoas, necessitando dos registros de dados clássicos combinados com arquivos, assim têm-se: i) imagem dos documentos pessoais, ii) identificação digital, iii) certificado digital A1 e/ou A3, iv) contratos, v) sentenças, vi) petições e outros. Percebe-se que a existência de grande volume de dados nas empresas podem ser analisados, combinados e relacionados objetivando criar a base de conhecimento.

Apesar dos sistemas evoluirem no sentido de terem relatórios para extração de dados, estas programações são realizadas de forma isoladas, gerando base de conhecimento com objetivos específicos, geralmente demandadas por pessoas com necessidades no tempo e espaço. Mas, não há, mesmo nos relatórios que fornecem dados para gestão, a criação de regras e estruturação de toda a base de conhecimento. Neste sentido, faz-se necessário aplicar a ontologia, ou seja, realizar estudo aprofundados na base de conhecimento, estabelecendo estrutura de classificação, separando os indivíduos, as classes, dizendo quais os atributos de cada indivíduo e das classes, gerando relacionamento entre eles.

A Figura 3.2 ilustra a estrutura **simples** da ontologia genérica de banco de dados relacional, onde a denominação **simples** é por não conter os conceitos de cada indivíduo, bem como os relacionamentos com as inferências entre eles e os atributos para cada um. Na Figura 3.3 é ilustrado a base de conhecimento gerada por indivíduo no espaço e tempo, trazendo os conceitos inerentes aos resultados. A título de exemplo, outras inferências externas poderiam ser extraídas da base de conhecimento da Figura 3.3, como a entidade em domínio externo no momento de estabelecimento de novos contratos, poder utilizar informações da base de dados do Judiciário sobre os questionamentos contratuais de cada indivíduo, objetivando seguir ou não com a transação contratual.

3.3.1 Componentes da Ontologia

A ontologia é composta por i) indivíduos, ii) classes, iii) atributos e iv) relacionamentos. Os indivíduos são os objetos, as classes são os conjuntos de indivíduos, os atributos são as características destes objetos e os relacionamentos são as formas como os objetos se relacionam. Esta classificação se aproxima da teoria dos conjuntos, como ilustrado na Figura 3.4, utilizando exemplo do Judiciário.

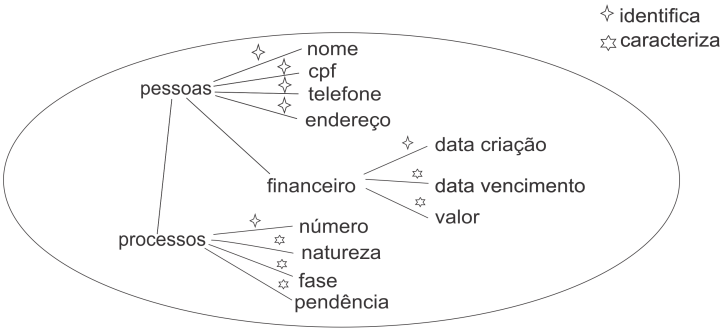

Figura 3.2 - Ontologia rudimentar de alto-nível de banco de dados relacional.

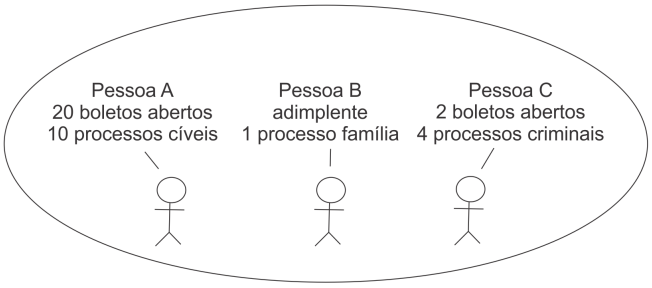

Figura 3.3 - Base de conhecimento gerada a partir da ontologia de domínio de banco de dados relacional.

Na Figura 3.4, o domínio é representado pelo conjunto/classe de dados de todas as sentenças proferidas de natureza revisional. Todas as outras classes estão contidas na classe revisional de contratos, desta forma têm-se: i) **revisional carro** $\subseteq$ **revisional contratos**, ii) **revisional imóvel** $\subseteq$ **revisional contratos**, iii) **revisional Dpvat** $\subseteq$ **revisional contratos** e iv) **reintegração posse** $\subseteq$ **revisional contratos**. Ainda, tem-se no modelo ontológico dois indivíduos da classe revisional de carro que se relacionam com um indivíduo da classe revisional de imóveis. Caso seja convencionado, a operação de interseção com o relacionamento pode ser apresentado no modelo de conjunto, assim: **dois indivíduos classe revisional carro** $\cap$ **um indivíduo classe revisional imovel**. Com estas inferências entre os indivíduos criadas no modelo, pode-se entender que há inteligência gerada no raciocínio do que

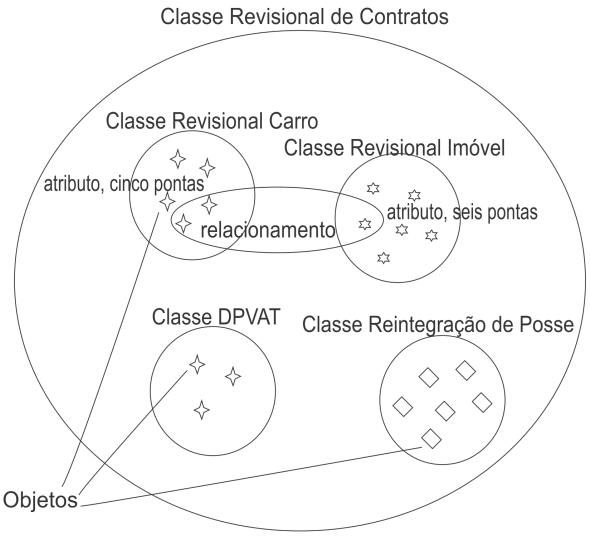

Figura 3.4 - Modelo usando teoria de conjuntos para representar os componentes da ontologia.

é correto e o que não é correto afirmar.

3.3.2 Classificação da Ontologia

Guarino (1998) classifica a ontologia como: i) ontologia genérica, ii) ontologia de domínio, iii) ontologia de tarefas e iv) ontologia de aplicação. A ontologia genérica cria as classes com conceitos diferentes e gerais, na maioria das vezes o relacionamento é dificultado, visto as diferenças existentes. A ontologia de domínio é a mais utilizada, pois trata de forma específica determinadas situações, como ontologia de veículos e habitações. Já a ontologia de tarefas e de aplicação são iniciativas mais recentes, sendo que a de tarefa descreve o vocabulário de ações que precisam ser executadas, enquanto a de aplicação está relacionada aos trabalhos vinculados aos sistemas de informação. A ontologia genérica também é chamada de ontologia de alto-nível.

As classificações apresentadas guardam relação entre elas, sendo que todas podem ser conhecidas como ontologia de alto-nível. A ontologia mais específica é a de aplicação, sendo que ela pode ser também, ontologia de domínio ou ontologia de tarefa. A Figura 3.5, adaptada de Guarino (1998), ilustra as classificações de ontologia e suas relações de pertinência. Existem outras classificações de ontologia, como a descrita por Welty (2001), onde o autor distingue dois tipos de ontologia: i) ontologia real e

ii) ontologia epistemológica. A ontologia real diz respeito a forma de organização do universo, correspondendo a abordagem filosófica, enquanto a ontologia epistemológica, relaciona a atividade de conceitualizar e estruturar os domínios específicos. No entanto, a mais utilizada na literatura é a proposta por Guarino (1998).

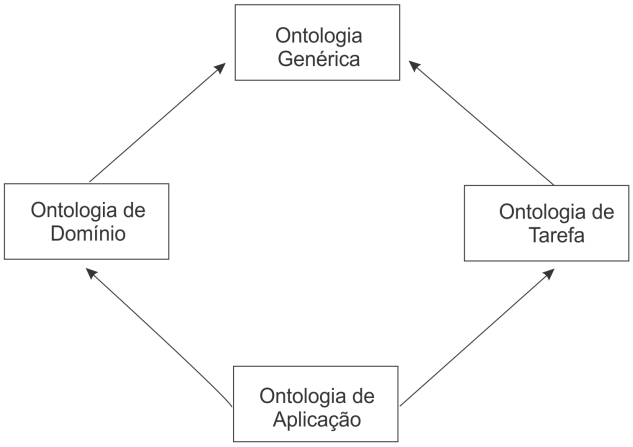

Figura 3.5 - Tipos de ontologias e suas relações de pertinência.

3.4 Metodologias de desenvolvimento

A construção da ontologia da base de conhecimento é atividade que necessita da inteira compreensão do negócio. Nesta compreensão insere-se o conhecimento nos metadados (conjunto de características do dado) e seus relacionamentos. O profissional que trabalha neste projeto deve agregar várias áreas do conhecimento, pois além de conhecer do negócio, da base de conhecimento, dos metadados e seus relacionamentos, deve conhecer metodologias para aplicar a ontologia. Dentre estes níveis de exigência, o conhecimento para aplicação da ontologia tem ficado para segundo plano, justificando a falta de padrão, ainda existente, para transformação do conhecimento implícito em explícito na elaboração da ontologia.

Os profissionais do conhecimento, na maioria das vezes quando aprendem o negócio e entendem a base de conhecimento, partem para a implementação da ontologia. Guimarães (2002) descreve os seguintes problemas ocasionados pela falta de conhecimento da ontologia: i) a conceitualização não fica clara; ii) dificuldade no reuso, visto a dificuldade de compreensão e iii) dificuldade na implementação. Assim, a adoção de metodologias é o caminho adequado a ser seguido na construção da ontologia.

Segundo Mattos (2007), existem alguns exemplos utilizados para ramos específicos do conhecimento, como: i) *Kactus*, ii) *Sensus*, iii) *On-to-Knowledge*, iv) *Uschold* e *King*, v) *Gruninger* e *Fox* e vi) *Methontology*.

Almeida (2010) apresenta modelo de criação da ontologia utilizando Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML). A Figura 3.6, adaptada de Almeida (2010), ilustra exemplo da metodologia utilizada para construção da ontologia em UML. Além da linguagem UML ser utilizada na elaboração da estrutura de projetos de software, ela guarda importante correspondência na apresentação e entendimento conceitual dos indivíduos, classes, atributos e relacionamentos, ou seja, nos componentes utilizados na ontologia. Mesmo utilizando o modelo UML como metodologia para desenvolvimento da ontologia, faz-se necessário o planejamento prévio do que será construído, bem como a aquisição do conhecimento, como ilustra a Figura 3.7, onde o fluxo apresenta a estruturação da base semântica.

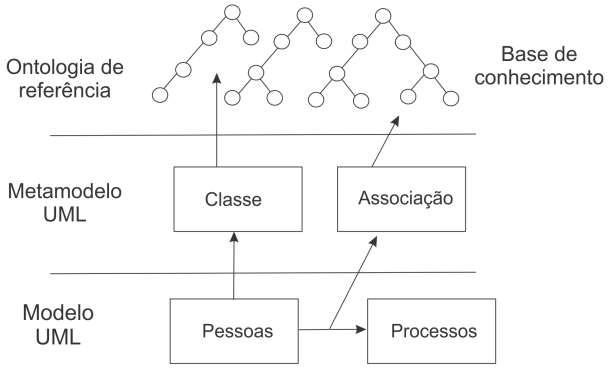

Figura 3.6 - Modelo aplicado na construção da ontologia.

3.4.1 Software utilizado para construção da ontologia

Atualmente existem várias ferramentas que podem ser utilizadas para construir a ontologia, como exemplo: i) *Protegé*, ii) *WebODE*, iii) *OntoEdit*, iv) *Vitro*, v) *Knoodl*, vi) *OntoKEM* e vii) outros. O *Protegé*® apresenta interface gráfica amigável, intuitivo, fácil manipulação, ambiente multiplataforma e interativo (MATTHEW, 2011). No *Protegé*® é possível agregar outros produtos, por exemplo: i) gerar grafo do rela-

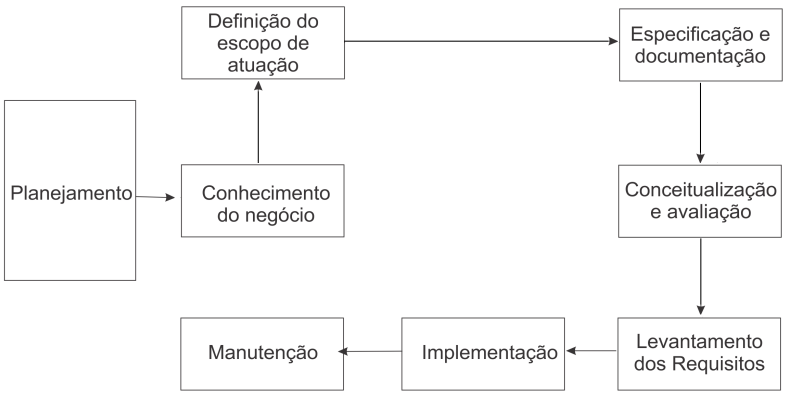

Figura 3.7 - Fluxo utilizado na elaboração da ontologia.

cionamento entre os indivíduos, ii) ferramenta para verificar e acusar erros na edição da ontologia e iii) outros.

Após a estruturação da ontologia, o *Protegé* gera arquivo em linguagem para definir e instanciar a ontologia na web (*Ontology Web Language* – OWL), permitindo sua exportação para outras ferramentas e outros softwares de integração ontológica. A OWL é linguagem de computador, que utiliza *tags* para separar sua sintaxe. Além da OWL, existem outras linguagens geradas pelo *Protegé*®: RDF, RDFS e XML.

3.5 Bases de conhecimento léxico-semântico

A ontologia permite estabelecer associações de sentido entre palavras. As relações semânticas no processo de classificação de documentos firma-se como processo relevante quando se pretende construir programas de computador capazes de lidar com conteúdo de textos (OLIVEIRA et al., 2015). A materialização da proposta de construção da ontologia é realizada pelos bancos de dados léxicos de relações semânticas entre palavras, conhecido como WordNet. A Wordnet trabalha desde o seu início com sinônimos, hipônimos e merônimos. Os sinônimos são agrupados em *synsets*, com sinônimos das palavras, definições curtas e exemplos de uso (OLIVEIRA et al., 2016). A Wordnet surge na década de 1990, criada apenas para o inglês. Na década de 2000 inicia o trabalho da Wordnet para língua portuguesa, mas somente na década de 2010 variantes da Wordnet no Brasil iniciam as abordagens no processamento de linguagem natural (PLN) (OLIVEIRA et al., 2015), (OLIVEIRA et al., 2016).

Atualmente, existem várias soluções de bases ontológicas lexical unificadas para o Português, como: PULO (*Portuguese Unified Lexical Ontology*), Onto.PT, OpenWordNet-PT e UfesWordNet (OLIVEIRA et al., 2015). Maiores detalhes da criação, maneiras de utilização, atualizações, itens lexicais e formas de acesso são exploradas nos trabalhos de Oliveira et al. (2015) e Oliveira et al. (2016).

3.6 Considerações Finais

É crescente o uso da ontologia na recuperação das informações e web semântica, sua relação com este trabalho é relevante como método para categorizar e relacionar os dados inerentes as decisões e sentenças judiciais. O próximo capítulo irá discutir a aplicação da recuperação da informação na ciência da informação e ciência da computação.

CAPÍTULO 4

RECUPERAÇÃO DA INFORMAÇÃO

Neste capítulo serão abordados os conceitos e métodos relacionados a recuperação da informação. É apresentado o termo recuperação da informação no contexto de ciência da informação, bem como os modelos quantitativos aplicados na recuperação da informação. São descritos brevemente os modelos booleano, vetorial e a recuperação da informação com sua aplicação no motor de busca de dados.

4.1 Ciência da Informação

A ciência da informação (CI) é o campo dedicado às questões científicas e à prática profissional voltadas para os problemas da efetiva comunicação do conhecimento e de seus registros entre os seres humanos, no contexto social, institucional ou individual do uso e das necessidades de informação. No tratamento destas questões são consideradas de particular interesse as vantagens das modernas tecnologias informacionais (SARACEVIC, 1996).

Robredo (2005) enfatiza o olhar moderno sobre a CI, não a considerando restrita ao campo da documentação ou evolução desta, mas sendo o estudo de todo tipo de documento e seus desdobramentos (sejam eles aplicados ou práticos) e igualmente as outras ciências que tenham a informação como objeto de estudo ou foco. Saracevic (1996) descreve a interdisciplinaridade da área, entendendo a necessidade de considerar a CI no seu sentido amplo de ciência inter, trans, multi e/ou pluridisciplinar, que faz com que as ciências cognitivas, as ciências da vida, as ciências humanas e sociais e as ciências físicas e exatas se relacionem.

Segundo Borko (1968) a CI é a disciplina que investiga as propriedades e o comportamento da informação, as forças que governam seu fluxo e o meio de processá-la, para otimizar sua acessibilidade e uso. A CI está ligada ao corpo de conhecimentos relativos à origem, coleta, organização, armazenagem, recuperação, interpretação, transmissão, transformação e uso de informação. Ela tem tanto componente de ciência pura, através de pesquisa dos fundamentos, sem atentar para sua aplicação, quanto componente de ciência aplicada, ao desenvolver produtos e serviços. Uma das subáreas da CI é a recuperação da informação, que é a responsável por tratar e recuperar os objetos de dados como: textos; imagens, sons e outros.

4.2 Revocação, Precisão e a Recuperação da Informação

A definição de recuperação da informação (RI) passa por vários aspectos que podem ir desde o usuário até o profissional da informação, por isto sua definição primária é diversificada. Saracevic (1996) descreve a RI como uma das maiores atividades da CI e a maior fonte de relações interdisciplinares com outras áreas. A partir de Mooers (1951), a recuperação da informação se torna refinada e avançada, e após cinquenta anos de evolução fica altamente sofisticada, trazendo maiores graus de interatividade e sendo acompanhada pelos problemas que há na interação homem-máquina. Robredo (2005) reforça dizendo que a RI é o objetivo final do processo de gestão da informação e conhecimento. Esta linha de pensamento é também reforçada por Delicato (2000), que descreve que a principal intenção, ao ser realizada a busca em base de dados, é encontrar documentos que sejam úteis para satisfazer a necessidade da informação, e evitar a recuperação de itens inúteis.

O documento relevante é todo aquele que contém informação relacionada à consulta. O objetivo do sistema de recuperação de informações é comparar a consulta com a coleção e retornar o conjunto de documentos relacionados para o usuário, frequentemente classificados de acordo com sua presumida relevância (DIAS, 2015). Além disto, a RI tornou-se visível e com foco em debates de modo amplo, pois o ser humano, naturalmente tem a necessidade da informação e dos mecanismos populares de busca.

A eficácia do sistema de RI é geralmente avaliada por duas medidas: i) revocação e ii) precisão. Revocação é a proporção de material relevante realmente recuperado do arquivo, enquanto que a precisão é a proporção do material recuperado que é encontrado para atender a necessidade do usuário. Revocação e precisão tendem a variar de forma inversa e é esta a dificuldade em recuperar tudo que é desejado enquanto se rejeita tudo que é indesejável. Estes dois conceitos também são trabalhados por Lancaster (2004) que os define como a designação da capacidade de recuperar documentos úteis (revocação) e a capacidade de evitar documentos inúteis (precisão).

Jones (1999) indica as dificuldades na RI e acrescenta que o próprio termo de busca realizado pelo usuário é apenas algo aproximado da sua real necessidade, pois o usuário ainda não está inteirado, na maioria dos casos, sobre o que realmente quer saber. Enquanto isto, o próprio documento e sua forma descritiva para ser recuperado

é apenas a expressão aproximada do seu real conteúdo. A RI está lidando com dois tipos de informações: i) a informação que o usuário necessita e ii) o conteúdo do documento. Além disto, a RI lida com a relação de relevância que ambos podem ter para a necessidade do usuário. A relevância é determinada pelo usuário ao fazer as conexões entre o conteúdo do documento e a sua própria necessidade de informação. Desta forma, a mediação realizada pela RI é entre dois fornecedores de informação, aquele que busca (o usuário) e aquele que fornece a resposta para esta busca (o documento).

4.2.1 Objetivo da Recuperação da Informação

Objetiva-se na RI encontrar e apresentar rapidamente a informação correta, do conteúdo do *corpus* do documento ao usuário, satisfazendo sua necessidade frente a expressão de busca realizada. A Figura 4.1, adaptada de Ferneda (2005), representa de forma simplificada o processo de recuperação da informação.

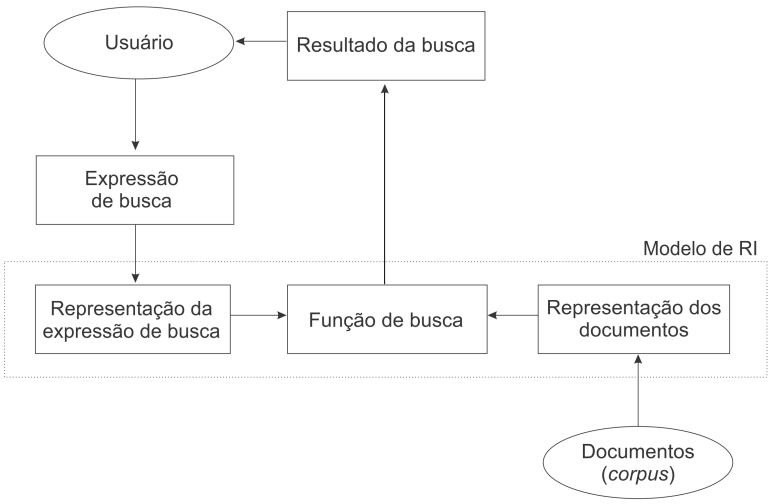

Figura 4.1 - Modelo do processo de recuperação da informação.

O modelo de RI, Figura 4.1, é a especificação formal de três elementos: i) representação dos documentos, ii) representação da necessidade de informação utilizando a expressão de busca e iii) função de busca.

Apesar de usar o termo usuário, é relevante informar que esta entidade pode ser tanto o ser humano quanto mecanismo de software, neste caso faz-se referência a

inteligência artificial (IA). Porém, em ambos os casos a dificuldade no processo de recuperação da informação está, na maioria das vezes, na forma como a busca é realizada no *corpus*, frente a expressão de busca informada pelo usuário ou por mecanismo de software inteligente. A expressão de busca geralmente é composta de palavras que tentam exprimir a semântica da necessidade de informação do usuário/sistema, assim, parte da responsabilidade da eficácia do processo de recuperação da informação é da entidade que inicia o fluxo na Figura 4.1. Por isto, a ontologia beneficia o processo de RI, visto que a modelagem, estruturação e inferências na informação são realizadas pela perspectiva do usuário.

4.3 Modelos quantitativos aplicados na Recuperação de Informação

Na Figura 4.1, a função de busca é um dos pontos de maior relevância, ela compara a expressão de busca dos usuários/sistemas com o *corpus*, recuperando itens que provavelmente são os corretos. A maioria das funções de busca é de natureza quantitativa, baseada em disciplinas como i) lógica, ii) estatística e iii) teoria de conjuntos.

A eficiência do sistema de RI está diretamente ligada ao modelo aplicado. Estes modelos de RI são conhecidos também como motores de busca. Alguns motores de buscas são criados nos anos 1960 e 1970, e aperfeiçoados nos anos 1980 e ainda são utilizados nos dias atuais, suas ideias estão presentes na maioria dos sistemas atuais de recuperação, bem como nos mecanismos de busca na internet. Ferneda (2005) e Martins (2010) afirmam que os modelos quantitativos impulsionaram o desenvolvimento do sistemas de recuperação da informação, dentre eles têm-se: i) booleanos, ii) vetoriais, iii) probabilísticos e iv) *clustering*.

Todos os modelos quantitativos trabalham na recuperação de documentos baseado em termos inseridos na entrada do processo, desta forma, pode-se considerar que cada documento no *corpus* é representado pelo conjunto de termos. Assim, a forma de construir a relação, a inferência, o grau e o peso dos termos $\times$ documentos é a essências dos modelos apresentados. O termo é a palavra que representa o conceito ou significado presente no documento. Identificar a relevância do termo na descrição do conteúdo do documento não é tarefa fácil, mesmo para o projetista do conhecimento que está acostumado a lidar com o negócio.

4.3.1 Modelo Booleano

Criado a partir da lógica de Boole, que se inspirou na distinção de verdadeiro e falso de Aristóteles, o modelo booleano assume que informações podem ser processadas de modo binário, 1 ou 0. Aristóteles, utilizando símbolos para representar expressões substantivas que fazem ligação com as demonstrações matemáticas, inspira Boole a criar sistema de símbolos e regras aplicáveis, desde números até enunciados. Com Boole, a lógica afasta-se da filosofia e aproxima-se da matemática (FACHIN, 2009).

São criados os operadores booleanos **and**, **or** e **not**, estes operadores são definidos: i) **and** estabelece a preferência por documentos onde tem a junção de dois ou mais termos, desta forma, caso o usuário busque termos t_1 **and** t_2 serão recuperados todos os documentos que contenham os termos t_1 e t_2, como ilustra a Figura 4.2, ii) **or** recupera documentos indexados por um ou outro termo, unindo-os, assim, se o interesse do usuário é buscar os termos t_1 **or** t_2 serão recuperados todos os documentos com o termo t_1 e todos os documentos com o termo t_2, como ilustra a Figura 4.3, iii) **not**, faz-se a exclusão do termo ou termos que o usuário não deseja recuperar, então se o usuário buscar t_1 **not** t_2, serão recuperados os documentos com o termo t_1 e serão excluídos os documentos que contenham o termo t_2, mesmo que nestes documentos contenham o termo t_1, como ilustra a Figura 4.4.

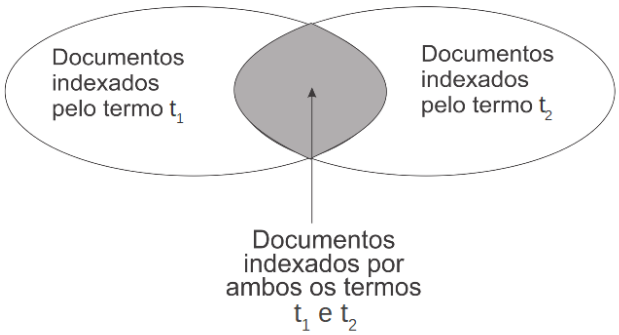

Figura 4.2 - Resultado da busca booleana com operador **and**.

O modelo booleano é relevante dentre outros modelos quantitativos, e o auxílio do parênteses agrega a ele importante capacidade ao entendimento da lógica, sendo

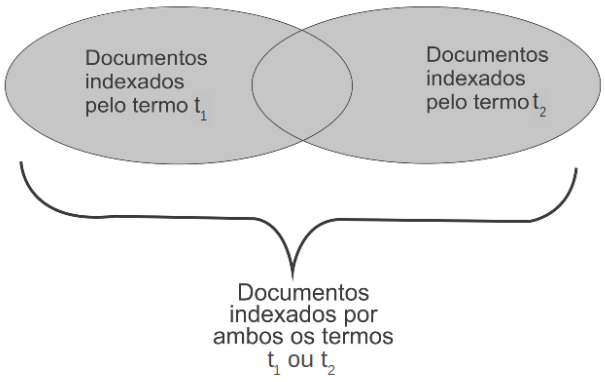

Figura 4.3 - Resultado da busca booleana com operador **or**.

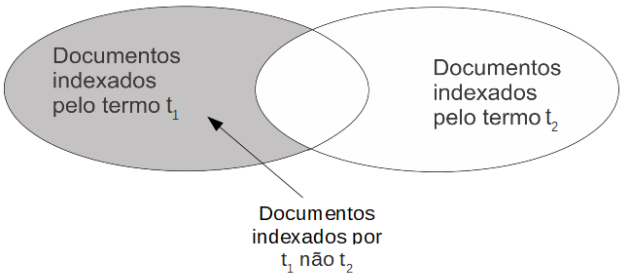

Figura 4.4 - Resultado da busca booleana com operador **not**.

que possibilita isolar termos ou unir outros, por exemplo: (inteligencia artificial **and** computação) **and** (Goiás **or** Distrito Federal) o que deve recuperar documentos sobre inteligência artificial na área de computação e com alguma relação com o Goiás ou com o Distrito Federal. A desvantagem apresentada neste modelo refere-se a incapacidade de permitir a atribuição de peso entre os termos ou ordenar os resultados por relevância para o usuário (DIAS, 2015). Salton (1986) informa que para melhorar o modelo booleano, faz-se necessário o mecanismo de aplicação de pesos para aumentar a precisão na busca, distinguindo os termos menos importantes dos mais importantes. Quando o sistema é capaz de atribuir peso nos termos, os algoritmos e funções matemáticas são capazes de entender esta relação termos $\times$ pesos, considerando a importância de cada termo.

4.3.2 Modelo Vetorial

Na tentativa de resolver o problema do modelo booleano, quanto a atribuição de pesos nos termos, é aplicado o modelo vetorial. Este modelo consegue associar pesos tanto nos termos de indexação quanto nos termos da expressão de busca. Estes pesos são utilizados para calcular o grau de similaridade entre expressões de busca estabelecida pelo usuário em cada documento. Assim, é possível obter os documentos ordenados pelo grau de similaridade frente a expressão de busca.

O modelo vetorial pode ser representado por vetor, em que cada elemento representa o peso ou a relevância do termo de indexação para o documento. Cada elemento do vetor é normalizado objetivando assumir valores entre 0 e 1, sendo que quanto mais próximo de 1, maior a importância do termo no documento. A Figura 4.5 ilustra o documento D_1 com termos t_1 e t_2, com pesos 0,4 e 0,7, respectivamente. Na Figura 4.5, como cada termo representa o eixo ou dimensão, o vetor descreve a posição do documento no espaço multidimensional.

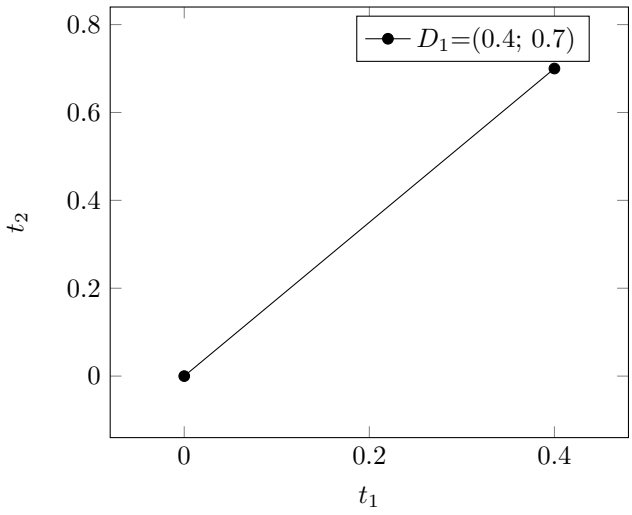

Figura 4.5 - Representação vetorial do documento com dois termos de indexação.

A Figura 4.6 ilustra dois documentos no mesmo espaço vetorial: D_1 e D_2, sendo os pesos dos termos t_1 e t_2 para $D_1 = (0,4; 0,7)$ e os pesos dos termos t_1 e t_2 para $D_2 = (0,6; 0,9)$. Da mesma forma, na Figura 4.7, é utilizada a mesma representação para os documentos e para a expressão de busca, ou seja, no mesmo espaço vetorial t_1 e t_2 são incluídos na expressão de busca $B_1 = (0,6; 0,4)$ junto com os documentos

D_1 e D_2. Esta expressão de busca é representada por vetor númerico, igual aos documentos, onde cada elemento tem seu peso na busca.

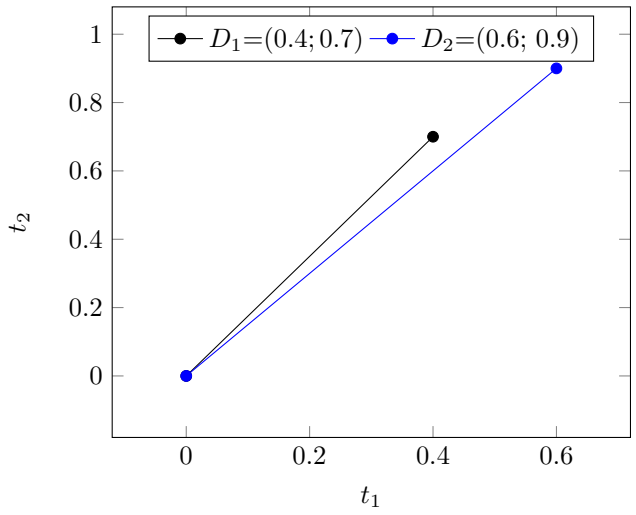

Figura 4.6 - Representação vetorial com dois documentos.

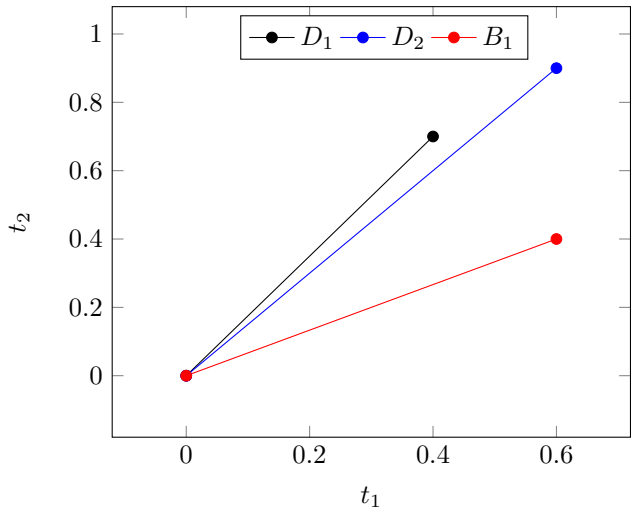

Figura 4.7 - Representação vetorial com dois documentos e expressão de busca.

No espaço vetorial bidimensional da Figura 4.7, a similaridade S_m entre o vetor da expressão de busca e os vetores de cada documentos, ou seja, a similaridade de B_1 com D_1 é calculada através do *cosseno* do ângulo formado pelos vetores. Assim, no modelo vetorial o resultado da busca é o conjunto de documentos ordenados pelo

grau de similaridade entre cada documento e a expressão de busca. Se for definido limite mínimo para o valor da similaridade, haverá restrinção na quantidade de documentos baseado neste limite.

Nos exemplos das Figura 4.5, Figura 4.6 e Figura 4.7, foram incluídos apenas dois termos para cada documento, porém em sistemas reais são vários números de termos de indexação por documento, gerando figura multidimensional. Antes de realizar o cálculo de similaridade entre os pesos dos termos nos vetores das expressões de busca e dos documentos, faz-se necessário preparar o *corpus*. Para Ferneda (2005) são necessárias algumas etapas importantes no procedimento de identificação dos termos no modelo vetorial, são elas: i) identificação e isolamento de cada palavra do documento ou de suas representações como resumos e/ou palavras-chaves, ii) eliminação de palavras com elevada frequência de aparecimento no documento e com pouco valor semântico, como artigos e preposições, iii) redução das palavras restantes ao seu radical, retirando sufixos e prefixos, iv) incorporação dos termos aos vetores dos documentos e v) atribuição do valor do peso para os termos. O cálculo do peso é outro aspecto que deve ser trabalhado nos modelos vetoriais, pois pode-se simplesmente inserir os pesos dos termos manualmente, porém necessitaria de pessoal especializado trabalhando durante certo tempo. Outra forma é contabilizar a quantidade de vezes que o termo aparece no *corpus* e utilizar esta frequência para identificar os pesos dos termos.

No entanto, existe o modelo *System for the Manipulation and Retrieval of Text* (SMART), construído por Gerard Salton nos anos 1960, que fornece método automático para o cálculo dos pesos, tanto das palavras nos documentos, quanto das expressões de busca, sendo a forma de cálcular descrita por Mcgill (1983). Inicialmente define-se a frequência do termo $t_f(D_j, t_i)$ como sendo o número de vezes que o termo t_i aparece no texto do documento D_j, com $i = 1, 2, \cdots, n$ e $j = 1, 2, \cdots, m$, onde n é o número de termos de indexação e m é o número de documentos.

Não é possível, apenas com a medida $t_f(D_j, t_i)$ saber se os termos ocorrem em todos os documentos do *corpus* ou várias vezes em apenas alguns documentos. Intuitivamente pode-se observar que se o termo aparecer em todos os documentos terá pouca utilidade na identificação da relevância dos documentos. Para atribuir maior precisão ao peso do termo, faz-se necessário realizar a estatística global que estime o termo em relação ao *corpus*. Esta medida $I_{df}(t_i)$, *inverse document frequency* (IDF), desenvolvida no modelo SMART é dada por:

$$I_{df}(t_i) = \log \frac{N}{n_t} \tag{4.1}$$

onde N é o número de documentos no *corpus* e n_t é o número de documentos que contém o termo t_i. Assim, quanto menor o número de documentos que contêm o termo, maior o $I_{df}(t_i)$ do termo. O $I_{df}(t_i) = 1$ se todos os documentos tiverem o termo t_i. A partir desta implementação, define-se o peso do termo t_i em relação ao documento D_j, que é $w_{(j,i)}$, pela multiplicação da medida $t_f(D_j, t_i)$ pela medida $I_{df}(t_i)$, dado por:

$$w_{(j,i)} = t_f(D_j, t_i) \cdot I_{df}(t_i) \tag{4.2}$$

Com este modelo os melhores termos de indexação, que tem maior peso, são aqueles que ocorrem com alta frequência em poucos documentos. Ferneda (2005) afirma que o modelo SMART ainda é referência em RI, conseguindo desempenho acima da média em relação a outras soluções.

De posse da expressão (4.2) é possível construir a matriz disposta na Tabela 4.1, que representa a coleção: i) documentos D_j, ii) termos de indexação t_i e iii) pesos $w_{(j,i)}$. A Tabela 4.1 dispõe o *corpus* com n termos de indexação $t_i = t_1, t_2, \cdots, t_n$ e m documentos $D_j = D_1, D_2, \cdots, D_m$. Cada documento é o vetor $D_j = w_{(j,1)}, w_{(j,2)}, \cdots, w_{(j,n)}$, na qual o valor de $w_{(j,i)}$ é a medida de ocorrência ou frequência do i-ésimo termo no j-ésimo documento do *corpus*. Na Tabela 4.1, as linhas representam os documentos e as colunas os termos nos documentos, os valores indicam a frequência do termo no documento.

Tabela 4.1 - Representação da associação dos termos aos documentos.

	t_1	t_2	t_3	...	t_n
D_1	$w_{(1,1)}$	$w_{(1,2)}$	$w_{(1,3)}$	...	$w_{(1,n)}$
D_2	$w_{(2,1)}$	$w_{(2,2)}$	$w_{(3,2)}$	...	$w_{(2,n)}$
...	...	...	...	...	...
D_m	$w_{(m,1)}$	$w_{(m,2)}$	$w_{(m,3)}$	...	$w_{(m,n)}$

Desta forma, calcula-se a similaridade S_m objetivando quantificar a semelhança de conteúdo entre dois documentos ou entre a expressão de busca em cada um dos

documentos. Apesar do cosseno ser apresentado como exemplo para aplicação do cálculo de similaridade S_m no modelo vetorial, existem outros modelos para definir a similaridade, como: i) modelo de Jaccard, ii) coeficiente de Dice e iii) outros. A similaridade utilizando o cálculo do cosseno é uma das medidas mais utilizadas para encontrar o valor de S_m entre dois documentos e é dado por:

$$S_{mc}(D_1, D_2) = \frac{\sum_{i=1}^{n} w_{(1,i)} \cdot w_{(2,i)}}{\sqrt{\sum_{i=1}^{n} [w_{(1,i)}]^2} \cdot \sqrt{\sum_{i=1}^{n} [w_{(2,i)}]^2}} \tag{4.3}$$

Este modelo é indicado para espaços de alta dimensionalidade e esparsos, ou seja, elevados valores de m e n. A similaridade de Jaccard S_{mj} é medida utilizada na estatística e é dada por:

$$S_{mj}(D_1, D_2) = \frac{\sum_{i=1}^{n} (w_{1,i} \cdot w_{2,i})}{\sum_{i=1}^{n} (w_{1,i})^2 + \sum_{i=1}^{n} (w_{2,i})^2 - \sum_{i=1}^{n} (w_{1,i} \cdot w_{2,i})} \tag{4.4}$$

Nas expressões (4.3) e (4.4) são retornados valores entre 0 e 1. Igualmente a similaridade do cosseno S_{mc}, a similaridade de Jaccard S_{mj} é indicado para comparar elevados valores de m. Nas expressões (4.3) e (4.4), $w_{(1,i)}$ é o peso do i-ésimo termo do vetor D_1 e $w_{(2,i)}$ é o peso do i-ésimo termo do vetor D_2. Com a criação dos termos e pesos baseados em domínio ontológico estabelecido, com o cálculo da similaridade no modelo vetorial, é possível classificar cada documento no *corpus* pela sua relevância. Desta forma, é possível identificar os documentos mais apropriados para cada subdomínio e criar o relacionamento pertinente entre o uso da ontologia e da RI.

4.4 Vetorização de Documentos em Texto

Os documentos precisam ser transformados em estrutura de dados para serem entendidos por softwares. A vetoriação é o processo de transformação do documento em vetor com uma ou várias dimensões. Este processo também tem o nome de construção de *features*, construção de pesos para os termos, ponderação dos termos e interfere na precisão e na recuperação dos resultados dos sistemas de processamento

de linguagem natural (PLN) (XIA; CHAI, 2011).

Na literatura, diversos trabalhos de vetorização de termos aplicam o conceito de saco-de-palavras (MURPHY, 2013), (GARG et al., 2011), (BOSCH et al., 2008), (LAZEBNIK et al., 2006), (FERGUS et al., 2003), outros saco-de-conceitos (LI et al., 2020), (MIKOLOV et al., 2013), (MILNE et al., 2007), (MIHALCEA et al., 2006), (GABRILOVICH; MARKOVITCH, 2005) e outros aplicam ambas as soluções (LI et al., 2020), (KIM, 2014), (HUANG et al., 2012). No modelo saco-de-palavras, o documento é transformado em vetor de tamanho n, em que n é o número de palavras usadas para representar o documento, cada campo vetorial está associado ao valor da palavra, calculado por métodos tradicionais como: i) frequência de termo tf, ii) frequência do termo-inverso da frequência nos documentos $tf - idf$ e iii) Okapi *best matching* 25 $BM25$.

No modelo de saco-de-conceitos, o documento é transformado em espaço vetorial de tamanho $n \times d$, em que n é o número de palavras e d é o número de dimensões, considerando que a dimensão d da palavra é geralmente definida pela própria palavra e pelas palavras que a acompanham no texto, criando a coocorrência entre as palavras. Além disto, se a representação do documento for separada por suas sentenças, o espaço vetorial pode ser expandido para $s \times n \times d$, onde s é o número de sentenças no documento. Os métodos conhecidos e aplicados na construção de saco-de-conceitos são word2vec (MIKOLOV et al., 2013), GloVe (PENNINGTON et al., 2014), ELMo (PETERS et al., 2018), BERT (DEVLIN et al., 2018) e outros.

Trabalhos recentes como Agarwal et al. (2020), Seo et al. (2020) e Li et al. (2020) informam as desvantagens na utilização do modelo saco-de-palavras, visto as limitações na aplicação do modelo *tf-idf*, demonstrando que o modelo não pode estabelecer pesos na coocorrência entre os termos nos documentos. Agarwal et al. (2020) compara duas soluções de vetorização, *tf-idf* e *lfw*, em documentos de *web services*, no formato WSDL, e aplica o modelo *k-médias* para agrupá-los. Seo et al. (2020) usa o modelo *tf-idf* para identificar termos incomuns no banco de dados de respostas do cliente. Li et al. (2020) propõe nova forma de aplicar saco-de-conceitos, usando base de conhecimento externa em conjunto com a análise multidimensional do texto, gerando dimensão adicional com base em conhecimento externo, embora os melhores resultados de Li et al. (2020) são obtidos aplicando o modelo de saco-de-conceitos em conjunto com saco-de-palavras.

Entretanto, no trabalho de Mandal et al. (2021), os autores observam que métodos

tradicionais, como *tf-idf* e Alocação Latente de Dirichlet (*Latent Dirichlet Allocation* – LDA), que dependem da representação de saco-de-palavras, têm melhor desempenho que métodos sensíveis ao contexto, como BERT e Law2Vec (CHALKIDIS, 2021). Observa-se nestes trabalhos que cada cenário ou área de atuação, o *corpus* possui características que requerem elevado ou baixo custo de processamento, bem como aplicação de técnicas simples ou complexas. Assim, soluções simples, com menor tempo de processamento podem atingir o objetivo da aplicação (CASTRO et al., 2017a), (CASTRO et al., 2017b). Gomaa e Fahmy (2013) informam que alguns modelos de construção de peso são tradicionalmente pesquisados e aplicados, como: i) tf, ii) $tf - idf$ e iii) $BM25$.

4.5 Classificação de documentos

Após a utilização de modelos de saco-de-palavras ou de saco-de-conceitos que conseguem representar documentos de texto em vetores unidimensionais ou multidimensionais, construindo os conjuntos de dados de treinamento, algoritmos de aprendizado de máquina podem ser treinados para reconhecer e classificar novos documentos, informando qual a categoria que ele pertence.

São vários os algoritmos que podem ser utilizados na classificação de documentos, como os tradicionais: floresta aleatória, redes neurais perceptron multicamadas, impulso adaptativo, reforço de gradiente, processo gaussiano, máquinas de vetores de suporte, Naive Bayes, k-vizinhos mais próximos, árvores de decisão, regressão logística, passivo-agressivo, entre vários outros. Cada algoritmo de classificação precisa de número mínimo de iterações na sua fase de treinamento para que tenha acurácia mínima na sua predição.

4.6 Considerações Finais

Várias são as aplicações da recuperação da informação na ciência da computação, ciência da informação e representação do conhecimento. As funções de busca são mecanismos essênciais para que a RI tenha sucesso no atendimento da demanda imposta pelo usuário. Este capítulo abordou os conceitos de CI, RI e no próximo capítulo será tratado da metodologia aplicada, sendo possível verificar como a ontologia pode ser utilizada na criação de modelo semântico relacionado as decisões e sentenças já proferidas no Judiciário.

CAPÍTULO 5

METODOLOGIA

Neste capítulo será apresentada a metodologia utilizada para o desenvolvimento deste trabalho. Serão descritas o levantamento de dados, a definição do escopo, as abordagens utilizadas para a construção da ontologia, definição das métricas utilizadas para medir os resultados, descrição do software de simulação, das bases de dados utilizadas, da metodologia de análise de mineração de dados e do modelo utilizado para criação da ontologia.

5.1 Levantamento de dados e definição do escopo

A metodologia desenvolvida neste trabalho tem como objetivo principal proporcionar agilidade nos julgamentos dos processos judiciais, pois identifica e classifica as ações em análise, permitindo a conexão com os processos julgados. O método proposto colabora com pesquisas na área de ontologia e aprendizado de máquina aplicado na classificação de textos, pois, inova em: i) resolver a limitação do modelo $tf - idf$ apresentado por Agarwal et al. (2020), Seo et al. (2020) e Li et al. (2020), ii) aplicar a coocorrência entre termos, atendendo de forma automática os componentes de relacionamento entre atributos de determinada categoria e iii) reduzir a não linearidade de termos semelhantes em diferentes categorias de documentos.

Inicia-se a metodologia proposta realizando o levantamento dos sistemas do Judiciário e as estruturas de dados (diagramas de entidades e relacionamentos) dos ambientes que permitem a pesquisa textual no banco de dados de sentenças e decisões na internet. Na Figura 5.1 é ilustrado o fluxograma de tarefas necessárias para concepção do ambiente de testes e simulação para mineração de dados usando *Ruby-on-Rails* e *PostgreSql*. Nestes ambientes são migrados tantos os dados do inventário de processos do primeiro grau de jurisdição, quanto a base de dados de sentenças/decisões.

De posse da estrutura de simulação, é iniciada a etapa de construção da ontologia jurídica. Para esta definição semântica é necessário estabelecer o escopo de atuação, visto o elevado volume de dados e naturezas processuais existentes no Judiciário. Para os primeiros testes, estabelece a criação da ontologia para as naturezas processuais de revisional e consignatória (contratos). Esta decisão é baseada no volume de processos judiciais existentes destas duas classes processuais. Depois são adicio-

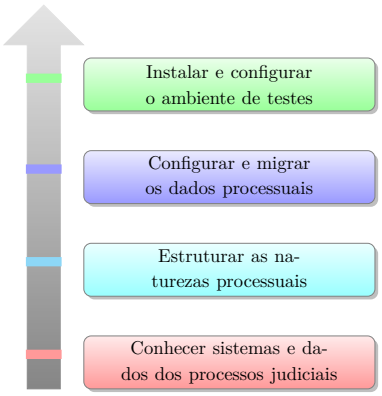

Figura 5.1 - Fluxo com estudo e preparação para construção do ambiente de testes.

nadas outras naturezas processuais. São utilizadas três abordagens de ontologia: i) ontologia com regras do especialista, ii) ontologia sem regras do especialista e iii) ontologia híbrida, com e sem as regras do especialista.

Para a construção da ontologia **com regras do especialista**, o conhecimento jurídico é utilizado para modelar os indivíduos, classes, atributos e relacionamentos. A ferramenta utilizada neste modelo é o software *Protegé*. Para a construção da ontologia **sem regras do especialista**, leva-se em conta que o processo é automático, sem intervenção humana. A nomenclatura **sem regras do especialista** é utilizado por fazer uso de modelo matemático e computacional para estruturar a matriz de similaridade de palavras, objetivando encontrar regra para estabelecer relacionamento entre as palavras, dentro do universo de decisões/sentenças de determinada categoria. Ainda, na metodologia proposta é utilizado o termo **híbrido** para empregar a ontologia com e sem regras do especialista. Para análise e verificação do aperfeiçoamento na aplicação da ontologia é utilizado banco de dados léxico externo, permitindo estabelecer relações semânticas entre as palavras.

5.2 Proposta de construção da ontologia automática

O modelo de ontologia sem a aplicação das regras do especialista é chamado neste trabalho de **ontologia automática** e é descrito resumidamente pelo Algoritmo 1, na qual ϕ é o limitador dado em porcentagem. O objetivo deste limitador é reduzir ou evitar a repetição de possíveis termos combinados em diferentes categorias de documentos, a fim de amenizar a não linearidade do problema, bem como reduzir o tamanho dos conjuntos de dados e os tempos de treinamento e testes. São realizados

testes com e sem limitador para averiguar o desempenho dos modelos de classificação. Neste processo é utilizado o modelo de recuperação da informação (RI), quantitativo vetorial.

Algoritmo 1: Resumo da aplicação da ontologia sem regras do especialista.

categorias$\leftarrow [contratos, posse, divórcio, previdenciário, ..., n]$

i$\leftarrow 0$

while $i < n$ **do**

$\Longrightarrow$ Calcular o peso dos termos, dentro do universo do inteiro teor das sentenças/decisões de categorias[i].

$\Longrightarrow$ Separar n termos com maior peso deste conjunto.

$\Longrightarrow$ Calcular a similaridade dos n termos.

$\Longrightarrow$ Separar a coocorrência dos termos com coeficiente de similaridade $\geq \phi$. Esses termos combinados $n \times n$ são utilizados na ontologia sem as regras do especialista.

$\Longrightarrow$ Usar esses termos combinados para gerar os conjuntos de dados para treinamento supervisionado e testes dos modelos de classificação.

i$\leftarrow i + 1$

end

A Figura 5.2 ilustra o fluxo de aplicação da ontologia sem regras do especialista, na qual as etapas implementadas são: i) conversão de arquivo binário para ASCII dos documentos judiciais assinado, ii) preprocessamento para remover elementos HTML e termos desnecessários, iii) separação de termos usando o modelo $tf - idf$, iv) aplicação da métrica de similaridade para encontrar a coocorrência de termos dois a dois e estabelecer o peso da coocorrência, v) criação de vetores binários e de frequência como representação dos documentos e vi) treinamento supervisionado de modelos de classificação.

5.2.1 Conversão de arquivos e preprocessamento

A primeira fase consiste na preparação dos documentos para a fase de preprocessamento. Os documentos das ações judiciais eletrônicas são assinados usando certificado digital e armazenados em banco de dados do tipo *blob*, em formato binário. Nesta etapa, é necessário retirar a assinatura digital dos documentos eletrônicos e transformá-los em formato de texto ASCII, em UTF-8. O resultado após a conversão é o documento em texto, no formato HTML ASCII, UTF-8.

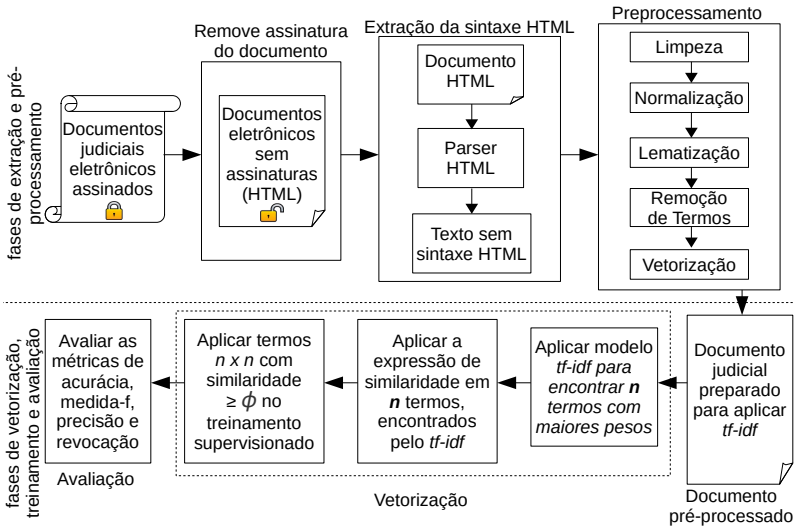

Figura 5.2 - Fluxograma da metodologia proposta.

O preprocessamento é realizado utilizando rotinas construídas em linguagem Ruby-on-Rails. No entanto, na atualidade, não há nenhuma biblioteca específica para tratar textos em sentenças da área jurídica em português para Python ou Ruby. Portanto, os métodos para as etapas de limpeza, normalização, lematização e remoção de termos são construídas e possivelmente disponibilizada no Github.

Os métodos de classe aplicados são: i) **HTML parser**: nesta etapa são eliminados os elementos HTML, ii) **limpeza**: elimina caracteres especiais, como: &, #, " ", ’ ’, ª, º, ( ), [ ], { }, |, /, ;, :, - entre outros, iii) **normalização**: todas as palavras são transformadas para minúsculas, os acentos são removidos, caracteres como ç são substituídos pela letra c e palavras diferentes com o mesmo significado jurídicos são tratadas, iv) **lematização**: reduz as palavras/termos flexionados ou derivados à sua base, v) **remoção de termos**: remove palavras como artigos, preposições, conjunções e palavras sem sentido, vi) **dicionarização**: substituição das palavras/termos por seus sinônimos em dicionários e vii) **vetorização**: transforma o texto em vetor de palavras ou termos, cada documento é transformado em vetor de termos, nas quais as palavras repetidas são mantidas dentro do vetor.

5.2.2 Identificação dos termos para aplicação do método

Nesta etapa são levantados os pesos dos termos dados pelo modelo $tf - idf$, no *corpus* de documentos em texto não estruturados em determinada categoria D, a

matriz de documentos × termos é estruturada pelo cálculo dos pesos dos termos em D dado pelo modelo *tf-idf*. Os pesos encontrados para cada um dos termos, em cada documento, são somados para chegar ao peso total de cada termo em todos os documentos, dado por:

$$W_{t_i}(t_i) = \sum_{i=1}^{m}(p_{t_i}) \tag{5.1}$$

no qual m é o total de documentos em D, p_{t_i} é o peso do termo t_i no documento D_i, sendo $i = 1, 2, \cdots, m$. Os resultados dos cálculos dos pesos para os termos dado por (5.1) são usados para identificar os n termos com maiores W_{t_i}. Estes n termos são usados na matriz termo × termo. Com o objetivo de comparar o modelo *tf-idf*, é aplicado também outro modelo para estabelecer pesos nos termos, o *Okapi BM25*. O *BM25* é modelo tradicionalmente aplicado em técnicas de saco-de-palavras. Esta comparação permitirá identificar qual o melhor modelo a ser utilizado em ambiente de produção no Judiciário.

5.2.3 Modificação da técnica de aprendizagem por similaridade para identificar os pesos na coocorrência dos termos

Este trabalho modifica a forma tradicional em aplicar a expressão de similaridade em (4.4), com objetivo de gerar conhecimento e identificar o *corpus* de determinada categoria. Conforme as técnicas de recuperação da informação, as expressões matemáticas para calcular similaridade são utilizadas, normalmente, para comparar documentos, D_1 e D_2, ou comparar expressões de busca e documentos. Neste trabalho, as expressões que calculam similaridade são alteradas para encontrar similariade entre n termos, t_1 e t_2, nos documentos D de determinada categoria. Os n termos usados para representar a categoria de documentos são aqueles que possuem os valores mais elevados de W_{t_i}, pelo modelo *tf-idf*. A expressão de Jaccard em (4.4) é alterada para construir a relação entre termos, encontrando a coocorrência deles para representar D. A expressão alterada de Jaccard S_α, é dada por:

$$S_\alpha(t_1, t_2) = \frac{\sum_{i=1}^{m}(w_{i,1} \cdot w_{i,2})}{\sum_{i=1}^{m}[(w_{i,1})^2 + (w_{i,2})^2] - \sum_{i=1}^{m}(w_{i,1} \cdot w_{i,2})} \tag{5.2}$$

Em (5.2), a similaridade entre os termos t_1 e t_2 é calculada em todos os documentos D do *corpus*, para a mesma categoria, onde t_1 é o primeiro termo e t_2 é o segundo termo, $w_{i,1}$ é o *tf-idf* do termo t_1 no i-ésimo documento, e $w_{i,2}$ é o *tf-idf* do termo t_2 no i-ésimo documento. Este processo é repetido, calculando a similaridade entre todos os n termos $t_j = t_1, t_2, \cdots, t_n$. Depois de calcular o coeficiente da coocorrência de todos os n termos em todos os m documentos em D, é possível inferir a relação de similaridade dos termos no *corpus* de determinada categoria, como disposto na Tabela 5.1. Considera-se a relação entre os termos t_1 e t_2 construída de forma automática, sendo ela parte dos componentes integrantes da ontologia.

Tabela 5.1 - Coocorrência de termos construída automaticamente, dado por S_α (5.2).

	t_1	t_2	t_3	$\cdots$	t_n
t_1	–	$S_\alpha(t_1, t_2)$	$S_\alpha(t_1, t_3)$	$\cdots$	$S_\alpha(t_1, t_n)$
t_2	$S_\alpha(t_2, t_1)$	–	$S_\alpha(t_2, t_3)$	$\cdots$	$S_\alpha(t_2, t_n)$
t_3	$S_\alpha(t_3, t_1)$	$S_\alpha(t_3, t_2)$	–	$\cdots$	$S_\alpha(t_3, t_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_n	$S_\alpha(t_n, t_1)$	$S_\alpha(t_n, t_2)$	$S_\alpha(t_n, t_3)$	$\cdots$	–

A Tabela 5.1 é a matriz hipotética que representa o conhecimento extraído do *corpus* D, de mesma categoria. Observa-se na matriz que a diagonal principal é nula e os valores acima da diagonal principal são idênticos aos valores abaixo da diagonal principal. Os resultados da aplicação da metodologia proposta criam a relação entre os termos, gerando a impressão digital do *corpus* de certa natureza de processos judiciais.

5.2.4 Treinamento supervisionado e modelos de classificação de texto

Após estabelecer o conhecimento do *corpus*, dois tipos de vetores, de coocorrência de termos encontrados pelo coeficiente de similaridade dado por (5.2), são usados para realizar treinamento supervisionado de modelos de classificação de texto. O objetivo é analisar qual dos dois vetores é melhor aplicado para realizar o treinamento supervisionado. Assim, cada documento judicial é representado por um vetor, cada campo em um vetor é representado por uma coocorrência de termos, gerando duas abordagens vetoriais: i) vetor binário e ii) vetor de frequência. Na abordagem de vetor binário, se o documento tem a coocorrência dos termos em seu conteúdo, o campo do vetor recebe um, caso contrário, o campo do vetor recebe zero. Na abordagem do vetor de frequência, se o documento tem a coocorrência dos termos em seu conteúdo,

o menor valor de frequência de um dos termos é inserido no campo do vetor, caso contrário, o campo do vetor recebe zero. Portanto, o vetor é a impressão digital do documento de texto.

Este vetor é a entrada para os modelos de classificação. O número de campos do vetor é o mesmo número da coocorrência de termos, na matriz termo × termo, com o coeficiente $\geq \phi$, encontrado pela expressão de aprendizagem por similaridade em (5.2). Neste trabalho são utilizadas e comparadas nove tecnologias de classificação de texto: i) floresta aleatória, ii) redes neurais perceptron multicamadas (MLPNN), iii) impulso adaptativo, iv) reforço de gradiente, v) processo gaussiano, vi) máquinas de vetores de suporte (SVM), vii) Naive Bayes, viii) k-vizinhos mais próximos e ix) árvores de decisão. Estes modelos são avaliados com relação a sua acurácia, precisão, revocação e medida-f.

O intuito para a aplicação destes nove modelos de classificação é avaliar qual abordagem é mais adequada para o problema em questão. O fluxograma ilustrado na Figura 5.3 detalha a sequência resumida e informada no Algoritmo 1, no qual é aplicado limitador para o coeficiente de similaridade $\geq \phi$ na escolha da coocorrência dos termos.

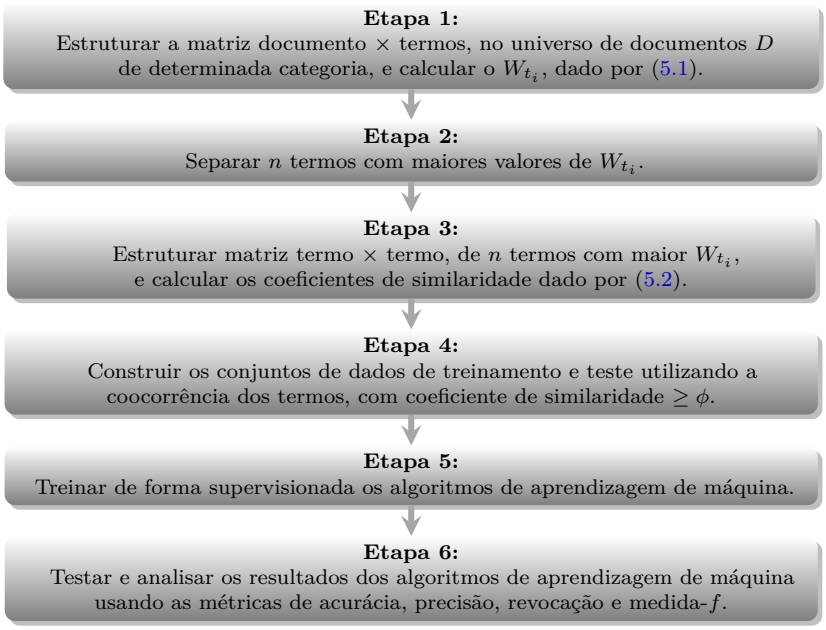

Figura 5.3 - Aplicação do método proposto de ontologia automática para representar os documentos e gerar o conjunto de dados para treinamento e teste dos algoritmos de classificação de texto.

5.3 Métricas de avaliação utilizadas

Para avaliar os resultados dos modelos de classificação aplicados, são utilizadas quatro métricas: acurácia, precisão, revocação e medida-f. A precisão é dada pela porcentagem de instâncias classificadas **corretamente** como positivas dentre todas as que foram classificadas como positivas, dado por:

$$P = \left| \frac{C_P}{C_P + F_P} \right| \tag{5.3}$$

A revocação é dada pela porcentagem de instâncias classificadas **corretamente** como positivas dentre todos os que realmente são positivas, dado por:

$$R = \left| \frac{C_P}{T_P} \right| \tag{5.4}$$

na qual C_P é a **corretamente** positiva, F_P é a **falsa** positiva e T_P são todas as positivas. Quando são realizados os testes de classificação nos documentos, é retornado o subconjunto de documentos, sendo que parte destes documentos estão realmente corretos no resultado e parte não estão corretos. Assim, a variável C_P corresponde a parte retornada de documentos que são realmente corretos, enquanto que a variável F_P corresponde a parte retornada de documentos que estão equivocados. A variável T_P é a relação de todos os documentos corretos dentro do conjunto de todos os documentos.

A acurácia é dada pela porcentagem de instâncias classificadas como corretas (as positivas e as negativas) dentre o número total de casos examinados, dado por:

$$A = \left| \frac{C_P + C_N}{C_P + C_N + F_P + F_N} \right| \tag{5.5}$$

A medida-f é a média harmônica entre a precisão e revocação, dada por:

$$F = \left| \frac{C_P}{C_P + \frac{1}{2} \cdot (F_P + F_N)} \right| \tag{5.6}$$

na qual C_N é a **corretamente** negativa e F_N é a **falso** negativo.

5.4 Ambiente de simulação e unificação das técnicas e modelos

O ambiente de simulação proposto é o Autosent, software construído para realizar a busca e classificação das sentenças judiciais. O Autosent é software WEB com padrão de arquitetura modelo-visão-controlador (*model-view-controller* – MVC) e estrutura de banco de dados relacional com duas tabelas principais: i) tabela inventários que armazena os metadados de todos os processos em tramitação no primeiro grau de jurisdição do Poder Judiciário Goiano e ii) tabela sentenças que armazena o inteiro teor das sentenças e decisões dos processos do primeiro grau de jurisdição do Poder Judiciário Goiano. A Figura 5.4 ilustra a infraestrutura do banco de dados utilizado na simulação.

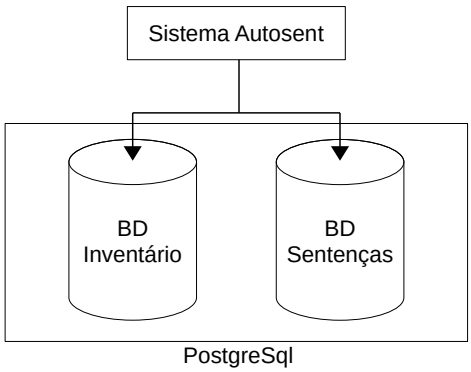

Figura 5.4 - Infraestrutura de Banco de Dados utilizado na Simulação.

A primeira simulação é realizada usando a metodologia proposta sem as regras do especialista. Depois, são aplicadas as regras do especialista e posteriormente o método híbrido. Dadas as metodologias apresentadas, busca-se realizar simulações nos dados reais carregados dos bancos de dados do Autosent. A Figura 5.5 resume as tecnologias aplicadas.

5.5 Considerações finais

A metodologia proposta possibilita realizar testes práticos de comparação entre os métodos aplicados, bem como avaliar os modelos construídos para a ontologia com as regras do especialista, sem as regras do especialista e o método híbrido. No próximo capítulo são apresentados os resultados obtidos da aplicação da metodologia proposta.

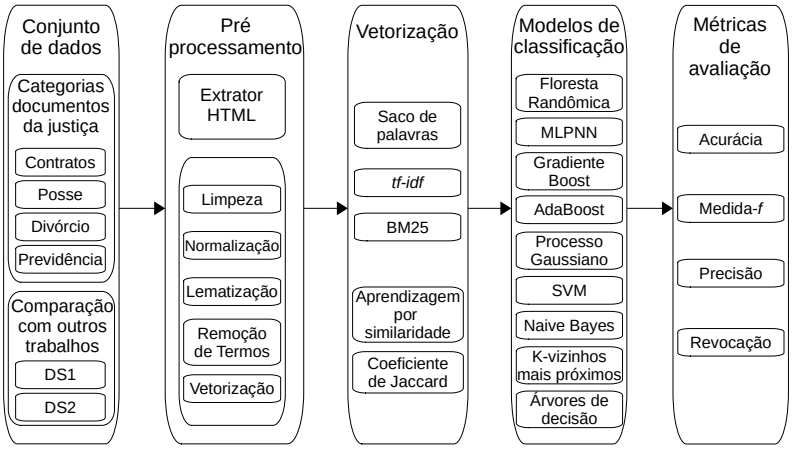

Figura 5.5 - Abordagem metodológica com conjuntos de dados e tecnologias aplicadas.

CAPÍTULO 6

RESULTADOS

Neste capítulo são apresentados os resultados deste trabalho. São apresentados os resultados nas etapas de: i) levantamento de dados e definição do escopo, ii) importação e armazenamento dos documentos, iii) resultados da aplicação do método proposto, iv) comparações com outros métodos e trabalhos de pesquisa, v) aplicação das regras de especialista, vi) interface de programação de aplicativos (API) de integração, vii) validação cruzada e viii) discussão.

6.1 Construção do ambiente e do conjunto de dados

A Figura 6.1 apresenta a sequência de ações para construção do ambiente e coleta dos dados. Inicia-se o trabalho conhecendo os sistemas de controle de andamento processual. Em 2016, o Poder Judiciário Goiano possuia duas estruturas de organização dos processos judiciais, uma para processos físicos (processos em papel) e outra para processos eletrônicos (processo digital). Os sistemas de processos em papel são conhecidos como Sistema de Primeiro Grau (SPG) e Sistema de Segundo Grau (SSG), enquanto que o sistema de processo eletrônico é chamado de Projudi.

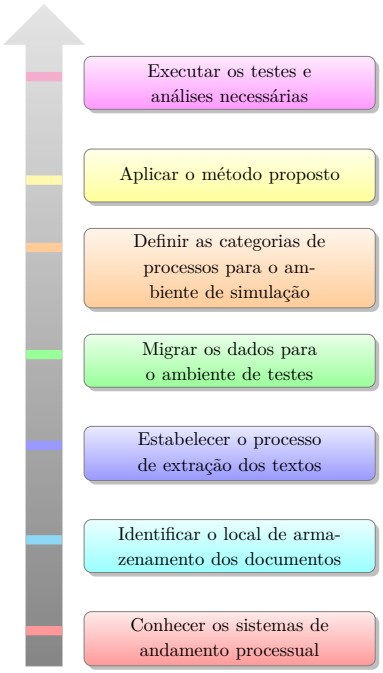

Figura 6.1 - Fluxo com as ações para construção do ambiente e coleta dos dados.

Como a matéria prima deste trabalho são documentos em que seu conteúdo são decisões judiciais, faz-se necessário saber onde estes textos são armazenados e como podem ser extraídos. Nesta fase de estudos dos sistemas, é identificado que o SPG armazena os metadados e os movimentos dos processos, mas não guarda os documentos com as decisões dos juízes. As decisões dos juízes são armazenadas em sistema auxiliar ao SPG, chamado de Sistema de Decisões Monocráticas (SDM). O SDM permite que os magistrados depositem o inteiro teor das suas sentenças, sendo os textos armazenados no formato HTML, UTF-8. Já no sistema Projudi, como o processo é inteiramente eletrônico, todos os documentos do processo judicial estão dentro do seu banco de dados. O Projudi exige a assinatura eletrônica dos documentos, visto não haver mais o uso de papel, assim, as decisões no Projudi são armazenadas com a assinatura eletrônica, no formato binário, em campo *blob* no banco de dados.

Após conhecer os sistemas e onde seus documentos são armazenados, faz-se necessário estudar os relacionamentos entre os campos nas tabelas, bem como sua relação com os documentos. Os documentos relacionam-se com categorias de processos, também conhecida como classes processuais. A orientação aplicada neste trabalho é utilizar pelo menos quatro categorias de processos nas simulações. Procurou-se usar categorias que sejam relevantes para a sociedade e que tenham grandes volumes de processos no Judiciário. As categorias ou naturezas escolhidas, bem como seu volume são dispostos na Tabela 6.1. Existem 581 categorias diferentes em quase dois milhões de processos judiciais no banco de dados do judiciário goiano.

Tabela 6.1 - Categorias de processos utilizadas e sua quantidade no banco de dados.

Descrição	Quantidade
Total de Processos	1.757.132
Contratos	34.495
Posse	12.344
Divórcio	15.246
Previdenciário	63.215

6.2 Importação e armazenamento dos documentos

Conhecidos os sistemas e os locais onde estão armazenados os arquivos no Poder Judiciário do Estado de Goiás, bem como o escopo dos documentos que são aplicados na pesquisa, inicia-se o processo de desenvovimento de rotinas para importação

dos textos de decisões judiciais para banco de dados relacional. Neste trabalho foi utilizado para o banco de dados relacional o PostgreSQL e as linguagens de programação Ruby-on-Rails e Python. Os documentos foram importados, principalmente, do SDM, sendo utilizadas as quatro categorias de processos judiciais: i) contratos, ii) posse, iii) divórcio e iv) previdenciário, como disposto na Tabela 6.2. Os documentos do Projudi, antes de serem importados para o banco de dados do ambiente de simulação, são transformados do seu formato de origem binário para ASCII, UTF-8 e são preprocessados.

Tabela 6.2 - Categorias de processos judiciais e quantidades de documentos importados.

Categorias	Quantidade
Contratos	1.948
Posse	774
Divórcio	838
Previdenciário	5.174

A Tabela 6.1 dispõe documentos de vários tipos produzidos pelos magistrados, como: despachos, decisões de mero expediente e sentenças. As sentenças são documentos mais elaborados, geralmente com mais conteúdos que os outros tipos. Na Tabela 6.2, são utilizados apenas as sentenças de magistrados, escolhidas aleatoriamente.

6.3 Aplicação da ontologia automática

Esta seção implementa de forma automática o modelo proposto na Figura 5.3. Em termos gerais, tem-se a parte de vetorização e classificação dos documentos jurídicos.

6.3.1 Aplicação do *tf-idf* e aprendizagem por similaridade

A Etapa 1 e a Etapa 2 apresentadas na Figura 5.3 são aplicadas no *corpus* de 1.948 decisões, da categoria de contratos. São separados dez termos, $n = 10$, com maiores valores de W_{t_i} no *corpus* da categoria de contratos, sendo que termos comuns ou sem relevância passaram pela fase de preprocessamento, sendo retirados. De posse dos termos, a matriz termo $\times$ termo é estruturada (Etapa 3 da Figura 5.3) utilizando (5.2). Com a criação da coocorrência automática dos termos, a combinação dos termos com coeficiente de similaridade $\phi \geq 50\%$ são utilizados para identificar documentos da categoria de contratos (Etapa 4 da Figura 5.3), como disposto na Tabela 6.3 e Tabela 6.4. O parâmetro $\phi \geq 50\%$ é aplicado com a intenção de avaliar o coeficiente de similaridade médio na separação da coocorrência de termos e o

valor fixo em 50% foi escolhido empiricamente. O objetivo é analisar se há ganhos para o modelo proposto, visto que este limitador irá reduzir ou evitar a repetição da coocorrência dos termos em diferentes categorias de documentos, amenizando a não linearidade do problema, bem como reduzir os conjuntos de dados no treinamento dos modelos de classificação e os tempos de computação. O mesmo processo é aplicado nas outras categorias de documentos: posse, divórcio e previdenciário.

Tabela 6.3 - Coocorrência dos termos indexados nos documentos de decisão- Matriz de Similaridade 1/2.

	juro	*contrato*	*revisional*	*cobrança*	*pagamento*
juro	-	0,85	0,37	0,68	0,39
contrato	0,85	-	0,48	0,67	0,49
revisional	0,37	0,48	-	0,51	0,65
cobrança	0,68	0,67	0,51	-	0,52
pagamento	0,39	0,49	0,65	0,52	-
capitalização	0,79	0,71	0,48	0,78	0,5
valor	0,47	0,61	0,66	0,54	0,69
taxa	0,83	0,74	0,41	0,76	0,41
cláusula	0,53	0,63	0,68	0,67	0,62
crédito	0,49	0,55	0,63	0,65	0,62

Tabela 6.4 - Coocorrência dos termos indexados nos documentos de decisão- Matriz de Similaridade 2/2.

	capitalização	*valor*	*taxa*	*cláusula*	*crédito*
juro	0,79	0,47	0,83	0,53	0,49
contrato	0,71	0,61	0,74	0,63	0,55
revisional	0,48	0,66	0,41	0,68	0,63
cobrança	0,78	0,54	0,76	0,67	0,65
pagamento	0,5	0,69	0,41	0,62	0,62
capitalização	-	0,52	0,76	0,69	0,6
valor	0,52	-	0,47	0,63	0,63
taxa	0,76	0,47	-	0,58	0,53
cláusula	0,69	0,63	0,58	-	0,65
crédito	0,6	0,63	0,53	0,65	-

O objetivo em utilizar dez termos, $n = 10$, é trabalhar com matriz termo $\times$ termo menor, consequentemente tem-se menor tempo de processamento na fase de vetorização, bem como na fase de treinamento e teste dos modelos de classificação.

O mesmo processamento realizado na categoria de contratos é aplicada para todas as outras categorias dispostas na Tabela 6.2. A Etapa 1 e a Etapa 2 apresentadas na Figura 5.3 são aplicadas, encontrando os termos com maiores W_{t_i}, utilizando o método *tf-idf*. De forma a padronizar a aplicação do modelo, o número de termos

é fixado em dez para cada categoria, $n = 10$. A Tabela 6.5 dispõe os 40 termos encontrados, com $n = 10$ em cada categoria de documentos.

Tabela 6.5 - Termos encontrados pelo modelo *tf-idf* nos *corpus* da Tabela 6.2.

Categorias	**Termos**
Contratos	contrato, juro, taxa, valor, capitalização, cobrança, revisional, pagamento, cláusula, crédito
Posse	posse, reintegração, prazo, bem, imóvel, forma, esbulho, prova, contrato, valor
Divórcio	divórcio, prazo, bem, alimento, acordo, assistência, benefício, valor, guarda, litigioso
Previdenciário	benefício, prazo, inss*, federal, rural, concessão, prova, nacional, aposentadoria, idade

* Intituto National de Seguridade Social.

Verifica-se que após aplicar o modelo *tf-idf*, termos iguais são encontrados em diferentes categorias, tornando o problema não-linear. Na Tabela 6.5, tem-se os termos: i) **valor** encontrado nas categorias de **contratos**, **posse** e **divórcio**; ii) **contrato** encontrado nas categorias de **contratos** e **posse**; iii) **prazo** encontrado nas categorias de **posse**, **divórcio** e **previdenciário**; iv) **bem** encontrado nas categorias de **posse** e **divórcio**; v) **prova** encontrado nas categorias de **posse** e **previdenciário**, e vi) **benefício** encontrado nas categorias de **divórcio** e **previdenciário**. Em porcentagem, tem-se que 50% dos termos da categoria de **posse**, também estão em outras categorias. Em mesma análise, tem-se que 40% dos termos na categoria de **divórcio**, que 30% dos termos na categoria de **previdenciário** e que 20% dos termos na categoria de **contratos** estão nas outras categorias estudadas. A Tabela 6.6 dispõe os termos repetidos encontrados e seu número de repetições.

Tabela 6.6 - Termos iguais da Tabela 6.5, encontrados em categorias diferentes de documentos, demonstram que o problema é não-linear.

Termos	**Repetições**
contrato, bem, prova, benefício	2
valor, prazo	3

Depois de encontrar os termos com maiores pesos, segue o fluxo para a Etapa 3, conforme Figura 5.3. Na Etapa 3 é utilizada a similaridade de Jaccard, em (5.2), na matriz termo $\times$ termo, para encontrar o coeficiente de similaridade dos $n = 10$ termos, 2×2, para cada categoria. A coocorrência dos termos com coeficientes de

similaridade $\phi \geq 50\%$ estão dispostos na Tabela 6.7. O propósito em usar o coeficiente $\phi \geq 50\%$ está tanto em usar termos combinados, 2×2, com alto grau de coocorrência no *corpus*, quanto suavizar ou até eliminar a não linearidade do problema de classificar os documentos nas diferentes categorias. Aplicar estes termos combinados com coeficiente de $\phi \geq 50\%$ de similaridade no *corpus* de mesma categoria, estabelece forte laço de classificação, facilitando a resolução do problema de predição para os algoritmos de aprendizagem de máquina.

A Tabela 6.5 dispõe a existência dos termos comuns encontrados pelo modelo *tf-idf* nas diferentes categorias, como bem, valor, prazo, benefício, contrato e outros. Com a aplicação da similaridade em (5.2), o problema torna-se linear, como disposto na Tabela 6.7, pois não há coocorrência de termos repetidos em diferentes categorias de documentos. Esta metodologia diminui tanto o tempo de processamento na construção dos pesos para o método de saco-de-palavras, com a redução da dimensão da matriz termo $\times$ termo, quanto o tempo de processamento no treinamento supervisionado dos métodos de classificação. Este conhecimento é usado na construção dos conjuntos de dados para treinamento e no próprio treinamento dos modelos de classificação utilizados neste trabalho.

Tabela 6.7 - Coocorrência de termos encontrados usando *tf-idf* e similaridade em (5.2) com coeficiente $\phi \geq 50\%$.

Categorias	**Coocorrência de termos encontrados usando *tf-idf* e similaridade em (5.2)**
Contratos	juro-contrato, juro-cobrança, juro-capitalização, juro-taxa, juro-cláusula, contrato-cobrança, contrato-capitalização, contrato-valor, contrato-taxa, contrato-cláusula, contrato-crédito, revisional-cobrança, revisional-pagamento, revisional-valor, revisional-cláusula, revisional-crédito, cobrança-pagamento, cobrança-capitalização, cobrança-valor, cobrança-taxa, cobrança-cláusula, cobrança-crédito, pagamento-capitalização,pagamento-valor, pagamento-cláusula, pagamento-crédito, capitalização-valor, capitalização-taxa, capitalização-cláusula, capitalização-crédito, valor-cláusula, valor-crédito, taxa-cláusula, taxa-crédito, cláusula-crédito
Posse	reintegração-esbulho, reintegração-bem, posse-esbulho, posse-imóvel, posse-bem, posse-reintegração
Divórcio	bem-alimento, divórcio-acordo, divórcio-bem, assistência-benefício
Previdenciário	inss*federal, benefício-nacional, benefício-rural, benefício-idade, prazo-inss, concessão-nacional, prova-aposentadoria, benefício-prova, concessão-prova, inss-aposentadoria, rural-prova, concessão-aposentadoria, inss-concessão, rural-aposentadoria, benefício-aposentadoria, aposentadoria-idade, benefício-inss, inss-nacional, benefício-concessão, rural-idade

* Instituto National de Seguridade Social.

6.3.2 Treinamento e modelos de classificação

Seguindo o fluxo apresentado na Figura 5.3, os termos combinados encontrados por (5.2), na matriz termo × termo com coeficiente de similaridade $\phi \geq 50\%$ são usados no treinamento supervisionado dos métodos de classificação. Estes termos combinados estão dispostos na Tabela 6.7. Neste cenário, o documento de texto do tribunal é representado por vetor, cada campo no vetor é representado pela coocorrência de termos, na combinação 2×2. Este vetor é a entrada para os modelos de classificação de texto. O número de campos no vetor é o mesmo número de termos combinados descobertos pelo método de aprendizado por similaridade aplicado. Os termos combinados dispostos na Tabela 6.7 são transformados em vetor, analisando se no conteúdo do documento jurídico existem as coocorrências dos termos ou não existem.

Duas abordagens de conjunto de dados são geradas a fim de analisar qual é o melhor conjunto de dados para treinamento, quais sejam, vetor binário ou vetor de frequência. Para a geração do conjunto de dados, os documentos de cada categoria disposto na Tabela 6.2 são separados aleatoriamente. O conjunto de dados de treinamento completo é composto por 1.619 documentos, divididos entre as categorias de contratos, posse, divórcio e previdenciário, como disposto na Tabela 6.8. Para gerar o conjunto de dados como vetor binário, aplica-se: se a coocorrência dos termos existir no documento de texto, o campo do vetor é definido como **um**, caso contrário como **zero**. Para gerar o conjunto de dados como vetor de frequência, aplica-se: se a coocorrência dos termos dentro do documento existir no documento, a frequência dos dois termos dentro do documento é contada, e a **frequência mais baixa** entre os dois termos é inserida no campo do vetor, caso contrário o campo do vetor é definido como **zero**.

Tabela 6.8 - Categorias de processos judiciais e quantidade documentos separados aleatoriamente para treinamento.

Categorias	Quantidade
Contratos	440
Posse	371
Divórcio	368
Previdenciário	440

Primeiramente, é realizado o treinamento supervisionado com o conjunto de dados

usando vetor binário, e os resultados são avaliados na base de teste contendo 200 documentos, dos quais 50 são de cada categoria. Após a obtenção dos resultados, é realizado novo treinamento supervisionado com o conjunto de dados com vetor de frequência dos termos combinados, e os resultados são avaliados na base de teste contendo os mesmos 200 documentos. Estes 200 documentos de teste são diferentes dos 1.619 documentos de treinamento. Para o treinamento supervisionado, nove modelos de classificação de texto são usados: i) rede neural perceptron multicamadas (MLP), ii) floresta aleatória (FA), iii) reforço de gradiente (RG), iv) impulso adaptativo (ImA), v) processo Gaussiano (PG), vi) máquina de vetores de suporte (MVS), vii) Naive Bayes (NB), viii) k-vizinhos mais próximos (KVP) e ix) árvores de decisão (AD). Classes da biblioteca *scikit-learn* em Python são usadas para implementar estes métodos.

A biblioteca `sklearn.neural_network` foi usada na linguagem Python para implementar o classificador rede neural MLP. Para o método `MLPClassifier()` da classe `MLPClassifier`, os seguintes parâmetros são usados: `activation='logistic'`, `solver='lbfgs'`, `alpha=1e-5`, `hidden_layer_sizes=(100,50)`, `random_state=1`, `max_iter=300`. A biblioteca `sklearn.ensemble` foi usada na linguagem Python para implementar os classificadores floresta aleatória, AdaBoost e Gradiente Tree Boosting. Para os métodos `RandomForestClassifier()`, `AdaBoostClassifier()` e `GradientBoostingClassifier()`, respectivamente das classes `RandomForestClassifier`, `AdaBoostClassifier` e `GradientBoostingClassifier`, foi usado o parâmetro `n_estimator=300`. A biblioteca `sklearn.gaussian_process` foi usada na linguagem Python para implementar o classificador Gaussian Process Classification (GPC). Para o método `GaussianProcessClassifier()` da classe `GaussianProcessClassifier`, foi usado o parâmetro `max_iter_predict=300`. A biblioteca `sklearn.svm` foi usada na linguagem Python para implementar a classificação de Support Vector Machines (SVM). Para o método `SVC()` da classe `SVC`, foi usado o parâmetro `max_iter=300`. Para os métodos `DecisionTreeClassifier()` e `GaussianNB()`, respectivamente das classes `árvore`, `GaussianNB`, seus parâmetros padrões são usados. Para o método `KNeighborsClassifier()` da classe `KNeighborsClassifier`, é usado o parâmetro `n_neighbors=3`.

Todos os métodos de classificação usados na avaliação foram configurados para utilizar no máximo 300 iterações. O software classificador, os conjuntos de dados de

treinamento e de testes estão disponíveis no GitHub[1], com a licença GNU General Public License v 3.0. O objetivo é analisar qual dos modelos de classificação é o mais adequado para o método proposto no *corpus* do Tribunal de Justiça. Acurácia, precisão, recuperação e medida-f são utilizadas para avaliar os melhores métodos usados.

6.3.3 Métricas de acurácia, medida-f, precisão e revocação

No processo de avaliação é necessário combinar dois tipos de conjuntos de dados de treinamento, binários e de frequência, com nove modelos de classificação, gerando dezoito resultados diferentes para cada métrica de avaliação. Como quatro métricas de avaliação são usadas, apresenta-se setenta e dois resultados. Os resultados obtidos nos testes são apresentados nas Figura 6.2 até Figura 6.5, respectivamente para as métricas de avaliação: acurácia; precisão; revocação e medida-f. Nas Figura 6.2 até Figura 6.5, os dados são apresentados comparando os resultados da classificação dos conjuntos de dados binários e frequência. Para os testes de classificação foram utilizados 200 documentos judiciais.

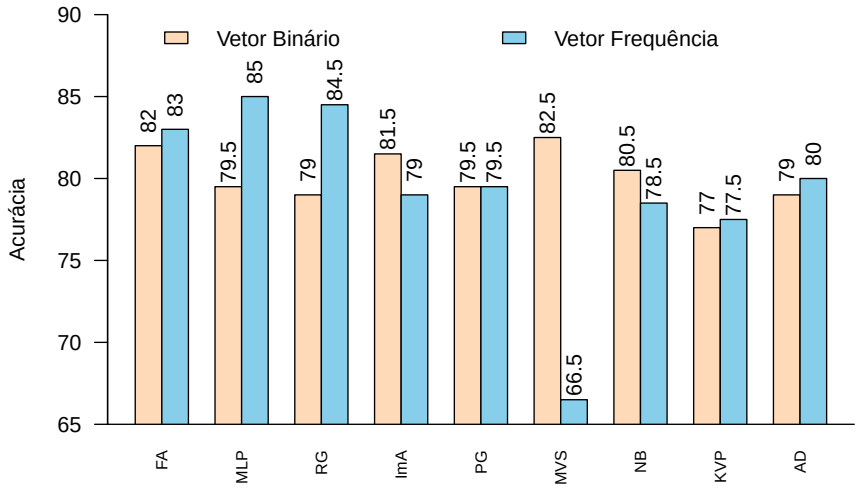

Figura 6.2 - Comparação dos valores de acurácia usando conjunto de dados binários e frequência.

[1] https://github.com/apcastrojr/court_of_law_datasets_and_text_classifiers2.

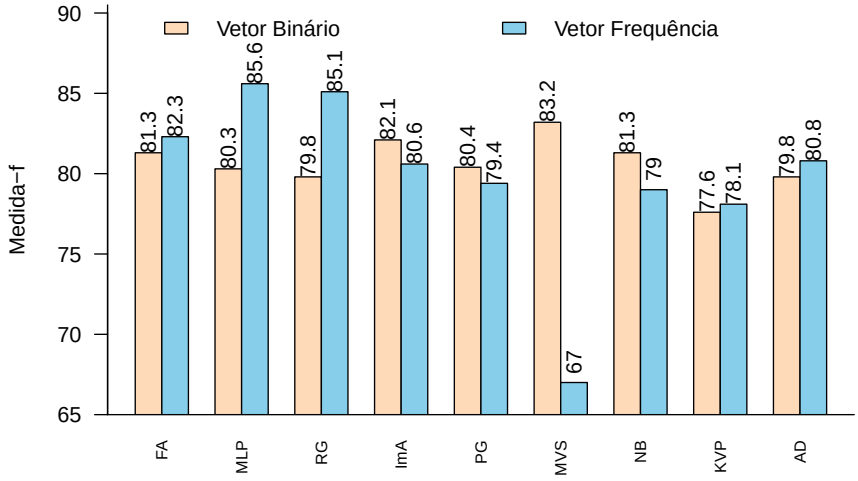

Figura 6.3 - Comparação dos valores de medida-f usando conjunto de dados binários e frequência.

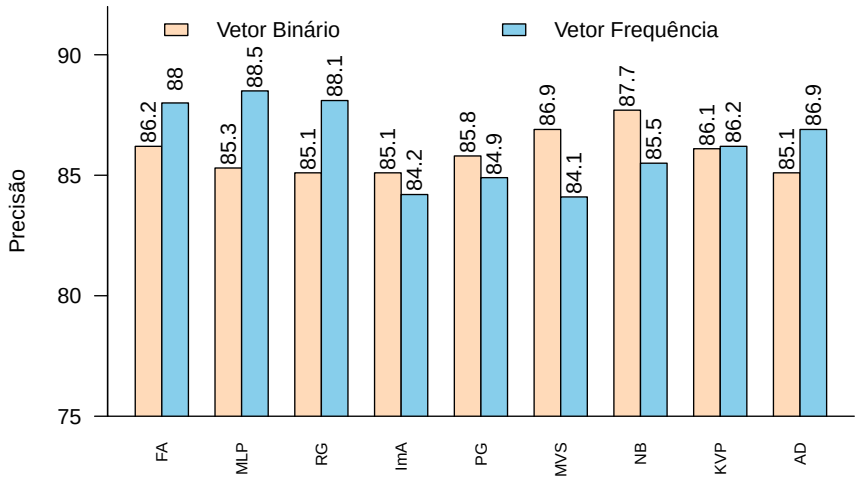

Figura 6.4 - Comparação dos valores de precisão usando conjunto de dados binários e frequência.

Os conjuntos de dados representam o conhecimento extraído do *corpus* das quatro categorias de documentos de texto não estruturados do tribunal de justiça. Para

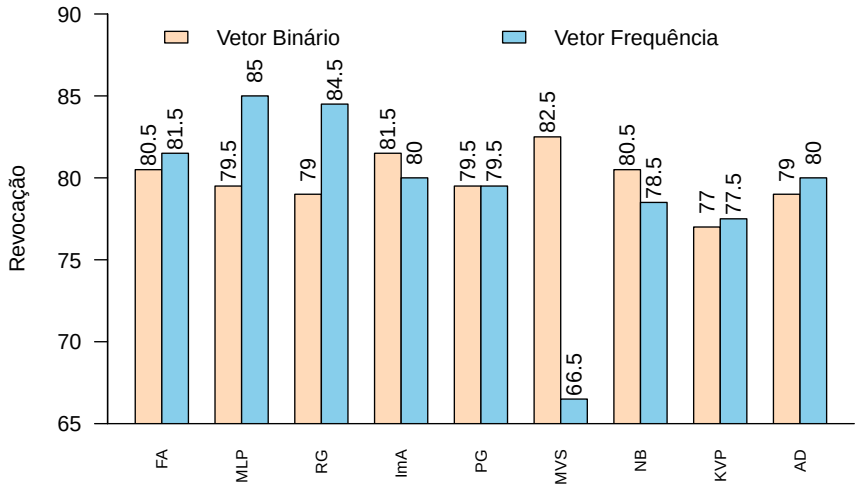

Figura 6.5 - Comparação dos valores de revocação usando conjunto de dados binários e frequência.

geração de conhecimento, os termos com os maiores pesos *tf-idf* foram separados primeiro. Com os $n = 10$ termos com o maior *tf-idf*, eles foram combinados 2×2 usando a similaridade de Jaccard em (5.2). Termos combinados com coeficiente $\phi \geq 50\%$ são usados para treinamento.

As Figura 6.2 até Figura 6.5 apresentam a capacidade da metodologia proposta em construir os pesos para representar os documentos, resultando em 85% para a acurácia e revocação, $85,6\%$ para a medida-f e $88,5\%$ para a precisão, utilizando MLP com conjunto de dados de frequência para treinamento. No conjunto de dados binário, nas métricas de acurácia, medida-f e revocação, os maiores valores são os do modelo MVS, mas na métrica de precisão, o maior valor foi usando NB. A Figura 6.3 apresenta a média harmônica entre precisão e revocação, reforçando os valores de acurácia obtidos na Figura 6.2, demonstrando a relevância do método proposto. O tempo médio de processamento foi de 23 segundos para aprendizagem supervisionada e predição dos nove modelos de classificação e conjuntos de dados usados. O maior tempo de processamento foi para o algoritmo ImA, com 31 segundos.

6.3.4 Consolidação da aplicação da metodologia proposta

A Figura 6.6 apresenta os resultados de forma resumida na aplicação do método proposto nos documentos do Tribunal de Justiça do Estado de Goiás, Brasil. Por meio do gráfico radar é possível verificar comparativamente todos os 72 resultados da aplicação do método proposto nos documentos do Tribunal de Justiça, utilizando vetor com valores binários na Figura 6.6 (a) e vetor com valores de frequência na Figura 6.6 (b), para os nove diferentes algoritmos de classificação.

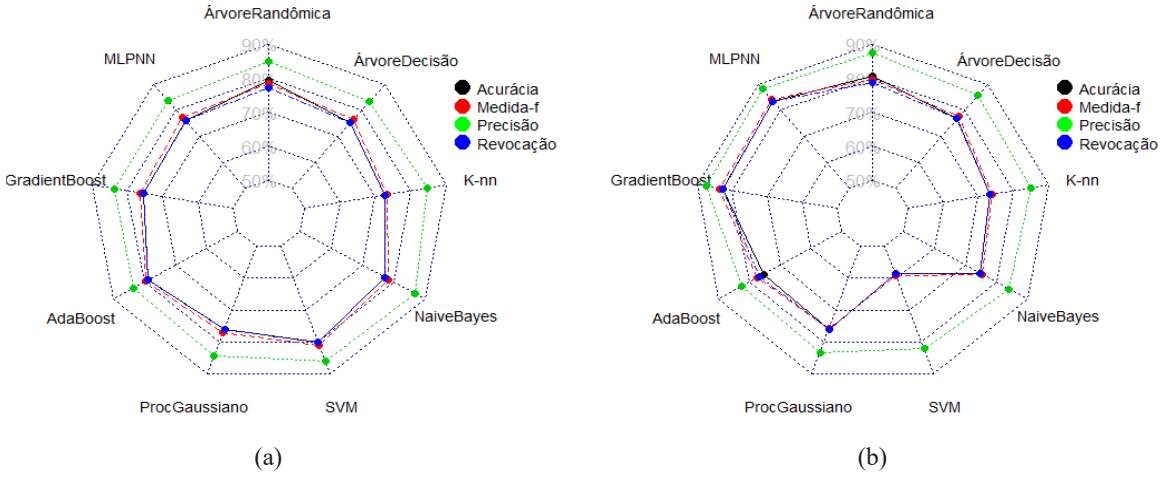

Figura 6.6 - Aplicação do método proposto: (a) vetor com valores binários e (b) vetor com valores de frequência.

A leitura resumida da Figura 6.6 é que: i) a média das métricas de avaliação foi de 80%, alguns casos de aproximadamente 90%, o que demonstra a eficiência do modelo proposto, ii) os melhores resultados são quando o treinamento supervisionado é aplicado com vetores de frequência, iii) no entanto, usando os vetores de frequência no treinamento, há variações mais significativas entre os resultados da métrica em algoritmos de classificação, ao contrário de quando vetores binários são usados, iv) os valores alcançados nas métricas de acurácia e revocação são próximos, v) avaliando todas as métricas de classificação juntas, para cada algoritmo de classificação, o melhor classificador encontrado foi o MLP, com resultados próximos a 90%.

6.4 Aplicação e comparação do método proposto com outra abordagem

Esta seção apresenta resultados comparativos do método proposto com métodos de classificação utilizados em outro trabalho de pesquisa. Conjunto de dados diferentes são usados, permitindo a comparação dos modelos de classificação dentro do contexto de aplicação com outra abordagem. O objetivo é verificar se o método proposto está aderente a outro trabalho de pesquisa.

Abualigah et al. (2020) apresenta e avalia algoritmos utilizados na solução do problema de classificação e agrupamento de textos, usa *corpus* disponíveis gratuitamente na Universidade de São Paulo (USP), Instituto de Matemática e Ciências da Computação (IMCC)[2]. Dois *corpus* de documentos de texto aplicados e avaliados por Abualigah et al. (2020), denominados DS1 e DS2, são usados pelos autores. O objetivo é comparar os resultados das métricas de acurácia, medida-f, precisão e revocação obtidos pelos métodos: algoritmo de busca harmônica (ABH), algoritmo genético (AG), enxames de partículas (PSO), colônia de formigas (ACO), rebanho de Krill (RK), busca do voo do cuco (CS), otimizador por lobo cinzento (LC), ecolocalização dos morcegos (BA) e a técnica k-médias (KM) utilizadas no estudo de Abualigah et al. (2020) e comparar com os algoritmos floresta aleatória (FA), rede neural perceptron multicamadas (MLP), reforço de gradiente (RG), impulso adaptativo (ImA), processo Gaussiano (PG), máquina de vetores de suporte (MVS), Naive Bayes (NB), k-vizinhos mais próximos (KVP), árvores de decisão (AD), utilizados nesta proposta. A Tabela 6.9 dispõe os conjuntos de dados utilizados nos experimentos.

Tabela 6.9 - *Corpus* usado nos experimentos.

Conjunto de dados	Número de Documentos	Categorias	Origem
DS1	299	4	Relatórios Técnicos*
DS2	333	4	Páginas Web*

*Abualigah et al. (2020).

As Figura 6.7 e Figura 6.8 apresentam os resultados comparativos entre as métricas de acurácia, medida-f, precisão e revocação em Abualigah et al. (2020) e nesta proposta para os conjuntos de dados DS1 e DS2, respectivamente. Na Figura 6.7, conjunto de dados DS1, observa-se que o método proposto obtém melhores resultados

[2] http://sites.labic.icmc.usp.br/text_collections/

nos algoritmos MLP e NB. Na Figura 6.8, conjunto de dados DS2, o algoritmo LC em Abualigah et al. (2020) apresenta os melhores resultados, no entanto, todos os outros algoritmos em Abualigah et al. (2020) tiveram valores inferiores aos obtidos na metodologia proposta. Analisando apenas a métrica de precisão, na Figura 6.8, conjunto de dados DS2, o algoritmo PG na metodologia proposta apresenta o melhor resultado entre todos os outros algoritmos.

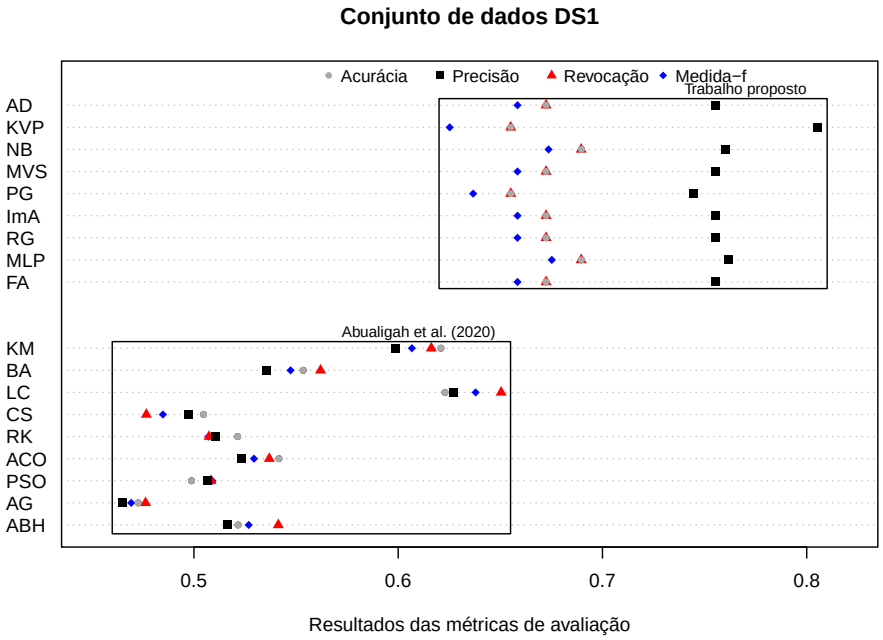

Figura 6.7 - Comparação entre Abualigah et al. (2020) e o método proposto para o conjunto de dados DS1.

As Tabela 6.10 até Tabela 6.13 dispõem os resultados da aplicação do método proposto nos conjuntos de dados DS1 e DS2 para as métricas: acurácia, medida-f, precisão e revocação, respectivamente. O código-fonte das rotinas, os conjuntos de dados e os documentos de texto utilizados neste estudo estão disponíveis no Github[3].

Analisando a origem do *corpus* DS1 e DS2 disposto na Tabela 6.9, observa-se que para os documentos técnicos, como os do conjunto de dados DS1 (Relatórios Técnicos), o método proposto é melhor que o utilizado por Abualigah et al. (2020), em comparação com os documentos genéricos, como os do conjunto de dados DS2

[3] https://github.com/apcastrojr/text_documents_ds1_ds2_to_compare.

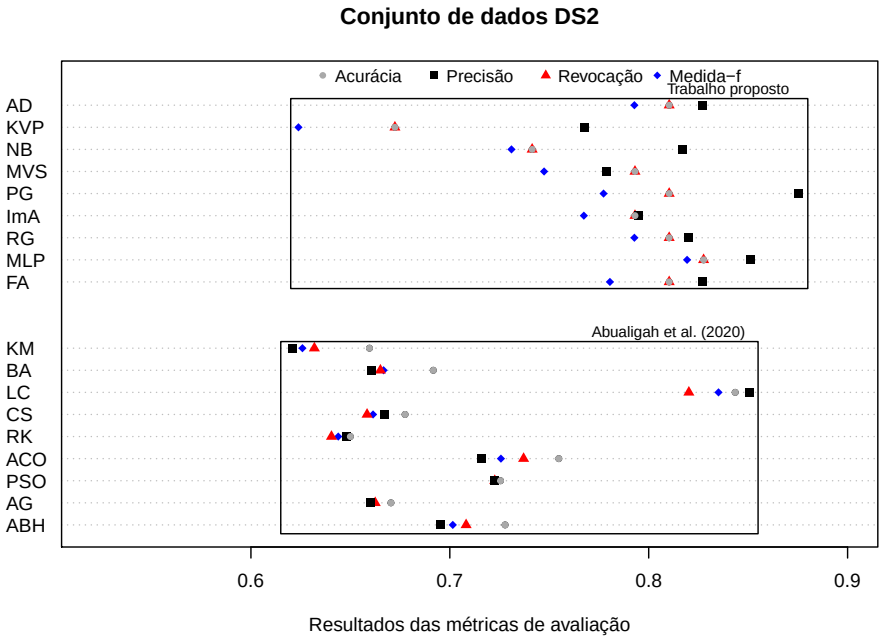

Figura 6.8 - Comparação entre Abualigah et al. (2020) e o método proposto para o conjunto de dados DS2.

Tabela 6.10 - Resultados para a métrica acurácia.

Dados	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
DS1	Binário	0,6724	0,6897	0,6724	0,6724	0,6552	0,6724	0,6897	0,6552	0,6724
	Frequência	0,6552	0,6724	0,6552	0,6724	0,6552	0,6724	0,6897	0,5517	0,6379
DS2	Binário	0,8103	0,7931	0,7931	0,7759	0,8103	0,7931	0,7414	0,6379	0,8103
	Frequência	0,8103	0,8276	0,8103	0,7931	0,8103	0,5345	0,7241	0,6724	0,8103

Tabela 6.11 - Resultados para a métrica medida-f.

Dados	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
DS1	Binário	0,6584	0,6752	0,6584	0,6584	0,6367	0,6568	0,6736	0,6252	0,6584
	Frequência	0,6360	0,6526	0,6360	0,6584	0,6360	0,6584	0,6691	0,4965	0,6144
DS2	Binário	0,7805	0,7674	0,7674	0,7393	0,7590	0,7474	0,7310	0,5677	0,7805
	Frequência	0,7773	0,8193	0,7928	0,7674	0,7773	0,4040	0,7163	0,6239	0,7928

Tabela 6.12 - Resultados para a métrica precisão.

Dados	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
DS1	Binário	0,7555	0,7618	0,7555	0,7555	0,7248	0,7312	0,7411	0,7430	0,7555
	Frequência	0,7446	0,7507	0,7446	0,7555	0,7446	0,7555	0,7603	0,8051	0,7188
DS2	Binário	0,8271	0,7947	0,7947	0,7574	0,8752	0,7790	0,8172	0,6587	0,8271
	Frequência	0,8169	0,8514	0,8200	0,7947	0,8169	0,3624	0,7826	0,7679	0,8200

Tabela 6.13 - Resultados para a métrica revocação.

Dados	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
DS1	Binário	0,6724	0,6896	0,6724	0,6724	0,6551	0,6724	0,6896	0,6551	0,6724
	Frequência	0,6551	0,6724	0,6551	0,6724	0,6551	0,6724	0,6896	0,5517	0,6379
DS2	Binário	0,8103	0,7931	0,7931	0,7759	0,8103	0,7931	0,7414	0,6379	0,8103
	Frequência	0,8103	0,8276	0,8103	0,7931	0,8103	0,5345	0,7241	0,6724	0,8103

(Páginas da WEB). Tendo em vista que a natureza dos documentos nos quais se utilizam nesta proposta é de caráter técnico, pois se refere a julgamentos do Tribunal de Justiça, respeitando as leis e normas governamentais para solucionar os problemas populacionais, a comparação com Abualigah et al. (2020) reforça a utilização do método proposto e sua aplicação no ambiente de produção, por meio de API.

6.5 Comparação das metodologias propostas para a ontologia

Após verificar os resultados da metodologia proposta, aplica-se a ontologia após o método estabelecer a coocorrência dos termos. A ontologia é integrada ao modelo de vetorização de duas formas: i) usando regras estabelecidas por especialista e ii) utilizando banco de dados léxico do PULO[4].

6.5.1 Ontologia usando regras do especialista na área jurídica

Objetiva-se nesta seção inserir o conhecimento humano, por meio da estruturação ontológica de domínio, definindo os indivíduos, as classes e principalmente os relacionamentos, referentes as tabelas e metadados. A Tabela 6.14 dispõe o escopo das classes de processos que são abordadas pelo especialista.

Tabela 6.14 - Tipos de naturezas de revisional e consignatória encontradas no banco de dados.

Naturezas do tipo Revisional e Consignatória
Rescisão Contratual
Restituição de Importâncias Pagas
Consignação em Pagamento (CPC)
Consignatória
Revisional de Locação
Revisional Cláusulas Contratuais com Restituição de Importância
Revisional
Revisional Cláusulas Contratuais com Restituição de Importância Pagas com Pedido de Liminar
Revisional de Aluguel

A Figura 6.9 apresenta a estrutura de conceito elaborada para identificar os compo-

[4] http://wordnet.pt.

nentes no domínio da ontologia. Esta estrutura ontológica de domínio foi construída para definir os indivíduos, classes e relacionamentos, referentes as diversas tabelas e metadados do Judiciário Goiano.

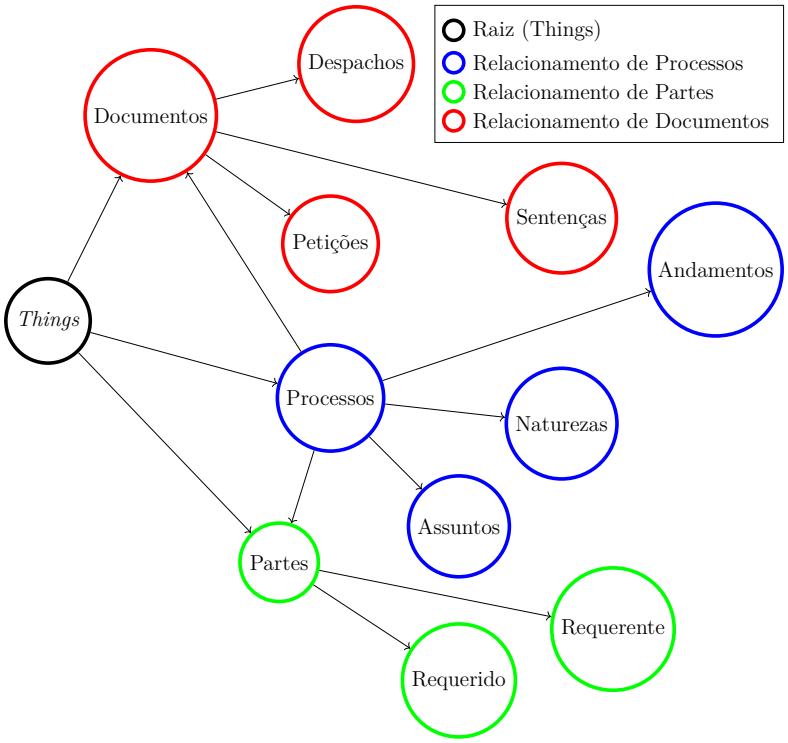

Figura 6.9 - Estrutura ontológica de domínio.

Com objetivo de realizar a criação da ontologia de aplicação com regras do especialista, é escolhido o software para criação, edição e manutenção das regras da ontologia jurídica. O software escolhido é o *Protegé*® (MATTHEW, 2011). Com a definição da ferramenta, é criada classificação em árvore da ontologia de aplicação jurídica, como apresentado na Figura 6.10. Esta primeira classificação taxonômica é realizada utilizando o conhecimento do autor.

A definição inicial apresentada na Figura 6.10 é revisada por profissional do direito, com vasta experiência na assessoria de magistrado. O profissional do direito realizou alterações na classificação, bem como auxiliou na construção dos relacionamentos e inferências nos conceitos analisados/levantados. O resultado desta análise é apresentado na Figura 6.11. Como resultado do trabalho no *Protegé*® é gerada a linguagem Web para ontologia (*Ontology Web Language* – OWL), que o computador entende

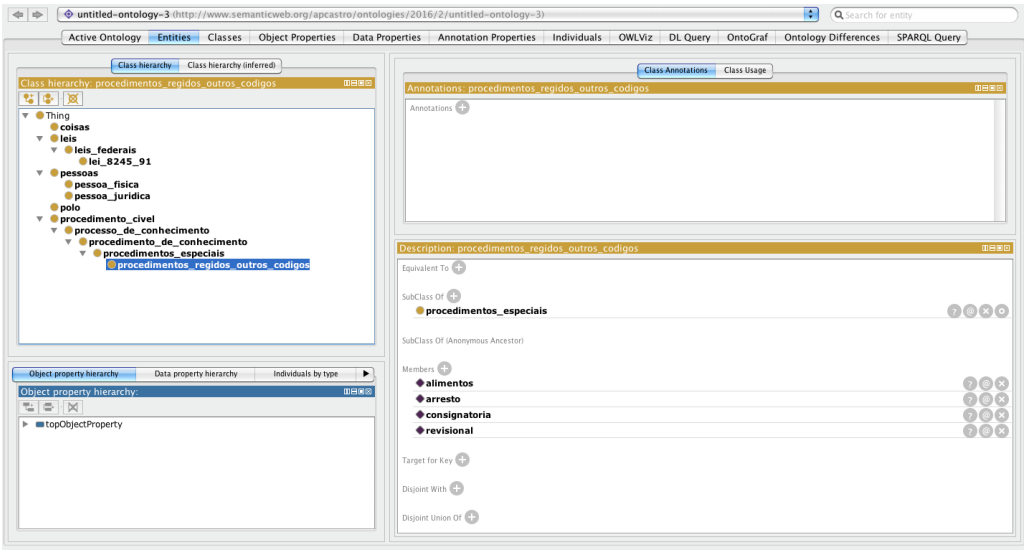

Figura 6.10 - Primeira classificação taxonômica utilizando o *Protegé*®.

e processa, apresentada na Figura 6.12.

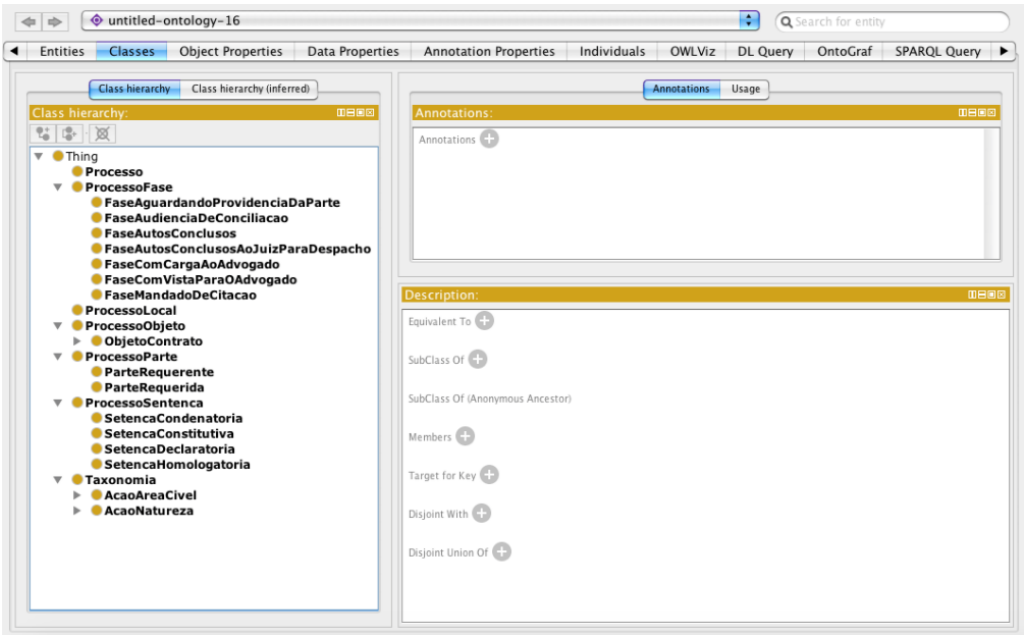

Figura 6.11 - Segunda classificação taxonômica utilizando o *Protegé*®.

A contribuição da ontologia com regras do especialista está no relacionamento das classes processuais (Tabela 6.14), para os tipos de processos de revisional e con-

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/apcastro/ontologies/2016/2/untitled-ontology-3"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://www.semanticweb.org/apcastro/ontologies/2016/2/untitled-ontology-3">
    <Prefix name="" IRI="http://www.semanticweb.org/apcastro/ontologies/2016/2/untitled-ontology-3#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <Class IRI="#coisas"/>
    </Declaration>
    <Declaration>
        <Class IRI="#lei_8245_91"/>
    </Declaration>
    <Declaration>
        <Class IRI="#leis"/>
    </Declaration>
    <Declaration>
        <Class IRI="#leis_federais"/>
    </Declaration>
    <Declaration>
        <Class IRI="#pessoa_fisica"/>
    </Declaration>
    <Declaration>
        <Class IRI="#pessoa_juridica"/>
    </Declaration>
    <Declaration>
        <Class IRI="#pessoas"/>
    </Declaration>
    <Declaration>
        <Class IRI="#polo"/>
    </Declaration>
    <Declaration>
        <Class IRI="#procedimento_civel"/>
    </Declaration>
    <Declaration>
        <Class IRI="#procedimento_de_conhecimento"/>
    </Declaration>
    <Declaration>
        <Class IRI="#procedimentos_especiais"/>
```

Figura 6.12 - Linguagem OWL produzida pela ontologia com regras do especialista.

signatória, em conjunto com os termos que não poderiam estar contidos dentro do inteiro teor das sentenças judiciais, como disposto na Tabela 6.15. A contribuição elaborada pelo especialista está na relação dos termos de exclusão para o domínio das classes e do indivíduo. Caso a mineração seja realizada por meio de filtros, na linguagem SQL em documentos de sentenças judiciais, a ontologia proposta é parcialmente descrita no trecho do código SQL apresentado na Figura 6.13. Conforme o especialista, a coocorrência dos termos **reintegração** + **alimento**, **família** + **alimento** e o termo **dpvat** não podem estar contidos na recuperação de decisões judiciais relacionadas as categorias de revisional e/ou consignatória, como disposto na Tabela 6.15.

Tabela 6.15 - Estrutura de domínio elaborada pelo especialista.

Indivíduo	**Classes relacionadas ao indivíduo**	**Objetos e os seus relacionamentos**	
		Termos aplicados	**Termos não aplicados**
Processos	Classes da Tabela 6.14	revisional, consignatória	reintegração-alimento, família-alimento, dpvat

```
...
and (sentenca ilike '%revisional%' or sentenca ilike '%consigna%')
and sentenca not ilike '%reintegra__o%' and sentenca not ilike '%alimento%'
and sentenca not ilike '%fam_lia%' and sentenca not ilike '%aliment%'
and sentenca not ilike '%dpvat%';
...
```

Figura 6.13 - Parte de código SQL para aplicação da ontologia com regras do especialista.

Observa-se que o conhecimento humano inserido estabeleceu a coocorrência de termos para serem excluídos na mineração do conjunto de documentos que deseja-se buscar. Apresenta-se que este conceito ontológico do especialista diminuirá os falsos positivos no processo de recuperação da informação, melhorando as métricas de avaliação.

Para verificar se há ganhos nas métricas, os termos aplicados e os não aplicados, dispostos na Tabela 6.15, são usados conjuntamente com o modelo de ontologia automática, sem regras do especialista, para vetorização dos documentos. Neste processo de avaliação, são utlizadas duas categorias de documentos, na qual a Tabela 6.16 dispõe as duas categorias utilizadas e suas quantidades para avaliação.

Tabela 6.16 - Categorias e quantidades utilizadas na vetorização e no treinamento supervisionado, com regras do especialista.

Categorias	Quantidade
Contratos	500
Posse, Divórcio e Previdenciário	1.500

As abordagens de vetores binário e frequência e os nove algoritmos de classificação são aplicados no treinamento supervisionado. Os resultados são dispostos nas Tabela 6.17 até Tabela 6.20. Observa-se nas Tabela 6.17 até Tabela 6.20 que a utilização das regras do especialista melhorou os resultados frente à não utilização. As regras estabelecidas pelo especialista nos relacionamentos dos objetos que não se

aplicam às classes, melhoram os valores das métricas de avaliação, principalmente na métrica precisão. Estes objetos são os termos que o especialista entende não se aplicar ao conjunto dos documentos das classes de revisional e/ou consignatória.

Tabela 6.17 - Resultados da ontologia usando a métrica acurácia.

Método	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
Híbrido	Binário	0,935	0,95	0,95	0,865	0,94	0,945	0,835	0,795	0,945
	Frequência	0,865	0,955	0,94	0,87	0,94	0,88	0,955	0,785	0,935
WordNet	Binário	0,83	0,845	0,845	0,84	0,845	0,92	0,775	0,755	0,81
	Frequência	0,82	0,82	0,83	0,835	0,825	0,865	0,92	0,855	0,805
Automático	Binário	0,845	0,85	0,85	0,845	0,845	0,92	0,785	0,76	0,825
	Frequência	0,835	0,85	0,855	0,85	0,845	0,86	0,925	0,77	0,82

Tabela 6.18 - Resultados da ontologia usando a métrica medida-f.

Método	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
Híbrido	Binário	0,934	0,948	0,948	0,872	0,939	0,944	0,845	0,808	0,944
	Frequência	0,872	0,953	0,939	0,876	0,939	0,872	0,953	0,799	0,934
WordNet	Binário	0,839	0,853	0,853	0,849	0,853	0,916	0,790	0,771	0,821
	Frequência	0,830	0,830	0,839	0,844	0,835	0,848	0,914	0,853	0,817
Automático	Binário	0,853	0,858	0,858	0,853	0,853	0,915	0,799	0,776	0,835
	Frequência	0,844	0,858	0,863	0,858	0,853	0,843	0,920	0,785	0,831

Tabela 6.19 - Resultados da ontologia usando a métrica precisão.

Método	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
Híbrido	Binário	0,934	0,951	0,951	0,898	0,940	0,945	0,901	0,881	0,945
	Frequência	0,898	0,958	0,940	0,901	0,940	0,880	0,958	0,878	0,934
WordNet	Binário	0,877	0,884	0,884	0,886	0,884	0,922	0,882	0,855	0,869
	Frequência	0,873	0,868	0,878	0,875	0,875	0,879	0,928	0,852	0,867
Automático	Binário	0,889	0,891	0,891	0,889	0,889	0,924	0,884	0,863	0,880
	Frequência	0,884	0,891	0,893	0,891	0,889	0,869	0,932	0,867	0,878

Tabela 6.20 - Resultados da ontologia usando a métrica revocação.

Método	Tipo	FA	MLP	RG	ImA	PG	MVS	NB	KVP	AD
Híbrido	Binário	0,935	0,95	0,95	0,865	0,94	0,945	0,835	0,795	0,945
	Frequência	0,865	0,955	0,94	0,87	0,94	0,88	0,955	0,785	0,935
WordNet	Binário	0,83	0,845	0,845	0,84	0,845	0,92	0,775	0,755	0,81
	Frequência	0,82	0,82	0,83	0,835	0,825	0,865	0,92	0,855	0,805
Automático	Binário	0,845	0,85	0,85	0,845	0,845	0,92	0,785	0,76	0,825
	Frequência	0,835	0,85	0,855	0,85	0,845	0,86	0,925	0,77	0,82

6.5.2 Ontologia usando base de dados lexical

A base de dados lexical utilizada neste trabalho é a Ontologia Lexical Unificada para o Português (*Portuguese Unified Lexical Ontology* – PULO). O PULO é variante da WordNet para a língua portuguesa (OLIVEIRA et al., 2015), (OLIVEIRA et al., 2016). O acesso a sua base de dados é realizada usando interface de consulta na web ou API Rest[5].

Como o método proposto relaciona a coocorrência dos termos, é possível de posse dos termos, buscar na base de dados do PULO os seus sinônimos. De posse dos sinônimos, é estabelecida novas combinações de termos, 2×2. Os termos sinônimos constituem novas relações, que são aplicadas na vetorização e no treinamento supervisionado. As abordagens de vetores binário e frequência e os nove algoritmos de classificação são utilizados nos testes da solução híbrida, sendo aplicados no treinamento supervisionado da ontologia usando base de dados lexical. Assim, é possível comparar os resultados da aplicação conjunta do método automático com a base de dados externa ontológica léxica do PULO.

Os resultados obtidos pela ontologia utilizando base de dados lexical são dispostos nas Tabela 6.17 até Tabela 6.20. Observa-se nos resultados das métricas que no geral, não ocorreram melhoria significativa aplicando a base de dados do PULO. No entanto, ocorreu pequena melhoria nas métricas de precisão e medida-f.

6.6 Implementação da interface de integração

Para que a solução seja efetiva na prática, no dia a dia dos departamentos de Justiça, é necessária a implantação de solução que permita avisar aos Juízes quando o método prever novos processos nas categorias: i) contrato; ii) possessório; iii) família e iv) pensão. O software que controla as ações judiciais eletrônicas utilizado na justiça Goiana é conhecido pelo nome de Projudi. A ferramenta Projudi controla e processa as ações judiciais. A API implementada busca, todos os dias, no banco de dados do Projudi o primeiro documento ajuizado em ação judicial, conhecido como petição inicial. Após buscar a petição incial, o modelo faz a predição da sua categoria. Após classificar a ação, a API informa o resultado ao Projudi. Assim, o usuário do Projudi pode verificar este aviso no próprio processo. Com o aviso, o usuário do Projudi fica ciente dos casos e poderá realizar os procedimentos necessários. As informações

[5] http://wordnet.pt/api.

aparecem tanto na tela principal do Projudi, quanto na própria tela do processo.

6.7 Aplicação e comparação com outras técnicas de vetorização

Esta seção apresenta resultados comparativos substituindo algumas tecnologias aplicadas no modelo proposto por outras tecnologias utilizadas na literatura. Os mesmos conjuntos de dados são utilizados, porém são substituídos os modelos de vetorização, como: i) usar o modelo *Okapi BM25* em conjunto com a similaridade de Jaccard proposta, ii) substituir Jaccard pelo Coseno, em conjunto com *tf-idf* e iii) retirar o limitador do coeficiente de similaridade $\phi = 50\%$ na separação da coocorrência dos termos.

6.7.1 Aplicação do *Okapi BM25*

O método *tf-idf* é substituído pelo *Okapi BM25* (*BM25*) na identificação dos termos. A Tabela 6.21 dispõe os termos encontrados aplicando o modelo *BM25*, Etapa 1 e Etapa 2 apresentadas na Figura 5.3 no mesmo *corpus* disposto na Tabela 6.2. A Tabela 6.22 dispõe os termos repetidos encontrados pelo modelo *BM25*. Observa-se que o problema é não linear. Aplicando a Etapa 3 e a Etapa 4 (Figura 5.3), tem-se a coocorrência dos termos, disposto na Tabela 6.23, encontrados por meio de (5.2), linearizando o problema.

Tabela 6.21 - Termos encontrados pelo modelo *BM25* no *corpus* na Tabela 6.2.

Categorias	**Termos encontrados**
Contrato	contrato, valor, revisional, prazo, pagamento, bem, forma, prova, crédito, cláusula
Posse	prazo, posse, esbulho, imóvel, contrato, área, propriedade, valor, concessão, requisitos
Divórcio	alimento, valor, guarda, prova, prazo, casal, filho, acordo, constituição, separação
Previdenciário	prazo, rural, segurado, idade, prova, aposentadoria, valor, concessão, atividade, pagamento

Tabela 6.22 - Termos não lineares encontrados nas diferentes categorias da Tabela 6.21.

Termos	**Repetições**
contrato, pagamento, concessão	2
prova	3
valor, prazo	4

Tabela 6.23 - Coocorrência de termos pela aplicação do *BM25* e similaridade em (5.2), com $\phi \geq 50\%$.

Categorias	**Coocorrência de termos aplicando *BM25* e Jaccard alterado em (5.2)**
Contrato	contrato-revisional, contrato-cláusula, contrato-forma, crédito-cláusula, revisional-pagamento, contrato-bem, contrato-crédito, revisional-forma, revisional-bem, bem-forma, contrato-prova, pagamento-forma
Posse	posse-esbulho, posse-imóvel, concessão-requisitos imóvel-propriedade
Divórcio	alimento-filho, guarda-filho, alimento-guarda
Previdenciário	rural-idade, aposentadoria-atividade, valor-pagamento, segurado-aposentadoria, idade-atividade, idade-aposentadoria, rural-aposentadoria, segurado-atividade, prova-atividade

A coocorrência dos termos encontrados e dispostos na Tabela 6.23, é utilizada no treinamento supervisionado para os modelos de classificação FA, MLP, RG, ImA, PG, MVS, NB, KVP e AD. Os mesmos conjuntos de dados utilizados nos testes para o modelo *tf-idf* são aplicados para o modelo *BM25*, sendo os resultados da métrica dispostos na Tabela 6.24. Os conjuntos de dados de treinamento e teste usando o modelo *Okapi BM25* estão disponíveis no GitHub[6], sob GNU General Public License v 3.0.

Tabela 6.24 - Resultados das métricas de avaliação nos modelos de classificação, com aplicação do *Okapi BM25* em conjunto com similaridade por (5.2).

Métodos de Classificação	**Tipo de Vetores**	**Acurácia**	**Medida-*f***	**Precisão**	**Revocação**
FA	Binário	71%	71,86%	80,43%	71%
	Frequência	67,5%	68,28%	76,22%	67,5%
MLP	Binário	71%	71,86%	80,43%	71%
	Frequência	68,5%	69,39%	77,28%	68,5%
RG	Binário	65%	65,44%	78,18%	65%
	Frequência	63%	62,75%	73,62%	63%
ImA	Binário	59,5%	59,58%	72,06%	59,5%
	Frequência	56,5%	56%	73,47%	56,5%
PG	Binário	64%	64,24%	77,01%	64%
	Frequência	64%	64,42%	75,59%	64%
MVS	Binário	69%	69,79%	79,66%	69%
	Frequência	58%	59%	73,19%	58%
NB	Binário	54%	53,11%	70,01%	54%
	Frequência	53,5%	52,38%	74,91%	53,5%
KVP	Binário	51,5%	47,96%	70,91%	51,5%
	Frequência	55%	52,15%	74,82%	55%
AD	Binário	61,5%	61,34%	74,79%	61,5%
	Frequência	59%	58,01%	72,98%	59%

Observa-se nos resultados obtidos que o modelo *tf-idf* apresenta melhores valores nas métricas de avaliação em comparação ao modelo *Okapi BM25*, Figura 6.14. Portanto, reforça a utilização do modelo *tf-idf* com a similaridade de Jaccard em

[6] https://github.com/apcastrojr/court_of_law_datasets_and_text_classifiers2.

(5.2) na implementação da API de integração do método proposto com a plataforma de gestão de processos do Tribunal da Justiça.

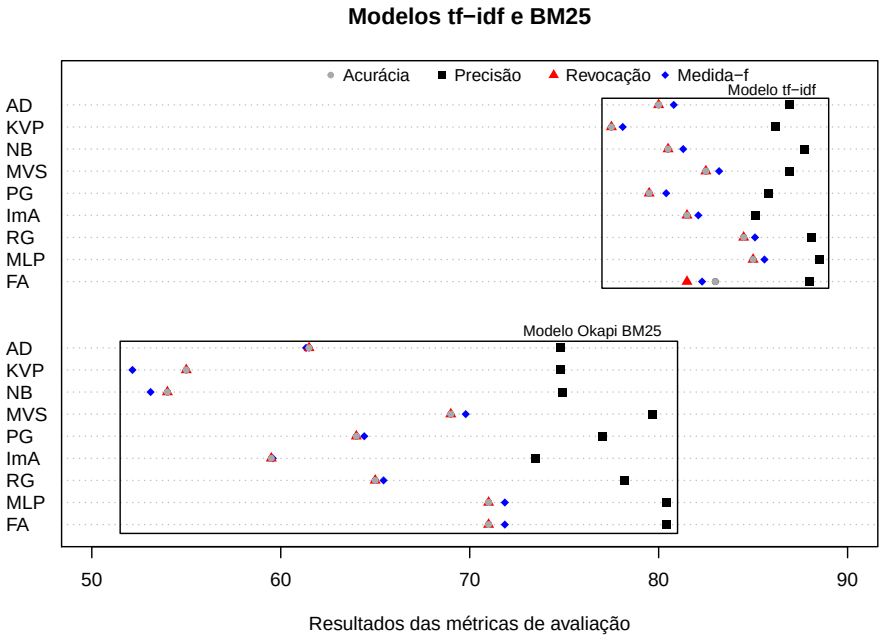

Figura 6.14 - Comparação das métricas de avaliação entre os modelos *tf-idf* e *Okapi BM25*.

6.7.2 Aplicação do limitador no coeficiente de similaridade de Jaccard

Na Etapa 4 da Figura 5.3 é utilizado o coeficiente de similaridade $\phi \geq 50\%$ como limitador para separar a coocorrência dos termos. Objetiva-se com este limitador separar menor quantidade de termos, 2×2, porém com maior capacidade de identificar documentos no *corpus*. Este limitador favorece o processamento na vetorização, treinamento supervisionado e testes dos modelos de classificação.

Desta forma, realiza-se comparações do método proposto usando o limitador $\phi \geq 50\%$ e sem usar o limitador, ou seja $\phi = 0\%$. Ao utilizar $\phi = 0\%$, considera-se que serão usadas todas as coocorrência entre os $n = 10$ termos, 2×2, da matriz termo $\times$ termo. Ou ainda, utilizar $\phi = 0\%$ significa utilizar a técnica 2-gramas nos $n = 10$ termos. Para o treinamento supervisionado e os testes, são utilizados os mesmos conjuntos de dados. Os resultados obtidos para este teste são dispostos na Tabela 6.25, na qual são comparados os dois tipos de vetores binário e de frequência.

Tabela 6.25 - Resultados da métrica de acurácia nos modelos de classificação, comparando o coeficiente de similaridade na coocorrência dos termos, com limitador $\phi \geq 50\%$ e sem limitador.

Vetores	Similaridade	Classificação	$\phi \geq 50\%$	$\phi \geq 0\%$
Binário	Jaccard	AD	79%	88%
		NB	80,5%	83,5%
		KVP	77%	78%
		MLP	79,5%	87,5%
		MVS	82,5%	87%
Frequência	Jaccard	AD	80%	88,5%
		NB	78,5%	82,5%
		KVP	77,5%	81,5%
		MLP	85%	92%
		MVS	66,5%	68%

Nos valores dispostos na Tabela 6.25, observa-se que quando não é aplicado o limitador do coeficiente de similaridade, ocorrem os melhores resultados. Na Tabela 6.26, observa-se que sem o limitador, aplicando a combinação simples de $n = 10$ termos, 2×2, tem-se o total de 45 coocorrências de termos por categoria de documentos, em quatro categorias, tem-se 180 coocorrências de termos, quase três vezes maior comparado com a aplicação do limitador ($\phi \geq 50\%$). Sem o limitador, as matrizes têm maiores dimensões, o processo de vetorização e treinamento supervisionado consome mais recursos computacionais e tempo de processamento. Assim, a diferença de quase três vezes mais termos combinados, sem aplicar o limitador, ocorre ganho, porém com alto custo computacional.

Tabela 6.26 - Quantidade de coocorrência de termos com e sem o limitador do coeficiente de similaridade de Jaccard.

Total de termos combinados usados na vetorização e treinamento	
Com limitador $\phi \geq 50\%$	65 termos combinados
Sem limitador $\phi = 0\%$	180 termos combinados

6.8 Validação cruzada do conjunto de dados e classificadores

Com o intuito de apresentar a confiança no conjunto de dados é aplicado o particionamento, utilizando o modelo de validação cruzada k-fold. Usando o k-fold é possível avaliar a capacidade de generalização dos classificadores utilizados. O conjunto de dados é dividido em $k = 8$ subconjuntos mutuamente exclusivos, com mesmo tamanho. Um dos subconjuntos é utilizado para validação e os outros $k - 1$ para treinamento, sendo medida a acurácia dos modelos. Este processo é repetido k vezes, alternando o subconjunto de validação. Terminando as k iterações é calculado

a acurácia média e sua variação.

O conjunto de dados aplicado na validação cruzada é composto pelos documentos dispostos na Tabela 6.8. Para implementação da validação cruzada é utilizada a biblioteca `sklearn.model_selection` do pacote *scikit-learn* em Python. As classes `cross_val_score` e `KFold` são as responsáveis pelas avaliações de acurácia dos modelos de classificação. Como o total de documentos utilizados são de 1.619, e sabendo que é aplicado $k = 8$, tem-se que cada subconjunto é composto de aproximadamente 202 documentos. Os resultados das validações cruzadas são apresentadas nas Figura 6.15 e Figura 6.16, para os conjuntos de dados binário e de frequência, respectivamente.

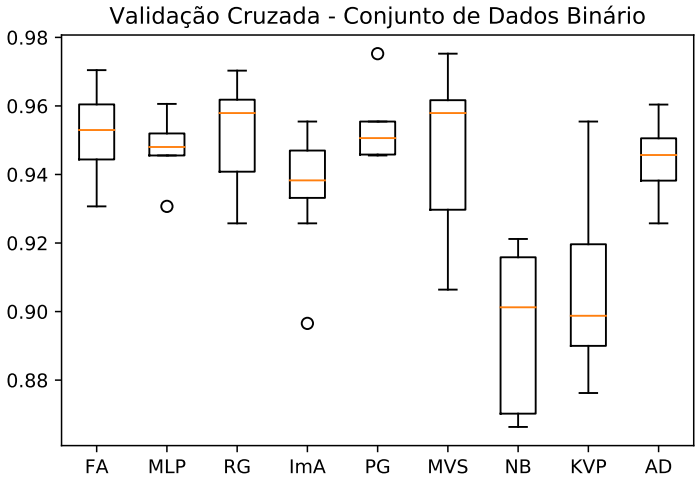

Figura 6.15 - Validação cruzada usando $k = 8$ para o conjunto de dados binário.

A variação da acurácia na validação cruzada em todos os modelos é baixa, apresentando a confiabilidade em aplicar o conjunto de dados. Nas Figura 6.15 e Figura 6.16 os valores atípicos de acurácia que fogem à normalidade, para as $k = 8$ iterações nos subconjuntos de dados, são ínfimos.

6.9 Discussão

Os resultados apresentam que a aplicação de Jaccard alterado em (5.2), dado por S_α, em conjunto com *tf-idf* consegue reconhecer documentos judiciais, e pode ser aplicada com diferentes modelos de classificação de texto para prever novos tipos de ações judiciais. A partir dos resultados apresentados nas Figura 6.2 até Figura 6.5,

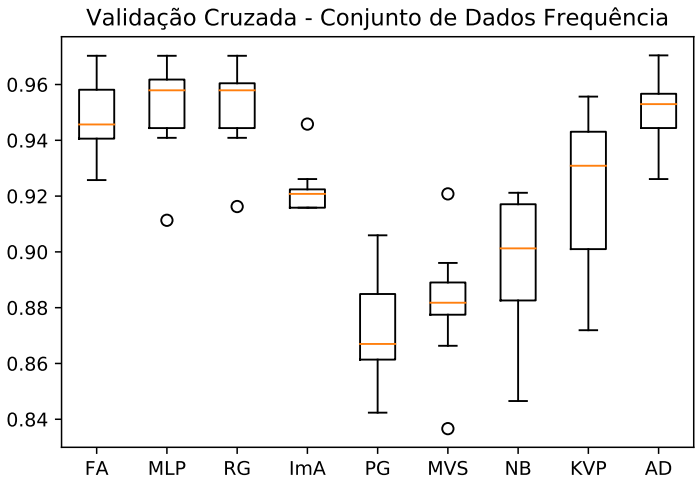

Figura 6.16 - Validação cruzada usando $k = 8$ para o conjunto de dados de frequência.

resumidos na Figura 6.6, observa-se que a combinação dos modelos tradicionais de construção de peso *tf-idf* com medida de similaridade de Jaccard para gerar conhecimento do *corpus* atingiu 85% na acurácia e revocação, $85,6\%$ em medida-f e $88,5\%$ em precisão na classificação de documentos, utilizando rede neural MLP, com treinamento supervisionado e com conjunto de dados estruturados pelo vetor de frequência. Estes valores obtidos para as métricas de avaliação são resultados significativos, pois trata-se de textos não estruturados, demonstrando o potencial da metodologia proposta.

O método proposto difere dos modelos de n-gramas tradicionais, pois a aplicação conjunta de técnicas de saco-de-palavras com aprendizado por similaridade estabelece critério/regra, para estabelecer a coocorrência de termos e seus pesos. A metodologia proposta, suaviza o problema dos modelos de n-gramas com relação a dimensionalidade das matrizes termo-documento e termo $\times$ termo.

A fase de preprocessamento é importante nesta proposta, principalmente devido às características não estruturadas dos documentos e aos termos comuns rotineiramente aplicados no *corpus* do Tribunal de Justiça. Como dificuldade inerente ao contexto de aplicação da pesquisa, não foram encontradas bibliotecas específicas em Python ou Ruby que implementassem limpeza, normalização, lematização e remoção de termos para textos jurídicos na língua portuguesa. Foi construído o preprocessamento para termos específicos deste ramo do conhecimento, na qual as bibliotecas desenvolvidas

estão disponíveis no Github[7], usando a licença pública (*General Public License* – GNU) v 3.0. O objetivo é compartilhá-las com outros pesquisadores, permitindo que haja continuidade no trabalho de investigação.

Nos resultados apresentados usando vetores binários para treinamento, observa-se nos modelos de classificação de texto usados ligeira variação de $5,5\%$ na acurácia e revocação, $5,6\%$ na medida-f e $2,6\%$ em precisão. O menor valor encontrado foi 77% na acurácia e revocação, $77,6\%$ na medida-f usando o modelo k-vizinhos mais próximos, enquanto o maior valor encontrado foi $82,5\%$ na acurácia e revocação, $83,2\%$ usando o modelo de máquina de vetores de suporte (MVS). Na métrica de precisão, o maior valor foi $87,7\%$ usando Naive Bayes e o menor valor foi $85,1\%$ usando reforço de gradiente, impulso adaptativo e árvores de decisão. Nos resultados apresentados usando vetores de frequência para treinamento, observa-se variação significativa entre os modelos de classificação usados, sendo $18,5\%$ na acurácia e na revocação, $18,6\%$ na medida-f. O menor valor encontrado foi $66,5\%$ na acurácia e revocação, 67% na medida-f usando o modelo de máquina de vetores de suporte (MVS). Em contraste, o maior valor encontrado foi 85% na acurácia e revocação, $85,6\%$ na medida-f e $88,5\%$ na precisão usando o modelo de rede neural perceptron multicamadas (MLP).

Pelos resultados apresentados, para vetores do tipo binários recomenda-se a aplicação do modelo MVS, e para os vetores do tipo frequência recomenda-se a aplicação do modelo MLP. Analisando os resultados obtidos, não recomenda-se o uso do conjunto de dados do vetor de frequência para o modelo MVS, pois, ele não apresenta resultados satisfatórios de acurácia, medida-f e revocação, uma vez que os resultados ficaram abaixo da média dos outros modelos aplicados nos estudos. Porém, excluindo o modelo MVS, o conjunto de dados usando vetor de frequência é o mais aconselhável para ser aplicado no treinamento supervisionado, e sugere-se seu uso no lugar do conjunto de dados usando vetor binário, principalmente com o modelo de classificação MLP.

Analisando os resultados gerais das métricas de acurácia, medida-f, precisão e revocação, observa-se que os modelos de classificação floresta aleatória, MLP, reforço de gradiente e árvores de decisão apresentaram melhores resultados com vetores de frequência. Em contrapartida, os modelos de classificação impulso adaptativo, MVS, Naive Bayes apresentaram melhores resultados com vetores binários. O pro-

[7] https://github.com/apcastrojr/court_of_law_pre_processing.

cesso Gaussiano e o k-vizinhos mais próximos obtiveram, praticamente, os mesmos resultados nos dois tipos de vetores. O tempo de processamento para aprendizagem supervisionada e predição foi em média de 23 segundos para todos os modelos de classificação e tipos de conjuntos de dados.

O ϕ é o valor de similaridade entre os termos, dado pelo modelo de Jaccard proposto. O coeficiente ϕ é parâmetro de sensibilidade para representar o *corpus* usando a coocorrência de termos. O $\phi = 0\%$ relaciona a coocorrência de todos os termos da matriz termo $\times$ temo, enquanto que o $\phi \neq 0\%$ traz inteligência ao modelo, por analisar o grau de similaridade da coocorrência dos termos, relacionando os melhores termos, 2×2 para respresentar o *corpus* e treinar os modelos de classificação. Por outro lado, $\phi \approx 100\%$ poderá não trazer nenhuma coocorrência de termos da matriz termo $\times$ temo, não trazendo nenhum ganho para representar o *corpus*.

Para trazer confiabilidade aos estudos e ao método proposto, os resultados são comparados com outros trabalhos de pesquisa, como Abualigah et al. (2020). Em Abualigah et al. (2020), dois conjuntos de dados são utilizados, um referente a relatórios técnicos e o outro referente a páginas Web. Estes dois conjuntos de dados em Abualigah et al. (2020), disponíveis gratuitamente na Internet, foram utilizados com o método proposto e os resultados avaliados. Nos resultados comparativos nas Figura 6.7 e Figura 6.8 observa-se que o método proposto apresenta melhores resultados nas métricas de avaliação utilizadas, com relevância para conjuntos de dados relativos a documentos técnicos. Das Figura 6.7 e Figura 6.8, pode-se afirmar que o método proposto é mais aderente na classificação de documentos técnicos do que os não técnicos. Considerando que os documentos do Tribunal de Justiça são técnicos, o metodologia proposta ajusta-se neste contexto de aplicação, reforçando assim, os benefícios sociais e acadêmicos desta proposta.

Trabalhos de pesquisa recentes, como Sulis et al. (2021), Mandal et al. (2021), Skrlj et al. (2021), Radygin et al. (2021), Hausladen et al. (2020), Waltl et al. (2018), Medvedeva et al. (2020), estão aplicando saco-de-palavras com *tf-idf* para vetorização e classificação de documentos na área jurídica. Embora aplicados a conjuntos de dados do direito, relativamente diferentes ao aplicado nesta proposta, estes estudos apresentam resultados semelhantes aos obtidos pelo método proposto. Para resumir as informações, a Tabela 6.27 apresenta os trabalhos, conjuntos de dados usados, seus países de aplicação, os valores alcançados pelas métricas de avaliação, os classificadores com melhores resultados, o modelo de saco-de-palavras aplicado e

o ano de publicação das obras. Exceto pelo resultado da métrica medida-f de 87% no trabalho de Radygin et al. (2021), os resultados alcançados nesta pesquisa são superiores aos artigos listados na Tabela 6.27. Entretanto, o resultado de 85,6% alcançado na medida-f, nesta pesquisa, é próximo ao valor 87% obtido por Radygin et al. (2021).

Tabela 6.27 - Pesquisas aplicando saco-de-palavras e classificadores com conjuntos de dados na área do Direito.

Pesquisas	**Ano**	**Saco-de-palavras**	**Dados**	**Métricas**	**Classificadores**
Trabalho proposto	–	*tf-idf* e Jaccard alterado em (5.2)	Poder Judiciário em Goiás	85,6% medida-f 88,5% precisão 85% acurácia 85% revocação	MLP
Sulis et al. (2021)	2021	*tf-idf* e método manual	União Européia	83% medida-f 76% acurácia	MVS
Mandal et al. (2021)	2021	*tf-idf* LDA and PScoreVect	Suprema corte Indiana	Acurácia média *tf-idf:* 80,8% LDA: 77,1% PScore: 67,1%	Por similaridade
Skrlj et al. (2021)	2021	*tf-idf* e tax2vec	Biomédico* Efeito e lado das drogas	medida-f 47% efeito drogas 52,3% lado drogas	MVS
Radygin et al. (2021)	2021	*tf-idf*	Lei federal Russa	87% medida-f	MVS
Hausladen et al. (2020)	2020	*tf-idf*	Corte dos Estados Unidos	74,5% medida-f 77,1% acurácia	Passivo agressivo
Medvedeva et al. (2020)	2020	*tf-idf*	Direitos humanos Europeu	75% acurácia	MVS
Waltl et al. (2018)	2018	*tf-idf* e método manual	Direito Civil Alemão	83% medida-f 85% precisão 84% revocação	MVS
Katz et al. (2017)	2017	manual baseado em metadados	Suprema Corte Estadus Unidos	70,2% acurácia 71,9% precisão	Floresta Aleatória

*dados em Drugs.com e Druglib.com.

Outra análise realizada é a substituição do modelo *tf-idf* pelo modelo *Okapi BM25*. Por serem dois modelos tradicionalmente utilizados na literatura e aplicados na vetorização de termos, segundo Mironczuk e Protasiewicz (2018), sua comparação torna-se relevante nesta proposta. Nos resultados obtidos, apresentados na Figura 6.14, observa-se que a técnica de aprendizagem por similaridade de Jaccard é melhor aplicada em conjunto com o modelo *tf-idf*, quando comparada ao modelo *Okapi BM25*.

Os resultados da aplicação da ontologia automática com ontologia das regras do especialista proporciona benefícios na classificação de textos. Esta combinação apresenta que há possibilidade de evolução das pesquisas agregando a ontologia. A dificuldade

encontrada na aplicação da ontologia foi em estabelecer o componente da ontologia que se ajustaria ao método proposto. O componente de relacionamento com os objetos e os atributos foi o item que aderiu, permitindo a sua incorporação na coocorrência dos termos encontrados de forma automática pela metodologia proposta. A utilização de base de dados lexical externa de forma conjunta com a ontologia automática, não apresentou melhorias significativas nos resultados.

O método proposto nesta pesquisa aprimora o modelo *tf-idf*, quando aplicado em conjunto com a similaridade de Jaccard em (5.2). Esta afirmação é consequência da solução na limitação relatada nos artigos de Agarwal et al. (2020), Seo et al. (2020) e Li et al. (2020), por encontrar e estabelecer pesos na coocorrência de termos no *corpus*. A coocorrência dos termos encontrados pelo cálculo de similaridade, dada por S_α em (5.2), dois a dois (2×2), pode ser modificada para executar cálculos com combinações maiores entre os termos. A pesquisa está em evolução, não se limitando aos modelos de classificação de textos aplicados nesta proposta. Pesquisas com outros modelos recentes, como redes neurais convolucionais (CNN), já estão em andamento. Também estão sendo realizados estudos com outros modelos de construção de pesos, como doc2vec e BERT.

CAPÍTULO 7

CONCLUSÃO

Neste trabalho, forma diferente de usar a técnica de similaridade de Jaccard é aplicada em conjunto com o modelo *tf-idf*, gerando conhecimento do *corpus* e permitindo o treinamento supervisionado de diferentes modelos de classificação de texto. Este método consegue prever categorias em novos processos no Tribunal de Justiça. O *corpus* jurídico tem a característica de não linearidade dos termos entre as categorias. Os resultados indicam que a metodologia proposta é viável de ser aplicada e utilizada no contexto de categorias de documentos jurídicos para, principalmente, ajudar a agilizar o trabalho dos juízes em seus julgamentos. A API é construída para permitir o uso da metodologia proposta em ambiente de produção no órgão governamental. Por meio da API, é possível informar o resultado na correlação de julgamentos já proferidos com novos processos que chegam no Poder Judiciário.

A metodologia de vetorização proposta é usada para gerar dois tipos de vetores: binários e de frequência. Estes dois modelos vetoriais de recursos são comparados e usados para treinamento supervisionado de nove tecnologias de classificação de texto: floresta randômica, redes neurais MLP, impulso adaptativo, reforço de gradiente, processo Gaussiano, SVM, Naive Bayes, k-vizinhos mais próximos e árvores de decisão. Em geral, os melhores resultados foram obtidos com o treinamento supervisionado usando o vetor de frequência. Porém, três modelos de classificação tiveram seus resultados prejudicados ao utilizar os vetores de frequência em comparação com os vetores binários.

Os resultados gerados nesta proposta são relevantes para conhecer o modelo de classificação de textos mais aderente à área do conhecimento jurídico. Os estudos auxiliam na escolha das melhores soluções preditivas a serem implementadas em ambientes de produção, principalmente em locais com intensa movimentação e fluxo de documentos não estruturados, como os Tribunais de Justiça do Brasil. Com a acurácia de 85%, a medida-f de $85,6\%$, a precisão de $88,5\%$ e a revocação de 85%, usando treinamento supervisionado com conjunto de dados de vetor de frequência, os resultados indicam que o melhor classificador, entre os aplicados neste trabalho, é a rede neural MLP.

A pesquisa realizada e o método de IA proposto consolidam a solução implemen-

tada em ambiente de produção no Poder Judiciário do Estado de Goiás, Brasil. Na esfera social, a partir dos resultados obtidos, a solução proposta preenche a lacuna apresentada, pois permite uniformizar julgamentos, reduzindo divergências, possíveis desigualdades e, em alguns casos, até mesmo discriminações. Assim, este trabalho atende o indicador 10.3 e o indicador 16.6, dentro de duas metas sustentáveis no mundo, Meta 10 e Meta 16, seguindo as dezessete metas estabelecidas pelas Nações Unidas.

Portanto, conclui-se que o método proposto pode ser aplicado em diversos ramos do conhecimento que possuem grandes volumes de documentos de texto e precisam automatizar o conhecimento, estabelecendo relacionamentos inteligentes e automáticos com novas entradas de documentos de texto.

7.1 Contribuições do trabalho

As contribuições podem ser assim descritas:

- Desenvolvimento de metodologia para estruturação automática da Ontologia;
- Aplicação da Ontologia na Mineração de Dados;
- Agilização dos trabalhos no Judiciário;
- Uniformização das decisões no Judiciário.

Trabalhos publicados em congressos, conferências ou simpósios:

CASTRO JUNIOR, A. P., CALIXTO, W. P., GOMES, V. M., VEIGA, E. F., SILVA, L. F. A., OLIVEIRA, L. L., BARBOSA, J. L. F. e CAMPOS, P. H. M. **Ontology Applied in the Judicial Sentences**. Conferência Internacional do IEEE no Chile - ChileCon 2017, Púcon, Chile, 18-20 de Outubro de 2017.

CASTRO JUNIOR, A. P. e CALIXTO, W. P. **Análise de Metodologia para Mineração de Dados em Sentenças Judiciais utilizando Ontologia**. Reunião Anual da Sociedade Brasileira para o Progresso da Ciência - SBPC 2017, UFMG, Belo Horizonte, 16-22 de Julho de 2017.

CASTRO JUNIOR, A. P., CALIXTO, W. P., GOMES, V. M., VEIGA, E. F., SILVA, L. F. A. e CAMPOS, P. H. M. **Ontology to Mining Judicial Sentence's Big**

Data. International Conference on Alive Engineering Education - ICAEEdu 2017, Rio de Janeiro, 21-23 de Junho de 2017.

CASTRO JUNIOR, A. P. e CALIXTO, W. P. **Mineração de Dados em Decisões Judiciais utilizando Ontologia**. Congresso de Pesquisa, Ensino e Extensão da UFG - CONPEEX 2016, UFG/EMC, Campos Samambaia, Goiânia, 17-19 de Outubro de 2016.

Trabalhos publicados em periódicos:

CASTRO JUNIOR, A. P., CALIXTO, W. P. e WAINER, G. A. **Weighting construction by bag-of-words with similarity-learning and supervised training for classification models in Court text documents**. Journal Applied Soft Computing, Elsevier, 2021. (ISSN 1568-4946, classificação CAPES A1).

CASTRO JUNIOR, A. P., CALIXTO, W. P. e WAINER, G. A. **Application of Artificial Intelligence in the automatic identification and classification repetitive demand resolution incident in the Brazilian Court of Justice**. Revista da Faculdade de Direito, UFG, 2021. (ISSN 0101-7187, classificação CAPES A3).

CASTRO JUNIOR, A. P., CALIXTO, W. P. e ARAÚJO, C. H. **Aplicação da Inteligência Artificial na identificação de conexões pelo fato e tese jurídica nas petições iniciais e integração com o Sistema de Processo Eletrônico**. Revista Desenvolvimento e Inovação Tecnológica, IBRACEDS, https://doi.org/10.29327/240349.2.1, 2020.

CASTRO JUNIOR, A. P., CALIXTO, W. P. e ARAÚJO, C. H. **Aplicação da Inteligência Artificial na identificação de conexões pelo fato e tese jurídica nas petições iniciais e integração com o Sistema de Processo Eletrônico**. Revista Eletrônica do CNJ, Conselho Nacional de Justiça, v. 4, n. 1, 2020.

CASTRO JUNIOR, A. P. e CALIXTO, W. P. **Assessor Jurídico Virtual Baseado em Ontologia**. Revista Eletrônica de Tecnologia da Informação Aplicada, FIEG/SENAI/FATESG, 4ª edição, 2016.

Trabalhos submetidos para periódicos:

CASTRO JUNIOR, A. P., CALIXTO, W. P. e WAINER, G. A. **Enhance the**

traditional weighting construction model with similarity-learning for text classification models in Court of Law. Journal of Experimental & Theoretical Artificial Intelligence, Taylor & Francis, 2021. (Está na segunda revisão).

CASTRO JUNIOR, A. P., CALIXTO, W. P. e WAINER, G. A. **Combining traditional bag-of-words with similarity-learning metrics and text classification models in Court of Law**. Journal of Artificial Intelligence and Law, Springer, 2021.

CASTRO JUNIOR, A. P., CALIXTO, W. P. e WAINER, G. A. **Application of Artificial Intelligence in the automatic identification and classification repetitive demand resolution incident in the Brazilian Court of Justice**. Revista Direito GV, FGV, 2021.

7.1.1 Notícias para a comunidade

Sistema Berna - Software que aplica inteligência artificial para agrupar ações idênticas no Judiciário Goiano. Lançamento em 2020.

Lançamento no Youtube em:

- https://www.youtube.com/watch?v=MY8OG7UGdhM

Notícias em:

- https://www.tjgo.jus.br/index.php/institucional/centro-de-comunicacao-social/17-tribunal/19972-artigo-de-representantes-do-tjgo-e-ufg-e-publicado-em-destaque-na-revista-comemorativa-aos-15-anos-do-cnj
- https://www.tjgo.jus.br/index.php/institucional/centro-de-comunicacao-social/20-destaque/19854-tjgo-lanca-sistema-de-inteligencia-artificial-que-agrupa-acoes-identicas

Sistema Berna - Evolução do sistema Berna no Judiciário Goiano, para aplicar em temas repetitivos, conforme noticiado em:

- https://www.tjgo.jus.br/index.php/institucional/centro-de-comunicacao-

social/17-tribunal/21876-nugepnac-se-reune-para-discutir-sistema-inteligencia

O objetivo é implementar no TJGO nova funcionalidade na ferramenta de Inteligência Artificial denominada Berna, que permite a leitura de peças processuais e rastreamento de assuntos que tenham semelhança com precedentes judiciais como o IRDR - Incidente de Resolução de Demandas Repetitivas, Repercussão Geral e Recursos Repetitivos.

7.2 Sugestões para trabalhos futuros

Como trabalhos futuros, têm-se:

- Implementação da proposta de minuta de decisão para o magistrado analisar,
- Desenvolvimento de função para reconhecimento de padrão com o intuito de automatizar o coeficiente ϕ,
- Aplicar modelos de multidimensões, como word2vec, law2vec e Bert, comparar os resultados,
- Evoluir os códigos disponibilizados no GitHub, principalmente o de pré-processamento,
- Integrar base de dados conceitual externa, como a WordNet na fase de preprocessamento, principalmente para buscar termos sinônimos aos da coocorrência identificados pelo método proposto,
- Utilizar o modelo estatístico de alocação latente de Dirichlet (LDA) no conjunto de dados desse trabalho, com objetivo de categorizar automaticamente os documentos,
- Evoluir a Berna no Judiciário Goiano para classificar temas repetitivos do Supremo e Superior Tribunal de Justiça,
- Aplicar o método proposto em base de dados de sentenças de outras naturezas processuais, como execução fiscal e na área criminal,

- Usar a metodologia proposta em outros domínios de conhecimento, como na área de segurança em redes, softwares de detecção de intrusão em redes entre outras.

APÊNDICE A

Cálculo do modelo *tf-idf*

Este anexo contém parte do código do Autosent utilizado para realizar o cálculo do modelo *tf-idf* das palavras/termos e sua ordenação. A pesquisa e a seleção é realizada no banco de dados relacional (PostgreSQL) de sentenças.

No código-fonte em A.1, são atribuídos pesos nas palavras dentro do *corpus*, em estrutura de dados `hash`, na variável `palavras_incidencias`.

Código-fonte A.1 - Cálculo do modelo *tf-idf* das palavras nas sentenças judiciais.

```
palavras_incidencias = {}
corpus = []
cont_doc=0
relacao_sentencas_certas = ActiveRecord::Base.connection.execute(
  # Aqui é inserido o código SQL em A.2
)
relacao_sentencas_certas.map do |documento|
 corpus << TfIdfSimilarity::Document.new(
   ActionView::Base.full_sanitizer.sanitize(
     preprocessamento(documento["sentenca"])).to_s
   )
end
model = TfIdfSimilarity::TfIdfModel.new(corpus)
relacao_sentencas_certas.map do |documento|
 ActionView::Base.full_sanitizer.sanitize(
      preprocessamento(documento["sentenca"])
 ).split.uniq.map do |palavra|
  if palavras_incidencias[palavra.to_s].blank?
   palavras_incidencias[palavra.to_s]=model.tfidf(corpus[cont_doc],palavra).abs
  else
   palavras_incidencias[palavra.to_s]+=model.tfidf(corpus[cont_doc],palavra).abs
  end
 end
 cont_doc+=1
end
```

Conforme comentário realizado no código-fonte em A.1, naquele ponto é inserido código-fonte em A.2, que é a estrutura `SQL`, usando `select`. A estrutura em `SQL` é utilizada antes de contabilizar as palavras nas sentenças judiciais selecionadas.

O código em A.2 encontrada todos números de processos com as condições de naturezas revisional e consignatória, no banco de dados de `inventarios`. Após encontrar todos os números de processos das naturezas referenciadas acima, é realizada a busca das sentenças, no caso em tela **aplicando as regras da ontologia com regras do especialista**, no inteiro teor das sentenças.

Código-fonte A.2 - Comando SQL utilizado para buscar as sentenças e aplicação da ontologia.

```
select sentenca from sentencas where processo in
(select processo from inventarios where natureza in
('RESCISAO␣CONTRATUAL','RESTITUICAO␣DE␣IMPORTANCIAS
PAGAS','CONSIGNACAO␣EM␣PAGAMENTO␣(␣CPC␣)',
'CONSIGNATORIA','REVISIONAL␣DE␣LOCACAO',
'REVISIONAL␣CLAUSULAS␣CONTRATUAIS␣C/C␣REST.IMPORTANCIAS
PAGAS␣C/PED.LIM','REVISIONAL','REVISIONAL␣CLAUSULAS
CONTRATUAIS␣C/C␣REST.IMP','REVISIONAL␣DE␣ALUGUEL␣(␣L.E.␣)'))
and sentenca not ilike '%reintegra%o%' and sentenca not ilike
'%alimentos%' and sentenca not ilike '%fam%lia%' and
sentenca not ilike '%aliment%' and sentenca not ilike '%dpvat%';
```

No código-fonte A.3 está a codificação utilizada para ordenar a estrutura em `hash`.

Código-fonte A.3 - Ordenação das palavras no Hash.

```
palavras_incidencias_ordenado = palavras_incidencias.sort {|a,b| a[1]<=>b[1]}
```

Transformando a estrutura em `hash` em `vetor` de `vetor`, ou seja, `matriz`, temos o resultado em A.4. Este é o resultado da quantidade de vezes que as palavras aparecem nas sentenças judiciais.

Código-fonte A.4 - Resultado da quantidade de vezes que as palavras aparecem nas sentenças judiciais.

```
[["documentos", 1013], ["Goiânia,", 1013], ["juros,", 1021], ["ainda", 1030],
["proteção", 1039], ["acordo", 1043], ["somente", 1052], ["deverá", 1075],
["devedor", 1079], ["Assim,", 1081], ["consignação", 1081], ["média", 1100],
["custas", 1106], ["correção", 1113], ["Defesa", 1159], ["inciso", 1162],
["Superior", 1163], ["contratual", 1172], ["julgamento", 1173], ["(fls.", 1177],
```

["sentença", 1181], ["Civil.", 1188], ["requerente", 1192], ["autos,", 1200],
["dias,", 1200], ["DECISÃO", 1211], ["aplicação", 1222], ["Civil,", 1238],
["contrato,", 1243], ["razão", 1265], ["processo", 1276], ["entendimento", 1342],
["crédito", 1345], ["Justiça", 1345], ["parcelas", 1354], ["quanto", 1364],
["Banco", 1389], ["qualquer", 1413], ["prova", 1428], ["efeitos", 1429],
["revisão", 1432], ["decisão", 1436], ["antecipação", 1467], ["mensal", 1469],
["relação", 1483], ["multa", 1496], ["remuneratórios", 1519], ["depósito", 1548],
["julgado", 1557], ["AGRAVO", 1560], ["encargos", 1655], ["comissão", 1686],
["presente", 1713], ["quando", 1723], ["direito", 1753], ["desde", 1755],
["partes", 1758], ["autor", 1782], ["mesmo", 1794], ["2016.", 1807],
["sendo", 1850], ["valores", 1856], ["contratos", 1929], ["Direito", 1999],
["conforme", 2004], ["entre", 2010], ["sobre", 2065], ["cláusulas", 2240],
["forma", 2261], ["pagamento", 2382], ["capitalização", 2384], ["Tribunal", 2423],
["autos", 2425], ["termos", 2523], ["tutela", 2542], ["cobrança", 2566],
["autora", 2647], ["prazo", 2825], ["Processo", 3027], ["pedido", 3066],
["artigo", 3231], ["valor", 4234], ["Código", 4319], ["contrato", 4706],
["parte", 7057], ["juros", 7222]]

APÊNDICE B

Matriz termo a termo

Este anexo contém parte do código utilizado para criar a matriz termo a termo, ou palavra a palavra. As dez palavras com maiores *tf-idf* nas sentenças judiciais, relacionadas as naturezas de revisional e consignatória, foram: i) juro, ii) contrato, iii) taxa, iv) valor, v) capitalização, vi) cobrança, vii) revisional, viii) pagamento, ix) cláusula e x) crédito.

No código-fonte em B.1 é calculado a similaridade dos termos/palavras no conjunto, dois a dois.

Código-fonte B.1 - Matriz termo a termo. Calcular a similaridade dos termos/palavras no conjunto, dois a dois.

```
#Passo 1 ==> Select nas sentenças relacionadas a Revisional e Consignatoria.
corpus = []
relacao_sentencas_certas = ActiveRecord::Base.connection.execute(
    #Aqui é inserido o código SQL em A.2, do anexo A
)
relacao_sentencas_certas.map do |documento|
    corpus << TfIdfSimilarity::Document.new(
        DescobrirTermosCorpus.new.preprocessamento(
            ActionView::Base.full_sanitizer.sanitize(documento["sentenca"])
        ).to_s
    )
end
model = TfIdfSimilarity::TfIdfModel.new(corpus)

#PASSO 2 ==> Criar a matriz da quantidade de vezes que a palavra aparece no documento.
matriz_termo_termo = []
(relacao_sentencas_certas.ntuples.to_i+1).times do
    matriz_termo_termo << []
end
matriz_termo_termo[0][0] = ""
matriz_termo_termo[0][1] = "contrato"
matriz_termo_termo[0][2] = "juro"
matriz_termo_termo[0][3] = "taxa"
```

```
matriz_termo_termo[0][4] = "valor"
matriz_termo_termo[0][5] = "capitalização"
matriz_termo_termo[0][6] = "cobrança"
matriz_termo_termo[0][7] = "revisional"
matriz_termo_termo[0][8] = "pagamento"
matriz_termo_termo[0][9] = "cláusula"
matriz_termo_termo[0][10] = "credito"
linha = 1
relacao_sentencas_certas.map do |sentenca|
 coluna=1
 matriz_termo_termo[linha][0] = linha.to_s
 while coluna < 11
  matriz_termo_termo[linha][coluna]=DescobrirTermosCorpus.new. \
   preprocessamento(
    ActionView::Base.full_sanitizer.sanitize(sentenca["sentenca"])
   ).split.uniq.map{|termo|
        model.tfidf(corpus[linha.to_i-1],termo).abs
          if termo==matriz_termo_termo[0][coluna]
   }.join.to_f
   coluna += 1
 end
 linha += 1
end
```

APÊNDICE C

Matriz de Similaridade

Este anexo contém parte do código utilizado para criar a matriz de similaridades, conforme método definido no Capítulo 5, aplicando a expressão (5.2).

Os resultados da aplicação do código-fonte C.1 estão nas matrizes estampadas na Tabela 6.3 e na Tabela 6.4.

Código-fonte C.1 - Cálculo do coeficiente de similaridade de Jaccard, dado por 5.2.

```
hash_similaridade_jaccard_alterado={}
array_termos = ["","contrato","juro","taxa","valor","capitalização",
"cobrança","revisional","pagamento","cláusula","credito"]
array_termos1 = 1
while array_termos1 < 10
 array_termos2 = array_termos1 + 1
 while array_termos2 < 11
  calc_multiplica_termos1e2=calc_adiciona_quadrado_termos1e2=0
  documento = 1
  while documento < (relacao_sentencas_certas.ntuples.to_i+1) #(FN1)
    calc_multiplica_termos1e2 += matriz_termo_termo[documento] \
    [array_termos1]*matriz_termo_termo[documento][array_termos2]
    calc_adiciona_quadrado_termos1e2 += \
        (matriz_termo_termo[documento][array_termos1]**2) \
          + (matriz_termo_termo[documento][array_termos2]**2)
    documento += 1
  end
  #calculo de jaccard alterados
   hash_similaridade_jaccard_alterado["#{array_termos[array_termos1]}-␣\
␣␣␣␣#{array_termos[array_termos2]}"]=(calc_multiplica_termos1e2.to_f/ \
     (calc_adiciona_quadrado_termos1e2.to_f-calc_multiplica_termos1e2. \
     to_f).to_f).to_f
   array_termos2 += 1
 end
 array_termos1 += 1
end
```

O código-fonte em C.2 faz a ordenação da matriz de similaridades.

Código-fonte C.2 - Ordenação da matriz de similaridades.

```
hash_similaridade_ordenado = \
    hash_similaridade_jaccard_alterado.sort {|a,b| a[1]<=>b[1]}
```

APÊNDICE D

Treinamento e testes dos modelos de classificação

Este anexo apresenta a classe em Python utilizada para realizar o treinamento supervisionado e testes dos nove modelos de classificação aplicados nesta tese. Na classe, observa-se também os métodos utilizados para calcular as métricas de avaliação dos modelos de classificação.

Código-fonte D.1 - Classe em Python que implementa os modelos de classificação e as métricas de avaliação.

```
class TreinamentoAvaliacao:
  import sys
  import numpy as np

  def ler_dataset(self, arquivo_dataset):
    f = open(arquivo_dataset, "r")
    dataset_array=[]
    label_array=[]
    for x in f:
        data = []
        label= 0
        for dado in x.split(';'):
            if "#" in dado:
                data.append(int(dado.split('#')[0]))
                label=int(dado.split('#')[1].split('\n')[0])
            else:
                data.append(int(dado))
        dataset_array.append(data)
        label_array.append(label)
    f.close()

    return dataset_array,label_array

  def tree(self, dataset_array, label_array, data_teste):
    from sklearn import tree
```

```
    clf = tree.DecisionTreeClassifier(max_depth=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def naivebayes(self, dataset_array, label_array, data_teste):
    from sklearn.naive_bayes import GaussianNB

    #Create a Gaussian Classifier
    clf = GaussianNB()
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def bnaivebayes(self, dataset_array, label_array, data_teste):
    from sklearn.naive_bayes import BernoulliNB

    #Create a Gaussian Classifier
    clf = BernoulliNB()
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def knn(self, dataset_array, label_array, data_teste):
    from sklearn.neighbors import KNeighborsClassifier

    #analise 2 resultados
    clf = KNeighborsClassifier(n_neighbors=3)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def mlp(self, dataset_array, label_array, data_teste):
    from sklearn.neural_network import MLPClassifier

    #analise 2 resultados
    clf = MLPClassifier(activation='logistic', solver='lbfgs', alpha=1e-5,
    hidden_layer_sizes=(100, 50), random_state=1, max_iter=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)
```

```
def svm(self, dataset_array, label_array, data_teste):
    from sklearn import svm

    #analise 2 resultados
    clf = svm.SVC(max_iter=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def rf(self, dataset_array, label_array, data_teste):
    from sklearn.ensemble import RandomForestClassifier

    clf = RandomForestClassifier(n_estimators=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def gbc(self, dataset_array, label_array, data_teste):
    from sklearn.ensemble import GradientBoostingClassifier

    clf = GradientBoostingClassifier(n_estimators=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def abc(self, dataset_array, label_array, data_teste):
    from sklearn.ensemble import AdaBoostClassifier

    clf = AdaBoostClassifier(n_estimators=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)

def gpc(self, dataset_array, label_array, data_teste):
    from sklearn.gaussian_process import GaussianProcessClassifier

    clf = GaussianProcessClassifier(max_iter_predict=300)
    clf.fit(dataset_array, label_array)
    return clf.predict(data_teste)
```

```
  def avaliar_acuracia(self, array_predicao, array_label_corretos):
    acuracia=cont=0
    for i in array_predicao:
      if i == array_label_corretos[cont]:
        acuracia+=1
      cont+=1
    return (acuracia / cont) * 100

  def avaliar_fmeasure(self, array_predicao, array_label_corretos):
    from sklearn.metrics import f1_score

    macro = f1_score(array_label_corretos,array_predicao, average='macro')
    micro = f1_score(array_label_corretos,array_predicao, average='micro')
    weighted = f1_score(array_label_corretos,array_predicao,
     average='weighted')

    return weighted

  def avaliar_all(self, array_predicao, array_label_corretos):
    from sklearn.metrics import precision_recall_fscore_support

    return precision_recall_fscore_support(array_label_corretos, array_predicao,
     average='weighted')

  def accuracy(self,array_predicao, array_label_corretos):
    from sklearn.metrics import accuracy_score

    return accuracy_score(array_label_corretos, array_predicao)

  if ___name___ == '___main___':
    if len(sys.argv) != 4:
        print("Erro␣na␣passagem␣dos␣argumentos")
        print("Correto:")
        print("python3␣<nome_arquivo_dataset>␣<nome_arquivo_teste>
␣␣␣␣␣<nome_metodo_predicao␣[mlp,␣svm,␣rf,␣gbc,␣abc,␣gpc,␣tree,␣nb,
```

```
␣␣␣␣␣␣bnb,␣knn]>")
    else:
        #constantes
        arquivo_dataset =sys.argv[1]
        arquivo_datateste=sys.argv[2]
        metodo_predicao =sys.argv[3]
        from TreinamentoAvaliacao import TreinamentoAvaliacao

        print("************************************")
        print("Classifier:␣{}".format(metodo_predicao))

        ta = TreinamentoAvaliacao()
        dataset_array, label_array = ta.ler_dataset(arquivo_dataset)
        data_teste, label_teste = ta.ler_dataset(arquivo_datateste)

        resultado_predicao=[]
        if metodo_predicao == "mlp":
            resultado_predicao = ta.mlp(dataset_array,label_array,data_teste)
        elif metodo_predicao == "svm":
            resultado_predicao = ta.svm(dataset_array,label_array,data_teste)
        elif metodo_predicao == "rf":
            resultado_predicao = ta.rf(dataset_array,label_array,data_teste)
        elif metodo_predicao == "gbc":
            resultado_predicao = ta.gbc(dataset_array,label_array,data_teste)
        elif metodo_predicao == "abc":
            resultado_predicao = ta.abc(dataset_array,label_array,data_teste)
        elif metodo_predicao == "gpc":
            resultado_predicao = ta.gpc(dataset_array,label_array,data_teste)
        elif metodo_predicao == "tree":
            resultado_predicao = ta.tree(dataset_array,label_array,data_teste)
        elif metodo_predicao == "nb":
            resultado_predicao = ta.naivebayes(dataset_array,label_array,
            data_teste)
        elif metodo_predicao == "bnb":
            resultado_predicao = ta.bnaivebayes(dataset_array,label_array,
            data_teste)
```

```
        elif metodo_predicao == "knn":
            resultado_predicao = ta.knn(dataset_array,label_array,data_teste)




        if resultado_predicao == []:
            print("************************************")
            print("  Erro:")
            print("Nome método de predição errado. Passar o tipo correto:
          [mlp, svm, rf, gbc, abc, gpc]")
            print("************************************")
        else:
            acuracia = ta.avaliar_acuracia(resultado_predicao,label_teste)
            print("************************************")
            print("Accuracy: {}".format(round(acuracia,2)))
            #print(resultado_predicao)
            #print(resultado_datateste)
            print("************************************")
            fmeasure = ta.avaliar_fmeasure(resultado_predicao,label_teste)
            print("F-measure: {}".format(round(fmeasure,2)))
            print("************************************")
            all = ta.avaliar_all(resultado_predicao,label_teste)
            print("Precision, recall, F-measure: {}".format(all))
            print("************************************")
            accuracy = ta.accuracy(resultado_predicao,label_teste)
            print("Accuracy: {}".format(round(accuracy,6)))
            print("Accuracy: {}".format(round(ta.accuracy(resultado_predicao,
            label_teste),6)))
            print("************************************")
```

APÊNDICE E

Implementação do preprocessamento

Este anexo apresenta a classe em Ruby-on-rails utilizada para realizar o pré-processamento dos textos. Esta fase é importante para a limpeza dos documentos, padronização de termos, remoção de termos irrelevantes, extração dos elementos HTML.

Código-fonte E.1 - Classe em Ruby-on-rails que implementa as fases do preprocessamento.

```
class CourtOfLawPreProcessing

  def begin(judge_text)
      return ActionView::Base.full_sanitizer.sanitize(
      preprocessamento(judge_text)).to_s
  end

  def preprocessamento(texto_sentenca)
      return normalization(fonetiza(stopwords(stemming(
      limpeza_texto(texto_sentenca)))))
  end

  private

  def stemming(texto)
      novo_texto = texto
      ['tenho','tens','tem','temos','tendes','têm','tinha','tinhas','tinhamos','tínheis',
      'tinham','tive','tiveste','teve','tivemos','tivestes','tiveram','tivera','tiveras',
      'tivéramos','tivéreis','tiveram','terei','terás','terá','teremos','tereis','terão',
      'teria','terias','teríamos','teríeis','teriam','tenha','tenhas','tenhamos','tenhais',
      'tenham','tivesse','tivesses','tivéssemos','tivésseis','tivessem','tiver','tiveres',
      'tiver','tivemos','tiverdes','tiverem','tendo'].map do |token|
          #novo_texto = novo_texto.gsub(" #{token} ",' ter ')
          novo_texto = novo_texto.gsub(" #{token} ",' ')
      end
      ['devo','deves','deve','devemos','deveis','devem','devia','devias','devíamos',
      'devíeis','deviam','devi','deveste','deveu','devemos','devestes','deveram',
```

```
'devera','devêramos','devêreis','deveram','deverei','deverás','deverá',
'deveremos','devereis','deverão','deveria','deverias','deveríamos',
'deveríeis','deveriam','deva','devamos','devais','devam','devesse',
'devesses','devesse','devêssemos','devêsseis','devessem','dever',
'deveres','devermos','deverdes','deverem','devendo'].map do |token|
    #novo_texto = novo_texto.gsub(" #{token} ",' dever ')
    novo_texto = novo_texto.gsub(" #{token} ",' ')
end
['estou','estás','está','estamos','estais','estão','estava','estavas','estávamos',
'estáveis','estavam','estive','estiveste','esteve','estivemos','estivestes',
'estiveram','estivera','estiveras','estivera','estivéramos','estivéreis',
'estiveram','estarei','estarás','estará','estaremos','estareis','estarão',
'estaria','estarias','estaríamos','estaríeis','estariam','esteja','estejas',
'estejamos','estejais','estejam','estivesse','estivesses','estivéssemos',
'estivésseis','estivessem','estiver','estiveres','estivermos','estiverdes',
'estiverem','estando'].map do |token|
    #novo_texto = novo_texto.gsub(" #{token} ",' estar ')
    novo_texto = novo_texto.gsub(" #{token} ",' ')
end
['sou','és','é','somos','sois','são','era','eras','éramos','éreis','eram','fui',
'foste','foi','fomos','fostes','foram','fora','foras','fôramos','fôreis','foram',
'serei','serás','será','seremos','sereis','serão','seria','serias','seríamos',
'seríeis','seriam','seja','sejas','sejamos','sejais','sejam','fosse','fosses',
'fôssemos','fôsseis','fossem','tor','fores','formos','fordes','forem',
'sendo'].map do |token|
    #novo_texto = novo_texto.gsub(" #{token} ",' ser ')
    novo_texto = novo_texto.gsub(" #{token} ",' ')
end

['digo','dizes','diz','dizemos','dizeis','dizem','dizia','dizias','dizíamos',
'dizíeis','diziam','disse','disseste','dissemos','dissestes','disseram',
'dissera','disseras','disséramos','dissêreis','disseram','direi','dirás',
'dirá','diremos','direis','dirão','diria','dirias','diríamos','diríeis','diriam',
'dissesse','dissesses','disséssemos','dissessem','disser','disseres',
'dissermos','disserdes','disserem'].map do |token|
    #novo_texto = novo_texto.gsub(" #{token} ",' dizer ')
```

```
    novo_texto = novo_texto.gsub(" #{token} ",' ')
end

novo_texto = novo_texto.gsub(' juros ',' juro ')
novo_texto = novo_texto.gsub(' multas ',' multa ')
novo_texto = novo_texto.gsub(' valores ',' valor ')
novo_texto = novo_texto.gsub(' contratos ',' contrato ')
novo_texto = novo_texto.gsub(' contratual ',' contrato ')
novo_texto = novo_texto.gsub(' contratuais ',' contrato ')
novo_texto = novo_texto.gsub(' embargos ',' embargo ')
novo_texto = novo_texto.gsub(' crédito ',' credito ')
novo_texto = novo_texto.gsub(' créditos ',' credito ')
novo_texto = novo_texto.gsub(' creditos ',' credito ')
novo_texto = novo_texto.gsub(' bens ',' bem ')
novo_texto = novo_texto.gsub(' filhos ',' filho ')
novo_texto = novo_texto.gsub(' alimento ',' alimentos ')
novo_texto = novo_texto.gsub(' divórcio ',' divorcio ')
novo_texto = novo_texto.gsub(' divórcios ',' divorcio ')
novo_texto = novo_texto.gsub(' divorcios ',' divorcio ')
novo_texto = novo_texto.gsub(' prazos ',' prazo ')
novo_texto = novo_texto.gsub(' acordos ',' acordo ')
novo_texto = novo_texto.gsub(' taxas ',' taxa ')
novo_texto = novo_texto.gsub(' pagamentos ',' pagamento ')
novo_texto = novo_texto.gsub(' cobranças ',' cobrança ')
novo_texto = novo_texto.gsub(' capitalizações ',' capitalização ')
novo_texto = novo_texto.gsub(' revisionais ',' revisional ')
novo_texto = novo_texto.gsub(' consignatórias ',' revisional ')
novo_texto = novo_texto.gsub(' consignatória ',' revisional ')
novo_texto = novo_texto.gsub(' consignatorias ',' revisional ')
novo_texto = novo_texto.gsub(' consignatoria ',' revisional ')
novo_texto = novo_texto.gsub(' esbulhos ',' esbulho ')
novo_texto = novo_texto.gsub(' área ',' area ')
novo_texto = novo_texto.gsub(' áreas ',' area ')
novo_texto = novo_texto.gsub(' areas ',' area ')
novo_texto = novo_texto.gsub(' benefícios ',' beneficio ')
novo_texto = novo_texto.gsub(' benefício ',' beneficio ')
```

```
    novo_texto = novo_texto.gsub('␣beneficios␣','␣beneficio␣')
    novo_texto = novo_texto.gsub('␣rurais␣','␣rural␣')
    novo_texto = novo_texto.gsub('␣aposentadorias␣','␣aposentadoria␣')
    novo_texto = novo_texto.gsub('␣idades␣','␣idade␣')
    novo_texto = novo_texto.gsub('␣segurados␣','␣segurado␣')
    novo_texto = novo_texto.gsub('␣provas␣','␣prova␣')
    novo_texto = novo_texto.gsub('␣federais␣','␣federal␣')
    novo_texto = novo_texto.gsub('␣sociais␣','␣social␣')
    novo_texto = novo_texto.gsub('␣systems␣','␣system␣')
    novo_texto = novo_texto.gsub('␣problems␣','␣problem␣')
    novo_texto = novo_texto.gsub('␣objects␣','␣object␣')
    novo_texto = novo_texto.gsub('␣results␣','␣result␣')
    novo_texto = novo_texto.gsub('␣images␣','␣image␣')
    novo_texto = novo_texto.gsub('␣transactions␣','␣transaction␣')
    novo_texto = novo_texto.gsub('␣songs␣','␣song␣')
    novo_texto = novo_texto.gsub('␣researchs␣','␣research␣')
    novo_texto = novo_texto.gsub('␣animals␣','␣animal␣')
    novo_texto = novo_texto.gsub('␣goats␣','␣goat␣')
    novo_texto = novo_texto.gsub('␣resources␣','␣resource␣')
    novo_texto = novo_texto.gsub('␣vocals␣','␣vocal␣')

    return novo_texto
end

def fonetiza(texto)
    novo_texto = texto

    ['farmácia','drogaria'].map do |token|
        novo_texto = novo_texto.gsub("␣#{token}␣",'␣farmácia␣')
    end

    ['mercado','supermercado','mercearia','hipermercado','atacadão',
    'carrefour'].map do |token|
        novo_texto = novo_texto.gsub("␣#{token}␣",'␣supermercado␣')
    end
```

```
    ['jogo futebol','jogar futebol','campeonato futebol','partida futebol',
    'jogar pelada','bater baba','estadio futebol','serra dourada',
    'estádio futebol','antonio accioly','antônio accioly',
    'estadio haile pinheiro','estadio hailé pinheiro'].map do |token|
        novo_texto = novo_texto.gsub(" #{token} ",' jogo futebol ')
    end

    return novo_texto
end

def stopwords(texto)
    novo_texto = texto

    #artigos, preposições, pronomes, numeral
    ['the','of','and','to','in','that','is','we','are','this','be','on','by','an','a','with',
    'from','can','it','which','these','such','not','or','also','between','using',
    'use','has','have','based','our','used','use','at','how','their','more','s',
    'show','some','about','re','you','may','will','i','all','div','they','t','than',
    'if','hi','into','them','your','c','first','only','been','what','who','each',
    'last','being','all','k','one','we','when','its','but','both','two','the','pm',
    'her','was','like','he','she','out','his','my','well','other'].map do |token|
        novo_texto = novo_texto.gsub(" #{token} ",' ')
    end

    #artigos, preposições, pronomes, numeral
    ['a','à','às','as','à','ás','jamais','agora','ainda','alguém','algum',
    'alguma''algumas','alguns','ampla','amplas','amplo','amplos',
    'ante','antes','ao','aos','após','aquela','aquelas','aquele','aqueles',
    'aquilo','as','até','através','cada','coisa','coisas','com','como','contra',
    'contudo','da','daquele','daqueles','das','de','dela','delas','dele','deles',
    'depois','desse modo','deste modo','dessa','dessas','desse','desses',
    'desta forma','desta','destas','deste','deste','destes','disto','dito','do','dos',
    'e','é','e','ela','elas','ele','eles','em','enquanto','entre','essa','essas','esse',
    'esses','esta','estas','este','estes','eu','feita','feitas','feito','feitos','for',
    'grande','grandes','isso','isto','já','la','la','lá','lhe','lhes','lo','mas','me',
    'mesma','mesmas','mesmo','mesmos','meu','meus','minha','minhas',
```

```
'muita','muitas','muito','muitos','na','não','nas','nem','nenhum','nessa',
'nessas','nesta','nestas','ninguém','no','nos','nós','nossa','nossas','nosso',
'nossos','num','numa','nunca','o','os','ou','outra','outras','outro','outros',
'outrossim','para','pela','pelas','pelo','pelos','pequena','pequenas',
'pequeno','pequenos','per','perante','pois','porque','pouca','poucas',
'pouco','poucos','primeiro','primeiros','própria','próprias','próprio',
'próprios','quais','qual','quando','quanto','quantos','que','quem','são',
'se','seja','sejam','sem','sempre','seu','seus','si','sido','só','sob','sobre',
'sua','suas','talvez','também','tampouco','te','teu','teus','ti','tido','toda',
'todas','todavia','todo','todos','tu','tua','tuas','tudo','última','últimas',
'último','últimos','um','uma','umas','uns','vez','vos','vós','dois','duas',
'três','treis','demais','além','alem','mais','assim sendo','assim','ja',
'nestes termos','nestes','qualquer','respectiva','respectivo',
'respectivas','respectivos',
'por'].map do |token|
    novo_texto = novo_texto.gsub(" #{token} ",' ')
end


#termos jurídicos comuns
['artigo','artigos','art','parágrafo','paragrafo','paragrafos','parágrafos',
'inciso','incisos','tipificacao legal','tipificacao','natureza','naturezas',
'classe','classes','assunto','assuntos','movimento','movimentos',
'lei federal','leis federais','lei complementar','leis complementares',
'lei','leis','codigo defesa consumidor','código defesa consumidor',
'codigo civil brasileiro','código civil brasileiro',
'codigo processo civil legislacao','código processo civil legislacao',
'codigo processo civil','código processo civil','codigo civil',
'código civil','codigo processo penal','código processo penal',
'codigo penal','código penal','cpp','cpc','cdc','súmula','súmulas',
'sumula','sumulas','ministerio publico','ministério público','requer',
'requerente','requerido','exequente','executado',
'promovente','promovido','reclamado','reclamante','certidao',
'certidão','especificação','especificacao','especie','especies','processo',
'ação','acão','acao','ré','autor','autora','grifei','grifo','data vênia',
'data venia','smj','fls','defensoria publica','defensoria pública','n',
'autos','caso','direito','parte','inicial','réu','reu','julgado',
```

```
        'juiz','justiça','termos','requerida','decisão','audiência',
        'processual','medida','juízo','rel','fato','julgamento','instrumento',
        'medida','pena','mérito','embargo','autores','vista','tjgo','exposto',
        'tribunal','sentença','face','pedido','pedidos','tutela','tutelas','agravo',
        'agravos'].map do |token|
            novo_texto = novo_texto.gsub(" #{token} ",' ')
        end

        return novo_texto
    end

    def limpeza_texto(texto)
        novo_texto = texto.downcase
        novo_texto = novo_texto.gsub('.','').gsub(',','').gsub('!',' ').
        gsub(':',' ').gsub(';',' ').gsub('?',' ').gsub('\"',' ').gsub('\'',' ').
        gsub('(',' ').gsub(')',' ').gsub('–','').gsub('r$',' ').gsub('$',' ').
        gsub('|',' ').gsub(' / ','').gsub('*',' ').gsub('§',' ').gsub('=',' ').
        gsub('nº','').gsub('{','').gsub('}','').gsub(/[.',‘?;<>_\/{}"]/,'')
    novo_texto = novo_texto.gsub('style@page','').gsub('style','').
    gsub('70px','').gsub('\"span ','').gsub('p&ampnbsp','').
    gsub(' p ','').gsub('\"lineheight"','').gsub('textindent','').
    gsub('justify','').gsub('150%','').gsub(' /p ','').gsub('100px','').
    gsub('\"fontsize','').gsub('marginleft','').
    gsub('70px','').gsub('&ampnbsp','').gsub('\"fontfamily','').
    gsub('80px','').gsub('30px','').gsub('100%','').
    gsub('marginright','').gsub(' margin ','').gsub('\"textalign','').
    gsub('\"fontfamily','').gsub('','').gsub(' times ','').gsub(' br ','').
    gsub(' span ','').gsub('lineheight','').gsub('fontsize','').
    gsub('fontfamily','').gsub('textalign','').gsub(' p ','').
    gsub('2016p','').gsub('3cm','').gsub('4cm','').gsub('p&ampnbsp','').
    gsub(' new ','').gsub('romanspan','').gsub('&ampnbsp','')

    return novo_texto
  end

  def normalization(texto)
```

```
␣␣␣␣␣novo_texto␣=␣texto.downcase
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('ç','c')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('á','a')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('à','a')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('ã','a')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('ô','o')

␣␣␣␣␣return␣novo_texto
␣␣end

␣␣def␣wordnet(texto)
␣␣␣␣␣novo_texto␣=␣texto
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣acordo␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣combinacao␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣contratacao␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣concordata␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣pacto␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣convenio␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣trato␣','␣contrato␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣interesse␣','␣juro␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣honorario␣','␣taxa␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣tarifa␣','␣taxa␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣tributo␣','␣taxa␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣tributacao␣','␣taxa␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣encargo␣','␣taxa␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣ordenado␣','␣pagamento␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣recompensa␣','␣pagamento␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣proposicao␣','␣cláusula␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣condicao␣','␣cláusula␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣possessao␣','␣posse␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣propriedade␣','␣posse␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣patrimonio␣','␣bem␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣pertence␣','␣bem␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣edificacao␣','␣imovel␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣construcao␣','␣imovel␣')
␣␣␣␣␣novo_texto␣=␣novo_texto.gsub('␣predio␣','␣imovel␣')
```

```
    novo_texto = novo_texto.gsub(' edificio ',' imovel ')
    novo_texto = novo_texto.gsub(' evidencia ',' prova ')
    novo_texto = novo_texto.gsub(' separacao ',' divorcio ')
    novo_texto = novo_texto.gsub(' alimentacao ',' alimento ')
    novo_texto = novo_texto.gsub(' comida ',' alimento ')
    novo_texto = novo_texto.gsub(' manja ',' alimento ')
    novo_texto = novo_texto.gsub(' mantimento ',' alimento ')
    novo_texto = novo_texto.gsub(' sustento ',' alimento ')
    novo_texto = novo_texto.gsub(' nutricao ',' alimento ')
    novo_texto = novo_texto.gsub(' ajuda ',' assistencia ')
    novo_texto = novo_texto.gsub(' auxilio ',' assistencia ')
    novo_texto = novo_texto.gsub(' proveito ',' beneficio ')
    novo_texto = novo_texto.gsub(' ganho ',' beneficio ')
    novo_texto = novo_texto.gsub(' lucro ',' beneficio ')
    novo_texto = novo_texto.gsub(' amparo ',' guarda ')
    novo_texto = novo_texto.gsub(' protecao ',' guarda ')
    novo_texto = novo_texto.gsub(' defesa ',' guarda ')
    novo_texto = novo_texto.gsub(' cuidado ',' guarda ')
    novo_texto = novo_texto.gsub(' seguranca ',' guarda ')
    novo_texto = novo_texto.gsub(' agrario ',' rural ')
    novo_texto = novo_texto.gsub(' agricola ',' rural ')
    novo_texto = novo_texto.gsub(' fundiario ',' rural ')
    novo_texto = novo_texto.gsub(' terreal ',' rural ')
    novo_texto = novo_texto.gsub(' campestre ',' rural ')
    novo_texto = novo_texto.gsub(' fazenda ',' rural ')
    novo_texto = novo_texto.gsub(' chacara ',' rural ')
    novo_texto = novo_texto.gsub(' aposentacao ',' aposentadoria ')
    novo_texto = novo_texto.gsub(' velhice ',' idade ')

    return novo_texto
  end


end
```

APÊNDICE F

Validação cruzada (k-fold)

Para implementação da validação cruzada é utilizada a biblioteca `sklearn.model_selection` do pacote *scikit-learn* em Python. As classes `cross_val_score` e `KFold` são as responsáveis por implementar o modelo de validação cruzada k-fold.

Código-fonte F.1 - Classes e métodos em Python para implementar o modelo de validação cruzada k-fold.

```
class TreinamentoAvaliacaoKfold:
  import sys

  from sklearn import tree
  from sklearn.naive_bayes import GaussianNB
  from sklearn.neighbors import KNeighborsClassifier
  from sklearn.neural_network import MLPClassifier
  from sklearn import svm
  from sklearn.ensemble import RandomForestClassifier
  from sklearn.ensemble import GradientBoostingClassifier
  from sklearn.ensemble import AdaBoostClassifier
  from sklearn.gaussian_process import GaussianProcessClassifier
  from sklearn.model_selection import cross_val_score
  #from sklearn.model_selection import StratifiedKFold
  from sklearn.model_selection import KFold

  def ler_dataset(self, arquivo_dataset):
    f = open(arquivo_dataset, "r")
    dataset_array=[]
    label_array=[]
    for x in f:
        data = []
        label= 0
        for dado in x.split(';'):
            if "#" in dado:
                data.append(int(dado.split('#')[0]))
```

```
                label=int(dado.split('#')[1].split('\n')[0])
            else:
                data.append(int(dado))
        dataset_array.append(data)
        label_array.append(label)
    f.close()

    return dataset_array,label_array

if ___name___ == '___main___':
    if len(sys.argv) != 3:
        print("Erro na passagem dos argumentos")
        print("Correto:")
        print("        python3 <nome_arquivo_dataset> <valor K de folds>")
    else:
        #constantes
        arquivo_dataset =sys.argv[1]
        k  =sys.argv[2]
        from TreinamentoAvaliacaoKfold import TreinamentoAvaliacaoKfold

        print("************************************")
        print("Valor do K folds: {}".format(k))

        ta = TreinamentoAvaliacaoKfold()
        X_train, Y_train = ta.ler_dataset(arquivo_dataset)
        #data_teste, label_teste = ta.ler_dataset(arquivo_datateste)

        modelos = []
        modelos.append(('rf',RandomForestClassifier(n_estimators=300)))
        modelos.append(('mlp',MLPClassifier(activation='logistic', solver='lbfgs', \
         alpha=1e-5, hidden_layer_sizes=(100, 50), random_state=1, \
         max_iter=300)))
        modelos.append(('gbc',GradientBoostingClassifier(n_estimators=300)))
        modelos.append(('abc',AdaBoostClassifier(n_estimators=300)))
        modelos.append(('gpc',GaussianProcessClassifier(max_iter_predict=300)))
        modelos.append(('svm',svm.SVC(max_iter=300)))
```

```
modelos.append(('nb',GaussianNB()))
modelos.append(('knn',KNeighborsClassifier(n_neighbors=3)))
modelos.append(('tree',tree.DecisionTreeClassifier(max_depth=300)))

resultados = []
nomes = []
for nome, modelo in modelos:
    #kfold = StratifiedKFold(n_splits=8, random_state=1)
    kfold = KFold(n_splits=8, shuffle=True)
    cv_results = cross_val_score(modelo, X_train, Y_train, cv=kfold, \
     scoring='accuracy')
    #cv_results = cross_val_score(modelo, X_train, Y_train, cv=kfold, scoring='f1_macro')
    resultados.append(cv_results)
    nomes.append(nome)
    print('%s: %f (%f)' % (nome, cv_results.mean(), cv_results.std()))
```

REFERÊNCIAS BIBLIOGRÁFICAS

ABUALIGAH, L.; GANDOMI, A. H.; ELAZIZ, M. A.; HUSSIEN, A. G.; KHASAWNEH, A.; ALSHINWAN, M.; HOUSSEIN, E. H. Nature-inspired optimization algorithms for text document clustering: A comprehensive analysis. **Algorithms**, MDPI, 2020. 18, 99, 100, 101, 102, 116

ACKRILL, J. L. **Aristotle: Categories and De Interpretatione.** [S.l.]: Oxford: Clarendon, 1963. 49

AGARWAL, N.; SIKKA, G.; AWASTHI, L. K. Enhancing web service clustering using length feature weight method for service description document vector space representation. **Expert Systems with Applications Journal**, 2020. 74, 77, 118

ALMEIDA, M. B. **Uma abordagem integrada sobre ontologias: Ciência da Informação, Ciência da Computação e Filosofia.** [S.l.]: Perpectivas em Ciência da Informação, vol. 19, n. 3, 2014. 48, 49, 50, 51, 53, 54, 55

ALMEIDA, V. N. P. O. e. K. C. C. M. B. **Estudo Exploratório sobre Ontologias Aplicadas a Modelos de Sistemas de Informação: Perspectivas de Pesquisa em Ciência da Informação**. [S.l.]: Revista Eletrônica Bibliotecon. Ci. Inf. ISSN 1518-2924, Florianópolis, v. 15, n. 30, p.32-56, 2010. 60

AMBROSIO, E. A. M. M. e A. P. L. **Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens**. [S.l.]: Relatório Técnico, INF, UFG, 2007. 27, 48, 54

ARAUJO E. FARIA, M. M. e. e. E. M. **Tribunal de Justiça do Estado de Goiás, 125 anos de sua Instalação**. [S.l.]: Tribunal de Justiça do Estado de Goiás, 2000. 39, 40

ARISTÓTELES. **A Política**. [S.l.]: São Paulo: Martins Fontes, 2001. 37

BAX, M. B. A. e M. P. **Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção**. [S.l.]: Revista Ciência da Informação, 32(3), 2003. 47

BEYER, C. **Edmund Husserl**. [S.l.]: Stanford, 2011. 51

BLAIR, D. C. **Wittgenstein, language and information: back to the rough ground**. [S.l.]: Amsterdan: Springer, 2006. 55

BORKO, H. **Information Science: what is it?** [S.l.]: American Documentation, Washington, v. 19, p. 3-5, 1968. 63

BORST, W. N. **Construction of Engineering Ontologies for Knowledge Sharing and Reuse**. [S.l.]: PhD thesis, University of Twente, P.O. Box 217 - 7500 AE Enschede - The Netherlands, 1997. 53

BOSCH, A.; ZISSERMAN, A.; MUNOZ, X. Scene classification using a hybrid generative/discriminative approach. **IEEE Trans. Pattern Analysis andMachine Intelligence, 30**, IEEE, 2008. 74

CASTRO, A. P.; CALIXTO, W. P.; ARAUJO, C. H. Application of artificial intelligence in the identification of connections by fact and thesis in the judicial complaint and integration with the electronic system of lawsuits (in portuguese). **CNJ Magazine**, v. 4, n. 1, p. 10, 2020. 33

CASTRO, A. P.; CALIXTO, W. P.; GOMES, V. M.; VEIGA, E. F.; SILVA, L. F.; CASTRO, L. L. O. P.; BARBOSA, J. L. F.; CAMPOS, P. H. Ontology applied in the judicial sentences. In: IEEE. **2017 CHILECON Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)**. [S.l.], 2017. p. 1–6. 33, 75

CASTRO, A. P.; CALIXTO, W. P.; GOMES, V. M.; VEIGA, E. F.; SILVA, L. F.; CAMPOS, P. H. Ontology to mining judicial sentence´s big data. In: **Alive Engineering Education**. [S.l.]: UFG, 2017. p. 187–196. ISBN 978-85-495-0151-6. 33, 75

CASTRO, W. P. C. e. B. F. F. A. P. **Gestão da Informação em Grandes Volumes de Dados no Poder Judiciário**. 5. ed. [S.l.]: V Coletânea Luso-Brasileira - Gestão da Informação, Cooperação em Redes e Competitividade, Cap. 2, p. 61-78, 2014, 2014. 28, 34, 41

CHALKIDIS, I. Law2vec - legal word embeddings. Archive.org, 2021. 31, 75

CURRAS, E. **Ontologies, Taxonomies and Thesauri in Systems Science and Systematics**. [S.l.]: Cambridge: Woodhead, 2003. 54

DELICATO, C. F. **Fenix - sistema de filtragem personalizada de informações para a WEB**. [S.l.]: Dissertação UFRJ, 2000. 64

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **Computing Research Repository (CoRR)**, Cornell University, arXiv, 2018. 74

DIAS, C. A. M. **Pesquisas em Inteligência Artificial: uma análise na biblioteconomia brasileira**. [S.l.]: Universidade de Brasília (UNB) - Monografia Faculdade de Ciência da Informação, 2015. 64, 68

FACHIN, B. R. G. **Recuperação inteligente da informação e Ontologias: um levantamento na área da Ciência da Informação**. [S.l.]: Biblos, Rio Grande, 23 (1): 259-283, 2009. 67

FERGUS, R.; PERONA, P.; ZISSERMA, A. Object class recognition by unsuper-vised scale-invariant learning. CVPR, 2003. 74

FERNEDA, E. **Recuperação de Informação: Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação**. [S.l.]: Tese, Escola de Comunicação e Artes, Universidade de São Paulo, 2005. 65, 66, 71, 72

FOX, C. J. **Information and Misinformation: An Investigation of the Notions of Information, Misinformation, Informing, and Misinforming**. [S.l.]: Westport: Greenwood, 1983. 54, 55

FUX, L. **Justiça em Números**. 2021. ed. [S.l.]: Revista Digital do Conselho Nacional da Justiça, Volume 1, Ano-base 2020, 2021. 38

GABRILOVICH, E.; MARKOVITCH, S. Feature generation for text categorization using world knowledge. **In Proceedings of the 19th International Joint Conference on Artificial Intelligence**, Kaufmann, 2005. 74

GANGEMI, M. C. e A. ***An OWL ontology library representing judicial interpretations***. [S.l.]: SEMANTIC WEB, v. 7, ed. 3, p. 229-253, DOI 10.3233/SW-140146, 2016. 30

GAO, N. Z. Y. P. S. Y. J. Z. e J. ***An Ontological Chinese Legal Consultation System***. [S.l.]: IEEE ACCESS, v. 5, p. 18250-18261, DOI 10.1109/ACCESS.2017.2745208, 2017. 30

GARG, V.; VEMPATI, S.; JAWAHAR, C. V. Bag of visual words: A soft clustering based exposition. **Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics**, 2011. 74

GIARETTA, N. G. e P. **Ontologies and Knowledge Bases: towards a terminological clarification**. [S.l.: s.n.], 1995. 53

GILCHRIST, A. **Thesauri, Taxonomies and Ontologies; an etymological note**. [S.l.: s.n.], 2003. 54

GOMAA, W. H.; FAHMY, A. A. A survey of text similarity approaches. **International Journal of Computer Applications**, 2013. 75

GOMES, S. M. B. e. A. J. R. E. R. **Representação do Conhecimento Jurídico através da Ontologia: Exercício do Governo Eletrônico**. [S.l.]: 6º CONTECSI - *International Conference on Information Systems and Technology Management*, p. 285-285, São Paulo, USP, 2009. 29

GRUBER, T. **What is an ontology?** [S.l.: s.n.], 2005. 53

GUARINO, N. **Formal Ontology in Information Systems**. [S.l.: s.n.], 1998. 53, 58, 59

GUIMARãES, e. a. U. **Constituição da República Federativa do Brasil**. [S.l.]: Presidencia da Republica, Casa Civil, 1988. 37

GUIMARãES, F. J. Z. **Utilização de ontologias no domínio B2C**. [S.l.]: Departamento de Informática, dissertação, PUC-Rio de Janeiro, 2002. 59

HAUSLADEN, C. I.; SCHUBERT, M. H.; ASH, E. Text classification of ideological direction in judicial opinions. **International Review of Law and Economics**, Elsevier, 2020. 31, 116, 117

HUANG, L.; MILNE, D.; FRANK, E.; WITTEN, I. H. Learning a concept-based document similarity measure. **Journal of the American Society for Information Science and Technology, ISSN 1532–2882**, Wiley, 2012. 74

JANSEN, L. **Categories: The Top Level Ontology**. [S.l.]: Berlin: Verlag, p. 173-196, 2008. 50

JASANOFF, S. Science, common sense & judicial power in u.s. courts. **Daedalus – Journal of the American Academy of Arts & Sciences**, Daedalus, 2018. 33

JONES, K. S. **Information retrieval and artificial intelligence**. [S.l.]: Artificial Intelligence, vol. 114 (1-2), p. 257-81, 1999. 64

KATZ, D. M.; BOMMARITO, M. J.; BLACKMAN, J. A general approach for predicting the behavior of the supreme court of the united states. **PLOS ONE**, PLOS, 2017. 32, 117

KIM, Y. Convolutional neural networks for sentence classification. **Computing Research Repository (CoRR)**, Cornell University, arXiv, 2014. 74

LANCASTER, W. F. **Indexação e resumos: teoria e prática**. [S.l.]: Briquet de Lemos, Brasília, 2a edição, p. 452, 2004. 64

LAZEBNIK, S.; SCHMID, C.; PONCE, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. CVPR, 2006. 74

LENNOX, J. **Aristotle's Philosophy of Biology: Studies in the Origins of Life Science**. [S.l.]: Cambridge University Press, 2000. 49

LI, P.; MAOA, K.; XU, Y.; LI, Q.; ZHANG, J. Bag-of-concepts representation for document classification based on automatic knowledge acquisition from probabilistic knowledge base. **Knowledge-Based Systems**, Elsevier, 2020. 74, 77, 118

MAGALHãES, M. T. e A. C. **Modifica o regime e dispõe sobre princípios e normas da Administração Pública**. [S.l.]: Casa Civil, Presidência da República, 1998. 27

MANDAL, A.; GHOSH, K.; GHOSH, S.; MANDAL, S. Unsupervised approaches for measuring textual similarity between legal court case reports. **Artificial Intelligence and Law**, Springer, 2021. 31, 74, 116, 117

MARCONDES, S. B. e D. **Dicionário Oxford de Filosofia**. [S.l.]: Rio de Janeiro, 1997. 48

MARQUES FILHO, G. **Plano de Gestão, biênio 2017/2019**. 2017. ed. [S.l.]: TJGO, 2017. 34

MARTINS, L. A. **Potenciais aplicações da Inteligência Artificial na Ciência da Informação**. [S.l.]: Informação e Informação (UEL. *Online*), v. 15, p. 1-16, 2010. 66

MATTHEW, H. **A Pratical Guide to Building OWL Ontologies using PROTÉGÉ and CO-ODE Tools**. [S.l.]: The University of Stanford, 2011. 60, 103

MATTOS, M. C. **A Metodologia *Methontology* na Construção de Ontologias**. [S.l.]: Revista Eletrônica de Iniciação Científica da UNESC, v. 5, n. 1, 2007. 60

MCGILL, G. S. e M. **Introdution to Modern Information Retrieval**. [S.l.]: Computer Science Series, USA. McGraw-Hill, 1983. 71

MEDVEDEVA, M.; VOLS, M.; WIELING, M. Using machine learning to predict decisions of the european court of human rightss. **Artificial Intelligence and Law**, Springer, 2020. 32, 116, 117

MIHALCEA, R.; CORLEY, C.; STRAPPARAVA, C. Corpus-based and knowledge-based measures of text semantic similarity. **In Proceedings of the 21st National conference on Artificial Intelligence**, AAAI Press, 2006. 74

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **Computing Research Repository (CoRR)**, Cornell University, arXiv, 2013. 74

MILNE, D. N.; WITTEN, I. H.; NICHOLS, D. M. A knowledge-based search engine powered by wikipedia. **In Proceedings of the 16th Association for Computing Machinery (ACM) Conference on Information and Knowledge Management**, ACM Press, 2007. 74

MILOSAVLJEVIC, I. C.-F. G. S. S. G. M. S. B. ***Semantic integration of enterprise information systems using meta-metadata ontology***. [S.l.]: INFORMATION SYSTEMS AND E-BUSINESS MANAGEMENT, v. 15, ed. 2, p. 257-304, DOI 10.1007/s10257-015-0303-6, 2017. 30

MIRONCZUK, M. M.; PROTASIEWICZ, J. A recent overview of the state-of-the-art elements of text classification. **Expert Systems With Applications**, Elsevier, 2018. 117

MOOERS, C. **Zatocoding applied to mechanical organization of knowledge**. [S.l.]: American Documentation, v. 2, n. 1, p. 20-32, 1951. 64

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. [S.l.]: The MIT Press (Cambridge Massachusetts), 2013. 74

NETTO, E. **A Influência da Jurisprudência no Direito Brasileiro**. [S.l.]: Direito Net, 2011. Acessado em 18/12/2016. 27

OLIVEIRA, H. G.; PAIVA, V.; FREITAS, C.; RADEMAKER, A.; REAL, L.; SIMOES, A. As wordnets do português. **OSLa**, Universitas Osloensis, 2015. 61, 62, 108

_____. An overview of portuguese wordnets. **8 Global WordNet Conference**, GWC2016, 2016. 61, 62, 108

OLIVEIRA, P. M. e A. B. **Organização e Informática no Poder Judiciário**. [S.l.]: Editora Juruá, 2008. 45

ORDONEZ, M. A. C. A. C. A. O. e H. ***Judicial Precedents Search Supported by Natural Language Processing and Clustering***. [S.l.]: 10TH COMPUTING COLOMBIAN CONFERENCE (10CCC), p. 372-377, 2015. 29

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. [S.l.]: Association for Computational Linguistics, 2014. p. 1532–1543. 74

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. In: **North American Chapter of the Association for Computational Linguistics - NAACL**. [S.l.]: ACM Digital Library, 2018. 74

PU, N. Z. P. W. e Y. ***Challenges and Related Issues for Building Chinese Legal Ontology***. [S.l.]: PROCEEDINGS OF THE 2015 INTERNATIONAL CONFERENCE ON MECHATRONICS, ELECTRONIC, INDUSTRIAL AND CONTROL ENGINEERING, p. 1260-1265, 2015. 29, 30

RADYGIN, V.; KUPRIYANOV, D.; BESSONOV, R.; IVANOV, M.; OSLIAKOVA, I. Application of text mining technologies in russian language for

solving the problems of primary financial monitoring. **Procedia Computer Science**, Elsevier, 2021. 31, 116, 117

ROBREDO, J. **Documentação de hoje e de amanhã: uma abordagem revisitada e contemporânea da ciência da informação e de suas aplicações biblioteconômicas, documentárias, arquivísticas e museológicas**. [S.l.]: Revista e Ampliada, Brasília-DF, 4a Edição, 2005. 63, 64

ROVER, H. S. R. J. e A. J. **O Ato Administrativo Eletrônico sob a Ótica do Princípio da Eficiência**. [S.l.]: II Conferência Sul-Americana de Ciência e Tecnologia Aplicada ao Governo Eletrônico-CONEGOV, p. 33-44, 2005. 27

SALTON, G. **Another look at automatic text-retrieval systems**. [S.l.]: Communications of the ACM, v. 29, n. 7, 1986. 68

SARACEVIC, T. **Ciência da informação: origem, evolução e relações**. [S.l.]: Perspectivas em Ciência da Informação, v. 1, n. 1, p. 41-62, 1996. 63, 64

SEO, S.; SEO, D.; JANG, M.; JEONG, J.; KANG, P. Unusual customer response identification and visualization based on text mining and anomaly detection. **Expert Systems with Applications Journal**, 2020. 74, 77, 118

SEWALD, E. J.; ROVER, A. J. **Modelagem de Sistema de Conhecimento para Apoio a Decisão Sentencial**. [S.l.]: Pós-Graduação em Engenharia e Gestão do Conhecimento, Dissertação, UFSC, Florianópolis, Santa Catarina, 2012. 28

SEWALD, P. F. S. e. E. R. G. S. E. J. **Gestão de Conhecimento para Administração Judiciária: Levantamento de Demandas de Conhecimento e Estabelecimento de Ontologias**. 5. ed. [S.l.]: Revista Democracia Digital e Governo Eletrônico, 2011. 28, 34

SILVA, E. R. G. **Representation of Legal Knowledge Through Ontologies: Exercise in Electronic Government**. [S.l.]: International Conference on Information Systems and Technology Management, 2009. 28

SILVEIRA, M. L. **Recuperação Vertical de Informação: um estudo de caso na área jurídica**. [S.l.]: Tese de Doutorado, Informática Pública, vol. 5 (1), p. 133-134, 2003. 29

SKRLJ, B.; MARTINC, M.; KRALJ, J.; LAVRAC, N.; POLLAK, S. tax2vec: Constructing interpretable features from taxonomies for short text classification. **Computer Speech & Language**, Elsevier, 2021. 116, 117

SMITH, B. **Ontology: Philosophical and computacional**. [S.l.: s.n.], 2006. 48

SMITH, B. S. e D. W. **The Cambrigde Companion to Husserl**. [S.l.]: Cambridge Press, 2005. 52

STUDER, S. S. e R. **Handbook on Ontologies**. [S.l.]: Berlin: Springer, 2004. 54

SULIS, E.; HUMPHREYS, L.; VERNERO, F.; AMANTEA, I. A.; AUDRITO, D.; CARO, L. D. Exploiting co-occurrence networks for classification of implicit inter-relationships in legal texts. **Information Systems**, Elsevier, 2021. 30, 116, 117

TAVARES, D. S. **O sofrimento no trabalho entre servidores públicos: uma análise psicossocial do contexto de trabalho em um Tribunal Judiciário Federal**. [S.l.]: Departamento de Saúde Ambiental da Faculdade de Saúde Pública da Universidade de São Paulo, 2003. 37

TRT, T. R. T. **Gestão Estratégica**. [S.l.]: Tribunal Regional do Trabalho, 10a Região, 2013. 42

VAZ, C. **Judiciário Goiano**. [S.l.]: Kelps, 2014. 40

VICKERY, B. **Ontologies**. [S.l.]: Journal of Information Science, London, v. 23, n. 4, p. 227-286, 1997. 54

VIEIRA, S. G. C. e C. L. Martinewski e L. J. M. **Mensuração da carga de trabalho de magistrados: uma análise comparativa do estudo realizado no Tribunal de Justiça do Estado do Rio Grande do Sul com experiências internacionais**. [S.l.]: Revista da Faculdade de Direito da UFRGS, n. 26, 2006. 37

WALTL, B.; BONCZEK, G.; SCEPANKOVA, E.; MATTHES, F. Semantic types of legal norms in german laws: classification and analysis using local linear explanations. **Artificial Intelligence and Law**, Springer, 2018. 32, 116, 117

WAND, V. S. e. R. W. Y. **An ontological analysis of the relationship construct in conceptual modeling**. [S.l.]: ACM Transactions on Database Systems, New York, v.24, n. 4, p. 494-528, 1999. 54

WEBER, Y. W. e R. **Mario Bunge's ontology as a formal foundation for information systems concepts**. [S.l.]: Amsterdam: Rodopi, 1990. 54

WELTY, B. S. e C. **Ontology: Towards a new synthesis**. [S.l.: s.n.], 2001. 54, 58

WOOD, A. **Kant**. [S.l.]: Oxford: Wiley-Blackwell, 2004. 50

XIA, T.; CHAI, Y. An improvement to tf-idf: Term distribution based term weight algorithm. **Journal of Software 6(3): 413-420**, 2011. 74

GLOSSÁRIO

Jurisprudência - é termo jurídico, que significa o conjunto das decisões, aplicações e interpretações das leis. É uma base de dados constanto apenas a decisão dos magistrados nos processos, geralmente, sem o inteiro teor da decisão. É considerada fonte não formal do direito, porém é bastante utilizada para reforçar a conclusão do julgador. Note-se que a jurisprudência poderá ter força equiparada à das normas jurídicas, tornando-se fonte formal, quanto transformar-se em súmula vinculante.

Súmula Vinculante - trata-se de pronunciamentos proferidos pelos Tribunais do nosso país, baseados em decisões reiteradas, que delimitam o entendimento e interpretação das leis sobre determinada matéria dada pelos nossos magistrados. É a união de várias decisões de um mesmo Tribunal, com idêntica interpretação sobre o mesmo tema. Pode ser conceituada também como um resumo da decisão.

Comarca - é o território ou circunscrição territorial em que o juiz de direito de primeira instância exerce sua jurisdição. Existem comarcas em três instâncias: i) entrância inicial, ii) entrância intermediária e iii) entrância final. O tipo da entrância depende do número de habitantes, número de eleitores, arrecadação da região e número médio de serviço forense mínimo ajuizados no triênio.

Tribunal - é orgão público que tem como objetivo a resolução de litígios. Estes órgãos são os principais meios de resolver disputas e se pressupõe que qualquer pessoa pode expor suas reivindicações perante o tribunal.

Decisão Monocrática - é a decisão proferida por apenas uma julgador.

Decisão Colegiada - é a decisão proferida por vários julgadores, sendo contado os votos para saber qual é a decisão da maioria. Os órgãos colegiados devem ser sempre em número impar.

Jurisdição - é o poder que o Estado detém para aplicar o direito a determinado caso, com o objetivo de solucionar conflitos de interesses e com isso resguardar a ordem jurídica e a autoridade da lei. No sentido coloquial, jurisdição é a área territorial (município, estado, região ou país) sobre o qual este poder é exercido por determinada autoridade ou juízo.

Web Semântica - é a extensão da Internet atual, que permitirá aos computadores e humanos trabalharem em cooperação. A Web Semântica interliga significados de palavras, tendo como finalidade conseguir atribuir significado (sentido) aos conteúdos publicados na Internet, de modo que seja perceptível tanto pelo humano como pelo computador.

Representação do Conhecimento - é o conjunto de sentenças em linguagem formal para a qual foram definidas sua semântica e um conjunto de regras de inferência, capazes de gerar novas sentenças a partir das sentenças disponíveis.

Balística - é a ciência que estuda o movimento dos projéteis, especialmente das armas de fogo, seu comportamento no interior e exterior destas, como: trajetória; impacto; marcas; explosão e etc.

Prolog - é linguagem de programação, se enquadra no paradigma de programação em lógica matemática. É utilizada, especialmente, com inteligência artificial e linguística computacional.

Sistemas Especialistas - são programas que têm como objetivo simular o raciocínio de um profissional experiente em alguma área de conhecimento específica.

Lógica *Booleana* - é a forma de lógica na qual os valores lógicos das variáveis podem ser apenas 0 (falso) ou 1 (verdadeiro).

Lógica *Fuzzy* - lógica difusa, é a forma de lógica multivalorada na qual os valores lógicos das variáveis podem ser qualquer número real, entre 0 (falso) e 1 (verdadeiro).

IoT - Internet das Coisas - é a idéia básica de que qualquer equipamento eletrônico poderá ser conectado a Internet gerando informações sobre uso, perfil de consumo dos usuários, estatísticas e etc. O conceito aborda ainda a capacidade de controlar pela Internet os equipamentos, ligando e desligando, controlando volume e/ou quantidade e outras medidas possíveis, respeitando as características de funcionamento de cada um.

Primeiro grau de jurisdição - também chamada de primeira instância, refere-se, em regra, ao juízo em que se iniciou a demanda judicial, ou onde foi proposta a ação inicial.

Segundo grau de jurisdição - também chamada de segunda instância, é aquela à qual se recorre quando se pretende modificar decisão ou sentença de primeira instância.

SQL - é a sigla inglesa de *Structured Query Language*, que significa, em português, Linguagem de Consulta Estruturada. É linguagem padrão de gerenciamento de dados que interage com os principais bancos de dados baseados no modelo relacional.

PostgreSQL - é o gerenciador de banco de dados baseado no modelo relacional. É um software livre, bastante utilizado pelas empresas.

ElasticSearch - é software livre, de código aberto, para busca e armazenamento de documento. Tem a capacidade de trabalhar com grandes volumes de dados em tempo real.

unigrama - é a técnica de saco-de-palavras que constrõe peso por termo.

n**-gramas** - é a técnica de saco-de-palavras que constrõe peso na coocorrência entre termos, sendo $2 \times 2, 3 \times 3, \cdots, n \times n$.