Análise Comparativa de Embeddings Jurídicos aplicados a Algoritmos de Clustering

José Alfredo Ferreira Costa
Dept. de Engenharia Elétrica
Universidade Federal do Rio Grande do Norte, Natal, RN, Brasil
E-mail: alfredo.costa@ufrn.br
https://orcid.org/0000-0002-1290-6454

Nielsen Castelo Damasceno Dantas
PEP – Programa de Pós-graduação em Engenharia de Produção.
Universidade Federal do Rio Grande do Norte. Natal, RN, Brasil
E-mail: nielsen.tekla@gmail.com
https://orcid.org/0000-0002-5699-2657

***Abstract*— Text clustering analysis plays an important role in the organization and comprehension of extensive amounts of textual data. By grouping semantically similar documents into coherent categories, or clusters, it is possible to extract pertinent information and the unearthing of latent patterns embedded within the text. Text clustering enables a deeper understanding of the underlying structure and relationships within textual data, therefore, unveiling patterns and thematic trends. This paper aims to evaluate the impact of different text embeddings in the task of clustering Brazilian legal documents. The embeddings were obtained from BERT (Bidirectional Encoder Representations from transformers) models: Jurisbert, Bert Law and Irisbert. Term Frequency-Inverse Document Frequency (TF-IFD) was also used as a base representation model for comparisons. Nine different clustering algorithms were tested, including methods such as MB Kmeans, DBSCAN, BIRCH. Experiments were conducted in a database of 30,000 documents in Brazilian Portuguese of judicial moves of the Tribunal de Justiça do Rio Grande do Norte. To evaluate the performance of the clustering algorithms, the Normalized Mutual Info and Jaccard coefficients were used. Processing time are also described for the different algorithms. Results suggest better results with embedding "Irisbert" and TF-IDF when considering NMI and "Bert Law" and TF-IDF when considering Jaccard coefficient, although "Irisbert" also produced good scores.**

Keywords—Text clustering, Embeddings, BERT, Legal documents, NLP

I. INTRODUÇÃO

A revolução digital acelerou o aumento no volume de dados e complexidade em várias áreas, incluindo o sistema judiciário. No Brasil, por exemplo, o Conselho Nacional de Justiça (CNJ) relata que o número de casos judiciais chegou a aproximadamente 77 milhões em 2019. Diante desse cenário, a digitalização judicial tem sido adotada como uma forma de lidar com a quantidade massiva de processos, sendo a plataforma PJe amplamente utilizada por tribunais brasileiros.

No entanto, a eficiência e celeridade no processamento desses casos tornam-se desafios para o sistema judiciário. É nesse contexto que técnicas de aprendizado de máquina podem desempenhar um papel fundamental, aproveitando os grandes volumes de dados disponíveis para auxiliar em atividades como classificação automática de documentos e organização por similaridade. A automação de etapas do processo, especialmente aquelas repetitivas, pode contribuir para a efetividade e rapidez nas decisões judiciais.

No Brasil, já existem diversos projetos que buscam aplicar Inteligência Artificial (IA) no âmbito jurídico, incluindo o SINAPSES desenvolvido pelo Tribunal de Justiça de Rondônia em colaboração com o CNJ. Esses projetos visam criar um ambiente colaborativo capaz de compartilhar modelos de aprendizado de máquina desenvolvidos por diferentes tribunais brasileiros, promovendo a automação e aprimoramento da recuperação de informações no contexto judicial.

Apesar dos avanços científicos em mineração de dados e IA, o processamento de texto e documentos no sistema legal apresenta desafios peculiares. Os dados são frequentemente não estruturados, não rotulados e diversificados, dificultando sua extração e análise manual. Além disso, a grande quantidade de informações inviabiliza a análise manual, tornando o estudo de algoritmos não supervisionados essencial para aprender padrões e obter representações densas dos dados, tanto para humanos quanto para máquinas.

O agrupamento de dados (*data clustering*) permite descobrir grupos dentro dos dados com base em similaridades, mesmo sem conhecimento prévio da estrutura. No âmbito jurídico, a capacidade de agrupar textos semelhantes é particularmente importante, pois pode trazer eficiência para tarefas que envolvem casos similares, promovendo equidade e justiça social. Além disso, os rótulos dos clusters podem ser utilizados como informações para outros modelos de aprendizado de máquina, potencializando a análise dos dados jurídicos.

Diversas técnicas de aprendizado de máquina, como BERT, ULM FIT e Word2Vec [1], têm se mostrado eficientes para capturar a sintaxe e semântica de textos complexos. No entanto, muitas vezes esses modelos complexos são difíceis de serem treinados e compreendidos. No contexto jurídico, é essencial que os algoritmos possam ser constantemente retreinados e auditados, garantindo que suas decisões estejam baseadas em dados atualizados e fornecendo transparência para justificar as decisões tomadas.

Neste estudo, nossa principal contribuição é fornecer insights sobre abordagens eficazes para a tarefa de agrupamento de textos jurídicos, considerando diferentes modelos de *embedding*, algoritmos de agrupamento e técnicas de pré-processamento. O processamento foi realizado utilizando bibliotecas da linguagem Python, incluindo Scikit-learn, NLTK, Numpy, entre outras. Para avaliar o desempenho dos algoritmos de agrupamento, foram utilizados os scores *Normalized Mutual Info* [18] e *Jaccard* [19], que foram aplicados ao resultado final dos agrupamentos. Essas métricas são amplamente reconhecidas e amplamente utilizadas para medir a qualidade dos agrupamentos obtidos. Essas informações podem beneficiar tanto pesquisadores quanto profissionais que trabalham com análise de dados jurídicos, fornecendo diretrizes para a seleção dos métodos mais adequados às suas necessidades. Além disso, conduzimos nossos experimentos em um conjunto de dados de

Esta pesquisa é parcialmente financiada pela CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

movimentos de processos, disponibilizado pelo Tribunal de Justiça do Rio Grande do Norte, composto por 30.000 documentos (movimentos) de processos judiciais de dez categorias, o que nos permite avaliar e verificar a coerência dos rótulos criados pelo algoritmo com os originais fornecidos por humanos.

II. Trabalhos Relacionados

Em Aguilar *et al*. [2] foram avaliadas quatro técnicas de extração de características: *Best Matching* 25 (BM25), Análise Semântica Latente (LSA), Doc2Vec e Alocação Latente de Dirichlet (LDA) para representação semântica de recursos educacionais digitais. Os autores avaliaram os resultados com métricas não supervisionadas e demonstraram que o desempenho dessas técnicas varia de acordo com o conteúdo dos textos.

Kawintiranon & Liu [3] também compararam quatro técnicas diferentes de *embedding* (GloVe Embeddings, Siamese-BERT, BERT e TF-IDF) para medir a semelhança entre diferentes legislações de proteção de dados de diferentes países, incluindo a legislação brasileira. Os resultados mostraram que, nesse caso, o BERT supera todos os outros modelos. Em contraste, nosso trabalho utiliza vários embeddings treinados especificamente em corpora jurídicos, ao invés de embeddings de linguagem geral. Isso permite representações textuais mais adequadas ao domínio jurídico.

Também foram realizados esforços para melhorar o desempenho final do agrupamento de documentos legais, como em [4], onde é demonstrada uma técnica de agrupamento suave em larga escala com segmentação de tópicos, conseguindo criar um algoritmo eficiente para recuperação.

As pesquisas e avanços mencionados na literatura nos motivam a desenvolver nosso trabalho, onde uma das principais contribuições será a investigação de modelos de *embedding* criados a partir de arquiteturas complexas baseadas em *transformers*. Essas arquiteturas têm demonstrado resultados promissores em várias tarefas de processamento de linguagem natural, e acreditamos que explorar seu potencial para a representação de textos jurídicos pode trazer informações valiosas para o agrupamento eficaz desses documentos.

III. Algoritmo utilizados

Esta seção é uma breve discussão dos algoritmos testados. Para os seguintes tópicos, considere $D = \{d_0, d_1, \dots, d_n\}$ o conjunto de documentos em um corpus, $V = \{v_0, v_1, \dots, v_n\}$ o vocabulário deste corpus, e $P = \{p_0, p_1, \dots, p_n\}$, onde $p_i \in \mathbb{R}^k$ são as representações vetoriais respectivas dos documentos D e k são as dimensões.

A. K-means++

O algoritmo k-means++ [5] é uma variante popular do k-means tradicional que busca melhorar a inicialização dos centroides dos clusters. Ao invés de inicializar os centroides aleatoriamente, o k-means++ seleciona centroides distintos e bem distribuídos nos dados.

Especificamente, o primeiro centroide é escolhido aleatoriamente dentre os pontos de dados. Cada ponto subsequente é escolhido probabilisticamente, com probabilidade proporcional à distância mínima desse ponto a algum centroide existente. Pontos mais distantes têm maior chance de serem escolhidos. Após selecionar k centroides iniciais dessa forma, o algoritmo segue normalmente o procedimento do k-means, atribuindo cada ponto ao cluster do centroide mais próximo e recalculando os centroides.

B. MB K-means

O MB Kmeans [6] é uma variação do k-means que usa uma estrutura de árvore binária para dividir recursivamente os clusters em subclusters menores. Especificamente, esse algoritmo constrói uma árvore binária para particionar recursivamente os dados em regiões menores. Essas partições são usadas para extrair mini-batches aleatoriamente e atualizar os centroides dos clusters. Essa abordagem ajuda a reduzir o tempo de execução do algoritmo.

C. K-means++ com o SOM

Esta abordagem combina o algoritmo k-means++ com SOM (Self-Organizing Map) [7]. O SOM é uma técnica de aprendizado não supervisionado que usa uma grade de neurônios para organizar os dados em um mapa. O k-means++ é então aplicado aos neurônios do SOM para realizar o agrupamento. Essa estratégia permite obter uma inicialização mais eficiente para o k-means++, acelerando a convergência e melhorando a qualidade da solução final. O SOM fornece uma representação que preserva topologia, enquanto o k-means++ refina os agrupamentos.

D. DBSCAN

O algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) é um método de agrupamento baseado em densidade [8]. Ele agrupa os pontos que estão próximos uns dos outros e separa os pontos que estão isolados em regiões de baixa densidade. Diferentemente de métodos tradicionais como k-means que requerem informar o número de clusters a priori, o DBSCAN é capaz de identificar clusters de forma automática a partir da densidade de pontos no espaço. O algoritmo define dois parâmetros principais: epsilon (ε) que determina o raio de vizinhança, e *MinPts* que é o número mínimo de pontos requerido para formar uma região densa.

E. Agglomerative Clustering

O Agglomerative Clustering [9] é um método hierárquico de agrupamento que começa com cada ponto como um cluster individual. O algoritmo começa tratando cada ponto de dado como um cluster individual. Em seguida, em cada iteração, os dois clusters mais similares são mesclados, formando partições maiores de forma bottom-up. Para decidir quais clusters mesclar, é utilizada alguma medida de distância ou similaridade entre clusters, como distância mínima, máxima ou média entre pontos. Esse processo aglomerativo continua até que resta apenas um cluster ou que algum critério de parada seja atingido, como número desejado de clusters ou limite na distância entre merges.

F. Gaussian Mixture Model

O Gaussian Mixture Model (GMM) [10] é um modelo estatístico probabilístico que assume que os dados são gerados a partir de uma mistura (mixture) de um número finito de distribuições gaussianas. Cada componente gaussiana representa um cluster ou subpopulação nos dados. O GMM tem dois conjuntos de parâmetros: os pesos de cada gaussiano na mistura, indicando suas probabilidades a priori, e os

parâmetros (média e covariância) de cada distribuição gaussiana individual. O algoritmo GMM estima esses parâmetros através da maximização da verossimilhança dos dados observados. Existem diferentes algoritmos para realizar essa estimação, como o popular Expectation-Maximization (EM).

G. Birch

O algoritmo Birch (Balanced Iterative Reducing and Clustering using Hierarchies) [11] constrói uma Árvore-B balanceada (Balanced Tree) para organizar os dados em subclusters de forma incremental e eficiente. Ao invés de usar distância euclidiana, ele utiliza medidas de clustering baseadas em densidade, como raio e diâmetro do cluster. Especificamente, cada nó da árvore representa um subcluster que armazena informações resumidas sobre os pontos, como o número de pontos, centroide e raio do agrupamento. Dados novos são inseridos descendo a árvore. O Birch pode incrementalmente construir uma estrutura de clustering para dados muito grandes que não caberiam na memória. Ele também é mais robusto a outliers do que métodos tradicionais.

H. Optics

O OPTICS (*Ordering Points To Identify the Clustering Structure*) [12] é um método de agrupamento baseado em densidade que cria uma ordenação dos pontos com base em sua densidade alcançável. O objetivo do OPTICS é superar algumas limitações dos algoritmos de clusterização baseados em densidade, como o DBSCAN, em identificar clusters de tamanhos e densidades diferentes.

O OPTICS funciona criando uma ordenação dos pontos de dados com base na densidade alcançável de cada ponto. A densidade alcançável de um ponto p é definida como o menor raio *eps* necessário para que o eps-vizinhança de *p* contenha pelo menos um número mínimo de pontos *MinPts*. Pontos com maior densidade alcançável são ordenados primeiro. Essa ordenação é então analisada para identificar regiões de alta densidade separadas por regiões de baixa densidade, correspondendo aos clusters. Os vales na plotagem da ordenação indicam fronteiras entre clusters.

I. HDBSCAN

O HDBSCAN (*Hierarchical Density-Based Spatial Clustering of Applications with Noise*) [13] é uma extensão do DBSCAN que constrói uma hierarquia de clusters. Ele determina automaticamente o número de clusters e permite a detecção de clusters de diferentes densidades. A vantagem do HDBSCAN sobre o DBSCAN é que ele não requer a especificação prévia dos parâmetros *eps* e *MinPts*. Em vez disso, o HDBSCAN seleciona automaticamente esses parâmetros com base na estabilidade dos clusters formados. Isso permite a detecção de clusters de diferentes densidades nos mesmos dados.

J. SOM

O Self-Organizing Map (ou mapa auto-organizável) é um algoritmo de aprendizado não supervisionado que mapeia os dados de entrada em uma grade bidimensional, na qual cada neurônio representa uma posição nesse mapa [14].

O treinamento do SOM é realizado usando a biblioteca "*somoclu*" [1]. O conjunto de dados de entrada, representado pela variável "x", pode ser um vetor de característica ou qualquer outro conjunto de dados, no nosso caso os *embeddings*. A dimensão do SOM é especificada através do parâmetro "dim_som", que define o tamanho da grade bidimensional em que os neurônios serão organizados. Durante o treinamento, o modelo SOM é criado utilizando a classe "somoclu" e é treinado usando o método "*train*". O número de épocas de treinamento é definido como 20 neste exemplo, bem como suas dimensões são 20x20

IV. Modelos de Embedding

Os três modelos apresentados a seguir são baseados na arquitetura do BERT (*Bidirectional Encoder Representations from Transformers*) [15], que é um modelo de aprendizado profundo amplamente utilizado para o processamento de linguagem natural. Os modelos são pré-treinados específicos para tarefas relacionadas ao processamento de textos jurídicos em língua portuguesa. No entanto, também existem diferenças entre os modelos. Essas diferenças podem estar relacionadas ao tamanho do modelo, ao tipo de pré-processamento dos dados realizados durante o treinamento, às técnicas adicionais utilizadas (como Meta-aprendizado e *Knowledge Distillation*) e aos conjuntos de dados específicos utilizados no treinamento. Os tamanhos dos modelos podem variar dependendo das versões específicas e das arquiteturas. No entanto, geralmente os modelos "large" tendem a ser maiores e mais complexos em comparação com os modelos "base" ou "small".

A. Jurisbert

Este modelo[2], tem cerca de 400 Mb e o tamanho de 768. O modelo em particular foi treinado em um grande corpus de textos jurídicos em português, com o objetivo de capturar o conhecimento específico e os padrões linguísticos encontrados nesse domínio. Ele utiliza uma abordagem de pré-treinamento por meio do aprendizado de representações contextuais, onde o modelo aprende a predizer palavras ou trechos de texto com base no contexto em que eles aparecem. Com o "jurisbert", é possível aproveitar seu conhecimento prévio sobre termos jurídicos, estrutura de sentenças e nuances da linguagem jurídica. Isso pode ser útil para tarefas como classificação de documentos, extração de informações jurídicas, entre outros.

B. Bert Law

Esse modelo [3] tem cerca de 1.4 Gb com 1024 de dimensão, é uma versão "large" do BERT e é "cased", o que significa que ele mantém a distinção entre letras maiúsculas e minúsculas durante o treinamento e inferência. Foi treinado em um grande corpus de textos jurídicos em português, com o objetivo de capturar os padrões e as características específicas da linguagem jurídica.

C. Irisbert

O STJ Irisbert[4] com tamanho de 1.2 Gb com uma dimensão de 1024 foi treinado especificamente para tarefas relacionadas ao processamento de textos jurídicos. Uma característica desse modelo é o uso de técnicas de Meta-aprendizado e *Knowledge Distillation* (MetaKD) durante o treinamento. Isso significa que ele foi treinado para não apenas aprender a linguagem jurídica, mas também incorporar conhecimento de modelos anteriores e otimizar o processo de aprendizado.

D. TF-IDF

O TF-IDF (Term Frequency-Inverse Document Frequency) [16] é uma técnica amplamente utilizada no processamento de linguagem natural para calcular a

1 - P. Wittek, S. C. Gao, I. S. Lim, L. Zhao. “Somoclu: An Efficient Parallel Library for Self-Organizing Maps”. *Journal of Statistical Software,* 78(9), pp.1–21. DOI:10.18637/jss.v078.i09. arXiv:1305.1422.

https://somoclu.readthedocs.io/en/stable/

importância relativa de uma palavra em um documento dentro de um conjunto de documentos. Ele é baseado na ideia de que as palavras que ocorrem com frequência em um documento e raramente em outros documentos têm um maior peso em termos de relevância para aquele documento específico. Usamos TfidfVectorizer da biblioteca sklearn para gerar uma matriz de 1582 features, para isso foi definido os seguintes parâmetros: max_df = 0.90, que significa que um termo deve ocorrer em no máximo 90% dos documentos para ser considerado relevante, e min_df = 0.021, que significa que um termo deve ocorrer em pelo menos 2,1% dos documentos para ser considerado relevante.

V. Base de dados

Foi utilizado neste estudo uma base de dados com 30.000 documentos legais provenientes do Tribunal de Justiça do Grande do Norte (TJRN30) igualmente em 10 classes [17]. Esses documentos são coletados e armazenados pelo tribunal como parte de seu acervo jurídico. O Tribunal de Justiça do Rio Grande do Norte (TJRN) é uma instituição do sistema judiciário brasileiro responsável por administrar a justiça estadual no estado do Rio Grande do Norte.

Portanto, os documentos legais presentes nessa base de dados representam movimentos processuais, podendo incluir sentenças, decisões judiciais, acórdãos, despachos, entre outros. Esses documentos abrangem movimentos processuais diversos, como descritos no quadro 1.

Quadro 1 – Códigos e movimentos processuais

Código	Tipo do movimento
196	Extinção da execução ou do cumprimento da sentença
198	Acolhimento de Embargos de Declaração
200	Não Acolhimento de Embargos de Declaração
219	Procedência
220	Improcedência
339	Liminar
458	Abandono de Causa
461	Ausência das condições da ação
463	Desistência
785	Antecipação de tutela

Os códigos provêm das tabelas processuais unificadas do CNJ[5]. O código 196 indica a sentença que põe fim ao processo de execução, enquanto o código 198 está associado ao caso em que todos os pedidos de declaração são conhecidos e é efetivada a declaração requerida. Já o código 200 se refere a casos em que os embargos de declaração são resolvidos não proferindo qualquer das declarações requeridas pelo embargante. O código 219 registra a solução do processo no juízo originário. Tratando-se de juízo recursal, registra em conhecido o recurso de parte e provido. Por outro lado, o código 220 (improcedência) é o caso contrário ao 219, quando, tratando-se de juízo recursal, registra em conhecido o recurso de parte e porém o mesmo não é provido. O código 339 indica quando a decisão que concede liminarmente a ordem cautelar. Já o código 458 está associado a abandono da causa, quando, por não promover os atos e diligências que lhe competir, o autor abandonar a causa por mais de 30 (trinta) dias. Por fim, o código 461 está associado a ausência das condições da ação, quando não concorrer qualquer das condições da ação.

Importante salientar que a base de dados foi obtida por amostragem simples, com extração do PJe (Processo Judicial Eletrônico – É um sistema desenvolvido no Brasil para a tramitação eletrônica de processos judiciais). A base foi utilizada para classificação (predição) do movimento a ser realizado. O uso em tarefas de agrupamento de dados traz várias dificuldades. Além da diversidade dos textos, mesmo dentro das categorias, há também, grande sobreposição em certas classes, como ocorre nos códigos 198 e 200. Assim, apesar de, a priori, imaginarmos 10 classes, associados a 10 categorias ou códigos de processos, há classes (ou movimentos) descritos, em vários casos, compartilham palavras e termos bastante semelhantes. Certos processamentos comuns em NLP, como exclusão de stopwords (palavras ou termos bastante comuns, como preposições, etc.) podem excluir determinados termos e aproximar, no espaço multidimensional, os vetores de representação das categorias. Isso ocorre, também, por ex., quando realizamos *steeming* (reduzir a palavra ao radical). Termos como 'procedente' e 'improcedente' geram um mesmo radical.

VI. Metodologia

Esta pesquisa segue uma abordagem baseada em NLP (*Natural Language Processing*), ou Processamento de Linguagem Natural, devido à natureza dos documentos jurídicos brasileiros. Esses documentos geralmente são extensos, complexos e apresentam características específicas da linguagem jurídica, como terminologia especializada, estruturas gramaticais complexas e uso de jargões legais.

Ao aplicar técnicas de NLP em agrupamento de textos jurídicos, busca-se identificar grupos de documentos que possuam similaridades em termos de tópicos, temas legais, precedentes, argumentos, entre outros aspectos relevantes. Essa abordagem permite uma organização mais eficiente e uma análise mais abrangente dos documentos, facilitando a busca por informações específicas, a identificação de padrões e a descoberta de insights relevantes no âmbito jurídico. Além disso, a avaliação dos *embeddings* jurídicos é de grande importância para garantir a qualidade e a eficácia dos resultados obtidos na tarefa de agrupamento. Assim, a utilização do NLP como parte integrante da metodologia desta pesquisa é fundamentada na necessidade de lidar com a complexidade e o volume de dados presentes nos documentos jurídicos brasileiros, bem como na busca por uma análise mais precisa, eficiente e abrangente dos mesmos.

A Figura 1 apresenta um fluxograma abrangente que ilustra as etapas-chave desse processo, desde a leitura dos dados da base até a criação de três tipos distintos de embeddings textuais. O fluxo começa com a leitura dos dados da base de dados, seguida por uma série de etapas de pré-processamento. Após o pré-processamento, destacamos a criação de três tipos distintos de embeddings, o que enriquece a representação textual para análises, a clusterização e finalmente uma avaliação. A Figura 1 encapsula as contribuições significativas deste estudo, apresentando de maneira visual como cada etapa contribui para a melhoria da qualidade e utilidade dos dados textuais para análise jurídica avançada. O objetivo principal é otimizar a representação

2-https://huggingface.co/alfaneo/jurisbert-base-portuguese-uncased/tree/main
3-https://huggingface.co/edwatanabe/bert-large-cased-pt-law
4-https://huggingface.co/stjiris/bert-large-portuguese-cased-legal-tsdae-gpl-nli-sts-MetaKD-v0
5-https://www.cnj.jus.br/programas-e-acoes/tabela-processuais-unificadas/

semântica dos textos jurídicos, permitindo análises posteriores mais eficazes.

A. *Pre-processamento*

Foram submetidos a um processo de pré-processamento para garantir a consistência e qualidade dos documentos antes da aplicação dos métodos de agrupamento. As seguintes etapas de pré-processamento foram realizadas, por funções desenvolvidas na linguagem Python:

- Substituição de números processuais: Foi aplicada a função “sub_numpro” para substituir os números processuais presentes nos textos por uma representação padronizada.
- Substituição de valores monetários: A função sub_dinheiro foi utilizada para substituir os valores monetários pelos termos "valor_monetario".
- Substituição de números: A função sub_num foi aplicada para substituir números por uma representação padronizada.
- Remoção de datas: Foi utilizada a função remover_datas para remover datas dos textos.
- Remoção de números por extenso: A função remove_numeros_por_extenso foi aplicada para remover números escritos por extenso dos textos.
- Remoção de palavras após datas por extenso: Foi utilizada a função remover_apos_data_por_extenso para remover palavras que aparecem após datas escritas por extenso nos textos.
- Remoção de acentos: A função remover_acentos foi aplicada para remover acentos das palavras nos textos.
- Remoção de *stopwords*: Foi utilizada a função remove_stopwords para remover palavras comuns que não agregam significado aos textos. Essas palavras foram consideradas irrelevantes para a análise e, portanto, removidas dos textos. Utilizamos o conjunto de stop words disponibilizado pela biblioteca NLTK, que contém uma lista de palavras comuns em português que normalmente não possuem relevância semântica. Além disso, realizamos um procedimento manual para identificar e adicionar à lista de stop words termos específicos encontrados em nosso corpus de textos jurídicos. Por exemplo, palavras como "ii", "iii", "iv", "cpc", "1a", "acao", "ad", "advocaticios" e "inciso" foram consideradas como stop words adicionais. Dessa forma, a combinação dessas duas fontes de stop words nos permite filtrar termos que não contribuem significativamente para a análise e o processamento dos textos.
- Remoção de palavras sozinhas: A função remover_palavras_sozinhas foi aplicada para remover palavras que não estão conectadas a outras palavras nos textos. Isso as vezes acontece quando se tem a remoção de alguns stopswords.
- Remoção de rodapé com data: Foi utilizada a função remover_rodape_data para remover o rodapé contendo a data de publicação dos textos.
- Palavras compostas: Além dessas etapas de pré-processamento, também foram consideradas técnicas específicas para lidar com palavras compostas, como: periculum_in_mora pode ser a mesma coisa que 'periculum', 'periculum in', 'periculum in mora'. Essas técnicas foram aplicadas para tratar palavras compostas e garantir que diferentes formas de as expressar.

B. *Geração de Embedding*

Foram aplicadas as técnicas de transformação de dados utilizando o Jurisbert, Bert law, Irisbert e IF-IDF, para gerar os *embeddings* correspondentes aos textos jurídicos.

C. *Aplicação dos métodos de agrupamentos*

Os métodos de agrupamento mencionados anteriormente foram aplicados aos *embeddings* gerados.

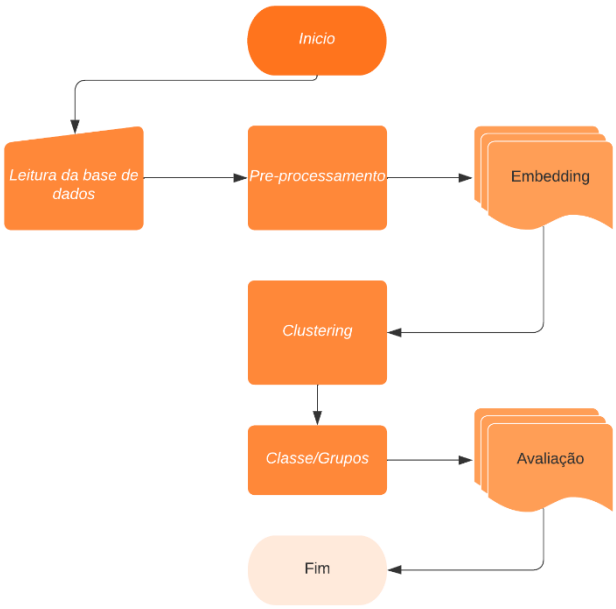

Figura 1 – Fluxo do processo de avaliação dos embedding

Cada algoritmo foi aplicado de forma individualizada a cada tipo de embedding. Os parâmetros desses algoritmos foram ajustados por meio de iterações experimentais e conhecimento prévio da área. No caso do algoritmo DBSCAN, adotou-se um valor de eps igual a 0.6, juntamente com um min_samples igual a 10. Quanto ao algoritmo OPTICS, as escolhas recaíram sobre min_samples com valor de 50, xi com valor de 0.05, e max_eps estabelecido em 0.2.

Para o algoritmo HDBSCAN, os parâmetros configurados foram min_cluster_size igual a 125, min_samples igual a 126, e a métrica utilizada foi a euclidiana. Por outro lado, nos demais algoritmos, optou-se por empregar as configurações padrão de seus respectivos métodos. É importante ressaltar que essas configurações passaram por extensivos testes e validações na literatura especializada.

D. *Avaliação dos resultados*

As métricas NMI (*Normalized Mutual Information*) [18] e Jaccard [19] são amplamente utilizadas para avaliar a qualidade dos agrupamentos.

O NMI mede a similaridade entre os agrupamentos gerados pelos algoritmos e as classes reais dos dados. Ele leva em consideração tanto a informação mútua quanto a entropia das classes e dos agrupamentos. Quanto maior o valor do NMI, mais similar é o agrupamento em relação às classes reais, indicando uma maior qualidade na formação dos grupos. O NMI varia entre 0 e 1.

O coeficiente Jaccard, por sua vez, mede a sobreposição entre os *clusters* encontrados pelo algoritmo e os agrupamentos. Ele é calculado como a proporção entre o tamanho da interseção dos clusters dividido pelo tamanho da união dos clusters. O coeficiente Jaccard varia entre 0 e 1, sendo que 1 indica uma sobreposição perfeita entre os clusters encontrados e os clusters de referência.

Essas métricas são úteis para avaliar a qualidade dos agrupamentos em relação às classes reais quando temos informações sobre as classes verdadeiras dos dados. Elas fornecem uma medida para avaliar o desempenho dos algoritmos e comparar diferentes abordagens. Ao calcular e analisar essas métricas, é possível verificar quão bem os agrupamentos obtidos representam as estruturas subjacentes nos dados e auxiliar na escolha dos melhores parâmetros para cada cenário.

E. Paralização dos Embeddings

Para melhorar o desempenho no processamento de dados, foram aplicadas técnicas de processamento paralelo devido às limitações de hardware. Foi utilizado o Google Colab com 12,7GB de RAM. A movimentação de dados entre CPU e GPU permitiu aproveitar a capacidade de processamento acelerado da GPU. Inicialmente, os textos foram processados na CPU em lotes. Esses lotes foram convertidos em tensores e transferidos para a GPU. O modelo também foi movido para a GPU. Em seguida, a inferência do modelo foi executada na GPU. Os embeddings resultantes foram movidos de volta para a CPU. Essa abordagem distribuiu eficientemente os dados entre CPU e GPU, aproveitando o processamento paralelo da GPU, enquanto a CPU gerenciou a iteração. Foram utilizadas bibliotecas Python para implementar as redes neurais.

F. Tempo de execução

Além das métricas de avaliação, também registramos o tempo de execução (em segundos) de cada modelo para registrar a eficiência computacional. Além disso, o algoritmo foi executado 20 vezes.

VII. Resultados

A Figura 2 apresenta uma projeção em duas dimensões, utilizando Análise de Componentes Principais (PCA), da representação TF-IDF com 300 termos das classes originais dos artigos. Já a Figura 2 exibe a projeção PCA em duas dimensões após a aplicação do algoritmo K-means com k = 10 para agrupamento. Analisando as duas figuras, é possível observar uma grande sobreposição entre as classes mesmo após a redução de dimensionalidade. Isso indica que há uma separação difusa dos dados originais, o que dificulta o processo de identificação de clusters claramente definidos e distintos. A sobreposição perceptível entre as projeções das diferentes classes evidencia a complexidade e desafio da tarefa de agrupamento semântico para os dados jurídicos utilizados. Classes semanticamente distintas acabam se misturando quando projetadas em espaços bidimensionais.

Portanto, as figuras ilustram visualmente a dificuldade de separabilidade dos dados devido à proximidade semântica entre algumas classes jurídicas analisadas. Isso corrobora a necessidade de investigação de algoritmos e representações mais sofisticadas, capazes de capturar nuances do domínio legal para aprimorar a qualidade dos clusters formados.

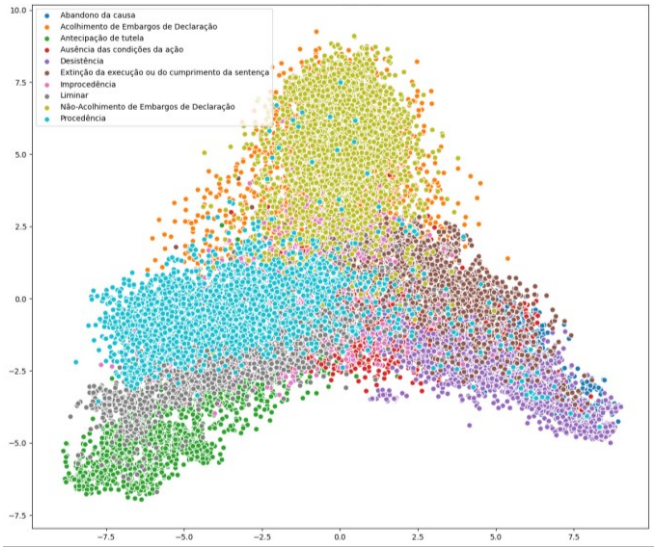

Figura 2 – Projeção (usando PCA) dos dados originais

As Tabelas 1 a 3 resumem resultados para os valores de NMI, Jaccard e tempo de processamento (em segundos). O modelo k-means++ mostrou um desempenho moderado, com valores NMI e Jaccard em torno de 0.4 a 0.5 para os três tipos de embedding (veja a Figura 4 e 5): Jurisbert, Bert law e Irisbert. O tempo de execução variou de cerca de 40 a 56 segundos (Figura 6). O modelo MB KMeans teve um desempenho semelhante ao k-means++, com valores NMI e Jaccard um pouco mais baixos, mas com tempos de execução muito menores, na faixa de 2 a 3 segundos.

Com a combinação do k-means++ e o SOM resultou em desempenho inferior, com valores NMI em torno de 0.2 a 0.4 para os diferentes embeddings. Entretanto, os tempos de treinamento do SOM foram bastante altos, variando de 87 a 185 segundos, enquanto os tempos de algoritmo de agrupamento foram menores, em torno de 2 a 5 segundos (Tabela 3).

O DBSCAN mostrou uma precisão mais baixa, com valores NMI em torno de 0.15 a 0.19 para os três tipos de embedding, mas alcançou um alto índice de Jaccard de 0.9960. Os tempos de execução foram relativamente rápidos, variando de 13 a 64 segundos.

O Agglomerative Clustering mostrou um desempenho moderado, com valores NMI em torno de 0.4 a 0.5 (veja a Figura 1) e valores Jaccard em torno de 0.7 para os três embeddings (Figura 4). No entanto, o tempo de execução foi consideravelmente alto, variando de 283 a 896 segundos.

O modelo Gaussian Mixture mostrou resultados semelhantes ao Agglomerative Clustering, com valores NMI em torno de 0.4 a 0.5 e valores Jaccard em torno de 0.68. Os tempos de execução foram um pouco menores, variando de 153 a 233 segundos.

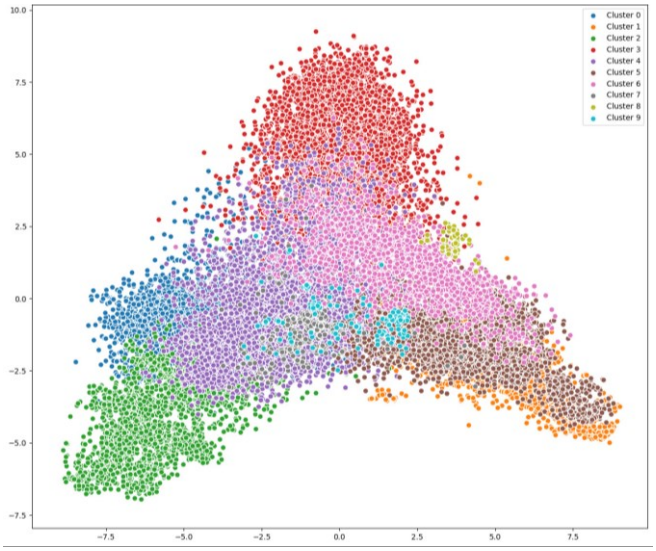

Figura 3 – Projeção (usando PCA) dos dados após agrupamento (kmeans)

Tabela 1 - Sumarização dos resultados com a métrica NMI.

	Embedding			
Algoritmo	Jurisbert	Bert law	TF-IDF	Irisbert
k-means++	0.4299	0.3822	**0.5734**	0.5098
MB KMeans	0.4239	0.3789	0.4884	**0.5816**
k-means++ with SOM	0.3340	0.2139	**0.4030**	0.3425
DBSCAN	0.1743	0.1861	0.1547	**0.3944**
Agglomerative Clustering	0.4234	0.3853	**0.5222**	0.5206
Gaussian Mixture	0.4462	0.3766	0.5159	**0.5276**
Birch	0.4205	0.4000	**0.5230**	0.4907
OPTICS	0.0839	0.0865	0.0802	**0.1324**
HDBSCAN	0.2167	0.1488	0.2169	**0.3059**

Tabela 2 - Sumarização dos resultados com a métrica Jaccard

	Embedding			
Algoritmo	Jurisbert	Bert law	TF-IDF	Irisbert
k-means++	0.8622	0.7677	0.7534	**0.8872**
MB KMeans	0.8581	0.9239	**0.8406**	0.7287
k-means++ with SOM	0.7708	0.8563	0.8096	**0.8815**
DBSCAN	**0.9960**	**0.9960**	**0.9960**	0.9899
Agglomerative Clustering	0.7049	**0.8510**	0.7284	0.5326
Gaussian Mixture	0.6805	**0.7509**	0.7180	0.7151
Birch	0.7432	0.5746	**0.8500**	0.8024
OPTICS	0.9960	**0.9966**	**0.9966**	0.9960
HDBSCAN	0.9663	0.9663	0.9663	**0.9765**

O modelo Birch teve um desempenho moderado, com valores NMI em torno de 0.4 para Jurisbert e Irisbert, mas um valor mais baixo de 0.4 para Bert law. Os valores Jaccard também foram moderados, variando de 0.57 a 0.85. Os tempos de execução foram semelhantes aos do Gaussian Mixture, variando de 184 a 235 segundos.

O modelo OPTICS teve um desempenho inferior, com valores NMI em torno de 0.08 a 0.09. No entanto, o índice de Jaccard foi alto em 0.9960. Os tempos de execução foram moderados, variando de 59 a 189 segundos.

O HDBSCAN mostrou um desempenho moderado, com valores NMI em torno de 0.15 a 0.22 e valores Jaccard em 0.96 para os três *embeddings*. No entanto, o tempo de execução foi consideravelmente alto, variando de 184 a 1286s.

Tabela 3 – Tempo de processamento (em segundos)

	Embedding			
Algoritmo	Jurisbert	Bert law	TF-IDF	Irisbert
k-means++	**39.04**	56.55	67.81	52,31
MB KMeans	2.81	**2.35**	2.43	5.57
k-means++ with SOM	168.51	191,15	**89.36**	294.5
DBSCAN	25.91	63.58	**12.95**	47.42
Agglomerative Clustering	896.33	292.59	**282.57**	778.88
Gaussian Mixture	232.84	181.71	**152.45**	711.57
Birch	234.67	183.57	228.21	**165.14**
OPTICS	58.57	189.49	**63.45**	195.8
HDBSCAN	1285.67	**183.57**	271.42	2500.54

O TF-IDF alcançou um valor de NMI de 0.5734 e um coeficiente de Jaccard de 0.7534 quando combinado com o algoritmo k-means++. Isso indica que o k-means++ em conjunto com o TF-IDF obteve um desempenho moderado em relação à similaridade dos clusters em relação aos rótulos de classe e à sobreposição dos elementos nos clusters. Já o MB KMeans alcançou um valor de NMI de 0.5816 e um coeficiente de Jaccard de 0.7287. Isso indica que o MB KMeans em conjunto com o TF-IDF apresentou um desempenho um pouco melhor em relação à similaridade dos clusters e à sobreposição dos elementos nos clusters em comparação com o k-means++. Os algoritmos k-means++ com SOM, Agglomerative Clustering, Gaussian Mixture, Birch, OPTICS e HDBSCAN, quando combinados com o TF-IDF, apresentaram valores de NMI e coeficientes de Jaccard variados. Não é possível afirmar que o TF-IDF é consistentemente superior nessas combinações, pois alguns algoritmos tiveram desempenho relativamente baixo em comparação com outros. Em relação ao tempo de execução, os algoritmos Agglomerative Clustering, Gaussian Mixture e HDBSCAN apresentaram tempos mais longos, enquanto os algoritmos MB KMeans, DBSCAN e Birch tiveram os menores tempos de execução.

As Figuras 4 a 6 resumem resultados para os valores de NMI, Jaccard e tempo de processamento.

Com relação ao OPTICS, ele teve um desempenho inferior no NMI, indicando que os rótulos atribuídos aos clusters não correspondem bem aos rótulos reais. No entanto, ele apresentou um bom desempenho no coeficiente de Jaccard, indicando que ele agrupou corretamente as amostras. Essas diferenças podem ocorrer devido às características específicas dos dados e às propriedades dos algoritmos de agrupamento. É importante considerar as limitações e as peculiaridades de cada métrica ao interpretar os resultados do agrupamento.

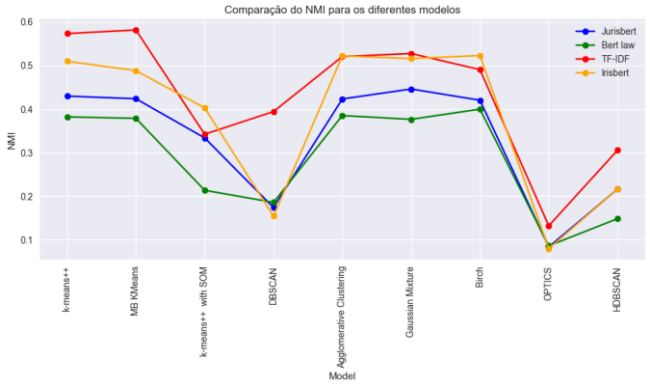

Figura 4 - Comparação dos métodos de embedding levando em consideração os valores do NMI.

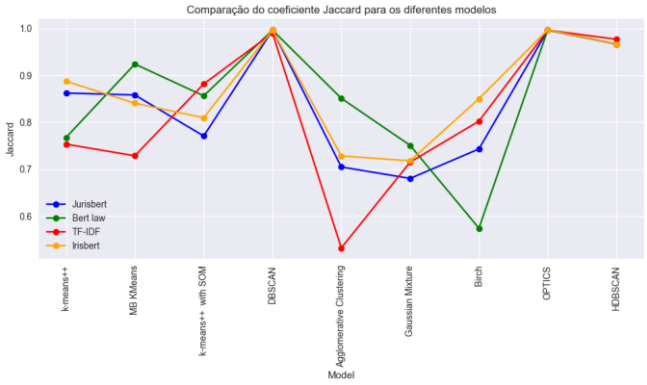

Figura 5 - Comparação dos resultados pelo índice Jaccard dos métodos de agrupamento com diferentes embeddings

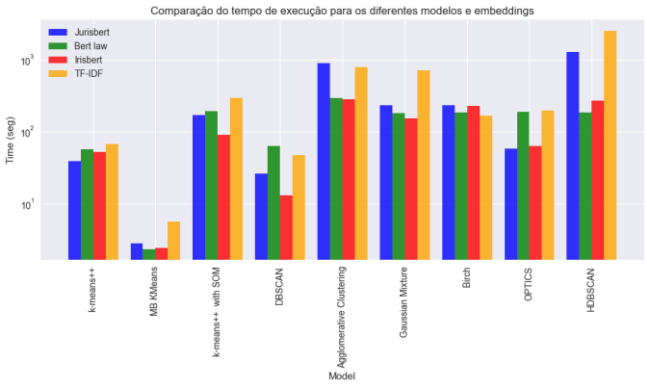

Figura 6 - Tempo de execução (em segundos) de cada modelo.

VIII. Considerações Finais e Trabalhos Futuros

A tarefa de agrupamento de textos desempenha um papel fundamental na organização e compreensão de grandes quantidades de documentos. É uma técnica fundamental na tarefa de agrupar textos semanticamente semelhantes em categorias ou agrupamentos coerentes, permitindo a extração de informações importantes e a descoberta de padrões latentes, objetivando identificar agrupamentos de tópicos coesos.

O artigo apresentou uma análise comparativa do desempenho de diferentes modelos de agrupamento aplicados a conjuntos de dados de textos jurídicos. Ao considerar diversos aspectos, como técnicas de pré-processamento, geração de *embeddings* e algoritmos de agrupamento, foi possível obter insights para a tarefa de agrupamento de textos jurídicos.

Os *embeddings* foram obtidos a partir dos modelos BERT (*Bidirectional Encoder Representations from transformers*), como Jurisbert, Bert Law e Irisbert. Também utilizamos como base para comparações a representação obtida por TFIFD. Nove diferentes algoritmos de agrupamento foram testados, entre eles, por exemplo, métodos como MB Kmeans, DBSCAN, BIRCH. Foram realizados experimentos em um banco de dados de 30.000 documentos obtidos de movimentações judiciais do Tribunal de Justiça do Rio Grande do Norte (TJRN), separados em 10 categorias.

Índices como NMI e Jaccard foram utilizados para avaliar o desempenho dos algoritmos de *clustering*, assim como o tempo de processamento foi descrito para os diferentes algoritmos. Os resultados sugerem melhores resultados com a incorporação de "Irisbert" e TF-IDF quando considerados NMI e "Bert Law" e TF-IDF quando considerados o coeficiente de Jaccard. Curiosamente, o IrisBERT também gerou bom resultado quando avaliado com Jaccard. Isso demonstra sua capacidade de capturar semântica jurídica tanto para a métrica NMI quanto Jaccard.

Várias etapas do pré-processamento foram discutidas, assim como alguns detalhes do uso de multiprocessamento para agilizar os algoritmos, dadas as limitações de hardware disponíveis. O processo de movimentação dos dados entre a CPU e a GPU foi empregado para aproveitar a capacidade de processamento acelerado da GPU (T4 do Colab).

Outro ponto a ser melhor discutido em outro artigo é na questão qualitativa dos agrupamentos obtidos pelos *embeddings* obtidos por modelos baseados em BERT e em representações convencionais. Uma análise de coocorrência das 20 palavras mais importantes de cada agrupamento mostrou maior correlação com as classes originais do que quando utilizamos representações tradicionais.

Como descrito na seção V, a base de dados foi extraída do PJe com foco no uso em classificação (predição) do movimento a ser realizado. O uso em tarefas de agrupamento de dados traz várias dificuldades, como grande diversidade dos textos, mesmo dentro das categorias, além da grande sobreposição das categorias (tipos de movimentos), como ocorre nos códigos 198 e 200. Certos processamentos comuns em NLP, como exclusão de *stopwords* e *steeming* podem excluir determinados termos e aproximar, no espaço multidimensional, os vetores de representação das categorias. Assim, apesar de, a priori, imaginarmos 10 classes, associados a 10 categorias ou códigos de movimentos de processos, há classes (ou movimentos) descritos, em vários casos, compartilham palavras e termos bastante semelhantes.

Trabalhos futuros focarão em outras bases de dados jurídicas, além da exploração de métricas diversas de avaliação de agrupamentos, objetivando perspectivas diferentes sobre a qualidade dos agrupamentos obtidos.

Os algoritmos possuem diferentes parâmetros que podem influenciar significativamente os resultados. Considera-se explorar diferentes configurações de parâmetros para cada método de agrupamento, como a realização de uma análise abrangente dos parâmetros e sua influência nos resultados, com uso, por exemplo, de frameworks como *Optuna* [20].

Finalmente, considera-se uso futuro de abordagens de ensemble, como descritas em [21], onde diferentes métodos são combinados para integrar os resultados dos métodos de agrupamentos.

Referências

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310. 4546, 2013.

[2] J. Aguilar, C. Salazar, H. Velasco, J. Monsalve-Pulido, & E. Montoya. Comparison and evaluation of different methods for the feature extraction from educational contents. *Computation*, 8(2), 30, 2020.

[3] K. Kawintiranon & Y. Liu. Towards Automatic Comparison of Data Privacy Documents: A Preliminary Experiment on GDPR-like Laws. *arXiv preprint arXiv:2105.10117*, 2021.

[4] Q. Lu, J. G. Conrad, K. Al-Kofahi, W. Keenan, Legal document clustering with built-in topic segmentation. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 383-392), 2011.

[5] D. Arthur, & S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, (pp. 1027-1035).

[6] D. Sculley. "Web Scale K-Means clustering", *Proceedings of the 19th international conference on World wide web,* 2010.

[7] T. Kohonen, *Self-organizing maps*. Springer series in information sciences, 30 Springer, Berlin, 3rd edition, (Dec 28, 2001)

[8] Ester, M., H. P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In *Proc. of the 2nd Intl Conference on Knowledge Discovery and Data Mining*, Portland, OR, AAAI Press, 1996, pp. 226–231.

[9] A.K. Jain, R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, 1998.

[10] A. P. Dempster, N. M. Laird, , D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B* (Methodological), 39(1), 1977, 1-22.

[11] T. Zhang, R. Ramakrishnan, & M. Livny. BIRCH: An efficient data clustering method for large databases. *ACM SIGMOD Record*, 25(2), 1996, 103-114.

[12] M. Ankerst, M. M. Breunig, H-P. Kriegel, and J. Sander. "OPTICS: ordering points to identify the clustering structure." *ACM SIGMOD Record 28*, no. 2, 1999, 49-60.

[13] L. McInnes and J. Healy. "Accelerated hierarchical density clustering." *arXiv preprint arXiv:1705.07321,* 2017.

[14] K. Kawintiranon, Y. Liu. Towards automatic comparison of data privacy documents: A preliminary experiment on gdpr- like laws. arXiv preprint arXiv:2105.10117, 2021.

[15] J. Devlin, M.-W. Chang, K. Lee, & K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL-HLT), Vol. 1, 2019, pp. 4171-4186.

[16] A. Aizawa. An information-theoretic perspective of TF–IDF measures. *Information Processing & Management* 39(1), 2003, 45–65.

[17] D. C. Araújo, A. Lima, J. P. Lima, J. A. F. Costa. A comparison of classification methods applied to legal text data. In: *EPIA Conf. on Artificia lIntelligence*, 2021, pp. 68–80. Springer.

[18] A. Strehl, & J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(Dec), 2003, 583-617.

[19] M. Steinbach, G. Karypis, e V. Kumar. "A comparison of document clustering techniques". *KDD Workshop on Text Mining*, 2000.

[20] Optuna. Available at https://optuna.org/. Access June, 14th, 2023.

[21] L. A. Pasa, J. A. F. Costa, M. G. de Medeiros (2017). An ensemble algorithm for Kohonen self-organizing map with different sizes. *Logic Journal of the IGPL*, Oxford Academic, Sept. 23, 2017, p. 1-14. https://doi.org/10.1093/jigpal/jzx046.