Aprendizado de Ranking de Entidades Aplicado aos Dados do Governo Brasileiro

Paulo Henrique Maia Soares

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2020

Paulo Henrique Maia Soares

Aprendizado de Ranking de Entidades Aplicado aos Dados do Governo Brasileiro

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Marcelo Keese Albertini

Uberlândia
2020

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

S676a 2020

Soares, Paulo Henrique Maia, 1991-
Aprendizado de ranking de entidades aplicado aos dados do Governo brasileiro [recurso eletrônico] / Paulo Henrique Maia Soares. - 2020.

Orientador: Marcelo Keese Albertini.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.6003
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Albertini, Marcelo Keese, 1984-, (Orient.). II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Rejâne Maria da Silva – CRB6/1925

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 32/2020, PPGCO				
Data:	03 de dezembro de 2020	Hora de início:	09:05	Hora de encerramento:	12:00
Matrícula do Discente:	11812CCP025				
Nome do Discente	Paulo Henrique Maia Soares				
Título do Trabalho:	Aprendizado de Ranking de Entidades Aplicado aos Dados do Governo Brasileiro				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Ciência de Dados				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Marcelo de Almeida Maia - FACOM/UFU; Ricardo Araújo Rios - UFBA e Marcelo Keese Albertini - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Ricardo Araújo Rios - São Carlos/SP; Marcelo de Almeida Maia e Marcelo Keese Albertini - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Marcelo Keese Albertini, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Ricardo Araújo Rios**, **Usuário Externo**, em 04/12/2020, às 14:31, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Marcelo de Almeida Maia**, **Professor(a) do Magistério Superior**, em 04/12/2020, às 15:59, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Marcelo Keese Albertini**, **Professor(a) do Magistério Superior**, em 04/12/2020, às 20:20, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2426462** e o código CRC **2823D9CD**.

Dedico este trabalho à minha esposa Dyessa e aos meus filhos Emanuel e Maria Clara.

Agradecimentos

Agradeço primeiramente a Deus, pelo dom da vida e por todas as oportunidades a mim concedidas. Agradeço também a Nossa Senhora, mãe de Jesus, por sua intermediação e auxílio na vida da graça.

À minha esposa Dyessa, pelos incansáveis incentivos para eu continuar a caminhada. Agradeço também pela compreensão, por seu carinho e amor nos momentos de ausências para dedicação deste trabalho.

Agradeço à minha mãe Maria Beatriz e aos meus avós Maria de Lourdes e Neomar(*in memoriam*), por todos os ensinamentos da vida e por me ajudar a perseverar diante das dificuldades. Agradeço minha irmã Ana Paula por todo companheirismo e carinho na jornada da vida. Agradeço também ao meu pai Modesto, por todos os ensinamentos.

Por fim deixo os meus agradecimentos ao meu orientador Marcelo Keese Albertini, pelos inúmeros ensinamentos, pela imensa paciência, pelo incentivo e por acreditar no meu trabalho.

"Todavia, coisa diferente prezo para mim, porque prefiro conhecer todas as coisas e ao mesmo tempo não ser por ninguém conhecido, do que o inverso, a saber, conhecer nada de nada, mas ser conhecido por todos!"

(Hugo de São Vitor)

Resumo

Com o crescimento da quantidade de informações referente à transparência governamental disponíveis nos últimos anos devido as exigências legislativas, o acesso à informação desejada torna-se cada vez mais difícil. Buscadores tradicionais como Google, Yahoo e Bing retornam os documentos ordenados pela relevância perante a consulta informada. A área cujo objetivo é retornar os documentos relevantes é conhecida como Recuperação de Informação à qual pode ser auxiliada por algoritmos de aprendizado de máquina para melhorar a ordenação dos documentos, denominada nesse contexto como Aprendizado de Ranking. Existem na literatura diversos algoritmos para resolver problemas de Aprendizado de Ranking, onde cada um busca resolver o problema de ordenação com base em diferentes critérios. No contexto de documentos governamentais observa-se a possibilidade de identificar quais são as principais entidades presentes nos documentos mais relevantes retornados em uma determinada consulta. Essa dissertação visou obter uma ordenação dos documentos disponíveis no Porta de Dados do Governo Brasileiro utilizando Aprendizado de Ranking e extrair informação de entidades de bases de dados não-estruturadas, semi-estruturas e tabulares, que são comuns entre as fontes disponibilizadas no Portal. Para atingir tal objetivo recorreu-se às técnicas disponíveis no estado da arte para reconhecimento de entidade nomeadas e utilizou-se das técnicas de otimização convexa para modelar o processo de aprendizado de ranking. Os resultados obtidos demonstraram-se superiores aos buscadores disponíveis no mercado (Google, Yahoo e Bing) visto que esses indexam somente o resumo dos conjuntos de dados do Portal de Dados.

Palavras-chave: Aprendizado de ranking, Recuperação de Informação, Aprendizado de máquina.

Abstract

With the growth in the amount of information, according to the governmental transparency available in recent years due to legislative requirements, access to information becomes increasingly difficult. Traditional search engines like Google, Yahoo and Bing return as desired information ordered by searching the document before the informed query. The area whose objective is to return relevant documents to the user is known as Information Retrieval which can be aided by machine learning algorithms to improve the ordering of documents, called in this context as Learning to Rank (L2R). There are several algorithms in the literature to solve L2R problems which each one seeks to solve the ranking problem in the best possible way. In the context of government documents, there is a possibility of identifying which are the main entities present in the most relevant documents relevant to a given query. This work aimed obtaining an ordering of the documents available on the Brazilian Government Data Portal using Learning to Rank and extracting information from entities from unstructured, semi-structured and tabular databases, which are common among the sources available on the Portal. To achieve this goal, used state-of-the-art techniques to recognize named entities and convex optimization models to model the L2R. The results obtained proved to be superior to the search engines available on the market (Google, Yahoo and Bing) since these index only the summary of data sets from the Data Portal.

Keywords: Learning to Rank, Information Retrieval, Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de siglas

RI - Recuperação de Informação

L2R - *Learning to Rank*

OGP - *Open Government Partnership*

GCU - Controladoria-Geral da União

tf - *Term Frequency*

idf - *Inverse Document Frenquency*

tf-idf - *Term Frenquency - Inverse Document Frequency*

BIM - Modelo de Independência Binária

HITS - *Hyperlink-Induced Topic Search*

SVM - *Support Vector Machine*

PRanking - *Perceptron-Based Ranking*

CKAN - *Comprehensive Knowledge Archive Network*

MIME - *Multipurpose Internet Mail Extensions*

REST - *Representational State Transfer*

NER - *Named Entity Recognition*

NLP - *Natural Language Processing*

TAE - Teoria do Aprendizado Estatístico

CVXR - *Disciplined Convex Optimization*

SCS - *Splitting Conic Solver*

MAP - *Mean Average Precision*

CG - *Cumulative Gain*

DCG - *Discounted Cumulative Gain*

nDCG - *Normalized Discounted Cumulative Gain*

Sumário

Capítulo 1

Introdução

Devido às exigências de legislação referente à transparência governamental, o governo brasileiro administra diversos portais que permitem a consulta de dados e informações relevantes para a administração pública. Exemplos desses portais são o Portal de Dados e o Portal da Transparência.

Assim como em diversas áreas e empresas (FACELI ANA CAROLINA LORENA, 2011), a quantidade de dados produzida pelos Governo Federal brasileiro é extremamente grande. Essa grande quantidade é essencial para garantir a transparência necessária para cidadãos entenderem como seus impostos estão sendo gastos e investidos na infraestrutura e serviços para a sociedade. Ao mesmo tempo, a grande quantidade de dados também configura um desafio do ponto de vista de acesso à informação. Isso porque muitas verificações necessárias para entender a atuação do governo exige o uso, acesso e transformação de múltiplas fontes de dados distintas que vão além da capacidade que pessoas tem em avaliar sem o uso de ferramentas computacionais.

Existem iniciativas que visam o desenvolvimento de ferramentas computacionais para o processamento automatizado de dados governamentais. Por exemplo, o projeto Serenata de Amor tem como objetivo a verificação automatizada de gastos de deputados federais e senadores utilizando técnicas de inteligência artificial (SERENATA, 2018). Em uma análise realizada, esse projeto foi capaz de identificar que um deputado pediu ressarcimento de 13 refeições em um só dia (SERENATA, 2018). Em outro projeto, cientistas sociais desenvolveram um pacote estatístico chamado electionsBR para analisar tendências políticas em votações utilizando dados do Tribunal Superior Eleitoral (ELECTIONSBR, 2018). Outra iniciativa é o estabelecimento de Observatórios Sociais que tem como um de seus objetivos a verificação de prestações de contas de órgãos públicos (OBSERVATóRIO, 2018). Atualmente, esses observatórios ainda usam muitas formas de verificação manual de dados.

O desenvolvimento de técnicas computacionais para processar e extrair informações de grandes volumes de dados começou desde o advento dos primeiros computadores digitais (LIU, 2011). Com o surgimento da *World Wide Web*, a necessidade de permitir a busca e

recuperação de informações de grandes coleções documentos textuais motivou o desenvolvimento da área de pequisa de Recuperação de Informação (RI). Nessa área, algoritmos, ferramentas técnicas, teoria e aplicações são estudadas e implementadas (MANNING, 2008).

Ainda hoje, muitos problemas de recuperação de informação estão sendo estudados e resolvidos. A disponibilidade de dados governamentais abre a possibilidade e exige o uso de ferramentas de recuperação de informação para possibilitar ampla e profunda análise dos dados.

No Portal de Dados verifica-se um alto grau de falta de estrutura para busca e acesso a dados. Tais dados são produzidos de maneira descentralizada, com pouca padronização, o que dificulta enormemente o uso de métodos manuais para descoberta de conhecimento.

Especificamente, muitas fontes envolvem informações de pessoas e empresas, denominadas *entidades* que possuem relacionamentos com o governo federal mas é difícil fazer consultas que permitem identificar quais dados são relacionados a essas entidades.

Observa-se a possibilidade da aplicação de técnicas para ordenação de resultados, cujos documentos identificam as bases de dados, que podem permitir a busca de assuntos específicos associados às entidades ou assuntos de interesse. Por exemplo, seria útil para cidadãos poderem buscar por palavras-chaves associadas à empresa Petrobrás e receber como resultado as bases que contêm dados relacionados e uma lista ordenada de quais entidades mais relevantes aparecem nessas bases.

A área de Recuperação de Informação tem desenvolvido técnicas de ordenação de documentos relevantes que auxiliam na identificação de entidades nomeadas como a Petrobrás. Do ponto de vista de ranking, técnicas avançadas de aprendizado de ranking fazem uso de aprendizado de máquina para obter ranking com máxima utilidade para o perfil de cada usuário (MANNING, 2008). As principais técnicas atualmente são baseadas em abordagens de aprendizado em *Pairwise*, *Pointwise* e *Listwise*. Seus principais desafios de implementação estão relacionados a: custo computacional, dependência de grande quantidade de dados e na organização desses dados. Essas técnicas são apresentadas no capítulo de fundamentação.

1.1 Motivação

A divulgação de dados do Governo brasileiro concede à sociedade a aprimoração desses dados para gerar valores coletivos. Dessa forma, a publicação de dados abertos permite a sociedade visualizar informações que tornam o estado transparente e fortalece a democratização da sociedade.

No Brasil, o acesso à informação é um direito previsto na Constituição Federal Brasileira desde o ano 1988. No ano de 2009 foi lançada a Lei da Transparência nº 131/2009, que reforçou os princípios de transparência das fundações públicas. Em 2011 o Brasil

passou a participar do *Open Government Partnership* (OGP) e firmou um contrato para dar os primeiros passos rumo a uma política de dados abertos. O acordo firmado na OGP consiste em disponibilizar uma ferramenta para que todos possam encontrar e utilizar os dados públicos. Dessa forma, em 2016 a presidenta Dilma Rousseff promulgou o decreto nº8.777/2016 que instituiu a Política de Dados Abertos do Governo brasileiro.

Com a Política de Dados Abertos, cada instituição governamental passou a ter a obrigação de elaborar um plano de dados abertos afim de disponibilizar seus dados. O objetivo primário da disponibilização dos dados abertos foi o total conhecimento e reutilização pela sociedade. O controle e monitoração do cumprimento das normas estabelecidas ficou a cargo da Controladoria-Geral da União (GCU) que garante a disponibilização das informações por parte das entidades envolvidas.

Alguns benefícios da utilização dos dados abertos incluem:

- ❏ Aumento do conhecimento da sociedade das ações e serviços do Governo;
- ❏ Clareza e transparência na prestação de contas no que diz respeito aos gastos públicos;
- ❏ Aumento do controle social das ações do Governo.

A atuação da sociedade nesse cenário de dados abertos representa um potencial para aumentar a eficiência e a eficácia do governo. Isso possibilita a melhoria dos serviços públicos e pode trazer diversos benefícios para a sociedade de um modo geral.

Devido à grande diversidade das entidades que disponibilizam os dados abertos, é notável no Portal de Dados a grande falta de estrutura e a dificuldade para encontrar as informações necessárias. Além do mais, os dados disponibilizados pelos órgãos governamentais conta com uma grande quantidade de informações referentes a pessoas e organizações, conhecido no contexto de Processamento de Linguagem Natural como *entidades*. Desta forma, acredita-se que a utilização dessas *entidades* para a composição de um *ranking* pode ajudar encontrar informações valiosas e impulsionar ainda mais a utilização dos dados abertos. Finalmente é importante ressaltar que os principais buscadores disponíveis no mercado (Google, Yahoo e Bing) não indexam todas as informações disponíveis no Portal de Dados, se limitando somente a um resumo disponível no HTML.

1.2 Objetivos e Desafios da Pesquisa

Com a intenção de prover um mecanismo de busca para o Portal de Dados que relaciona entidades relevantes a bases de dados relevantes, o objetivo deste projeto é propor e implementar um modelo de Aprendizado de Ranking que extraia informação de entidades de bases de dados não-estruturadas (puramente textual), semi-estruturadas (por exemplo linguagens de marcação XML) e tabulares, que são comuns entre as fontes disponibilizadas

no Portal. Grande parte do desafio técnico desse problema provém da falta de preparo das técnicas de aprendizado de ranking atuais em considerar as entidades que os documentos se referem, por exemplo, um buscador preparado para o Portal de Dados seria capaz de utilizar informações dos documentos e das entidades para buscar e traçar ligações entre diferentes bases não-estruturadas e semi-estruturadas com o intuito de melhorar a capacidade de encontrar documentos relevantes a entidades incluídas nas consultas.

Com o intuito de resolver esse problema, foi proposto e avaliado o uso de técnicas *off-the-shelf* para reconhecimento de entidade nomeadas, técnicas para identificar e compatibilização de indexação de documento de diferentes formatos de organização e modelos de otimização convexa para o processo de aprendizado de ranking considerando essas características (BOYD; VANDENBERGHE, 2004).

1.3 Hipótese

Este trabalho pretende validar a seguinte hipótese:

- Utilizar as entidades nomeadas presentes nos conjuntos de dados do Portal de Dados ajudam a melhorar a qualidade do ranking de documentos e facilita a busca por informações.

1.4 Contribuições

As contribuições deste trabalho incluem:

- Disponibilizar um novo método para obter um *ranking*, considerando as entidades nomeadas do conjunto de dados como agregação das técnicas de Aprendizado de *Ranking*;
- Construção de um *dataset* utilizando as informações presentes no Portal de Dados, que possibilita a execução de qualquer algoritmo de Aprendizado de *Ranking* implementado segundo a abordagem *pairwise*.

1.5 Organização da Dissertação

O presente trabalho está disposto da seguinte forma:

- **Capítulo 2:** Apresenta a fundamentação teórica deste trabalho, apresentando as principais técnicas e algoritmos da área de Recuperação de Informação. O capítulo apresenta tanto os modelos tradicionais quanto os modelos baseado em Aprendizado de Máquina;

- **Capítulo 3:** Apresenta detalhes das etapas seguidas para o desenvolvimento desta pesquisa. O capítulo apresenta detalhes das características dos conjuntos de dados presentes no Portal de Dados e mostra como foi feito a coleta de dados até a etapa da construção de um *dataset* específico para problemas de *Learning to Rank*;
- **Capítulo 4:** Descreve como foi conduzido os experimentos para validar a hipótese levantada, apresenta quais foram os métodos de avaliação utilizados e faz uma análise dos resultados obtidos;
- **Capítulo 5:** Contém a conclusão deste trabalho, bem como as principais contribuições e indica possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Sistemas de Recuperação de Informação tradicionais recuperam os documentos de acordo com a relevância à uma consulta de um usuário. Uma vez que todos os documentos foram encontrados baseados na consulta, o sistema ordena os documentos por meio de uma função de *ranking* $f(q, d)$, onde q representa a consulta do usuário e D o conjunto de documentos conforme ilustrado na Figura 1. Uma lista de documentos ordenados $d_{q,n}$ é retornada como resultado, onde n representa o total de documentos relacionado a consulta q. As funções de *ranking* dos Sistemas de Recuperação de Informação geralmente utilizam técnicas como BM25 ou modelos vetoriais (distância do cosseno) e podem também utilizar técnicas de Aprendizado de Máquina conforme detalhado a seguir.

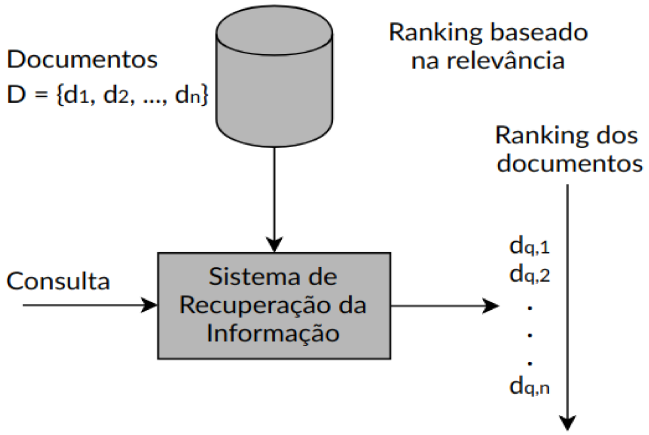


Figura 1 – Organização de um sistema de Recuperação de Informação tradicional (LI, 2011).

As Seções 2.1 e 2.2 apresentam informações sobre as principais técnicas de Recuperação da Informação tradicional. Em seguida, na Seção 2.4 será apresentado os modelos de Recuperação da Informação que utilizam Aprendizado de Máquina, mais especificamente Aprendizado de *Ranking* e suas principais abordagens.

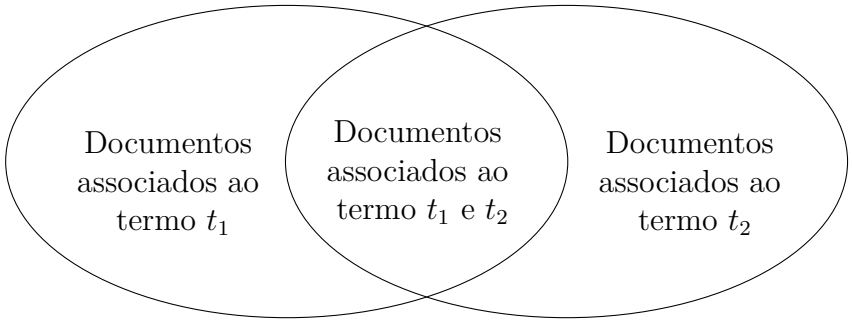


Figura 2 – Representação do modelo booleano.

2.1 Modelo Booleano

O modelo booleano é baseado na teoria de conjuntos e na álgebra booleana e foi a base dos primeiros sistemas de Recuperação da Informação (MANNING, 2008). Apesar desse modelo não definir medidas de relevância para os documentos, ele foi usado como base para os outros sistemas de Recuperação da Informação. Devido sua simplicidade e precisão, esse modelo foi e tem sido aplicado em vários cenários, tais como sistemas bibliográficos e de advocacia (MANNING, 2008).

No modelo booleano a indexação é feita utilizando o conceito de índice invertido, onde cada termo t é associado uma lista de todos os documentos que contêm t. A consulta é formada por todos os termos conectados por expressões lógicas AND, OR e NOT. Dessa forma, dada uma consulta q, o primeiro passo é recuperar a lista dos documentos associados a cada um dos termos contidos na consulta. Uma vez que temos todas as listas dos documentos pertencentes a cada termo, o próximo passo é obter a lista final de documentos utilizando as expressões lógicas da consulta.

A Figura 2 ilustra o modelo booleano, onde é possível representar todos os retornos para o usuário em uma consulta com dois termos. Por exemplo, caso a expressão lógica que conecta os dois termos seja AND o retorno será a intersecção dos dois conjuntos.

Logo, o resultado para uma determinada consulta são os documentos que satisfazem as expressões lógicas contidas na consulta. Os documentos são recuperados analisando se está ou não contido para um determinado termo. Dessa forma, não existe *ranking* para essa abordagem, ocasionando que não exista uma prioridade entre os documentos, como se todos estes fossem igualmente relevantes. O grande problema do Modelo Booleano é conhecido como tudo-ou-nada, devido a consulta pode retornar poucos ou muitos documentos.

2.2 Modelo Vetorial e Probabilístico

O Modelo Vetorial foi a primeira forma proposta para resolver o problema do tudo-ou-nada do Modelo Booleano.

A frequência de termo $tf_{t,d}$ indica o número de vezes que o termo t ocorre no documento d. Podemos utilizar a pontuação de tf para ranquear os documentos cujo os termos da consulta mais se repetem. Porém somente a pontuação de frequência não é suficiente, visto que a relevância do documento não aumenta proporcionalmente com o tf e não sabemos a raridade do termo na coleção de documentos. Sabemos que um documento com vários termos da consulta é mais relevante do que um outro documento que contém muitas repetições de somente um termo. O peso *idf* (*inverse document frequency*) mede essa raridade, sendo utilizado como um complemento do tf e é apresentado na Equação 1, onde N representa a quantidade de documentos presente na coleção e df_t representa o numero de documentos que o termo t aparece.

$$idf_t = \log \frac{N}{df_t} \tag{1}$$

Uma vez que definimos os pesos de tf e idf, podemos construir o peso $tf - idf$ que é o produto de peso tf e o seu peso idf conforme a Equação 2. O valor de $tf - idf$ aumenta com o número de ocorrências em um documento juntamente com a raridade do termo na coleção de documentos.

$$tf - idf = tf_{t,d} \times idf_t \tag{2}$$

O Modelo Vetorial representa os documentos e consultas como vetores de pesos $tf-idf$. Desta forma temos um espaço vetorial onde os termos são eixos do espaço e os documentos são os pontos ou vetores. Quanto maior a similaridade entre o documento e a consulta, menor será o angulo θ, podendo chegar a 0 quando a consulta e o documento possuem similaridade máxima.

A similaridade entre a consulta e o documento é calculada utilizando a distância cosseno do ângulo apresentada na Equação 3.

$$cos(\mathbf{q}, \mathbf{b}) = \frac{\mathbf{q_i} \cdot \mathbf{d_i}}{|\mathbf{q_i}||\mathbf{d_i}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}} \tag{3}$$

O valor de q_i é dado pelo peso $tf - idf$ do termo i na consulta. O valor de d_i é o peso $tf - idf$ do i-ésimo termo no documento. O valor de q_i e d_i são as magnitudes de $\mathbf{q}$ e $\mathbf{d}$, ou seja, é o produto das distâncias euclidianas dos dois vetores. O limite $|V|$ é o número total de termos no sistema. Caso os vetores estejam normalizados, o cosseno é equivalente ao produto escalar dos vetores. Existem outras formas de avaliar a importância de termos de consultas em documentos e podem ser encontradas em (MANNING, 2008).

O Modelo Probabilístico é uma melhoria do Modelo Vetorial. Esse modelo tem sólidos fundamentos teóricos baseados na teoria da probabilidade que ajudam a obter um

ranking. As primeiras implementações desse modelo é o Modelo de Independência Binária (BIM) (ROBERTSON; JONES, 1976) e o BM25 (ROBERTSON; WALKER, 1994) também conhecido como Okapi.

No BIM, os documentos e as consultas são representados como vetores binários. Dessa forma, um documento d é representado como um vetor $\mathbf{x} = (x_1, ..., x_M)$, onde x_t=1 representa que o termo está presente no documento d, e x_t=0 representa que o termo não está em d. A consulta segue a mesma representação do documento, sendo $\mathbf{q}$ o vetor de termos binários que podem ser $q_t = 1$ caso o termo esteja presente na consulta q, e $q_t = 0$ caso contrário.

Utilizando o BIM, a relevância do documento é calculada sobre os vetores de incidências de termos $P(R|\mathbf{x}, \mathbf{q})$, então usando o modelo de *Naïve Bayes* tem-se a Equação 4.

$$\begin{aligned} P(R = 1|\mathbf{x}, \mathbf{q}) &= \frac{P(\mathbf{x}|R = 1, \mathbf{q})P(R = 1|\mathbf{q})}{P(\mathbf{x}|\mathbf{q})} \\ P(R = 0|\mathbf{x}, \mathbf{q}) &= \frac{P(\mathbf{x}|R = 0, \mathbf{q})P(R = 0|\mathbf{q})}{P(\mathbf{x}|\mathbf{q})} \end{aligned} \tag{4}$$

Na Equação 4 temos $P(\mathbf{x}|R = 1, \mathbf{q})$ que é a probabilidade do documento x, aqui representado por seu vetor, ser relevante. Respectivamente, $P(\mathbf{x}|R = 0, \mathbf{q})$ é a probabilidade do documento ser não relevante para a consulta. Temos ainda $P(R = 1|\mathbf{q})$ e $P(R = 0|\mathbf{q})$ que são as probabilidades a *priori* de recuperarmos um documento relevante ou não relevante para a consulta q, e podem ser encontrados a partir da porcentagem de documentos relevantes na coleção. Portanto a atribuição de R representa a relevância ou não do documento.

No sistema BM25 foi proposto o uso de $tf - idf$ com parâmetros para estimar essas equações e obter um *ranking* conforme a Equação 5.

$$score(d) = \sum_{t \in q}^{n} log\left[\frac{N}{df_t}\right] \cdot \frac{(k_1 + 1)tf_{t,d}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{t,d}} \tag{5}$$

Tem-se que tf_{td} é a frequência de termos no documento d. L_d representa o tamanho do documento d, e L_{ave} representa o tamanho médio dos documentos na coleção. O parâmetro k_1 controla o ajuste para a frequência de termos dos documentos e b controla o ajuste para o comprimento do documento (ROBERTSON; ZARAGOZA; TAYLOR, 2004).

2.3 Modelo Baseado em *Hyperlinks*

Com o objetivo de melhorar a eficiência de um Sistema de Recuperação de Informação, os algoritmos em *hyperlinks* consideram a ligação entre os documentos de uma coleção

(BRIN; PAGE, 1998) (KLEINBERG, 1999) (MANNING, 2008). Essa ligação é dada por meio de um *hyperlink*, comumente encontrada em estruturas HTML e que possui duas propriedades relevantes:

- O *hyperlink* indica a relação entre dois documentos. Dado um documento d_1 que trata sobre um determinado assunto, se esse faz referência ao documento d_2, então o documento d_2 de alguma forma contém informações sobre o mesmo assunto que o d_1;
- O texto do *hyperlink*, encontrado entre as *tags*, descreve de forma resumida o conteúdo do documento referenciado.

Existem diversas implementações baseadas em *hyperlinks*, sendo o *PageRank* e HITS as principais delas.

2.3.1 *PageRank*

A proposta do algoritmo *PageRank* é encontrar a relevância de um documento dentro de uma coleção através das citações que ocorrem entre eles (BRIN; PAGE, 1998). Para isso, o *PageRank* contabiliza a quantidade e a qualidade dos links que apontam para um determinado documento. O cálculo do *PageRank* é definido pela Equação 6.

$$PR(d) = (1-p) + p\left(\frac{PR(d_1)}{C(d_1)} + \frac{PR(d_2)}{C(d_2)} + ... + \frac{PR(d_n)}{C(d_n)}\right) \tag{6}$$

Temos que $PR(d)$ é o *PageRank* do documento D. Os documentos que contém links para o documento D são representados por d_n. Os documentos que são apontados por D são representados por $C(d_n)$. Por fim, p é a probabilidade do usuário sair da página antes de visualizar o link.

2.3.2 HITS

O algoritmo HITS (*hyperlink-induced topic search*) (KLEINBERG, 1999) tem como objetivo utilizar os *hyperlinks* dos documentos para encontrar a relevância por meio de um conceito de *hubs* e autoridades. Quando informamos uma consulta para um buscador, pode ser que o número de palavras presentes em um site não sejam suficientes para que a página receba uma boa posição. Por isso, segundo Jon Kleinberg, encontrar um resultado relevante para uma determinada consulta vai além de apenas comparar uma lista de palavras que são comuns tanto para consulta quanto para os documentos.

Para o algoritmo HITS, as autoridades são as páginas que podem ser encontradas com as palavras contidas na consulta. Porém, existe uma segunda categoria de resultados que podem ser obtidos, chamados *hubs*, que são as páginas relacionadas as autoridades.

O algoritmo inicia seu funcionamento a partir de um conjunto de dados S, denominado raiz, que é obtido por uma função de ordenação baseada na consulta informada. O segundo passo é obter o conjunto de dados T, expandindo o conjunto S através da adição dos documentos referenciados ou que fazem referência a S. O HITS faz a associação dos documentos através do peso de hub $h(d_n)$ [1] e do peso de autoridade $a(d_n)$ conforme as Equações 7 e 8.

$$a(d_i) = \sum_{d_j \to d_i} h(d_j) \tag{7}$$

$$h(d_i) = \sum_{d_i \to d_j} a(d_j) \tag{8}$$

onde, $d_j \to d_i$ indica cada par documento d_j, d_i tal que o documento d_j possui um *hyperlink* para o documento d_i.

2.4 Aprendizado de *Ranking*

Como mencionado nas seções anteriores, existem diversas formas de se obter um *ranking* utilizando modelos tradicionais. Grande parte desses modelos contém parâmetros que necessitam ser ajustados para se obter um bom *ranking*, de acordo com as medidas de avaliação. Por exemplo, no sistema BM25 explicado na seção anterior, temos os parâmetros k_1 e b e para obter os ajustes precisamos utilizar um conjunto para validação. Porém, fazer os ajustes de parâmetros não é uma tarefa trivial, especialmente se considerarmos que as medidas de avaliação não levam em consideração os parâmetros (LI, 2011). Além disso, parâmetros bem ajustados para um conjunto de treinamento podem não funcionar bem para conjuntos novos, caso tenha ocorrido um sobre ajuste que também é conhecido como $overfitting$. Desta forma, parece bastante promissor utilizar técnicas de Aprendizado de Máquina para resolver os problemas de *ranking* mencionados.

No contexto de busca na *web*, o conceito de relevância pode ser representado por diferentes características, tais como a quantidade de referência que a página possui, PageRank e as avaliações que uma determinada página recebeu. Os mecanismos de busca na *web* registram várias informações sobre a iteração do usuário, como *clicks*, posições acessadas no retorno da busca, assuntos mais pesquisados e região do usuário. Essas informações possibilitam utilizar tais dados para treinar um modelo e obter a função de *ranking*.

Chamamos de Aprendizado de *ranking* (*Learning to Rank*) a área na qual utilizamos técnicas de Aprendizado de Máquina para resolver problemas de *ranking*. Técnicas de Aprendizado de *ranking* também podem ser utilizadas em diversas áreas tais como Processamento de Linguagem Natural, Mineração de Dados e Filtragem Colaborativa.

[1] Hub representa as páginas portal

2.4.1 Definição

Podemos formalizar o Aprendizado de *Ranking* como sendo uma tarefa de aprendizado de máquina supervisionado, onde espera-se receber exemplos de entradas e saídas geralmente fornecidas por um especialista ou a partir da observação do comportamento de usuários reais (LIU, 2011).

O conjunto de treinamento utilizado é formado por documentos e consultas, onde para cada consulta temos os documentos pertinentes a estes e também um grau dizendo o quão relevante é o documento dado a consulta. O grau de relevância é geralmente dividido em níveis, por exemplo, ótimo, excelente, bom, razoável e ruim (LI, 2011). Podemos também acrescentar outras informações para o treinamento, como, por exemplo, o vetor de características que representa o par documento e consulta. Essas características podem ser BM25, PageRank, $tfidf$, dentre outras.

A Figura 3 ilustra um sistema de Recuperação da Informação utilizando Aprendizado de *Ranking*. Inicialmente temos o conjunto de treinamento dado como entrada. Um exemplo desse conjunto consiste em n consultas $q_i (i = 1, ..., n)$, os documentos associados são representados pelo vetor de características $d^i = \{d^i_j\}^{d^i}_{j=1}$, onde d^i representa o número de documentos associados com a consulta q_i e o grau de relevância correspondente. Como trata-se de um processo de aprendizado supervisionado, precisamos também de um conjunto de teste que é estruturado da mesma forma que o conjunto de treinamento. Por fim temos um algoritmo de aprendizado que aprenderá a função de *ranking* capaz de predizer a relevância dos documentos para as novas consultas.

Desta forma, podemos formalmente definir o Aprendizado de *ranking* como sendo:

- Q o conjunto de todas as consultas q_1 ,q_2 , . . . , q_n;
- D o conjunto dos documentos d_{i1} ,d_{i2} , ..., d_{im} que estão associados a consulta q_i;
- Y o conjunto com os graus de relevância y_1 ,y_2, ..., y_l , cujos valores vão de 1 até l, sendo que $l > l - 1 > \ldots 1$, uma vez que $>$ indica uma maior relevância perante a consulta.

A maioria dos algoritmos de Aprendizado de *ranking* atuam de forma similar ao fluxograma apresentado na Figura 3. Esses algoritmos são categorizados em três diferentes abordagens, sendo elas a *pointwise*, *pairwise* e *listwise* onde, a Tabela 1 mostra os detalhes de cada abordagem. As seções a seguir apresentam os detalhes de cada uma dessas abordagens.

2.4.2 *Pointwise*

A abordagem *Pointwise* aplica as técnicas de aprendizado de máquina diretamente para resolver problemas de *ranking* (LIU, 2011). Essa abordagem realiza o treinamento do

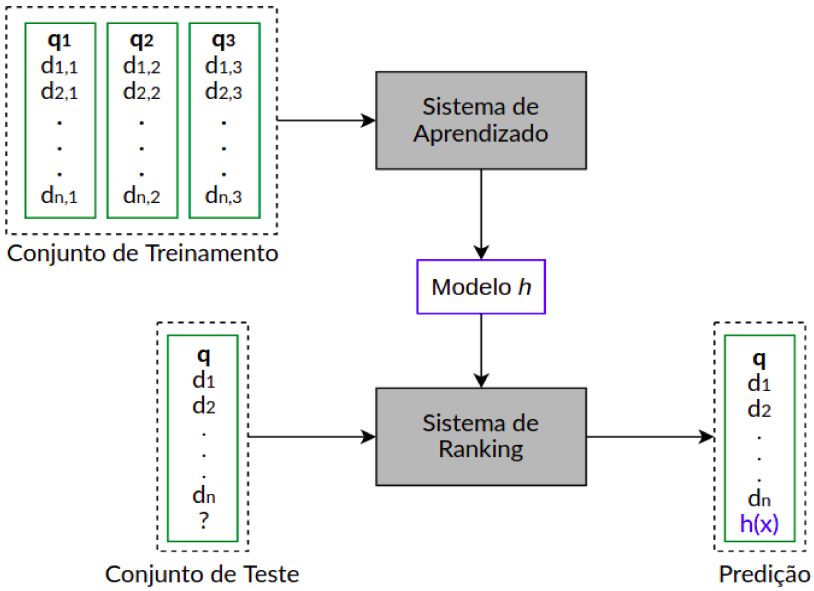


Figura 3 – Esquema de um Sistema de Recuperação da Informação utilizando Aprendizado de *ranking* (LI, 2011).

Abordagem	***Pointwise***		
	Regressão	Classificação	Regressão Ordinal
Entrada	Único documento x_i		
Saída	Valor real y_i	Categoria não ordenada y_i	Categoria ordenada y_i
Modelo	Função de ranking $f(x_i)$		
Função de perda	$L(f; x_i, y_i)$		
Abordagem	***Pairwise***		
Entrada	Par de documentos (x_i, x_j)		
Saída	Rótulo y_{ij}		
Modelo	$I\{f(x_i) > f(x_j)\}$ -1		
Função de perda	$L(f; x_i, x_j, y_{i,j})$		
Abordagem	***Listwise***		
Entrada	Conjunto de documentos $\mathbf{x} = \{x_j\}_{j=1}^{m}$		
Saída	*Ranking* π_y		
Modelo	$f(\mathbf{x})$		
Função de perda	$L(f; \mathbf{x}, \pi_y)$		

Tabela 1 – Resumo das características das abordagens *Pointwise*, *Pairwise* e *Listwise*.

algoritmo de classificação ou regressão utilizando cada documento dada uma determinada consulta. Nessa abordagem a relevância de cada documento é independente dos outros documentos e o *ranking* final é resultado da ordenação de documentos de acordo com o grau de relevância aprendida.

Nessa abordagem cada par consulta-documento no conjunto de treinamento tem uma pontuação numérica ou ordinal. Conforme apresentado na Figura 4, o algoritmo recebe n documentos relacionados a uma consulta q e retorna um grau de relevância para cada

documento. Dessa forma, estamos interessados em prever o grau de relevância de cada documento perante a consulta e, para tal, podemos utilizar por exemplo, o vetor de características de cada documento. A predição do grau de relevância do documento nessa abordagem geralmente é feita utilizando uma função de perda. O objetivo da função de perda é medir até que ponto a previsão gerada pela hipótese está de acordo com a real relevância do documento (LIU, 2011).

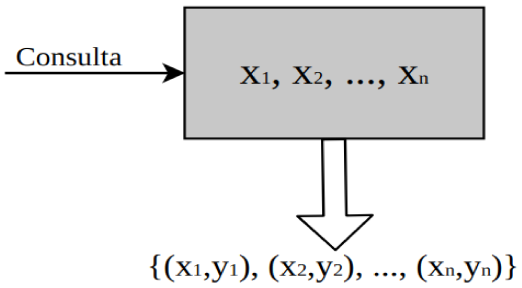


Figura 4 – Estrutura de um algoritmo de *Learning to Rank* utilizando a abordagem *pointwise*. O algoritmo recebe uma consulta e retorna para cada documento associado um grau de relevância.

Visto que a maioria das medidas de avaliação para sistemas de Recuperação de Informação baseiam-se em consultas juntamente com a posição do documento no *ranking*, essa abordagem possui suas limitações (LIU, 2011).

Exemplos de algoritmos dessa abordagem podem ser encontrados em (CHU, 2005), (COSSOCK, 2006), (KRAMER, 2000), (LI, 2008), (NALLAPATI, 2004), (SHASHUA, 2003), (VELOSO, 2008). A aplicação dessa abordagem pode ser dividida em três subcategorias, sendo elas, algoritmos baseados em regressão, classificação e regressão ordinal no qual serão descritas a seguir.

2.4.2.1 Algoritmos Baseados em Regressão

Em um problema de regressão, tentamos prever os resultados em uma saída contínua. O principal objetivo desta técnica é inferir a relação entre duas variáveis obtendo uma equação que explique tal situação. Nesse caso, o modelo construído irá retornar um valor real que indica o grau de relevância de um documento perante a consulta.

Em (COSSOCK, 2006) o problema de *ranking* é reduzido a um problema de regressão da seguinte maneira: seja $D = \{x_j\}_{j=1}^{m}$ o conjunto de documentos associados com a consulta q e $y = \{y_j\}_{j=1}^{m}$ a relevância associada a cada documento.

Dessa forma o objetivo é encontrar uma função f que seja capaz de retornar a relevância dos documentos dada a consulta. Para isso, Crossock e Zhang propuseram a Equação 9.

$$L(f; x_j, y_j) = (y_j - f(x_j))^2 \tag{9}$$

Como trata-se de uma função quadrática, temos que a perda mínima é dado quando o retorno da função é exatamente igual ao valor esperado (LIU, 2011). Ou seja, haverá perda 0 somente quando a função predizer 1 para um documento relevante. Caso contrário a perda aumenta em ordem quadrática.

2.4.2.2 Algoritmos Baseados em Classificação

Do mesmo modo que apresentamos na seção anterior, podemos transformar o problema de *ranking* em um problema de classificação. Nesse assunto lidamos com os documentos, denotados como d e os rótulos, representados por y. Desta forma, o principal objetivo desta técnica é predizer uma classe y baseado nas entradas informadas d. Existem vários algoritmos aplicado ao contexto de *ranking* que são baseados em classificação, onde um deles é a SVM que detalharemos a seguir.

Support Vector Machine (SVM) (VAPNIK, 1995) (VAPNIK, 1998) é um algoritmo de classificação robusto e amplamente utilizada no campo de aprendizado de máquina (NALLAPATI, 2004). Seu funcionamento é baseado na ideia de encontrar um hiperplano entre os dados afim de separá-los em diferentes classes. Entre dados de classes distintas podemos encontrar vários hiperplanos e nesse caso a SVM escolhe o hiperplano separador com maior margem. Quanto maior a margem em relação ao vetor de suporte do hiperplano, melhor será na previsão de novos registros. Entende-se por vetores de suporte os vetores que estão na margem do hiperplano separador (MANNING, 2008). Esses vetores são utilizados para delimitar o hiperplano separador.

Enquanto a etapa de treinamento do processo de classificação tradicional é um conjunto de objetos juntamente com os rótulos correspondentes, no processo de *ranking* o conjunto de treinamento é uma ordenação dos dados para uma determinada consulta. Dessa forma, $d = \{d_j\}_{j=1}^{m}$ representa o conjunto de documentos associado a uma determinada consulta q e $y = \{y_j\}_{j=1}^{m}$ o conjunto de relevâncias associadas a cada um dos documentos, podendo ser $+1$ para documentos relevantes e -1 para documentos não relevantes. A formulação da SVM aplicada ao contexto de *ranking*, utilizando *kernel* linear e a abordagem *Pointwise* é apresentado na Equação 10 (LIU, 2011).

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m^{(i)}} \xi_j^{(i)} \\ &\text{s.t.} \quad w^T x_j^{(i)} \leq -1 + \xi_j^{(i)}, \quad \text{if} \quad y_j^{(i)} = 0 \\ &\qquad\quad w^T x_j^{(i)} \geq 1 - \xi_j^{(i)}, \quad \text{if} \quad y_j^{(i)} = 1 \\ &\qquad\quad \xi_j^{(i)} \geq 0, \quad j = 1, ..., m^{(i)}, i = 1, ..., n, \end{aligned} \tag{10}$$

As variáveis $x_j^{(i)}$ e $y_j^{(i)}$ são os j-ésimos documentos e seus respectivos rótulos em relação à consulta q_i e, $m^{(i)}$ é o número de documentos associados a consulta q_i. O vetor que

define o hiperplano ótimo para separação dos dados é representado por w. As restrições da equação correspondem a se cada documento do conjunto de treinamento $x_j^{(i)}$ é classificado na classe correta. A função de perda nesse problema de otimização é definida em cada documento. Por exemplo, se o rótulo $y_j^{(i)}$ for -1 e a saída do modelo $w^T x_j^{(i)}$ não for maior que -1 então a perda para esse documento será 0. Caso contrário, a perda será $\xi_j^{(i)}$, também conhecida como margem suave do SVM. Dessa forma, a constante $\xi_j^{(i)}$ está relacionada com a classificação, correta ou não, do documento (LI, 2011). A função de perda utilizada nesse algoritmo é a *hinge*, que remete a um formato de dobradiça conforme a Figura 5.

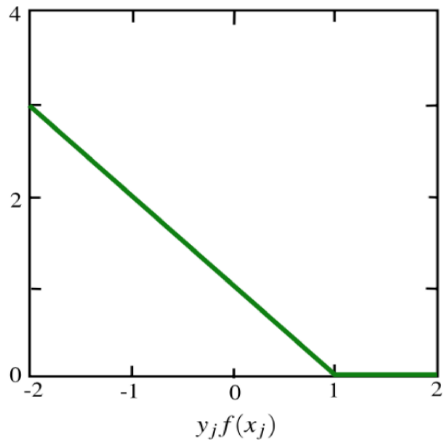


Figura 5 – Função de perda *hinge* usada na formulação (JOACHIMS, 2002).

2.4.2.3 Algoritmos Baseados em Regressão Ordinal

Algoritmos baseados em regressão ordinal são muito similares à classificação e à regressão tradicional (LI, 2011). Essa abordagem recebe como entrada um vetor de características e retorna uma nota de relevância de forma ordinal como, por exemplo, bom, muito bom, ruim e médio. O objetivo da regressão ordinal é encontrar uma função f que seja capaz de predizer uma pontuação dado o vetor de características recebido como entrada.

Em (CRAMMER, 2001) é apresentado um algoritmo de regressão chamado *PRanking* (Perceptron-Based *ranking*). Seu objetivo é encontrar uma direção definida por um vetor de parâmetros w por meio de um processo de aprendizado iterativo. A cada iteração t é obtida uma instância x_j que está associado a uma consulta q. Em seguida, o algoritmo faz a predição da classe $\hat{y}_j$ e compara o valor obtido com o rótulo de entrada y_j. Caso a predição feita pelo algoritmo esteja incorreta, ou seja, $\hat{y}_j \neq y_j$, então o valor de $w^T x_j$ está do lado errado de b_k. Para efetuar a correção é preciso mover os valores de $w^T x_j$ e b_k em direção ao outro lado. Por fim, o parâmetro do modelo w é ajustado iterativamente por $w = w + x_j$ e o processo de treinamento é repetido até convergir (CRAMMER, 2001).

Posteriormente (HARRINGTON, 2003) propôs a utilização de *ensemble* para melhorar o desempenho do *PRanking*. Para tal, primeiramente os dados de treinamento são sub-amostrados e um modelo de *PRanking* é aplicado para cada amostra. Em seguida os pesos associados a cada amostra são calculados para chegar ao *ranking* final. (HERBRICH, 2001) provou que desta forma o poder de generalização alcançado é maior do que o algoritmo tradicional.

2.4.3 Pairwise

A abordagem *pairwise* diferente da *pointwise* não foca em obter o grau de relevância de cada documento. Nessa abordagem busca-se obter o *ranking* reduzindo a um problema de classificação em pares de documentos. O objetivo nesse caso é classificar os pares de documentos predizendo qual dos dois é o mais importante dada a consulta e, minimizar o número de pares de documentos classificados de modo errôneo. Desta forma, se todos os pares de documento forem classificados corretamente, temos a solução ótima para o *ranking* (LIU, 2011). Como os pares de documentos são dependentes uns dos outros, não conseguimos aplicar algoritmos de classificação diretamente para solucionar o problema. A função de perda utilizada nessa abordagem considera somente a ordem relativa entre dois documentos e o objetivo principal é minimizar os pares de documentos não classificados corretamente.

Exemplos de algoritmos dessa abordagem podem ser encontrados em (JOACHIMS, 2002), (FREUND, 2003), (GAO, 2003), (BURGES, 2005), (SHEN, 2005), (TSAI, 2007), (ZHENG, 2007), (ZHENG, 2008) e (WU, 2010).

2.4.3.1 RankNet

O algoritmo RankNet foi criado por (BURGES, 2005) e soluciona o problema de *ranking* utilizando técnicas de redes neurais. Sejam dois documentos x_i e x_j associados com uma consulta q e suas respectivas probabilidades indicando qual documento do par é mais relevante, uma rede neural é formada por duas camadas e utiliza a função de perda inspirada na Entropia Cruzada, exibida na Equação 11.

$$c_{ij} = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}) \tag{11}$$

onde P_{ij} é modelado utilizando a função logística da Equação 12.

$$P_{ij} = \frac{e^{o_{ij}}}{1 + e^{o_{ij}}} \tag{12}$$

Assim, P_{ij} é a probabilidade do documento x_i ser mais relevante que o documento x_j e, em $\bar{P}_{ij}$, temos o valor esperado para essa probabilidade. O ajuste de pesos dos neurônios é feito utilizando *back propagation* calculando o gradiente descendente para aprender a função de pontuação (BURGES, 2005).

2.4.3.2 RankBoost

RankBoost criado por (FREUND, 2003) é um algoritmo baseado em técnicas de *boosting*, onde por meio da criação de vários *weak rankers* (em aprendizado de máquina são *weak learners*) é identificado qual documento é mais importante dado um par (x_i, x_j) perante a consulta q. Esse algoritmo adota o AdaBoost (FREUND, 1995) com a diferença de que o RankBoost é aplicado a um par de documento (LI, 2011).

O algoritmo atua de forma iterativa. Em cada etapa é realizado um treinamento do conjunto de pares de documentos D, afim de se obter um *weak ranker* f_t e um peso α_t atribuído a f_t. A escolha de f_t deve ser realizada afim de minimizar a Equação 13.

$$Z_t = \sum_{x_i, x_j} D_t(x_i, x_j) \exp(\alpha_t(f_t(x_i) - f_t(x_j)) \tag{13}$$

Nas próximas iterações o conjunto D será novamente considerado, porém priorizando os pares de documentos que tiveram o ranqueamento incorreto pelo ranker f. Desta forma, outro *weak ranker* f_t será criado e outro peso α_t descoberto. Finalmente a saída produzida $H(x)$ será efetuada através da combinação de todos os *weak rankers* disponíveis conforme a Equação 14.

$$H(x) = \sum_{t=1}^{T} \alpha_t f_t(x) \tag{14}$$

2.4.3.3 *Support Vector Machine*

A *Support Vector Machine* (SVM) adaptada à abordagem *pairwise* foi proposta por (JOACHIMS, 2002). A proposta intitulada *query-log* tem como objetivo registrar as interações dos usuários em um Sistema de Recuperação da Informação e utilizar desses registros para melhorar a qualidade do *ranking*. A ideia central da proposta se fundamenta no conhecimento de que o usuário interage com os documentos que o mesmo considera mais relevantes.

Na abordagem *pairwise*, a SVM trabalha com uma tupla (d_i, d_j) conforme a Equação 15. Nesse caso temos que o documento d_i tem maior relevância para a consulta q do que o documento d_j. O vetor $\vec{w}$ será ajustado pelo aprendizado do *ranking*.

$$(d_i, d_j) \in f_{\vec{w}}(q) \Leftrightarrow \vec{w}\Phi(q, d_i) > \vec{w}\Phi(q, d_j) \tag{15}$$

O vetor de características representado pela função $\Phi(q, d)$ descreve a relação de relevância entre a consulta e o documento (FUHR, 1992). A relação de relevância pode ser representada por exemplo como, Pagerank, BM25 e TF-IDF (BUCKLEY; SINGHAL; MITRA, 1995) (VITTAUT; GALLINARI, 2006).

O objetivo desse método então é encontrar um vetor que minimize o números de pares discordantes da verdade fundamental, conforme a Equação 16.

$$
\begin{aligned}
V(\vec{w}, \vec{\xi}) = \min \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \\
\text{s.t} \\
\forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) > \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\
\cdots \\
\forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) > \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \\
\forall_i \forall_j \forall_k : \xi_{i,j,k} \geq 0
\end{aligned}
\tag{16}
$$

Encontrar os pares concordantes é um problema NP-hard e por isso a SVM faz uma aproximação da solução ótima com a adição das variáveis de folga $\xi_{i,j,k}$. O par de documentos é representado pelas variáveis d_i e d_j e a consulta pela variável q_i. A constante C indica a regularização, onde a intensidade é inversamente proporcional ao valor atribuído. A principal diferença da SVM na abordagem *pointwise* e *pairwise* encontra-se nas definições das restrições do problema de otimização.

2.4.4 Listwise

A abordagem *listwise* examina todos os documentos em busca da ordenação ideal. Diferente da abordagem *pointwise* que analisa a relevância de cada documento perante a consulta e da abordagem *pairwise* que analisa a relevância relativa entre dois documentos dado a consulta, essa abordagem analisa a relevância de todo o conjunto de documentos para encontrar o *ranking*.

Na fase de treinamento dessa abordagem um conjunto $Q = \{q_1, q_2, ..., q_m\}$ de consultas é processado. Para cada consulta q_m, temos também a lista dos documentos associados $D = \{d_1, d_2, ..., d_n\}$. Além disso, temos para cada um dos documentos o grau de relevância $Y = \{y_1, y_2, ..., y_n\}$, que nos indica o quão relacionado o documento está à consulta q. Finalmente, a lista com as pontuações do *ranking* é criada utilizando a função f que é obtida do vetor de características juntamente com o documento correspondente (CAO, 2007).

Existem duas subcategorias na qual a *listwise* pode ser dividida. Na primeira categoria, a função de perda busca otimizar diretamente o valor de uma medida de avaliação apresentadas nas seções 2.1 e 2.2. Na segunda categoria, a função de perda não está diretamente relacionada com as medidas de avaliação de Sistemas de Recuperação da Informação, mas sim na diferença da lista obtida na predição com a existente na base de treinamento. A abordagem *listwise* busca minimizar uma função de perda levando em consideração a ordem de todos os documentos pertencente à lista.

Exemplos de algoritmos dessa abordagem podem ser encontrados em (BURGES, 2006), (CAO, 2007), (CHAPELLE, 2010), (HUANG; FREY, 2009) e (QIN TIE-YAN LIU, 2009).

2.4.4.1 ListNet

A principal dificuldade na abordagem *listwise* é como definir uma função de perda que possa representar a diferença entre a lista de documentos obtidas no processo de aprendizagem com a lista informada com os dados de entrada. Para responder a essa pergunta, o autor do algoritmo ListNet (CAO, 2007) propôs a função de perda apresentado na Equação 17

$$\sum_{i=1}^{M} L(y^{(i)}, z^{(i)}) \tag{17}$$

,onde L é a função de perda, $y^{(i)} = (y_1^{(i)}, \ldots, y_{n(i)}^{(i)})$ é o conjunto de pontuações para a coleção de documentos e $z^{(i)} = (f(x_1^{(i)}), \ldots, f(x_{n(i)}^{(i)}))$ é a lista das pontuações geradas no processo de treinamento. Dessa forma, as listas de pontuações são mapeadas para uma distribuição de probabilidades afim de utilizar alguma métrica entre as duas distribuições. Nesse cenário o autor propôs utilizar dois modelos: a probabilidade de permutação e a probabilidade top k. O algoritmo utiliza uma rede neural como modelo juntamente com o gradiente descendente conforme descrito no Algoritmo 1.

Algoritmo 1 ListNet

Require: Conjunto de treinamento $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\}$
  Parametros: Número de iterações T e a taxa de aprendizagem η
Ensure: Lista de segmentos X
  for $t = 1$ até T **do**
    for $i = 1$ até m **do**
      Calcular a lista de pontuação $z^{(i)}(f_\omega)$ informando $x^{(i)}$ da consulta $q^{(i)}$ para a Rede Neural com o ω atual
      Calcular o gradiente descendente $\Delta\omega$, onde ω é o peso da Rede Neural
      Atualizar $\omega = \omega - \eta \times \Delta\omega$
    end for
  end for
  return Modelo da Rede Neural ω =0

O algoritmo *ListNet* foi inspirado no *RankNet*. A maior diferença entre ambos está em o primeiro utilizar a listas de documentos como entrada (*ListNet*) enquanto o segundo utiliza pares de documentos (*RankNet*). Quando existe apenas dois documentos na abordagem *ListNet* a função de perda torna-se equivalente à do *RankNet* (CAO, 2007).

2.4.5 Considerações Finais

Esse Capítulo apresentou a fundamentação teórica dos sistemas de Recuperação de Informação, abordando as principais definições e modelos disponíveis na literatura. O próximo Capítulo apresenta a metodologia utilizada para a construção de uma base de dados própria para Aprendizado de *Ranking* e em seguida a aplicação de uma *Support Vector Machine (SVM)* adaptada à abordagem *pairwise* utilizando documentos e entidades.

Capítulo 3

Coleta de Dados e Indexação

A disponibilização e publicação dos dados governamentais permite que a sociedade utilize de tais dados a favor do bem comum (PIRES, 2020). Dessa forma, qualquer cidadão com acesso aos dados abertos deveria ser capaz de analisá-los e para isso é necessário que antes encontre tais informações. A área de Recuperação da Informação fornece meios e técnicas para que um usuário consiga buscar informações em um conjunto de dados.

Os princípios dos dados abertos disponíveis em (TAUBERER, 2007), diz que os dados devem ser razoavelmente estruturados para possibilitar o seu processamento automatizado. Podemos encontrar também nos mesmos princípios, que os dados não devem ser disponibilizados em formatos proprietários. Porém, existem dezenas de formatos não proprietários disponíveis para tal utilização (ELETRONICO, 2019), e que em um contexto de busca de informações pode dificultar bastante o processo.

No contexto do Portal de Dados, podemos encontrar informações publicadas por diversos setores governamentais, o que favorece a predominância de dezenas de formatos e padrões diferentes. No Portal de Dados, temos a possibilidade de encontrar um conjunto de dados através do campo de busca, porém existe uma grande limitação do seu funcionamento, estando limitada somente aos resumos que são disponibilizados pelos respectivos autores. Fundamentado nessas limitações, este trabalho busca possibilitar encontrar documentos do Portal de Dados na sua totalidade, considerando todas as informações presentes nos conjuntos de dados, bem como seus documentos anexos. Consideramos as entidades presentes em tais *datasets* com o objetivo de melhorar o ranking, visto que para tal contexto apresenta uma grande importância.

Neste capítulo, será descrito o método adotado para construção de um ranking utilizando técnicas de aprendizado de máquina e considerando as entidades presentes nos documentos.

3.1 Características dos Conjuntos de Dados

No último trimestre do ano de 2019 observou-se que o Portal de Dados contava com 6.791 conjuntos de dados disponíveis e 34.417 recursos. Cada conjunto de dados está vinculado a uma categoria, e pode ou não conter documentos anexos. Dos 6.791 conjuntos de dados presentes no Portal, 3.315 possuem algum documento anexo, representando 48,77% dos dados. Na tabela 2 é apresentado todos os formatos disponíveis e a quantidade de documentos pertencentes ao respectivo formato.

Formato	Quantidade	Percentual
CSV	10549	30.651%
HTML	4928	14.319%
JSON	4679	13.595%
PDF	3316	9.635%
WSDL	3063	8.900%
XML	1343	3.902%
GEOJSON	1024	2.975%
KML	1018	2.958%
ZIP	950	2.760%
XLSX	812	2.359%
SHP	749	2.176%
XLS	421	1.223%
ODS	401	1.165%
PNG	325	0.944%
ESRI REST	196	0.569%
TXT	167	0.485%
ODT	142	0.413%
SAS	70	0.203%
RDF	45	0.131%
API	44	0.128%
DOCX	38	0.110%
OGC WMS	24	0.070%
ODATA	22	0.064%
SHP-ZIP	15	0.044%
ZIP+ODS	12	0.035%
WMS	11	0.032%
PDF/A	11	0.032%
DOC	9	0.026%
KMZ	9	0.026%

URL	5	0.015%
XLM	3	0.009%
SQL	3	0.009%
ODF	2	0.006%
XPS	2	0.006%
OWL	2	0.006%
RAR	2	0.006%
ONLINE	2	0.006%
SVG	1	0.003%
CVS	1	0.003%
OTS	1	0.003%

Tabela 2 – Formatos presentes nos Conjuntos de Dados do Portal de Dados.

3.2 *Crawling*

Crawling é o processo automatizado de visitar páginas na Web com o objetivo de coletar seu conteúdo (OLSTON; NAJORK, 2010). Essa técnica é comumente utilizada no campo de Recuperação da Informação e é considerado o ponto de partida de onde os dados serão indexados. O conteúdo coletado pode conter vários formatos diferentes e a arquitetura do crawler deve ser capaz de lidar com esses diferentes formatos.

Conforme observado na seção anterior, o Portal de Dados possui algumas características próximas do universo Web. Podemos observar na tabela 2, que o Portal de Dados possui pelo menos 40 formatos diferentes de documentos, além das páginas HTML que contém as informações gerais dos conjuntos de dados. Portanto a utilização de ferramentas de *crawling* torna-se bastante útil para recuperar as informações.

A Figura 6 mostra a arquitetura de coleta de dados em alto nível. Primeiramente, na etapa de *crawler*, fez-se necessário a análise de como o Portal de Dados está estruturado e quais as informações estão disponíveis. No total, contabiliza-se 59 campos que descrevem os conjuntos de dados, onde 52% está presente em mais de 90% dos conjuntos de dados, tais como, título, autor, tags, etc. Entender a organização dos dados nesse processo foi crucial, pois como veremos nas próximas seções foi projetado um *dataset* específico para funcionar com as técnicas de *Learning to Rank*.

Para a recuperação dos dados, utilizou-se as APIs do CKAN, que é a maior plataforma para portal de dados em software livre do mundo (ASSOCIATION, 2014). Os dados dos *datasets* recuperados pelo *crawler* foram agrupados em diferentes diretórios juntamente com todos os seus documentos anexos para facilitar a indexação. Ao final do processo de *crawling*, chegamos em um total de 138 *gigabytes* de dados coletados.

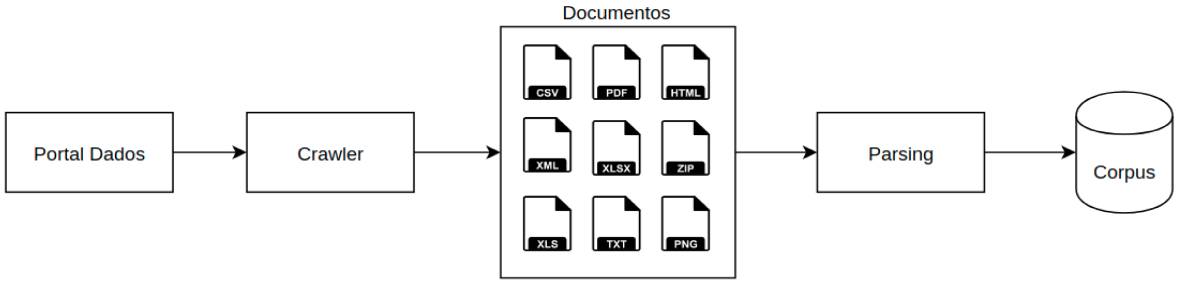


Figura 6 – Arquitetura da Coleta de Dados.

3.3 *Parsing*

A etapa de *parsing* ilustrado na Figura 6 foi realizada utilizando o projeto *open source Apache Tika* (FOUNDATION, 2007). O projeto foi escrito na linguagem Java e segundo a documentação oficial, possibilita extrair os metadados e o conteúdo de diferentes tipos de arquivos, tais como, XLS, PDF e PPT.

A arquitetura do *Apache Tika* é dividida em quatro principais módulos. O primeiro módulo é o *Language Identifier*, responsável por detectar a linguagem de um determinado documento. O segundo, chamado *Multipurpose Internet Mail Extensions (MIME)*, é responsável por detectar o tipo do documento, como por exemplo PDF e XML. O terceiro é o *Parser Interface*, responsável por extrair o conteúdo e os metadados de um documento. Por fim o *Tika Facade* é o módulo responsável por permitir a interação com a biblioteca, que foi implementado utilizando o padrão *facade* (FOUNDATION, 2007).

A etapa de *parsing* utilizou o módulo *Parser Interface* para extrair o conteúdo dos diversos formatos disponíveis no Portal de Dados. Uma vez extraído o conteúdo, executou-se uma limpeza dos dados, removendo pontuações, números e caracteres especiais.

3.4 Indexação

A última etapa ilustrada na Figura 6 refere-se a criação do *corpus*. Esta etapa permitirá executar consultas utilizando modelos tradicionais de Recuperação da Informação. Como visto na seção anterior, o resultado da etapa de *parsing* foi a junção do conteúdo de todos os documentos anexos a um conjunto de dados, bem como suas informações gerais presentes nas páginas HTML do Portal de Dados. Este resultado foi utilizado como entrada para a criação do corpus e foi indexado utilizando o projeto *Elasticsearch*.

Elasticsearch é um projeto *open source* que provê um mecanismo de busca e análise de dados distribuídos através de uma interface REST (FOUNDATION, 2010). O projeto foi desenvolvido sobre o Apache Lucene, que provê uma biblioteca na linguagem Java para indexação e busca utilizando os principais conceitos da área de Recuperação da

Informação (FOUNDATION, 2011).

A ingestão dos dados no *Elasticsearch* é feita utilizando um conceito conhecido como *schema-free*, significando que não é necessário definir uma estrutura de dados previamente, bem como acontece em bancos de dados relacionais. Os dados são armazenados nos *shardings*, que são as partições e podem ser armazenadas em diferentes servidores (*nodes*). Isso possibilita que a arquitetura seja redimensionada horizontalmente, permitindo distribuir e paralelizar os processamentos em diferentes servidores. Os *shards* podem ter réplicas, possibilitando a utilização de *fallbacks* conforme ilustrado na Figura 7. Por fim, vale ressaltar que o *Elasticsearch* utiliza o índice invertido como estrutura de dados, no qual o funcionamento foi explicado na seção 2.1.

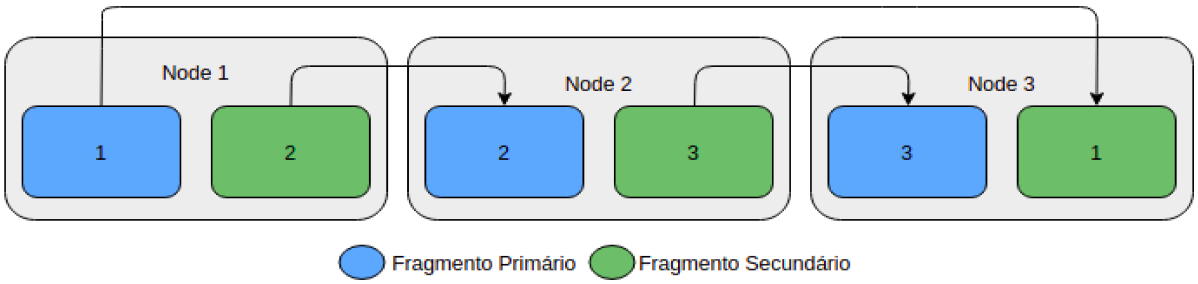


Figura 7 – Exemplo de três *nodes*, com três *shards* e uma réplica no *Elasticsearch* (FOUNDATION, 2010).

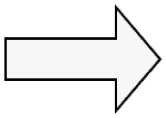
O Apache Lucene permite realizar as buscas utilizando diferentes algoritmos que foram explicados na Seção 2, tais como TF-IDF e BM25.

3.5 Construção da base de treinamento

A utilização de técnicas de Aprendizado de Máquina nos Sistemas de Recuperação de Informação pressupõe uma base de dados própria que siga os padrões estabelecidos. Dentre os vários componentes necessários para aplicação de Aprendizado de *Ranking* o registro de utilização é um que está presente na maioria dos Sistemas de Informação modernos. O objetivo nesse caso é monitorar e registar a interação do usuário com o sistema em um arquivo de log. As informações contidas nesse arquivo podem ser utilizadas para construir uma base de dados conforme exibido na Figura 8.

A primeira etapa para a construção da base de treinamento foi construir um sistema para buscar as informações no *corpus* indexado. O objetivo desta aplicação é permitir realizar buscas nos documentos que foram indexados no *Elasticsearch* e manter um histórico das consultas realizadas, bem como a relação das consultas e os documentos que foram retornados conforme ilustrado na Figura 9. A aplicação construída foi feita utilizando a linguagem Java e as consultas realizadas foram persistidas em um banco de

Consulta	Documento	Relevância
c1	d5	2
c1	d2	1
c1	d1	0
c2	d4	1
c2	d3	1
c2	d6	0
c3	d8	2
c3	d1	1
c3	d4	1



Consulta	Documento	a_1	a_2	a_3	...	a_N
c1	d5	0.2	0.7	0.9	...	0.4
c1	d2	0.6	0.1	0.3	...	0.4
c1	d1	0.0	0.4	0.2	...	0.1
c2	d4	0.1	0.1	0.5	...	0.3
c2	d3	0.8	0.9	0.4	...	0.1
c2	d6	0.6	0.1	0.3	...	0.7
c3	d8	0.4	0.3	0.8	...	0.9
c3	d1	0.9	0.1	0.4	...	0.5
c3	d4	0.1	0.3	0.6	...	0.6

Figura 8 – Exemplo de transformação dos dados de log para um *dataset* de Aprendizado de *Ranking*. A esquerda, dados coletados durante a interação do usuário. A direita transformação dos resultados em atributos.

dados MySQL. O histórico salvo seguiu o padrão utilizado em (JOACHIMS, 2002) com o objetivo de gerar uma base de dados própria para problemas de *Learning to Rank*.

Uma vez que o sistema foi construído e sua arquitetura preparada para armazenar as consultas e os resultados atrelados, a próxima etapa foi executar as consultas nos conjuntos de dados indexados. Foram selecionadas 69 consultas de forma aleatória, relacionadas aos diversos assuntos presentes no Portal de Dados. Cada consulta executada foi salva juntamente com os resultados atrelados na sua respectiva ordem.

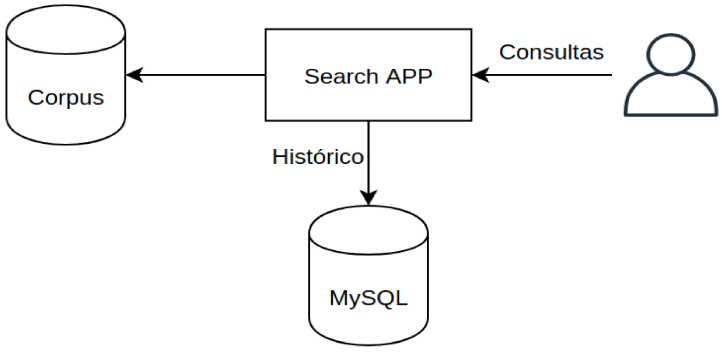


Figura 9 – Aplicação construída para realizar as buscas no *corpus*. Toda interação do usuário foi salva no padrão exibido da Figura 8.

Devido ao tamanho total dos dados coletados no Portal de Dados, não é factível extrair a relevância de todos os conjuntos de dados vinculados as consultas executadas. Uma prática comum nessa situação é extrair as *features* dos conjuntos de dados que possivelmente são relevantes (QIN TIE-YAN LIU; LI, 2010). Por esse motivo, os documentos factíveis de serem relevantes para uma determinada consulta, são os que foram retornados pela

aplicação construída e armazenada no histórico. Os documentos que não estão vinculados a nenhuma consulta são considerados, dessa forma, como irrelevantes. Para não cometer o erro de classificar um conjunto de dados como irrelevante, foram selecionados os top 100 conjuntos de dados para cada consulta executada, conforme o exemplo encontrado em (QIN TIE-YAN LIU; LI, 2010). É importante ressaltar que para algumas consultas, os documentos vinculados podem ser abaixo de 100 devido a falta de informação que vincule o documento à consulta.

No contexto de Recuperação da Informação, dado um conjunto de documentos e uma determinada consulta, o objetivo é encontrar quais documentos são mais relevantes para a consulta informada. Para resolver esse problema com *Learning to Rank*, cada par consulta-documento precisa ser representado em um vetor de *features* multidimensional, onde cada dimensão indica o quão relevante é o documento para uma determinada consulta.

O *dataset* criado contem 752 dimensões e 633 instâncias. A primeira coluna representa o ID da consulta realizada. A segunda coluna representa o ID do documento vinculado a consulta, seguido pelas *features* que foram extraídas. A estrutura das *features* geradas está organizada em três tipos, sendo o primeiro as *features* relacionadas ao conjunto de dados e seus documentos. O segundo tipo são as *features* que relacionam o documento com a consulta e o terceiro as *features* das entidades presentes nos documentos do conjunto de dados, sendo este último o de maior dimensão. A ordenação esperada como resultado das consultas aplicadas está representado pela ordem das linhas da matriz gerada, desta forma, o conjunto de dados da linha 1 tem maior relevância que a linha 2. A união das dimensões, composta pelas três divisões de *features* é o conjunto final utilizado para treinamento da função de *ranking*.

O par consulta-documento é representado pelo vetor de *features* que contém 752 dimensões. As 11 primeiras dimensões fazem parte do primeiro grupo de *features*, que representa as informações do documento. As nove dimensões seguintes representa o vínculo entre a consulta e o documento. Por fim, as últimas 732 são as entidades presentes nos conjuntos de dados. O terceiro grupo de *features* representa a parte central da proposta deste trabalho, pois tem como objetivo representar as entidades que estão presentes nos conjuntos de dados e consequentemente priorizar os documentos que possuem uma maior quantidade de entidades. A Tabela 3 contém um mapeamento das *features* do grupo um e dois.

ID da Dimensão	**Descrição da *Feature***
1	Tamanho do título
2	Tamanho do resumo
3	Quantidade de tags
4	Quantidade de grupos
5	Tamanho do título da organização
6	Tamanho do autor

7	Tamanho do responsável pelo conjunto de dados
8	Menor *Term Frequency* (TF)
9	Maior *Term Frequency* (TF)
10	Soma dos *Term Frequency* (TF)
11	Média dos *Term Frequency* (TF)
12	Menor *Inverse Document Frequency* (IDF)
13	Maior *Inverse Document Frequency* (IDF)
14	Soma dos *Inverse Document Frequency* (IDF)
15	Média dos *Inverse Document Frequency* (IDF)
16	Menor TF-IDF
17	Maior TF-IDF
18	Soma dos TF-IDF
19	Média dos TF-IDF
20	BM25

Tabela 3 – Descrição das *features* utilizadas na base de dados construído.

O terceiro grupo de *features* foi construído utilizando técnicas de Reconhecimento de Entidades Nomeadas. Entende-se por entidade nomeada uma sequência de palavras que faz referência a alguma entidade do mundo real (JIANG, 2012). A tarefa de Reconhecimento de Entidades Nomeadas tem como objetivo processar um *corpus* de escrita livre e identificar as entidades como pessoas, locais e organizações. Para realizar a extração das entidades dos conjuntos de dados, foi utilizado a ferramenta *open source Spacy*, escrito na linguagem *Python* e que oferece suporte a mais de 33 línguas (HONNIBAL, 2015). Essa biblioteca tem suporte a várias técnicas de NLP e no contexto deste trabalho foi utilizado somente o módulo *Named Entity Recognition* (NER). O módulo NER reconhece 18 tipos diferentes de entidades, sendo que foi utilizado para construção do terceiro grupo de *features* os tipos, pessoas e organizações.

Finalmente a matriz de *features* gerada passou por um processo de normalização. Este processo é de extrema importância pois variáveis com escalas muito diferentes podem atrapalhar o processo de aprendizagem. Utilizou-se a normalização *minimax* conforme apresentado na Equação 18. O valor é obtido subtraindo o valor atual pelo valor mínimo da dimensão e dividindo pela diferença entre o valor máximo e mínimo, fornecendo um valor entre 0 e 1.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{18}$$

3.5.1 Considerações Finais

Esse Capítulo apresentou a metodologia utilizada para a construção de uma base de dados própria para aplicação das técnicas de Aprendizado de *Ranking*. É possível observar na literatura (TROTMAN, 2005) (QIN TIE-YAN LIU; LI, 2010) (LIU, 2011) (TAN XUECHENG NIE, 2019) (PASUMARTHI R. K.; WOLF, 2019), que as principais base de dados disponíveis são organizadas utilizando os algoritmos tradicionais, como TF-IDF, BM25 e PageRank. A construção da base de dados desse trabalho considerou, além das técnicas disponíveis na literatura, as entidades presente nos documentos afim de melhorar a relação entre a consulta e o documento. O conjunto de dados próprio para Aprendizado de *Ranking* com todos os dados disponíveis no Portal de Dados serviu como entrada para a aplicação da SVM. O próximo capítulo exibirá detalhes do Aprendizado de *Ranking* aplicado a documentos e entidades, afim de comparar o resultado utilizando somente as informações presente na literatura.

Capítulo 4

Aprendizado de Ranking de Documentos e Entidades

A política de dados abertos do Poder Executivo Federal exige que todos os dados não sigilosos, referente a administração pública, sejam disponibilizados no Portal de Dados. É possível encontrar por exemplo, dados do sistema de transporte, de segurança pública, gastos governamentais, processos eleitorais, indicadores da educação e dados relacionados ao sistema de saúde. Todos os dados disponibilizados possuem vínculo com alguma instituição pública e tem potencial de conter informações referente à alguma pessoa física. Por consequência, essas entidades presente nos conjuntos de dados representa uma grande importância e podem ser utilizadas para melhorar o resultado em um sistema de Recuperação da Informação.

Na era da informação, a coleta das interações do usuário surge como uma abordagem relevante para ajustar um *ranking* de acordo com as preferências de cada usuário (CRASWELL; SOBOROFF, 2005) (SOBOROFF; CRASWELL, 2006) (QIN TIE-YAN LIU, 2009). Conforme apresentado no Capítulo 2, ao contrário das técnicas tradicionais, a área de Aprendizado de *Ranking* propõe encontrar uma função de ordenação dinamicamente utilizando algoritmos de Aprendizado de Máquina. Dessa forma, podemos encontrar diferentes *rankings* para diferentes perfis de usuários. No entanto, para que isso seja possível, é necessário que se tenha um conjunto de dados previamente construído para que o aprendizado aconteça. A construção da base de dados utilizada nesse trabalho foi detalhada no Capítulo 3.

A ordenação dos documentos por meio das técnicas de Aprendizado de *Ranking* é feita utilizando técnicas de aprendizado supervisionado. Assim, o aprendizado acontece condicionado a um conjunto de treinamento. No contexto de Aprendizado de *Ranking*, o conjunto de treinamento consiste no conjunto consulta-documento, onde cada consulta está associada a um número de documentos. A relevância dos documentos perante a consulta pode ser representada de diversas maneiras e no contexto desse trabalho, seguiu a ordem preestabelecida no conjunto de treinamento.

Encontrar um bom *ranking* por meio de técnicas de Aprendizado de Máquina tem sido vastamente pesquisado nas últimas décadas (MANNING, 2008). Isso é motivado pelo fato de que no contexto de buscas na WEB, a relevância pode ser representada por diferentes aspectos (LIU, 2011). O resultado dessas pesquisas trouxe o desenvolvimento de novas técnicas como, *Support Vector Machine*, Redes Neurais, Árvores de Decisão, Regressão e *Random Forest*.

Dentre as diversas técnicas desenvolvidas para lidar com o problema de Aprendizado de *Ranking*, a SVM tem demonstrado bons resultados e uma vasta utilização (FINE; SCHEINBERG, 2001) (JOACHIMS, 2002) (JOACHIMS HANG LI; ZHAI, 2007) (YU; JOACHIMS, 2008) (JOACHIMS; SCHNABEL, 2017) (YADAV; JOACHIMS, 2019). O algoritmo SVM é embasado pela Teoria do Aprendizado Estatístico (TAE) proposto por (VAPNIK, 1995). A TAE estipula algumas condições matemáticas que auxiliam a aprendizagem com o objetivo de encontrar um bom classificador a partir de um conjunto de dados.

Diferente da formulação da SVM apresentada no Capítulo 2, que considera a relação consulta-documento, a formulação proposta nesse trabalho considera também as entidades presente no documento e desta forma é representado pela tripla consulta-documento-entidades. Seja Q o conjunto de consultas e D o conjunto de documentos obtidos conforme explicado no Capítulo 3. O par consulta-documento (q_i, d_j) representa a correspondência do documento d_j para a consulta d_i. Considere que para a consulta q_1 foi retornado os documentos d_4, d_2 e d_7 em suas respectivas ordens. Dessa forma temos o conjunto de treinamento $S = \{(q_1, d_4), (q_1, d_2), (q_1, d_7)\}$. Em uma abordagem de Aprendizado de *Ranking* tradicional, para cada par (q_i, d_j), temos o vetor de *features* $\Phi(q_i, d_j)$ que representa a relevância do documento perante a consulta. As *features* podem incluir por exemplo, BM25, *PageRank* e TF-IDF. Na proposta deste trabalho as *features* são representadas com uma tripla $\Phi(q_i, d_j, e_j)$, onde a relevância do documento perante a consulta também considera as entidades presentes no documento. A Equação 19 mostra a formulação matemática da SVM considerando as entidades do documento, com a finalidade de melhorar o resultado da busca.

$$
\begin{aligned}
V(\vec{w}, \vec{\xi}) &= \min \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \\
&\text{s.t} \\
\forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i, e_i) &> \vec{w}\Phi(q_1, d_j, e_j) + 1 - \xi_{i,j,1} \\
&\cdots \\
\forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i, e_i) &> \vec{w}\Phi(q_n, d_j, e_j) + 1 - \xi_{i,j,n} \\
\forall_i \forall_j \forall_k : \xi_{i,j,k} &\geq 0
\end{aligned}
\tag{19}
$$

A Figura 10 mostra a estrutura das *features* organizada de acordo com a tripla

$\Phi(q_i, d_j, e_j)$ para aplicação do Aprendizado de *ranking* de documentos e entidades. As colunas de posição 21 à 752 representam todas as 731 entidades presentes no *corpus*.

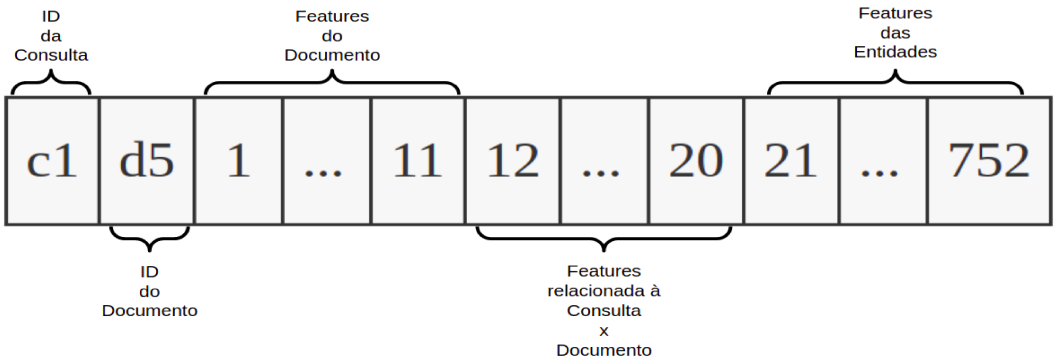


Figura 10 – Estrutura das *features* construída para aplicação do Aprendizado de *Ranking* de documentos e entidades.

A proposta de um método de Aprendizado de *Ranking* utilizando SVM nesta dissertação foi realizada utilizando o pacote de otimização convexa *Disciplined Convex Optimization* (CVXR) (GRANT; BOYD, 2008) (GRANT; BOYD, 2014). O pacote CVXR possibilita formular problemas de otimização convexa de uma forma natural, seguindo a convenção matemática. Seu funcionamento consiste em primeiramente analisar a formulação matemática informada, em seguida é verificado se o problema informado possui características de um problema convexo e, por fim, o pacote converte a formulação matemática informada para sua forma canônica e retorna um solucionador adequado para obter a solução. Existem diferentes *solvers* que podem ser utilizados para resolver problemas de otimização convexa no pacote CVXR (GRANT; BOYD, 2008) (GRANT; BOYD, 2014). Neste trabalho foi utilizado o *solver splitting conic solver* (SCS) (O'DONOGHUE et al., 2016) (BOYD, 2019). O *solver* SCS proposto or Boyd, apresenta excelentes resultados para resolução de problemas de otimização convexa e em diversos trabalhos apresenta um melhor desempenho comparado aos demais (NECOARA; GLINEUR, 2018). Para todos os experimentos, a execução da SVM foi realizada utilizando o *kernel* linear. O parâmetro C que é utilizado para regularização assumiu nos experimentos desta dissertação o valor 1. Os passos utilizados para obter o *ranking* estão descritos no Algoritmo 2.

Algoritmo 2 Aprendizado de *Ranking* de Documentos e Entidades

begin
Carrega os dados de treinamento;
Constrói a relação *pairwise* entre os documentos;
Aplica a SVM no pacote CVXR ;
Carrega os dados de teste;
Aplica o vetor ótimo no conjunto de teste;
Ordena os resultados de acordo com a pontuação;
return Resultado ordenado
end=0

Os experimentos realizados e os resultados obtidos são detalhados no Capítulo seguinte.

Capítulo 5

Experimentos e Análise dos Resultados

Este capítulo descreve os experimentos realizados para validação da hipótese levantada no Capítulo 1, a saber: Utilizar as entidades nomeadas presentes nos conjuntos de dados do Portal de Dados ajudam a melhorar a qualidade do ranking de documentos e facilita a busca por informações. Também apresenta os resultados obtidos pelo método proposto. Por fim, é feita uma análise dos resultados obtidos.

5.1 Método para a Avaliação

A abordagem padrão para avaliação de um sistema de *ranking* é dada pela noção de um documento ser relevante ou não relevante (LIU, 2011). Desta forma, o principal objetivo de medir a avaliação de um sistema nesse contexto é saber se o resultado obtido por um determinado *ranking* está condizente com a consulta realizada.

Existem diversas abordagens para avaliar um sistema de *ranking* disponíveis na literatura, tais como *Precision*, *Recall* e MAP (RUSSEL; NORVIG, 2003). Grande parte das abordagens de avaliação foram desenvolvidas para os modelos de *ranking* tradicional e só conseguem tratar situações onde o nível de relevância é binário (relevante ou não relevante). Uma técnica de avaliação que tem sido amplamente utilizada no contexto de *Learning to Rank* é a *Normalized Discounted Cumulative Gain* (nDCG) (LIU, 2011).

A medida nDCG foi desenvolvida para o contexto onde existem diferentes graus de relevância para os documentos retornados em uma consulta. Os graus de relevância neste caso se dão através de uma escala, como por exemplo 0 (não relevante), 1 (pouco relevante), 2 (relevante) e 3 (muito relevante) (BARTH, 2013). Segundo (SAKAI, 2007) a medida nDCG pode ser considerada com uma das melhores para o contexto de *ranking*.

A principal vantagem da medida nDCG é o fato dela priorizar os documentos retornados na sua respectiva ordem, o que as outras medidas não consideram. Dessa forma a medida considera que os documentos retornados no começo possuem maior relevância que os retornados em seguida.

Para o entendimento do nDCG considerando o cálculo do Ganho Acumulado (CG) apresentado na função 20. O CG é calculado recursivamente, somando a relevância do documento encontrado a partir da posição 1 até a posição i na lista ordenada.

$$CG[i] = \begin{cases} \text{G}[i] & \text{se } i = 1 \\ \text{CG}[i-1] + \text{G}[i] & \text{caso contrario} \end{cases} \tag{20}$$

Consideremos o vetor $D = \{3, 2, 3, 0, 0, 1, 2, ...\}$ que representa os documentos retornados como exemplo. Aplicando a função 20 no vetor, teríamos o seguinte CG como resultado: $CG = \{3, 5, 8, 8, 8, 9, 11, ...\}$.

O nDCG informa que os documentos retornados nas primeiras posições são mais importantes do que os documentos retornados nas posições inferiores (JäRVELIN; KEKäLäINEN, 2002). Para isso, é necessário diminuir a importância do documento progressivamente calculando o Ganho Acumulado Descontado (DCG) conforme a Equação 21, onde b é a base do logaritmo. Aplicando o DCG no vetor D, com o logaritmo na base 2, chegamos ao resultado $DCG = \{3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, ...\}$.

$$DCG[i] = \begin{cases} \text{CG}[i] & \text{se } i < b \\ \text{DCG}[i-1] + \text{G[i]} \ / \ log_b \text{i} & \text{se } i > b \end{cases} \tag{21}$$

Finalmente, podemos calcular o Ganho Acumulado Descontado Normalizado (nDCG) conforme a função 22. A equação considera o resultado obtido pelo DCG para um resultado ótimo $O = \{o_1, o_2, o_3, ...\}$ e o DCG retornado por um ranking qualquer $R = \{r_1, r_2, r_3, ...\}$.

$$nDCG = \{o_1/r_1, o_2/r_2, o_3/r_3, ...\} \tag{22}$$

5.2 Experimentos e Avaliação dos Resultados

Os experimentos a seguir foram conduzidos com a finalidade de validar as hipóteses levantadas na Seção 1.3. A base de dados utilizada em todos os experimentos foi construída com os dados do Portal de Dados e disponível na Seção 3.5. Para efeito de comparação, foi utilizado a base de dados construída com e sem as *features* de entidades nomeadas. A base de dados é composta por 633 instâncias, no qual dividiu-se em um conjunto de treinamento e um de teste. A divisão foi realizada de maneira aleatória, onde separou-se 75% dos dados para o conjunto de treinamento e 25% para o conjunto de teste (RUSSEL; NORVIG, 2003).

A primeira etapa do experimento consistiu em executar a SVM no conjunto de dados completo, contendo todas as 752 dimensões. A execução da SVM retornou um vetor ótimo e o mesmo foi aplicado em todos os conjuntos de dados. Em seguida o ranking

para a base completa foi obtida. A segunda etapa do experimento foi executar a SVM no conjunto de dados sem as entidades, contendo somente as *features* do primeiro e do segundo grupo apresentado na Seção 3.5. Por fim os dois *rankings* obtidos também foi comparado com o resultado do *ranking* obtido pelo algoritmo BM25.

A tabela 4 mostra os resultados dos rankings obtidos quantificado pela métrica nDCG. A SVM aplicada para o conjunto de dados completo, contendo as entidades, obteve um melhor resultado que a SVM executada no conjunto de dados sem as entidades. A medida nDCG também mostra que os 10 primeiros resultados da SVM aplicado com as entidades obteve um resultado superior ao algoritmo BM25. A aplicação da SVM no conjunto de dados sem as entidades obteve resultados muito próximos do *ranking* BM25.

	SVM dataset com entidades	**SVM dataset sem entidades**	***BM25***
nDCG@1	0,9642	0,8632	0,8144
nDCG@3	0,9167	0,8206	0,7911
nDCG@5	0,8817	0,7474	0,7494
nDCG@10	0,8542	0,7132	0,7062

Tabela 4 – Comparação dos rankings obtidos utilizando a métrica nDCG.

A Figura 11 mostra a comparação dos rankings obtido utilizando a métrica nDCG. É possível observar que a SVM aplicado ao conjunto de dados com entidades foi melhor comparada a SVM aplicada no conjunto de dados sem as entidades e ao algoritmo BM25. É interessante notar também que os melhores resultados estão concentrados nas primeiras posições, o qual para o contexto de Recuperação da Informação representa ótimos resultados (LIU, 2011). Os resultados da execução dos experimentos confirma a hipótese levantada na seção 1.3, que para os contexto do Portal de Dados considerar as entidades presentes na busca melhora a ordenação do *ranking* obtido.

Um exemplo de *ranking* obtido para a consulta "Servidores da UFOP"é apresentado na Tabela 5. A aplicação da SVM adaptada à abordagem *pairwise*, com a utilização das entidades apresentou melhor resultado. É possível notar que a aplicação da SVM com entidades obteve seis resultados idênticos ao resultado esperado, sendo os três primeiros os mais representativos. Já a aplicação da SVM sem a utilização das entidades conseguiu somente um resultado na mesma posição esperada.

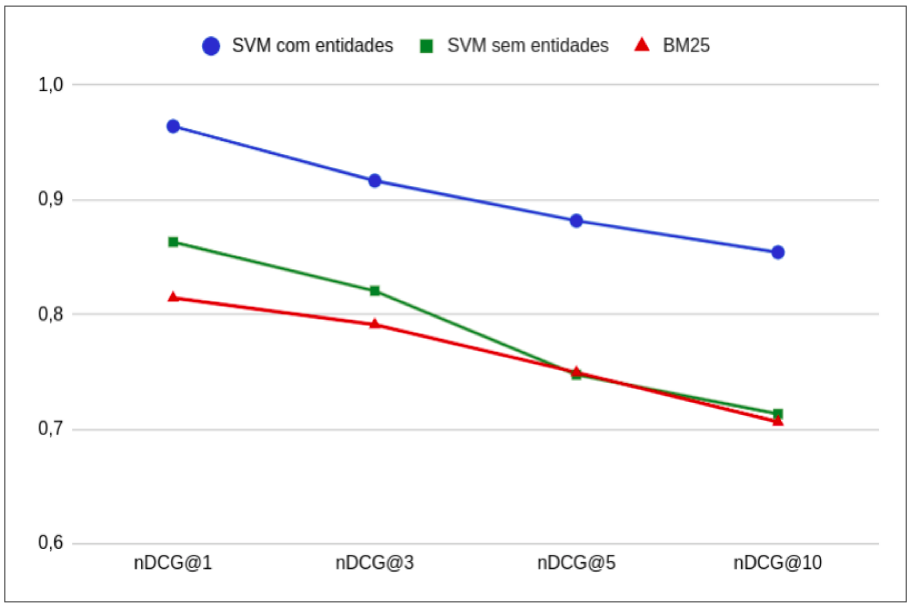


Figura 11 – Comparação dos rankings obtidos. Os melhores resultados foram alcançados com a aplicação da SVM com as entidades, que obteve um melhor resultado nas 10 primeiras posições.

Consulta	Ranking com Entidades	Ranking sem Entidades	Ranking Esperado
Servidores da UFOP	Obras executadas pela UFOP	Programas de Extensão	Obras executadas pela UFOP
	Residentes nas moradias estudantis da UFOP	Conjunto de dados de pesquisadores da UFOP	Residentes nas moradias estudantis da UFOP
	Conjunto de dados de pesquisadores da UFOP	Residentes nas moradias estudantis da UFOP	Conjunto de dados de pesquisadores da UFOP
	Relação de projetos de TI geridos pelo Núcleo de Tecnologia da Informação	Relação de projetos de TI geridos pelo Núcleo de Tecnologia da Informação	Bolsistas Transporte 2011-2
	Programas de Extensão	Bolsistas Transporte 2011-2	Programas de Extensão
	Bolsistas Alimentação 2015.1	Bolsistas Alimentação 2015.1	Bolsistas Alimentação 2015.1
	Estudantes contemplados com bolsa alimentação	Estudantes contemplados com bolsa alimentação	Relação de projetos de TI geridos pelo Núcleo de Tecnologia da Informação
	Bolsistas Transporte 2011-2	Quantitativo de matriculados na UFOP	Estudantes contemplados com bolsa alimentação
	Quantitativo de matriculados na UFOP	Obras executadas pela UFOP	Quantitativo de matriculados na UFOP
nDCG	**0,9906**	**0,9732**	**1**

Tabela 5 – Exemplo de um ranking obtido com e sem as entidade para a consulta Servidores da UFOP.

Capítulo 6

Conclusão

Este trabalho investigou o problema relacionado a busca de informações no Portal de Dados Brasileiro. Uma das características mais relevantes do Portal de Dados é a falta de estrutura e a dificuldade de encontrar as informações ali presentes.

Na etapa de análise dos conjuntos de dados disponíveis no Portal de Dados, identificou-se que na maioria das informações disponibilizadas pelo governo brasileiro contém algum tipo de entidade nomeada e que essas informações podem ser utilizadas para melhorar a qualidade dos resultados em um sistema de Recuperação de Informação. Desta forma, a proposta do trabalho voltou-se para a hipótese levantada, que considerou utilizar as entidades nomeadas no problema de ranking melhoraria o resultado da ordenação e a qualidade das consultas.

Afim de validar a hipótese levantada, aplicou-se o BM25 em um *corpus* construído que não considera as entidades nomeadas e a SVM aplicada a *Learning to Rank* em um conjunto de dados com e sem as entidades nomeadas. Dessa forma, foi possível comparar a qualidade do ranking utilizando técnicas tradicionais, técnicas de aprendizado de máquina e também a consideração das entidades nomeadas no problema, que foi a proposta deste trabalho.

Foi utilizado o nDCG como medida de avaliação e nos experimentos executados, a utilização das entidades nomeadas obteve resultados superiores tanto das técnicas tradicionais quanto das técnicas de aprendizado de máquina sem considerar as entidades. É importante ressaltar que ambos os resultados são superiores aos buscadores disponíveis no mercado (Google, Yahoo e Bing) visto que nenhum deles indexam todos os dados utilizados neste trabalho, se limitando somente ao resumo do conjunto de dados presente no HTML do Portal de Dados.

6.1 Principais Contribuições

A principal contribuição deste trabalho foi comprovar que a composição das entidades nomeadas para a aplicação de técnicas de Aprendizado de *Ranking* melhora o a qualidade

do *ranking* obtido para o conjunto de dados do Portal de Dados do governo brasileiro. Também foi construído e disponibilizado um *dataset* utilizando os dados do Portal de Dados que possibilita executar qualquer algoritmo de Aprendizado de Ranking implementado segundo a abordagem de *pairwise*.

Os código-fontes utilizados nesse projeto estão disponibilizados no repositório br-gov-learning-to-rank.

6.2 Trabalhos Futuros

Como trabalhos futuros, indica-se:

- Indica-se implementar outros algoritmos de Aprendizado de *Ranking* no *dataset* construído afim de verificar o resultado da sua aplicação com e sem as entidades nomeadas;
- Utilizar outras medidas de avaliação disponíveis no estado da arte para problemas de Aprendizado de *Ranking*;
- As entidades nomeadas extraídas foram selecionadas de maneira aleatória. Indica-se utilizar-se de outros meios para tentar melhorar o ranking;
- Explorar o uso de *feature* semânticas;
- Utilizar técnicas de análise de importância das *features* utilizadas.

Referências

ASSOCIATION, C. **CKAN**. 2014. Disponível em: <https://ckan.org/>. Acesso em: abr. 2020.

BARTH, F. J. Uma introdução ao tema recuperação de informações textuais. **Revista de Informática Teórica e Aplicada**, 2013. <https://doi.org/10.22456/2175-2745.26055>.

BOYD, S. **SCS: Splitting Conic Solver, version 2.1.2**. 2019. <https://github.com/cvxgrp/scs>.

BOYD, S.; VANDENBERGHE, L. **Convex Optimization**. [S.l.]: U.K.: Cambridge Univ. Press, 2004. <https://doi.org/10.1017/CBO9780511804441>.

BRIN, S.; PAGE, L. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 1998. <https://doi.org/10.1016/S0169-7552(98)00110-X>.

BUCKLEY, C.; SINGHAL, A.; MITRA, M. New retrieval approaches using SMART: TREC 4. In: **Proc. TREC**. [S.l.: s.n.], 1995.

BURGES, C. Learning to rank using gradient descent. 05 Proceedings of the 22nd international conference on Machine learning, 2005. <https://doi.org/10.1145/1102351.1102363>.

_____. Learning to rank with nonsmooth cost functions. NIPS, 2006.

CAO, Z. Learning to rank: From pairwise approach to listwise approach. ICML '07 Proceedings of the 24th international conference on Machine learning, 2007. <https://doi.org/10.1145/1273496.1273513>.

CHAPELLE, M. W. O. Gradient descent optimization of smoothed information retrieval metrics. Information Retrieval Journal. Special Issue on Learning to Rank 13, 2010. <https://doi.org/10.1007/s10791-009-9110-3>.

CHU, W. Gaussian processes for ordinal regression. Journal of Machine Learning Research 6, 2005. <https://doi.org/10.1007/978-3-540-28650-9_4>.

COSSOCK. **Subset Ranking Using Regression**. [S.l.]: International Conference on Computational Learning Theory, 2006. <https://doi.org/10.1007/11776420_44>.

CRAMMER, K. Pranking with ranking. Advances in Neural Information Process- ing Systems, 2001. <https://doi.org/10.1145/775047.775067>.

CRASWELL, A. d. V. N.; SOBOROFF, I. Overview of the trec 2005 enterprise track. **Proceedings of the Fourteenth Text REtrieval Conference**, 2005. <https://doi.org/10.1145/1401890.1402008>.

ELECTIONSBR. 2018. Disponível em: <https://www.electionsbr.com>. Acesso em: dez. 2018.

ELETRONICO, G. **Padrões de Interoperabilidade de Governo Eletrônico**. 2019. Disponível em: <http://eping.governoeletronico.gov.br/>. Acesso em: mai. 2020.

FACELI ANA CAROLINA LORENA, J. G. A. C. P. L. F. d. C. K. **Inteligência Artificial. Uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

FINE, S.; SCHEINBERG, K. Efficient svm training using low-rank kernel representations. **JMLR**, 2001. <https://doi.org/10.1162/15324430260185619>.

FOUNDATION, A. S. **Apache Tika**. 2007. Disponível em: <https://tika.apache.org/>. Acesso em: abr. 2020.

_____. **Elasticsearch**. 2010. Disponível em: <https://www.elastic.co/pt/>. Acesso em: mar. 2020.

_____. **Lucene**. 2011. Disponível em: <https://lucene.apache.org/>. Acesso em: mar. 2020.

FREUND, Y. A decision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Sciences, 1995. <https://doi.org/10.1006/jcss.1997.1504>.

_____. An efficient boosting algorithm for combining preferences. Journal of machine learning, 2003. <https://doi.org/10.1162/1532443041827916>.

FUHR, N. Probabilistic models in information retrieval. **Computer Journal**, v. 35, n. 3, p. 243–255, 1992. <https://doi.org/10.1093/comjnl/35.3.243>.

GAO, J. Linear discriminant model for information retrieval. Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003. <https://doi.org/10.1145/1076034.1076085>.

GRANT, M.; BOYD, S. Graph implementations for nonsmooth convex programs. In: BLONDEL, V.; BOYD, S.; KIMURA, H. (Ed.). **Recent Advances in Learning and Control**. [S.l.]: Springer-Verlag Limited, 2008, (Lecture Notes in Control and Information Sciences). p. 95–110. <https://doi.org/10.1007/978-1-84800-155-8_7>.

_____. **CVX: Matlab Software for Disciplined Convex Programming, version 2.1**. 2014. <http://cvxr.com/cvx>. <https://doi.org/10.1007/978-1-84800-155-8_7>.

HARRINGTON, E. F. Online ranking/collaborative filtering using the perceptron algorithm. Proceedings of the 20th International Conference on Machine Learning, 2003. <https://doi.org/10.1007/s10994-005-0918-9>.

HERBRICH, R. Bayes point machines. Journal of Machine Learning, 2001. <https://doi.org/10.1162/153244301753683717>.

HONNIBAL, M. **Spacy**. 2015. Disponível em: <https://spacy.io/>. Acesso em: mar. 2020.

HUANG, J. C.; FREY, B. J. Structured ranking learning using cumulative distribution networks. Advances in Neural Information Processing Systems 21 (NIPS 2008), 2009. <https://doi.org/10.1109/CVPR.2008.4587699>.

JIANG, J. Information extraction from text. **AGGARWAL**, 2012. <https://doi.org/10.1007/978-1-4614-3223-4_2>.

JOACHIMS, A. S. T.; SCHNABEL, T. Unbiased learning-to-rank with biased feedback. **WSDM**, 2017. <https://doi.org/10.1145/3018661.3018699>.

JOACHIMS HANG LI, T.-Y. L. T.; ZHAI, C. Learning to rank for information retrieval. **SIGIR Forum**, 2007. <https://doi.org/10.1561/1500000016>.

JOACHIMS, T. Optimizing search engines using clickthrough data. Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2002. <https://doi.org/10.1145/775047.775067>.

JäRVELIN, K.; KEKäLäINEN, J. Cumulated gain-based evaluation of ir techniques. **ACM Transactions on Information Systems**, 2002. <https://doi.org/10.1145/582415.582418>.

KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. Journal of the ACM, 1999. <https://doi.org/10.1145/324133.324140>.

KRAMER, S. Prediction of ordinal classes using regression trees. Funfamenta Informaticae 34, 2000. <https://doi.org/10.1007/3-540-39963-1_45>.

LI, H. A short introduction to learning to rank. EICE TRANSACTIONS on Information and Systems, 2011. <https://doi.org/10.1587/transinf.E94.D.1854>.

LI, P. Mcrank: Learning to rank using multiple classification and gradient boosting. Advances in Neural Information Processing Systems 20, 2008. <https://doi.org/10.1093/comjnl/35.3.243.31>.

LIU, T.-Y. **Learning to rank for information retrieval**. [S.l.]: Springer, 2011. <https://doi.org/10.1561/1500000016>.

MANNING, C. D. **Introduction to Information Retrieval**. [S.l.]: Cambridge University, 2008. <https://doi.org/10.1017/CBO9780511809071>.

NALLAPATI, R. Discriminative models for information retrieval. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, 2004. <https://doi.org/10.1145/1008992.1009006>.

NECOARA, Y. N. I.; GLINEUR, F. Linear convergence of first order methods for non-strongly convex optimization. **Mathematical Programming**, 2018. <https://doi.org/10.1007/s10107-018-1232-1>.

OBSERVATóRIO. 2018. Disponível em: <http://osbrasil.org.br/>. Acesso em: dez. 2018.

O'DONOGHUE, B. et al. Conic optimization via operator splitting and homogeneous self-dual embedding. **Journal of Optimization Theory and Applications**, v. 169, n. 3, p. 1042–1068, June 2016. <https://doi.org/10.1007/s10957-016-0892-3>.

OLSTON, C.; NAJORK, M. **Web crawling**. [S.l.]: Journal of Foundations and Trends in Information Retrieval, 2010. <https://doi.org/10.1561/9781601983237>.

PASUMARTHI R. K., W. X.-L. C. B. S. B. M. N. M. P. J. G. N. A. R.; WOLF. Tf-ranking: Scalable tensorflow library for learning-to-rank. **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining**, 2019. <https://doi.org/10.1145/3292500.3330677>.

PIRES, M. T. **Guia de Dados Abertos**. 2020. Disponível em: <https://ceweb.br/guias/dados-abertos/>. Acesso em: abr. 2020.

QIN TIE-YAN LIU, H. L. T. A general approximation framework for direct optimization of information retrieval measures. Springer Netherlands, 2009. <https://doi.org/10.1007/s10791-009-9124-x>.

QIN TIE-YAN LIU, J. X. T.; LI, H. Letor: A benchmark collection for research on learning to rank for information retrieval. **Information Retrieval Journal**, 2010. <https://doi.org/10.1007/s10791-009-9123-y>.

ROBERTSON, S. E.; JONES, K. S. Relevance weighting of search terms. Journal of the American Society for Information science, 1976. <https://doi.org/10.1002/asi.4630270302>.

ROBERTSON, S. E.; WALKER, S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. Proc. of the 17th ACM SIGIR, 1994. <https://doi.org/10.1007/978-1-4471-2099-5_24>.

ROBERTSON, S. E.; ZARAGOZA, H.; TAYLOR, M. Simple bm25 extension to multiple weighted fields. Proceedings of the thirteenth ACM international conference on Information and knowledge management, 2004. <https://doi.org/10.1145/1031171.1031181>.

RUSSEL, S. J.; NORVIG, P. Artificial intelligence: a modern approach. **Prentice-Hall**, 2003. <https://doi.org/10.1016/j.artint.2011.01.005>.

SAKAI, T. On the reliability of information retrieval metrics based on graded relevance. **Information Processing Management**, 2007. <https://doi.org/10.1016/j.ipm.2006.07.020>.

SERENATA. 2018. Disponível em: <https://serenata.ai/>. Acesso em: dez. 2018.

SHASHUA, A. Ranking with large margin principles: two approaches. Advances in Neural Information Processing Systems 15, 2003. <https://doi.org/10.1155/2017/4629534>.

SHEN, L. Ranking and reranking with perceptron. Journal of Machine Learning, 2005. <https://doi.org/10.1007/s10994-005-0918-9>.

SOBOROFF, A. P. d. V. I.; CRASWELL, N. Overview of the trec 2006 enterprise track. **The Fifteenth Text REtrieval Conference Proceedings**, 2006.

TAN XUECHENG NIE, Q. Q. N. L. H. L. Z. Learning to rank proposals for object detection. **Proceedings of the IEEE/CVF International Conference on Computer Vision**, 2019. <https://doi.org/10.1109/ICCV.2019.00836>.

TAUBERER, J. **Eight principles of open government data**. 2007. Disponível em: <https://opengovdata.org/>. Acesso em: mai. 2020.

TROTMAN, A. Learning to rank. information retrieval. **Springer**, 2005. <https://doi.org/10.1007/s10791-005-6991-7>.

TSAI, M.-F. Frank: a ranking method with fidelity loss. 07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007. <https://doi.org/10.1145/1277741.1277808>.

VAPNIK. The nature of statistical learning theory. Springer, Berlin, 1995. <https://doi.org/10.1007/978-1-4757-2440-0>.

_____. Statistical learning theory. Wiley-Interscience, New York, 1998.

VELOSO, A. A. Learning to rank at query-time using association rules. Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008. <https://doi.org/10.1145/1390334.1390381>.

VITTAUT, J.-N.; GALLINARI, P. Machine learning ranking for structured information retrieval. In: **Proc. ECIR**. [S.l.: s.n.], 2006. p. 338–349. <https://doi.org/10.1007/11735106_30>.

WU, Q. Adapting boosting for information retrieval measures. Information Retrieval, 2010. <https://doi.org/10.1007/s10791-009-9112-1>.

YADAV, Z. D. H.; JOACHIMS, T. Fair learning-to-rank from implicit feedback. **arXiv**, 2019.

YU, C.-N. J.; JOACHIMS, T. Training structural svms with kernels using sampled cuts. **Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining**, 2008. <https://doi.org/10.1145/1401890.1401985>.

ZHENG, Z. A regression framework for learning ranking functions using relative relevance judgments. 07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007. <https://doi.org/10.1145/1277741.1277792>.

_____. Query-level learning to rank using isotonic regression. Workshop on Learning to Rank for Information Retrieval, 2008. <https://doi.org/10.1109/ALLERTON.2008.4797684>.