CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CEFET-MG
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

SISTEMAS DE RECOMENDAÇÃO BASEADOS EM VETORES DE PALAVRAS E PROPRIEDADES DE REDES COMPLEXAS

RENATO BARROS ARANTES

Orientador: Prof. Dr. Anisio Mendes Lacerda
Centro Federal de Educação Tecnológica de Minas Gerais

Coorientador: Prof.ª Dra. Elizabeth Fialho Wanner
Centro Federal de Educação Tecnológica de Minas Gerais

Belo Horizonte, Junho de 2018

RENATO BARROS ARANTES

SISTEMAS DE RECOMENDAÇÃO BASEADOS EM VETORES DE PALAVRAS E PROPRIEDADES DE REDES COMPLEXAS

Texto de mestrado submetido à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito final para obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de Concentração: Modelagem Matemática e Computacional
Linha de Pesquisa: Sistemas Inteligentes

Orientador: Prof. Dr.Anísio Mendes Lacerda
Co-orientador: Prof. Dr. Elizabeth Fialho Wanner

Belo Horizonte
Junho de 2018

Resumo

Com o rápido crescimento e a abundância de serviços na internet, as pessoas se vêem em frente a uma enorme gama de opções. Embora esse crescimento tenha permitido que os usuários da internet consumam uma grande quantidade de recursos com apenas um clique, também tornou mais difícil para os mesmos encontrar informações relevantes para seus interesses. Em muitas situações, as pessoas possuem pouca ou quase nenhuma experiência pessoal para escolher uma dentre as inúmeras opções que são oferecidas. Neste contexto, os sistemas de recomendação desempenham um papel importante auxiliando no aumento da capacidade e eficácia deste processo de indicação. Neste trabalho abordamos a recomendação de *tags* e a recomendação de artigos científicos, que têm em comum a utilização de propriedades de redes complexas como meio auxiliar no processo de recomendação. A recomendação de *tags* é feita de duas maneiras distintas. Para determinar a relevância de uma *tags* em um documento, o vocabulário do documento é representado usando um grafo gerado a partir de um modelo de vetores de palavras. Utilizando este grafo, métricas de rede complexas são aplicadas para calcular a relevância das palavras do documento e as palavras mais relevantes são selecionadas como as *tags* recomendadas. Alternativamente, as *tags* podem ser recomendadas usando as métricas de redes complexas como dados de entrada para técnicas de aprendizado de máquinas. Para a recomendação de artigos científicos, para cada usuário, pretende-se recomendar artigos que não estão em sua biblioteca, mas são de potencial interesse para o mesmo. As informações sobre as bibliotecas de outros usuários e informações sobre os artigos disponíveis, tais como título e resumos, utilizados como entrada para um modelo de vetor de palavras, serão usadas para gerar uma rede complexa e, usando propriedades de redes complexas, palavras deste grafo são selecionadas para expandir um documento. A partir dos resultados concluímos que combinar as métricas de rede é mais eficiente do que usá-las separadamente para recomendar *tags* nos filmes do conjunto de dados IMDb. Para os métodos não supervisionados, o melhor resultado foi obtido pelo algoritmo TF-IDF, com uma precisão de aproximadamente 31%. Este resultado é muito inferior ao que foi obtido usando o algoritmo *Random Forest* que foi de aproximadamente 85% no cenário supervisionado. Os resultados obtidos com a metodologia proposta para recomendação de *tags* mostraram o potencial da abordagem. A partir dos experimentos realizados para a recomendação de artigos científicos, a técnica proposta melhora o desempenho do modelo CTR para o problema de *cold start*.

PALAVRAS-CHAVE: Sistemas de recomendação, Redes Complexas, Vetores de Palavras, Aprendizado de Máquinas, Processamento de LInguagem Natural.

Abstract

With the rapid growth and abundance of services on the Internet, people are faced with a huge range of options. While this growth has allowed Internet users to consume a large amount of resources with just one click, it has also made it more difficult for them to find information relevant to their interests. In many situations, people have little or no personal experience to choose from among the many options that are offered. In this context, recommendation systems play an important role in helping to increase the capacity and effectiveness of this referral process. In this work, we discuss the recommendation of *tags* and the recommendation of scientific articles, which have in common the use of complex network properties as an auxiliary medium in the recommendation process. The *tags* recommendation is made in two distinct ways. To determine the relevance of a tag in a document, the vocabulary of the document is represented using a graph generated from a word vector model. Using this graph, complex network metrics are applied to calculate the relevance of the document's words, and the most relevant words are selected as the recommended tags. Alternatively, tags can be recommended using complex network metrics as input data for machine learning techniques. For the recommendation of scientific articles, for each user, it is intended to recommend articles that are not in your library, but are of potential interest for the same. Information about other users libraries and information about available articles, such as title and abstracts, used as input to a word vector template, will be used to generate a complex network and, using complex network properties, words from this graph are selected to expand a document. From the results we conclude that combining network metrics is more efficient than using them separately to recommend *tags* in movies in the IMDb dataset. For unsupervised methods, the best result was obtained by the TF-IDF algorithm, with an accuracy of approximately 31%. This result is much lower than that obtained using the *Random Forest* algorithm which was approximately 85% in the supervised scenario. The results obtained with the methodology proposed for recommendation of *tags* showed the potential of the approach. From the experiments performed for the recommendation of scientific articles, the proposed technique improves the performance of the CTR model for the cold start problem.

KEYWORDS: Recommendation Systems, Complex Networks, Word Vectors, Machine Learning, Natural Language Processing.

LISTA DE ABREVIATURAS E SIGLAS

NLP	Natural Language Processing
TF	Term Frequency
IDF	Inverse Document Frequency
SVM	Support Vector Machine
CV	Cross-Validation
LDA	Latent Dirichlet Allocation
CTR	Collaborative Topic Regression
SLIM	Supersparse Linear Integer Model
BPR	Bayesian Personalized Ranking

Sumário

1 INTRODUÇÃO **1**

1.1 Objetivos 3

1.2 Organização do Trabalho 3

2 SISTEMAS DE RECOMENDAÇÃO **4**

2.1 Filtragem Baseada em Conteúdo 4

2.2 Filtragem Colaborativa 6

2.2.1 Filtragem Baseadas em Memória 6

2.2.2 Filtragem Baseada em Modelo 7

2.3 Filtragem Híbrida 8

3 TRABALHOS RELACIONADOS **9**

3.1 Recomendação de *Tags* 9

3.2 Recomendação de Artigos Científicos 11

4 FUNDAMENTAÇÃO TEÓRICA **12**

4.1 Vetores de Palavras 12

4.1.1 *Skip-Gram (SG)* 13

4.1.2 *Continuous Bag of Words (CBOW)* 13

4.2 Redes Complexas 14

4.2.1 Métricas de Redes Complexas 15

4.2.1.1 *Preferential Attachment* 16
4.2.1.2 *Clustering* 16
4.2.1.3 *Betweenness Centrality* 16
4.2.1.4 *Edge Betweenness Centrality* 17
4.2.1.5 *Betweenness Centrality Subset* 17
4.2.1.6 *Edge Betweenness Centrality Subset* 17
4.2.1.7 *Degree Centrality* 17
4.2.1.8 *Closeness Centrality* 18
4.2.1.9 *Eigenvector Centrality* 18
4.2.1.10 *Katz Centrality* 18
4.2.1.11 *Global Degree* 18
4.2.1.12 *Local Degree* 19
4.2.1.13 *Random* 19
4.2.1.14 TF-IDF 19
4.3 *Máquinas de Vetor Suporte (SVM)* 19
4.4 Regressão Logística 21
4.4.1 Estimando os parâmetros β_0 e β_1 23
4.5 *Random Forest* 23
4.5.1 Classificador 23
4.5.2 Árvores de Decisão 23
4.5.3 Métodos Ensemble 24
4.6 Alocação Latente de Dirichlet – LDA 25
4.7 Regressão Colaborativa de Tópicos – CTR 26

5 RECOMENDAÇÃO DE TAGS 28

5.1 Etapas da Metodologia Proposta 28

5.1.1 Normalização 28
5.1.2 Remoção de Dados Não relevantes 29
5.1.3 Construção de Vetores de Palavras 29
5.1.4 Representação da Rede Complexa 30
5.1.5 Cálculo das Métricas 30
5.1.6 Recomendação de Tags 30
5.2 Experimentos e Resultados 31
5.2.1 Aquisição dos Dados 31
5.2.2 Filtragem e Tratamento dos Dados 31
5.2.3 Construção de Vetores de Palavras 32
5.2.4 Representação da Rede Complexa 33
5.2.5 Aprendizado de máquina não supervisionado 33
5.2.6 Aprendizado de Máquina 33
5.2.7 Resultados Obtidos 34

6 RECOMENDAÇÃO DE ARTIGOS CIENTÍFICOS 39

6.1 Etapas da Metodologia Proposta 39
6.1.1 Conjunto de Dados 39
6.1.2 Vetores de Palavras 40
6.1.3 Representação da Rede Complexa 40
6.1.4 Cálculo das Métricas 41
6.1.5 Expansão dos Documentos 41
6.1.6 Separação do Conjunto de Dados 42
6.1.7 Experimentos e Resultados 42
6.1.8 *Cold Start* 44
6.1.9 Expansão em 5% 47

6.1.10 Comparação com Outros Métodos 48

7 CONCLUSÕES E TRABALHOS FUTUROS 52

7.1 Recomendação de *Tags* 52

7.2 Recomendação de Artigos Científicos 53

7.2.1 Trabalhos Futuros 53

Lista de Figuras

2.1 Principais técnicas de Recomendação 5

4.1 Rede Neural para o modelo Skip-Gran (Fonte http://alexminnaar.com/) 14

4.2 Rede Neural para o modelo CBOW (Fonte http://alexminnaar.com/) 14

4.3 SVM possíveis soluções 20

4.4 SVM Hiperplano Ótimo 20

4.5 Função Logística 22

4.6 Exemplo de tópicos 25

4.7 Modelo gráfico LDA em notação de placa 26

4.8 Processo gerador do LDA 26

4.9 Modelo gráfico CTR em notação de placa 27

4.10 Processo gerador do CTR 27

5.1 Visão geral da metodologia de recomendação de *tags* proposta neste trabalho. . . 29

5.2 Exemplo de filme após normalização e tratamento 32

5.3 Resultado obtido com métricas individualmente 34

5.4 Amostra do filme Toy Story 36

5.5 Viés no conjunto de dados 37

5.6 Desempenho dos métodos 38

6.1 Visão geral da metodologia de recomendação de artigos científicos 40

6.2 Processo gerador para o documento expandido 42

6.3 Comparação entre métricas quando P = 1 e P = 10 43

6.4 Degree centrality e Regular 45

6.5 Ganho Percentual em Relação ao Conjunto Regular quando P = 1 45

6.6 Ganho Percentual em Relação ao Conjunto Regular quando P = 10 46

6.7 Comparação de Percentual de Expansão 46

6.8 Comparação de Percentual de Expansão: ganho obtido em relação ao conjunto regular 47

6.9 Comparação entre Percentuais de Expansão 47

6.10 Expansão de 5% 48

6.11 Modelos MyMediaLite 49

6.12 Comparação CTR e MyMediaLite 50

Lista de Tabelas

5.1 Resultados obtidos pelos métodos *SVM*, *Random Forest* e *Regressão Logística*. . 37

6.1 Comparação entre métricas quando P = 1 e P = 10 44

6.2 Revocação Média por Conjunto de Dados 45

6.3 Comparação de Percentual de Expansão 51

Capítulo 1

INTRODUÇÃO

Com o rápido crescimento e a grande disponibilidade de serviços na internet, as pessoas têm acesso a grandes quantidades de conteúdo on-line, como músicas, filmes, notícias e artigos [1]. Embora esse crescimento tenha permitido que os usuários da internet consumam uma grande quantidade de recursos com apenas um clique, também tornou mais difícil para os mesmos encontrarem informações relevantes para seus interesses. O número de opções de uma pessoa frente a um determinado serviço é enorme e, em muitas das vezes, as pessoas possuem pouca ou quase nenhuma experiência pessoal para escolher uma dentre as inúmeras opções que são oferecidas. Neste contexto, sistemas de recomendação tornaram-se um tema não só popular mas também desafiador.

Os usuários podem não estar cientes da existência de filmes de seu interesse e que gostariam de assistir. Portanto, a recomendação adequada de um filme, por exemplo, pode tanto conquistar o usuário como perdê-lo. Sistemas que conseguem fazer o melhor uso possível da informação conquistam o usuário e aumentam valor de vendas para as empresas através da qualidade das recomendações oferecidas. Dessa forma, os sistemas de recomendação estão se tornando cada vez mais importantes para atrair usuários e tornar efetivo o uso da informação disponível [2].

A estrutura básica de um sistema de recomendação tem como objetivo identificar como os relacionamentos entre usuários e itens, elementos básicos de todo sistema de recomendação, se assemelham e realizam a recomendação. Para que seja possível calcular a relevância de cada tópico (um determinado produto ou conteúdo) é preciso que haja uma forma de analisar o comportamento do usuário obtendo um *feedback* de suas preferências.

Os sistemas de recomendações são classificados em categorias relativas às abordagens usadas para a recomendação [3]:

- abordagens baseadas em conteúdo: os itens recomendados são similares aos itens que o usuário mostrou preferência no passado;
- abordagem colaborativa: os itens recomendados são escolhidos por pessoas com preferências similares às dos usuários;
- abordagens híbridas: combinam técnicas das duas abordagens anteriores para realizar a recomendação.

Este trabalho aborda duas vertentes de um sistema de recomendação: a recomendação de *tags* [1] e a recomendação de artigos científicos.

O processo conhecido como *tagging* surge como uma das melhores maneiras de associar metadados a objetos. Em sistemas de informação, uma *tag* é uma palavra-chave ou termo atribuído a uma informação (como uma imagem digital, registros de banco de dados ou um arquivo de computador). Esse tipo de metadado ajuda a descrever um item e permite que ele seja encontrado novamente através de uma pesquisa. As *tags* geralmente são escolhidas informalmente e pessoalmente pelo criador do item ou por outras pessoas. *Tagging* é útil em muitos aspectos. Por exemplo, as *tags* ajudam a descrever o conteúdo de um documento, foto, vídeo, etc., revelando seu significado semântico. As *tags* não só enfatizam os termos-chave de um documento, mas também representam informações adicionais que não necessariamente estão presentes no texto desse documento [4]. Estudos recentes demonstraram que, entre outras características textuais, tais como título, descrição e comentários dos usuários, as *tags* são as mais eficazes para dar suporte a serviços de Recuperação de Informações como a busca [5], classificação automática [6] e recomendação de conteúdo [7]. A marcação ou *tagging* foi popularizada por sites associados à Web 2.0 e é uma característica importante de muitos serviços da Web 2.0 atualmente.

De acordo com [8], o problema de recomendação de *tags* representa um exemplo de um sistema de recomendações baseados em conteúdo e pode ser definido como se segue. Dado um objeto e um conjunto de recursos de texto, pretende-se gerar um conjunto de possíveis recomendações, ordenadas de acordo com sua relevância para o objeto. Neste trabalho, a recomendação de *tags* é feita de duas maneiras. Inicialmente para determinar a relevância de uma *tag* em um documento o vocabulário deste documento é representado usando um grafo gerado a partir de um modelo de vetores de palavras. As métricas de rede complexas podem então ser aplicadas para calcular a relevância das palavras do documento e em seguida as palavras mais relevantes são selecionadas como as *tags* a serem recomendadas. Alternativamente, as *tags* podem ser recomendadas usando as métricas de redes complexas como dados de entrada para técnicas de aprendizado de máquinas.

Outra forma de recomendação proposta nesse trabalho é a recomendação de artigos científicos. Devido a enorme quantidade de artigos científicos existentes online, sistemas de recomendação de artigos científicos podem ser utilizados para filtrar e sugerir conteúdos de interesse em uma pesquisa bibliográfica. Existem mecanismos de busca especializados em literatura científica tais como o Google Scholar e o CiteSeer e serviços de buscas disponibilizados pelas editoras e sociedades cientificas para realizar buscas em suas coleções tais como IEEE e ACM. Nestes sistemas de buscas, os usuários realizam uma consulta para encontrar documentos de interesse e a responsabilidade de usar os termos corretos para recuperar os documentos relevantes é toda do usuário.

O problema de recomendação de artigos científicos é um exemplo de sistemas de recomendações baseados em abordagem colaborativa. O problema que será abordado neste trabalho pretende, para cada usuário, recomendar artigos que não estão em sua biblioteca, mas são de potencial interesse para o mesmo. As informações sobre os artigos disponíveis, tais como título e resumos, após serem utilizados como entrada para um modelo de vetor de palavras, serão usadas para gerar uma rede complexa e, usando as propriedades dessa rede complexa, palavras deste grafo são

[1] Tags e outros termos em inglês que são utilizados frequentemente, sem tradução para o português, em artigos sobre processamento de linguagem natural, redes neurais ou aprendizado de máquina serão utilizados neste trabalho sem tradução para a língua portuguesa.

selecionadas para expandir um documento. Para extrair os tópicos e as recomendações, técnicas de processamento de linguagem natural serão aplicadas.

1.1 Objetivos

Este trabalho tem como o objetivo geral criar sistemas de recomendações que utilizam um modelo de vetores de palavras de um conjunto de dados para gerar uma rede complexa. Através das propriedades das redes complexas, as recomendações são realizadas. Estes sistemas de recomendação serão aplicados em recomendações de *tags* e recomendações de artigos científicos.

Os objetivos específicos deste trabalho são:

1. Uma revisão da literatura sobre os métodos existentes para recomendação de *tags* e sistemas de recomendação.
2. O desenvolvimento de uma nova metodologia que explora vetores de palavras para criar um grafo que posteriormente terá suas propriedades exploradas para a recomendação de *tags*. O processo de recomendação de *tag* é totalmente baseada em propriedades de redes complexas.
3. O desenvolvimento de uma nova metodologia que explora técnicas de aprendizado de máquina para recomendação de *tag*. Neste caso as propriedades da rede complexa geradas na metodologia anterior são usadas como entrada das técnicas de aprendizado de máquina e usadas no processo de recomendação.
4. O desenvolvimento de uma metodologia que utiliza propriedades de redes complexas e metodologias de aprendizado de máquinas na construção de sistemas de recomendação de artigos científicos.
5. Avaliação experimental das estratégias propostas.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2 os conceitos básicos relacionados aos Sistemas de Recomendação são definidos e discutidos. O Capítulo 3 apresenta os trabalhos relacionados ao problema de recomendação de *Tags* e ao problema de recomendação de artigos científicos. O Capítulo 4 apresenta os principais métodos e técnicas que serão aplicados no desenvolvimento deste trabalho. O Capítulo 5 apresenta as duas propostas para recomendação *Tags* e discute os resultados obtidos. A metodologia de recomendação de artigos científicos proposta será apresentada no Capítulo 6. Os resultados obtidos serão apresentados e discutidos neste capítulo. Finalmente, o Capítulo 7 conclui este trabalho e apresenta trabalhos futuros.

Capítulo 2

SISTEMAS DE RECOMENDAÇÃO

Na internet, a quantidade de informação existente é gigantesca e a sobrecarga de informações representa um problema aos usuários. É necessário filtrar, priorizar e fornecer informações relevantes de maneira eficiente a fim de aliviar a sobrecarga de informações. Os sistemas de recomendação resolvem esse problema pesquisando um grande volume de informações para fornecer aos usuários conteúdo e serviços personalizados.

Um sistema de recomendação é uma tecnologia em que itens (produtos, filmes, eventos, artigos, etc.) são recomendados para usuários (clientes, usuários de aplicativos, leitores). Os sistemas de recomendação aplicam técnicas de mineração de dados e algoritmos de previsão para prever o interesse dos usuários em informações, produtos e serviços entre a enorme quantidade de itens disponíveis. Normalmente, há muitos itens e muitos usuários presentes no ambiente, tornando o problema difícil de se resolver. Devido a grande escala de informação disponível, os desafios de um sistema de recomendação são: produzir uma recomendação precisa, lidar com muitas recomendações de forma eficiente e lidar com o grande crescimento do número de participantes no sistema [9].

O uso de técnicas de recomendação eficientes e adequadas é muito importante para um sistema que fornecerá uma recomendação de alta qualidade e útil para seus usuários. Isso explica a importância de se entender os recursos e as diferenças entre as metodologias de recomendação. A Figura 2.1 mostra as diferentes técnicas utilizadas em sistemas de recomendação. Os tipos de sistemas de recomendação são classificados em três categorias, filtragem baseada em conteúdo, filtragem colaborativa e filtragem híbrida, e dizem respeito a forma na qual a recomendação é feita.

2.1 Filtragem Baseada em Conteúdo

Um sistema de recomendação baseado em filtragem de conteúdo tem por objetivo recomendar itens semelhantes aos itens que o usuário selecionou no passado. Neste tipo de sistema, o problema de recomendação é tratado como uma busca por itens relacionados, filtrando e analisando o conteúdo e a descrição de itens para identificar os que possam interessar ao usuário [10]. Os

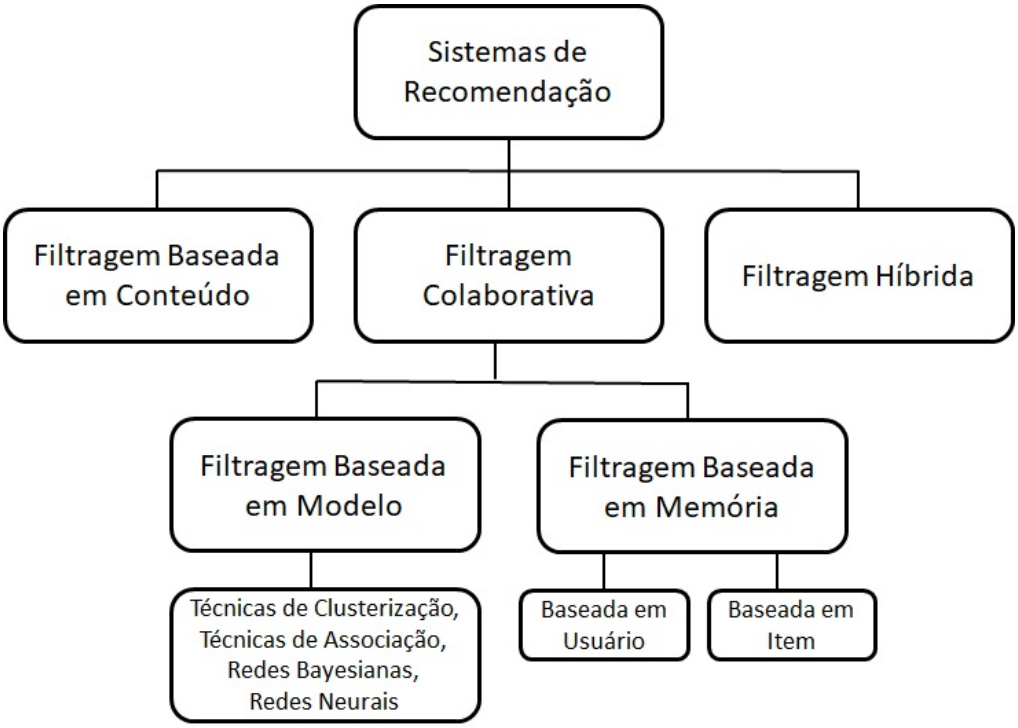

Figura 2.1: Principais técnicas de Recomendação

itens selecionados para recomendação são itens com os quais o conteúdo se relaciona com as preferências do usuário [11][12]. Por exemplo, sempre que um usuário avalia um item, o sistema de recomendação cria uma consulta para encontrar outros itens populares do mesmo autor, artista, diretor, *tags* ou assuntos similares.

As técnicas de recomendação baseadas em filtragem de conteúdo são dependentes do domínio e enfatizam a análise dos atributos dos itens para gerar recomendações. Quando documentos como páginas da Web, publicações, filmes e notícias são recomendados, as técnicas de recomendação baseadas em filtragem de conteúdo são as mais bem-sucedidas. A recomendação é feita com base no perfil do usuário usando recursos extraídos do conteúdo dos itens que o usuário avaliou no passado [11]. Apenas os itens que são relacionados aos itens que foram positivamente classificados pelo usuário são recomendados. As técnicas de recomendação baseadas em filtragem de conteúdo utilizam diferentes tipos de modelos para encontrar semelhanças entre os documentos, a fim de gerar recomendações relevantes. Por exemplo, modelos probabilísticos tais como Classificadores Beysianos Ingênuos ou Simples [13], Árvores de Decisão [14] ou Redes Neurais [15] podem ser usados para modelar a relação entre documentos diferentes dentro de um conjunto. Estas técnicas fazem recomendações aprendendo o modelo subjacente com análises estatísticas ou técnicas de aprendizado de máquina. As técnicas de filtragem baseadas em conteúdo não precisam do perfil de outros usuários, já que eles não influenciam a recomendação. Além disso, se o perfil do usuário para o qual se está recomendando mudar, estas técnicas ainda terão o potencial de ajustar suas recomendações.

As técnicas de recomendação baseadas em filtragem de conteúdo têm a capacidade de recomendar novos itens, mesmo se não houver classificações fornecidas pelos usuários. Portanto, mesmo que o banco de dados não contenha preferências do usuário, a precisão da recomendação não

será afetada. Além disso, se as preferências do usuário mudarem, elas têm a capacidade de se ajustarem. Elas podem gerenciar situações em que usuários diferentes não compartilham os mesmos itens, mas apenas itens idênticos de acordo com seus recursos. Os usuários podem obter recomendações sem compartilhar seu perfil e isso garante a privacidade [16]. As técnicas de recomendação baseadas em filtragem de conteúdo também podem fornecer explicações sobre como as recomendações são geradas para os usuários. No entanto, estas técnicas sofrem vários problemas, como discutido na literatura [3]. Estas técnicas dependem dos metadados dos itens exigindo uma descrição rica dos itens e um perfil de usuário muito bem organizado antes que uma recomendação possa ser feita. Assim, a eficácia destas técnicas depende da disponibilidade de dados descritivos. A superespecialização de conteúdo [17] é outro problema. Os usuários estão restritos a receber recomendações semelhantes aos itens já definidos em seus perfis.

2.2 Filtragem Colaborativa

A filtragem colaborativa, também conhecida como filtragem social, filtra informações usando as recomendações de outras pessoas. Baseia-se na ideia de que as pessoas que concordaram com a avaliação de certos itens no passado provavelmente concordarão novamente no futuro. As técnicas de filtragem colaborativa são técnicas de recomendação independente de domínio, desenvolvidas para conteúdo que não pode ser adequadamente descrito por metadados, como músicas e filmes. Estas técnicas criam um banco de dados (matriz de itens por usuário) dos itens preferidos por usuários. Em seguida, associa os usuários a interesses e preferências relevantes calculando as semelhanças entre seus perfis para fazer recomendações [18]. Um usuário recebe recomendações para os itens que ele não avaliou antes, mas que já foram avaliados positivamente pelos usuários em sua vizinhança. As recomendações que são produzidas pela filtragem colaborativa podem ser de previsão ou recomendação. A previsão é um valor numérico, R_{ij}, expressando a pontuação prevista do item j para o usuário i, enquanto a recomendação é uma lista dos N principais itens que o usuário mais gosta. As técnica de filtragem colaborativa podem ser divididas em duas categorias: **baseado em memória** e **baseado em modelo** [12] [19].

2.2.1 Filtragem Baseadas em Memória

Os itens que já foram classificados por um usuário anteriormente desempenham um papel relevante na busca por um vizinho que compartilhe suas preferências [20]. Uma vez que um vizinho de um usuário é encontrado, diferentes algoritmos podem ser usados para combinar as preferências de vizinhos para gerar recomendações. Devido à eficácia dessas técnicas, elas obtiveram sucesso generalizado em aplicações da vida real. A filtragem colaborativa baseada em memória pode ser obtida de duas maneiras diferentes: através de técnicas baseadas em usuário ou em técnicas baseadas em itens.

A técnica de filtragem colaborativa baseada no usuário calcula a similaridade entre usuários comparando suas avaliações do mesmo item. A classificação prevista para um item j, pelo usuário i, é calculada como uma média ponderada das classificações do item j, dada por usuários semelhantes ao usuário i, onde os pesos são as semelhanças desses usuários com o item j.

A técnica de filtragem colaborativa baseadas em itens calcula as recomendações usando a similaridade entre os itens e não a similaridade entre os usuários. Primeiro, esta técnica executa um estágio de construção, encontrando a semelhança entre todos os pares de itens. Essa função de similaridade pode assumir várias formas, como a correlação entre classificações ou o cosseno desses vetores de classificação. Assim como na filtragem colaborativa baseada no usuário, as funções de similaridade podem usar classificações normalizadas corrigindo, por exemplo, a classificação média de cada usuário.

A filtragem colaborativa baseada em itens está sujeita a menos erros do que a filtragem colaborativa baseada no usuário. Além disso, seu modelo menos dinâmico pode ser computado com menos frequência e armazenado em uma matriz menor, de modo que o desempenho do sistema baseado em itens é geralmente melhor que o dos sistemas baseados no usuário [21].

2.2.2 Filtragem Baseada em Modelo

Nessa abordagem, os modelos são desenvolvidos usando algoritmos de mineração de dados e de aprendizado de máquina para prever a classificação dos usuários para itens ainda não classificados. Existem muitos algoritmos de filtragem colaborativa baseados em modelos: Redes Bayesianas, modelos de agrupamento, modelos semânticos latentes, como decomposição de valores singulares e modelos baseados em processos de decisão de Markov [22]. Estes modelos podem permitir que o sistema aprenda a reconhecer padrões complexos com base nos dados de treinamento e fazer previsões inteligentes, para dados de teste ou dados do mundo real, baseado nos modelos aprendidos. Algoritmos de filtragem colaborataiva baseados em modelo foram investigados para solucionar as deficiências dos algoritmos de filtragem colaborativa baseados em memória [19] [23].

A filtragem colaborativa tem algumas vantagens importantes em relação a filtragem baseada em conteúdo, pois ela pode ser executado em domínios nos quais não há muito conteúdo associado a itens e nos quais o conteúdo do sistema de recomendação é difícil de analisar, como opiniões e idéias. Além disso, as técnicas de filtragem colaborativa tem a capacidade de fornecer recomendações inesperadas, podendo recomendar itens que sejam relevantes para o usuário, mesmo sem o conteúdo estar no perfil do usuário [24]. Apesar do sucesso das técnicas de filtragem colaborativa, seu uso difundido revelou alguns problemas potenciais tais como:

1. **Cold Start** - Refere-se a situação em que um sistema de recomendação não tem informações adequadas sobre um usuário ou um item para fazer previsões relevantes [25]. Este é um dos principais problemas que reduzem o desempenho do sistema de recomendação. O perfil desse novo usuário ou item estará vazio, pois ele não classificou nenhum item e então suas preferências não são conhecidas pelo sistema.

2. **Esparcidade** - Esse é o problema que ocorre como resultado da falta de informações suficientes, quando apenas alguns dos itens disponíveis em um banco de dados são classificados pelos usuários [1]. Isso sempre leva a uma matriz esparsa de itens por usuário, o que tem como consequência a incapacidade de localizar vizinhos bem-sucedidos que, por sua vez, leva à geração de recomendações fracas.

2.3 Filtragem Híbrida

As técnicas de filtragem híbrida combinam diferentes técnicas de recomendação para obter uma melhor otimização do sistema, a fim de evitar as limitações e os problemas dos sistemas de recomendação tradicionais [26]. A idéia por trás das técnicas híbridas é que uma combinação de algoritmos fornecerá recomendações mais precisas e efetivas do que um único algoritmo, já que as desvantagens de um algoritmo podem ser superadas por outro algoritmo [24]. Usar várias técnicas de recomendação pode eliminar as fraquezas de uma técnica individual em um modelo combinado. A combinação de abordagens pode ser feita de qualquer uma das seguintes formas:

1. Implementação separada de algoritmos e combinação de resultados.
2. Utilização de alguma filtragem baseada em conteúdo em abordagem colaborativa.
3. Utilização de alguma filtragem colaborativa em abordagem baseada em conteúdo.
4. Criação um sistema de recomendação unificado que utiliza ambas as abordagens.

Capítulo 3

TRABALHOS RELACIONADOS

Neste capítulo, os trabalhos relacionados ao problema de recomendação de *Tags* e ao problema de recomendação de artigos científicos são apresentados e discutidos. Uma análise crítica das semelhanças e diferenças entre o trabalho proposto nesta dissertação e os trabalhos listados será realizada.

3.1 Recomendação de *Tags*

Tags são palavras-chave livremente atribuídas pelos usuários para descrever conteúdo web e tornaram-se muito populares em aplicações Web 2.0 devido à grande facilidade existente para que seus usuários criem e descrevam seu próprio conteúdo. Este aumento de popularidade levou a uma vasta literatura sobre métodos de recomendação de palavras-chave ou *tags*.

Embora exista uma vasta literatura sobre métodos de recomendação de *tags*, são poucos os métodos que exploram propriedades de redes complexas para automatizar esta tarefa. Abaixo são citados os principais trabalhos que utilizam propriedades de redes complexas, em alguma etapa do desenvolvimento, para recomendação de *tags*.

Uma abordagem baseada em clusters foi proposta em [27] para agregar *tags* semanticamente relacionadas em clusters semelhantes. As *tags* são representadas como grafos onde cada vértice é uma *tag* e a aresta entre dois vértices corresponde à sua co-ocorrência nos mesmos documentos. As *tags* no mesmo cluster são recomendadas aos usuários com base em medidas de similaridades existentes.

Um método chamado *FolkRank*, baseado no algoritmo *PageRank*, é proposto em [28]. A ideia do método é que um objeto que recebe *tags* relevantes de usuários importantes também se torna importante, ou seja, uma boa fonte para uma recomendação. Reciprocamente, uma *tags* é relevante se está associada a objetos importantes por usuários importantes. Assim, um grafo de reforço mútuo é construído, permitindo a pontuação e recomendação de *tags*.

Em [29] os autores aprimoraram sua metodologia para recomendação de *tags* com base em grafos

explorando links sociais entre usuários. As relações sociais entre usuários e as similaridades de conteúdo entre diferentes fontes são expressas usando um grafo. Usando metodologia de passeio aleatório para similaridades entre todos os objetos do grafo, um algoritmo combinando informação colaborativa e *tags* personalizadas é apresnetado para realizar a recomendação.

Um método que recomenda a um usuário u as *tags* atribuídas pelos k usuários mais similares a u é proposto em [30]. Neste trabalho, os autores criaram um grafo onde os vértices são usuários e cada aresta conecta dois usuários que compartilham *tags*. Medidas de similaridades, tais como *Jaccard coefficient*, são usadas para ponderar o grafo.

Em [31], os autores propuseram usar a divergência de *Kullback-Leibler* para estimar a semelhança entre dois usuários distintos. Neste trabalho os autores propuseram um método probabilístico que explora o vocabulário e os objetos do usuário alvo bem como o vocabulário de usuários similares.

Utilizando o *OptRank* como estratégia de otimização para calcular os pesos dos vértices, em [32] os autores modelaram a folksonomia, que é uma maneira de indexar informações, como um grafo heterogêneo contendo *tags*, usuários e objetos como vértices.

Em [33], os autores exploraram técnicas de filtragem colaborativa e de recomendação de *tags* baseadas em conteúdo. O primeiro explora o comportamento do usuário e da comunidade para produzir recomendações de *tags*, enquanto que o último explora algumas heurísticas para extrair *tags* diretamente do conteúdo textual do objeto.

Em [34] o problema de recomendação de *tags* é modelado como um problema de "consulta e classificação". Um algoritmo de *ranking* baseado em grafos para entidades relacionadas, que são *tags*, usuários e documentos é proposto. Quando um usuário envia uma solicitação de *tag*, o documento e o usuário são tratados como parte da consulta. As *tags* são classificadas pelo algoritmo de *ranking* que por sua vez leva em consideração a relevância do documento e a preferência do usuário.

Em [35] os autores abordaram não apenas o problema de recomendar *tags*, mas também de prever diferentes tipos de relacionamentos (como relações entre usuários, comentários, itens e *links* sociais entre usuários). Ao explorar um modelo de fator latente generalizado e uma inferência bayesiana, os autores descobriram que a conexão de comentários e *tags* dentro do mesmo modelo permite o reforço mútuo e melhora a precisão da predição.

Os autores propõem em [36] uma metodologia que visa acelerar o tempo de execução de recomendadores de *tags* baseados em grafos para redes de folksonomia em grande escala. Um algoritmo de detecção de comunidade em redes multiplex é aplicado para aumentar o hipergrafo representando uma folksonomia.

Neste trabalho propomos uma metodologia totalmente baseada em propriedades de redes complexas para a tarefa de recomendação de *tags*. Os trabalhos listados acima utilizam esta metodologia apenas de forma auxiliar no processo de recomendação. Neste presente trabalho, a metodologia proposta utiliza como base um grafo onde os vértices representam as palavras do vocabulário de um corpus e o peso das arestas indicam o quão semelhantes tais palavras são entre si. Posteriormente este grafo terá suas propriedades exploradas para a recomendação de *tags* diretamente usando as métricas de redes ou como entrada para uma abordagem de aprendizado de máquina.

3.2 Recomendação de Artigos Científicos

Os sistemas de recomendação estão se tornando cada vez mais populares e são utilizados em diversas áreas [1]. Os trabalhos de pesquisa mais comuns estão focados em estudos de recomendação de filmes [37], no entanto, um grande volume da literatura existente para sistemas de recomendação está centrado em diferentes tópicos, como música [38], televisão [39], documentos [40][41], livros [42], *e-learning* [43], *e-comerce* [44] e pesquisa na web [45], entre outras.

Mesmo a literatura sobre sistemas de recomendação sendo vasta, não foi possível localizar um grande número de trabalhos onde propriedades de redes complexas foram utilizadas para auxiliar no processo de recomendação. Abaixo são citados os principais trabalhos que utilizam propriedades de redes complexas, em alguma etapa, no processo de recomendação em geral.

Em [46], os autores representam o conjunto de dados, NetFlix dataset, como um grafo bipartido, onde de um lado estão os usuários e do outro estão os itens a serem recomendados. Utilizando uma métrica baseada em *preferential attachment*, a compatibilidade entre o item i e o usuário j é calculada. Com base nessa compatibilidade, uma lista de classificação é criada e então os itens, que realmente foram selecionados pelo usuário, são pesquisados nessa lista e uma pontuação lhes é atribuída de acordo com a sua posição.

É empregado em [47] uma estrutura de rede de compra que envolve as transações de compra dos usuários. Nesse caso, cada nó representa um usuário e uma aresta entre dois nós é criada quando os usuários correspondentes adquirem a mesma transação. Intuitivamente, o acordo se torna mais provável de ser relevante para um usuário em particular se ele estiver próximo a usuários que também compraram esse contrato recentemente.

Em [48], os autores modelam e caracterizam um *e-market* como uma rede complexa e usam a estrutura de rede para investigar a credibilidade dos vendedores. Os autores propõem um novo algoritmo, baseado na estrutura da rede de negociação, para recomendar se o vendedor é confiável ou não. Um *e-market* fornece um meio *on-line* para realizar transações entre compradores e vendedores, potencialmente apoiando todas as etapas de todo o processo de atendimento de pedidos e obviamente a credibilidade é um requisito importante para o seu sucesso.

Diferentemente dos trabalhos listados nessa seção, que utilizam uma rede complexa como principal método de recomendação, o trabalho aqui proposto utiliza a rede complexa apenas como método auxiliar do processo de recomendação de artigos científicos. A rede complexa proposta aqui é criada para selecionar palavras que posteriormente serão adicionadas ao conjunto de dados original para expandi-lo. O conjunto expandido será processado e avaliado pelos modelos utilizados em [2] para então ser feita a recomendação. As técnicas que serão aplicadas são *Latent Dirichlet Allocation* (LDA) e *Collaborative Topic Regression* (CTR) e serão descritas no próximo capítulo.

Capítulo 4

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os principais conceitos utilizados no desenvolvimento deste trabalho. O conceito de *embedding* de palavras, ou como as palavras são representadas por meio de vetores, é apresentado e discutido. As redes complexas também serão apresentadas e as métricas para tais redes são apresentadas e discutidas. Técnicas de aprendizado de máquinas, tais como Máquinas de Vetor Suporte, Regressão Logística e Random Forest, serão apresentadas e discutidas. As técnicas de Processamento de Linguagem Natural, tais como Alocação Latente de Dirichlet e Regressão Colaborativa de Tópicos, serão apresentadas.

4.1 Vetores de Palavras

A maioria das técnicas tradicionais de processamento de linguagem natural consideram as palavras como unidades atômicas de processamento [49]. Embora esta abordagem tenha produzido resultados em muitos domínios, ela pode ser melhorada com a exploração de significados e semelhanças de palavras, especialmente para termos de uso ocasional. A representação distribuída de palavras foi concebida para capturar a semântica, codificando cada palavra e seu contexto em um espaço vetorial.

As representações de vetores de palavras são consistentes com a álgebra vetorial, no sentido de que algumas operações no domínio vetorial (como a soma ou subtração) devem manter algum grau de consistência com manipulações semânticas semelhantes. Por exemplo, em [50], são mostrados modelos de vetores de palavras em que o vetor resultante da operação

$$\overrightarrow{V}(rei) - \overrightarrow{V}(homen) + \overrightarrow{V}(mulher)$$

produz um vetor que é próximo ao vetor

$$\overrightarrow{V}(rainha)$$

.

De fato, os vetores de palavras capturam simetrias sintáticas e semânticas significativas de uma maneira muito simples. Especificamente, tais simetrias são observadas como um deslocamento

constante entre pares de vetores de palavras que compartilham um relacionamento particular. Por exemplo, se focamos na relação singular/plural, observamos que

$$\overrightarrow{V}(maça) - \overrightarrow{V}(maças) \approx \overrightarrow{V}(carro) - \overrightarrow{V}(carros)$$

e

$$\overrightarrow{V}(família) - \overrightarrow{V}(famílias) \approx \overrightarrow{V}(carro) - \overrightarrow{V}(carros)$$

e assim por diante.

Existem vários trabalhos publicados sobre fundamentos e aplicações de vetores de palavras e aqui citamos dois modelos: *Skip-Gram (SG)* [50] e *Continuous Bag of Words* (CBOW) [50], que são os modelos codificados pelo programa *word2vec*, que é utilizado em nossos experimentos. Estes dois modelos são detalhados abaixo.

4.1.1 *Skip-Gram (SG)*

O modelo *Skip-Gram (SG)* é uma rede neural artificial cuja arquitetura é mostrada na Figura 4.1. O algoritmo usa uma codificação denominada *one-hot* para representar palavras em sua entrada e saída. Esta codificação representa palavras como um vetor binário de tamanho S, onde apenas um bit é ligado (definido como 1). Em nosso cenário, o tamanho S é igual ao tamanho do vocabulário V (número de palavras distintas no corpus) e para cada palavra w, a representação *one-hot* correspondente é um vetor binário de tamanho V com apenas o *bit* associado a palavra w ligado.

A entrada do modelo é uma palavra e a saída é um conjunto de palavras dentro de uma janela de contexto de tamanho c. Portanto, a tarefa da rede neural é inferir o contexto para uma determinada palavra. Este mapeamento possibilita a transformação de qualquer corpus de texto em um conjunto de dados de treinamento para um problema de classificação.

Todas as palavras na camada de saída compartilham a matriz W_{NxV} de pesos que produz uma distribuição multinominal usando a função *softmax* [1]. Vetores de palavras de tamanho N (que é um parâmetro e define o tamanho da camada oculta) são extraídos da matriz W'_{VxN} de pesos, um vetor por linha. Os parâmetros são aprendidos utilizando o algoritmo gradiente descendente estocástico e *backpropagation*.

4.1.2 *Continuous Bag of Words (CBOW)*

O modelo CBOW, apresentado na Figura 4.2, também é uma rede neural artificial que aprende a tarefa inversa do modelo SG descrito acima. A tarefa desta rede é, portanto, inferir uma palavra dada o seu contexto de tamanho C. O processo de aprendizagem é semelhante ao modelo SG. As camadas de entrada e a camada oculta são conectadas por uma matriz de pesos compartilhados W'_{VxN}, a partir da qual todos os vetores de palavras são extraídos, um vetor por linha.

[1] A função *softmax* calcula a distribuição de probabilidades de um evento sobre n eventos distintos. De maneira geral, esta função calcula as probabilidades de cada classe alvo sobre todas as classes possíveis.

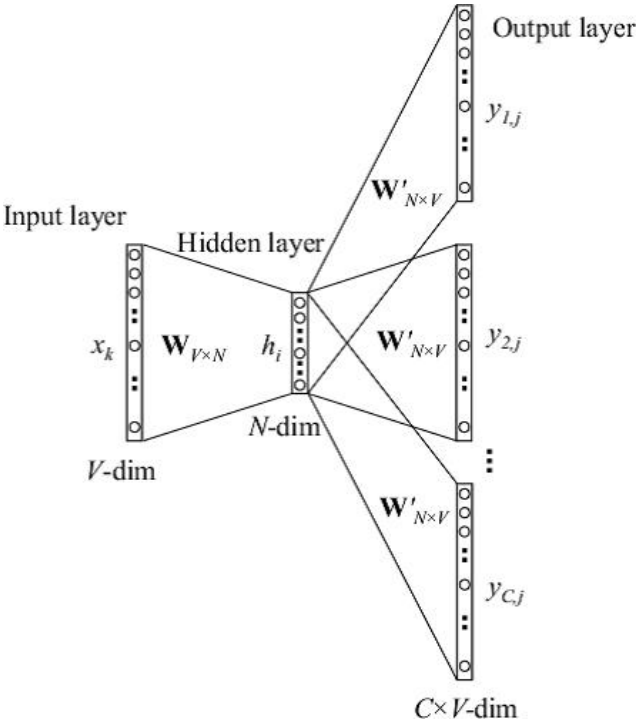

Figura 4.1: Rede Neural para o modelo Skip-Gran (Fonte http://alexminnaar.com/)

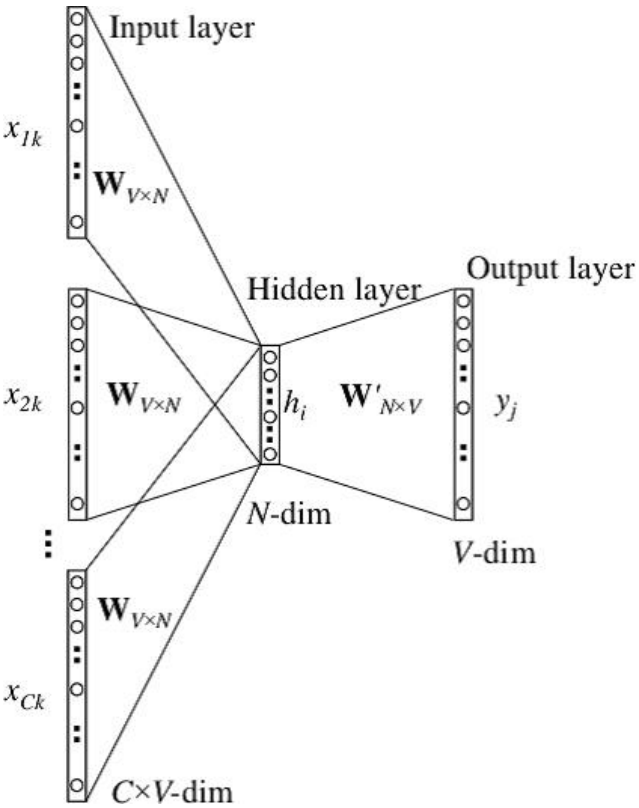

Figura 4.2: Rede Neural para o modelo CBOW (Fonte http://alexminnaar.com/)

4.2 Redes Complexas

O estudo de redes complexas é um tema interdisciplinar que abrange diversas áreas de conhecimento, tais como a ciência da computação, matemática, física, biologia e sociologia. O termo

redes complexas refere-se a um grafo que apresenta uma estrutura topográfica não trivial, composto por um conjunto de vértices que são interligados por meio de arestas [51]. O estudo de redes na forma de grafos é um dos pilares da matemática discreta e teve início em 1735, quando Euler propôs uma solução para o problema das pontes de Königsberg [52], originando a teoria dos grafos.

Uma rede complexa pode ser definida como um grafo que é composto por, relativamente, muitos nós mutuamente relacionados. Também pode ser definida como uma rede que possui características topológicas não observáveis em redes simples, como as aleatórias, mas que ocorrem frequentemente em grafos modelados a partir de sistemas reais [53]. Vários eventos do mundo real podem ser representados por meio de redes complexas para resolução de problemas diversos. É possível modelar toda a estrutura física de uma rede de computadores tal como a internet, por exemplo. Nesse caso, os computadores conectados à internet são os vértices enquanto que os cabos e outros meios de transmissão representam as arestas do grafo. Outras analogias podem ser também utilizadas, tais como o conteúdo de páginas WEB — *World Wide Web* — relações sociais entre grupos de pessoas, redes organizacionais, redes neurais e muitas outras.

Os estudos das redes complexas foram iniciados em meados de 1930, quando sociólogos utilizavam essas redes com a finalidade de estudar o comportamento da sociedade e a relação entre os indivíduos. Essas pesquisas eram baseadas em características muito peculiares das redes, como a centralidade (o vértice mais central) e a conectividade (vértices com maior número de conexões). As redes sociais eram constituídas por indivíduos, que eram representados por vértices, e pelas interações entre eles, representadas pelas arestas. A centralidade e a conectividade eram usadas, por exemplo, para determinar os indivíduos que melhor se relacionavam com os demais ou para identificar os indivíduos mais influentes.

Com o avanço da tecnologia de informação e a disponibilidade de computadores, que permitem a análise de dados em grandes quantidades, houve uma mudança significativa na área. As pesquisas, antes focadas nas pequenas redes e nas propriedades de vértices individuais ou arestas, passaram a considerar propriedades estatísticas em larga-escala. Atualmente, são comuns estudos com redes envolvendo milhões ou bilhões de vértices, as quais antes eram compostas por dezenas ou, em casos extremos, centenas. A mudança de paradigma revelou várias características que diferem, substancialmente, as redes do mundo real das redes aleatórias, tidas por muitos anos como o principal modelo de redes [54]. Descobriu-se que a topologia e a evolução das redes do mundo real apresentam propriedades organizacionais bastante robustas e distintas das redes aleatórias. Essa é a principal razão pela qual estas redes passarem a ser chamadas de redes complexas.

4.2.1 Métricas de Redes Complexas

Após a construção da rede complexa, várias métricas relacionadas a grafos podem ser calculadas e usadas para a caracterização de uma rede complexa. Neste trabalho estamos interessados em métricas relativas aos vértices da rede [2]. Estas métricas podem ser usadas para selecionar palavras no processo de expansão de um documento para realizar o processo de recomendação de artigos científicos.

[2] Os nomes das métricas serão todas mantidas em seu nome original.

4.2.1.1 *Preferential Attachment*

A pontuação dos vértices u e v de um grafo é dado por

$$score(u, v) = |\Gamma(u)||\Gamma(v)| \tag{4.1}$$

em que $\Gamma(u)$ representa o número de ligações desse vértice.

A premissa básica do *preferential attachment* é que a probabilidade de que uma nova aresta seja criado com o vértice u é proporcional a $\Gamma(u)$, o número atual de vizinhos de u. Em [55] e em [56], os autores propuseram, com base em evidências empíricas, que a probabilidade de ligação de u e v está correlacionada com o produto do número de vizinhos de u e v. Este produto do número de vizinhos de u e v corresponde à equação (4.1).

4.2.1.2 *Clustering*

Para grafos ponderados, *clustering* ou coeficiente de agrupamento, mede o grau com que os vértices de um grafo tendem a agrupar-se e é definido como a média geométrica dos pesos das arestas de um subgrafo [57],

$$c_u = \frac{1}{deg(u)(deg(u) - 1))} \sum_{uv} (w_{uv} w_{uw} w_{vw})^{1/3}$$

em que $deg(u)$ é o grau do vértice u. Os pesos das arestas w_{uv} são normalizados pelo maior peso existente no grafo $w_{uv} = w_{uv}/max(w)$. O valor c_u é igual a 0 se $deg(u) < 2$.

4.2.1.3 *Betweenness Centrality*

Na teoria dos grafos, *betweenness centrality* [58] é uma métrica de centralidade em um grafo baseado em caminhos mais curtos. *Betweenness centrality* de um dado vértice v é igual ao número de caminhos mais curtos de todos os vértices para quaisquer outros vértices que passam por v.

Betweenness centrality é calculada da seguinte forma:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

em que V é o conjunto de vértices do grafo, $\sigma(s, t)$ é o número de caminhos mais curtos (s, t), e $\sigma(s, t|v)$ é o número desses caminhos que passam pelo vértice *v*, além de *s,t*.

4.2.1.4 *Edge Betweenness Centrality*

Edge betweenness centrality [58] de uma aresta e é a soma da fração de todos os pares de caminhos mais curtos que passam por e e é definida por:

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s,t|e)}{\sigma(s,t)}$$

onde V é o conjunto de vértices do grafo, $\sigma(s,t)$ é o número de caminhos mais curtos (s,t) e $\sigma(s,t|e)$ é o número desses caminhos que passa por e.

4.2.1.5 *Betweenness Centrality Subset*

Para calcular o *betweenness centrality subset* [58] para um subconjunto de vértices de um grafo é utilizado a equação

$$c_B(v) = \sum_{s \in S, t \in T} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

em que S é o conjunto de origem, T é o conjunto de destino, $\sigma(s,t)$ é o número de caminhos mais curtos (s,t) e $\sigma(s,t|v)$ é o número desses caminhos que passam por algum vértice v, além de s, t. Se $s = t$ então $\sigma(s,t) = 1$ e se $v \in s,t$ então $\sigma(s,t|v) = 0$.

Para grafos ponderados, os pesos das bordas devem ser maiores que zero. Os pesos de borda iguais a zero podem produzir um número infinito de caminhos de igual comprimento entre pares de vértices.

4.2.1.6 *Edge Betweenness Centrality Subset*

Para calcular o *edge betweenness centrality subset* [58] entre as bordas para um subconjunto de vértices é utilizado a equação

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s,t|e)}{\sigma(s,t)}$$

em que S é o conjunto de origem, T é o conjunto de destino e $\sigma(s,t|e)$ é o número desses caminhos que passam pelo vértice e.

4.2.1.7 *Degree Centrality*

O *degree centrality* para um dado vértice v é a fração de vértices que ele está conectado e é calculado utilizando a equação

$$C_v = \frac{deg(v)}{n-1}$$

em que $deg(v)$ é o grau do vértice v. Os valores do *degree centrality* são normalizados dividindo-os pelo grau máximo possível em um grafo, onde n é o número de vértices em G.

4.2.1.8 *Closeness Centrality*

A métrica *closeness centrality* [59] de um vértice é o recíproco da soma das distâncias do caminho mais curto de u para todos $n-1$ outros vértices. Como a soma das distâncias depende do número de vértices no grafo, *closeness centrality* é normalizada pela soma das distâncias mínimas possíveis $n-1$. Para calcular a métrica *closeness centrality* é utilizado a equação

$$C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)}$$

em que $d(v,u)$ é a distância do caminho mais curto entre v e u, e n é o número de vértices no grafo.

4.2.1.9 *Eigenvector Centrality*

Eigenvector Centrality calcula a centralidade de um vértice baseado na centralidade de seus vizinhos. A centralidade do vetor próprio para o vértice i é dada por:

$$Ax = \lambda X$$

em que A é a matriz de adjacência do grafo G, com autovalor λ. Em virtude do teorema de Perron Frobenius existe uma solução única e positiva se λ é o maior autovalor associado ao vetor próprio da matriz adjacente A [60].

4.2.1.10 *Katz Centrality*

Katz Centrality [61] calcula a centralidade de um vértice baseado na centralidade de seus vizinhos. É uma generalização da métrica *eigenvector centrality*. *Katz Centrality* para o vértice i é definido como

$$x_i = \alpha \sum_j A_{ij} x_j + \beta$$

em que A é a matriz de adjacência do grafo G com autovalores λ. O parâmetro β controla a centralidade inicial. Conexões feitas com vizinhos distantes, no entanto, são penalizados por um fator de atenuação α que deve ser estritamente menor que o maior autovalor inverso da matriz de adjacência:

$$\alpha < \frac{1}{\lambda_{\max}}.$$

4.2.1.11 *Global Degree*

O grau $deg(v)$ de um vértice v de um grafo G é definido como sendo o número de arestas adjacentes a esse vértice.

4.2.1.12 *Local Degree*

O grau local $deg'(v)$ de um vértice v é o grau deste vértice em relação a um subgrupo G' de um grafo G.

As duas métricas a seguir, *Random* e TF-IDF, não são métricas de grafo mas foram desenvolvidas para serem utilizadas como *baseline* básico.

4.2.1.13 *Random*

O método randômico escolhe aleatoriamente N termos distintos do vocabulário de um documento. Acreditamos que métricas cujos resultados sejam semelhantes ao obtido pelo randômico são pobres e não acrescentam nada ao objetivo deste trabalho que é melhorar as técnicas existentes.

4.2.1.14 TF-IDF

Em recuperação de informação, TF-IDF, abreviação para *Term Frequency* e *Inverse Document Frequency*, é uma métrica que se destina a computar a importância de uma palavra para um documento em uma coleção ou corpus. Muitas vezes é usado como fator de ponderação em *Information Retrieval*, *Text Mining*, e *User Modeling*. O valor do TF-IDF aumenta proporcionalmente ao número de vezes que uma palavra aparece no documento, mas é compensado pela frequência da palavra no corpus, o que ajuda a ajustar o fato de que algumas palavras aparecem mais frequentemente em geral [62].

Existem várias fórmulas propostas para o cálculo do TF-IDF [63]. Para este trabalho foi selecionado

$$tfidf_{t,d} = tf_{t,d} \log(\frac{N}{n_t})$$

em que N é o número de documentos na coleção. $tf_{t,d}$ é a frequência do termo t no documento d e n_t é a frequência do termo t em toda a coleção de documentos ou corpus.

4.3 *Máquinas de Vetor Suporte (SVM)*

Máquinas de Vetor Suporte ou *Support Vector Machine (SVM)* é um classificador discriminativo formalmente definido por um hiperplano separador [64]. Em outras palavras, dado um conjunto de dados de treinamento rotulados (aprendizagem supervisionada), o algoritmo produz um hiperplano ótimo que categoriza novos exemplos. Em que sentido o hiperplano obtido é otimizado? Consideremos o problema abaixo:

Para um conjunto linearmente separável de pontos 2D que pertencem a uma das duas classes, encontre uma linha reta separadora.

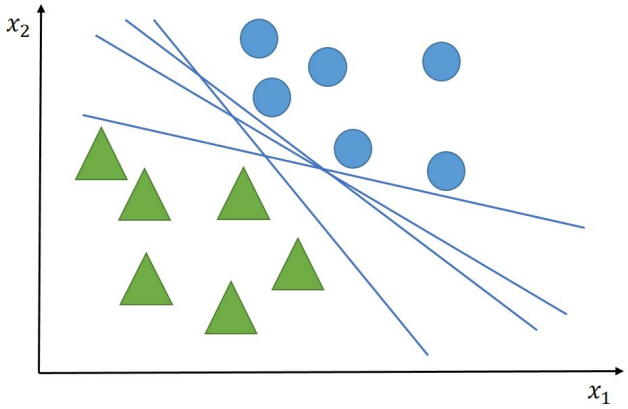

Figura 4.3: SVM possíveis soluções

Na imagem acima existem várias linhas que oferecem uma solução para o problema. Qual é a melhor? Podemos definir intuitivamente um critério para estimar o valor das linhas:

Uma linha é ruim se ela passa muito perto dos pontos, porque será sensível ao ruído e não generalizará corretamente. Portanto, o objetivo é encontrar a linha que maximiza a distância de todos os pontos.

A operação do algoritmo SVM baseia-se na busca do hiperplano que maximiza a distância aos exemplos de treinamento. Essa distância recebe o nome da margem. Portanto, o hiperplano de separação ideal maximiza a margem dos dados de treinamento.

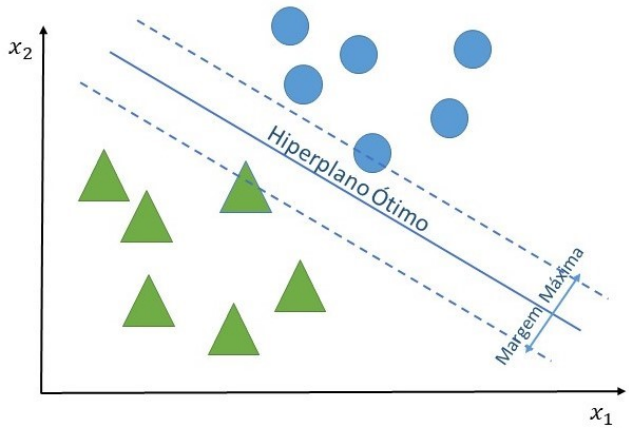

Figura 4.4: SVM Hiperplano Ótimo

Um hiperplano é definido formalmente como

$$f(x) = \beta_0 + \beta^T x$$

em que β é o vetor peso e β_0 é o viés.

O hiperplano ótimo, mostrado na Figura 4.4, pode ser representado um número infinito de maneiras diferentes por uma escala de β e β_0. Por convenção, entre todas as possíveis representações do

hiperplano, o escolhido é

$$|\beta_0 + \beta^T x| = 1$$

em que x representa os exemplos de treinamento mais próximos do hiperplano. Em geral, os exemplos de treinamento mais próximos do hiperplano são chamados de vetores de suporte. Esta representação é conhecida como hiperplano canônico.

Agora, usaremos o resultado geométrico que dá a distância entre um ponto x e o hiperplano

$$Distância = \frac{|\beta_0 + \beta^T x|}{||\beta||}$$

Em particular, para o hiperplano canônico, o numerador é igual a um e a distância para os vetores de suporte é

$$Distância_{VetoresSuporte} = \frac{|\beta_0 + \beta^T x|}{||\beta||} = \frac{1}{||\beta||}$$

A margem introduzida anteriormente, aqui denotada como M, é o dobro da distância para os exemplos mais próximos

$$M = \frac{2}{||\beta||}.$$

Finalmente, o problema de maximizar M é equivalente ao problema de minimizar uma função $L(\beta)$ sujeita a algumas restrições. As restrições modelam o requisito para o hiperplano classificar corretamente todos os exemplos de treinamento x_i. Formalmente,

$$\min_{\beta\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \textit{ sujeito a } y_i(\beta^T x_i + \beta_0) \geq 1 \forall i$$

em que y_i representa cada um dos rótulos dos exemplos de treinamento.

Este é um problema de otimização de Lagrange que pode ser resolvido usando multiplicadores de Lagrange para obter o vetor de peso β e o viés β_0 do hiperplano ideal.

4.4 Regressão Logística

Regressão logística é um modelo de regressão que mede a relação entre a variável categórica dependente e uma ou mais variáveis independentes estimando probabilidades usando uma função logística [64]. Em estatística, uma variável categórica é uma variável que pode assumir um número limitado e geralmente fixo de valores possíveis.

A função logística é definida como

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \tag{4.2}$$

Os termos β_0 e β_1 neste modelo representam parâmetros desconhecidos que serão estimados com base nos dados amostrais utilizando o método chamado máxima verossimilhança que maximiza a probabilidade de obter o grupo observado de dados.

A função logística sempre produzirá uma curva em forma de S, como mostrado na Figura 4.5, e, independentemente do valor de X, obteremos uma previsão no intervalo $[0, 1]$.

Após manipular a função logística acima obtemos

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X} \tag{4.3}$$

A quantidade $\frac{p(X)}{1-p(X)}$ é chamada de *odds*, e pode assumir qualquer valor de entre 0 e ∞. Valores de *odds* próximos de 0 indicam probabilidades muito baixas e valores de *odds* próximos de ∞ indicam probabilidades muito altas.

Tirando o logaritmo de ambos os lados da Equação (4.3) acima obtemos

$$log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X \tag{4.4}$$

O lado esquerdo é chamado de *log-odds* ou *logit*. Observamos que o modelo de regressão logística na Equação (4.3) possui um *logit* linear em X.

Em um modelo de regressão linear, β_1 dá a mudança média em Y associada ao aumento de uma unidade em X. Em contraste, no modelo de regressão logística, o aumento de X por uma unidade altera o *logit* por β_1 na Equação (4.4). No entanto, como a relação entre $p(X)$ e X em (4.2) não é uma linha reta, β_1 não corresponde à alteração em $p(X)$ associada a um aumento de uma unidade em X. A quantidade que $p(X)$ altera, devido a uma mudança de uma unidade em X, dependerá do valor atual de X. Mas, independentemente do valor de X, se β_1 for positivo, o aumento de X será associado ao aumento de $p(X)$, e se β_1 for negativo, o aumento de X será associado à diminuição de $p(X)$.

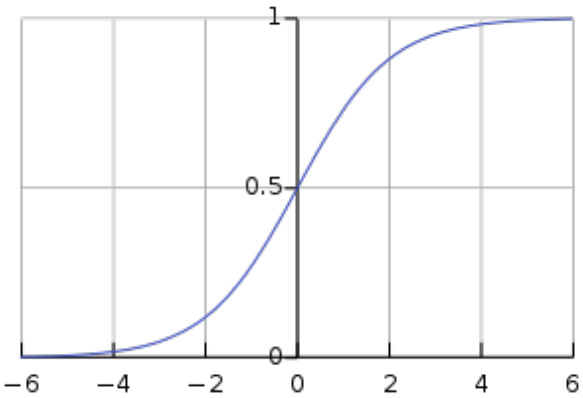

Figura 4.5: Função Logística

4.4.1 Estimando os parâmetros β_0 e β_1

Os coeficientes β_0 e β_1 na Equação (4.2) são desconhecidos e devem ser estimados com base nos dados de treinamento disponíveis. O modelo de regressão logística utiliza o método chamado *maximum likelihood* ou máxima verossimilhança [64].

A intuição básica por trás do uso da máxima verossimilhança é a seguinte: buscamos estimativas para β_0 e β_1, de modo que a probabilidade prevista $p(x_i)$ de cada indivíduo, usando a equação (4.2) acima, corresponde o mais próximo possível do valor observado de cada indivíduo. Em outras palavras, tentamos encontrar β_0' e β_1' de modo que ao utilizarmos estas estimativas no modelo $p(X)$, dado em (4.2), o modelo produzirá um número perto de um para todas as palavras-chave, e um número próximo a zero para todos as palavras que não são palavras-chave. Essa intuição pode ser formalizada usando uma equação matemática chamada de função de verossimilhança

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \tag{4.5}$$

Os valores de β_0' e β_1' são escolhidos de tal maneira que eles maximizem a função (4.5) acima.

4.5 *Random Forest*

O algoritmo *Random Forest* é um tipo de *ensemble learning*, método que gera muitos classificadores e agrega o seu resultado. No caso do *Random Forest*, ele gera múltiplas árvores de decisão que depois são utilizadas na classificação de novos objetos.

Abaixo faremos o detalhamento dos elementos citados acima.

4.5.1 Classificador

Dado um conjunto de exemplos de treinamento, um indutor (ou algoritmo de aprendizado) gera como saída um classificador de forma que, dado um novo exemplo, ele possa predizer sua classe. Formalmente, um exemplo é um par $(x_i, f(x_i))$ onde x_i é a entrada e $f(x_i)$ é a saída. A tarefa de um indutor é, dado um conjunto de exemplos, induzir uma funçao $h(\cdot)$ que aproxima $f(\cdot)$, normalmente desconhecida. Neste caso, $h(\cdot)$ é chamada uma hipótese sobre a função objetivo $f(\cdot)$, ou seja, $h(x_i) \approx f(x_i)$.

4.5.2 Árvores de Decisão

Uma árvore de decisão é uma estrutura semelhante a um fluxograma em que cada vértice interno representa um "teste" em um atributo (por exemplo, se um *flip* de moeda aparece cara ou coroa), cada ramo representa o resultado do teste e cada vértice de folha representa um rótulo de classe. Os caminhos da raiz para a folha representam regras de classificação [65].

Uma árvore de decisão utiliza uma estratégia de dividir-para-conquistar, decompondo um problema maior em sub-problemas mais simples, de forma recursiva. A construção de uma árvore de decisão baseia-se na escolha de um atributo que servirá para particionar os exemplos em subconjuntos correspondentes a cada valor do atributo. Passa-se então os exemplos para os subconjuntos de acordo com o valor do atributo escolhido para cada subconjunto. É necessário observar se todos os exemplos pertencem a uma mesma classe. Se isso ocorrer, associa-se essa classe à uma folha contendo todos os exemplos desse subconjunto. Caso contrário, é escolhido um novo atributo para particionar os exemplos e o processo é repetido até que haja uma homogeneidade das classes dos exemplos de um determinado subconjunto.

Uma árvore de decisão pode ser usada para classificar novos exemplos. Assim, quando for necessário fazer uma nova classificação, percorre-se a árvore a partir da raiz e desvia-se em cada vértice de decisão, segundo o atributo correspondente até que se atinja um vértice folha e, então, associa-se a esse novo exemplo a classe correspondente a esse vértice folha.

4.5.3 Métodos Ensemble

Métodos *ensemble* são algoritmos de aprendizado que constroem um conjunto de classificadores e combinam seus votos para classificar um novo exemplo [66]. O voto corresponde à classe fornecida por cada um dos classificadores para esse novo exemplo. Os métodos ensembles amplamente conhecidos são *boosting* [67], *bagging* [68] e *Random Forests* [69].

Boosting é um meta-algoritmo *ensemble* de aprendizado de máquina cuja principal função é diminuir o viés, e também a variância em aprendizado supervisionado. *Boosting* baseia-se na pergunta colocada por Kearns e Valiant [70]: *"Um grupo de aprendizes fracos pode criar um único aprendiz forte?"*. Um aprendiz fraco é definido como um classificador que é apenas ligeiramente correlacionado com a classificação correta (ele pode rotular exemplos melhor do que uma classificação aleatória). Em contraste, um aprendiz forte é um classificador que é arbitrariamente bem correlacionado com a classificação correta.

No início o método *boosting* cria diferentes classificadores base e inicializa todos os exemplos com pesos iguais. Cada exemplo classificado erroneamente por um classificador recebe um peso maior na próxima iteração do treinamento, afim de tentar classificá-lo corretamente. O erro é computado, o peso dos exemplos classificados corretamente é reduzido e o peso dos exemplos classificados incorretamente é aumentado. O voto de cada classificador individual é ponderado proporcionalmente ao seu desempenho [71].

Bagging ou *bootstrap aggregation* [68] é um dos algoritmos de aprendizado *ensemble* mais antigos. Ele usa amostras *bootstrap* do conjunto de treinamento [72]. Amostragem *bootstrap* é uma técnica de amostragem com reposição: a partir do conjunto de treinamento inicial, são selecionados aleatoriamente exemplos para um novo subconjunto de treinamento. Assim, no método *bagging* diferentes subconjuntos são aleatoriamente construídos, com reposição, a partir do conjunto original. Devido à reposição, alguns exemplos podem ser escolhidos mais de uma vez e outros podem não ser escolhidos.

Random Forest é um método *ensemble* que constrói muitas árvores de decisão que serão usadas

para classificar um novo exemplo por meio do voto majoritário. Cada árvore de decisão usa um subconjunto de atributos selecionados aleatoriamente a partir do conjunto original, contendo todos os atributos. Além disso, cada árvore usa uma amostra *bootstrap* diferente da base de dados da mesma maneira que o algoritmo *bagging*.

4.6 Alocação Latente de Dirichlet – LDA

LDA (Latent Dirichlet Allocation) é um método probabilístico criado com o objetivo de descrever coleções de documentos usando um conjunto de tópicos [73]. Um tópico é formalmente definido como uma distribuição de probabilidade sobre um vocabulário fixo, definido por uma coleção de documentos, em que as palavras com maior probabilidade são mais propensas a serem selecionadas para descrever o tópico. Por exemplo, considerando o tópico esportes, ele pode ser descrito por uma distribuição de probabilidade concentrada em termos como jogo, futebol, tênis, bola e partida. A Figura 4.6 mostra alguns exemplos de tópicos extraídos usando o algoritmo LDA. Note que cada palavra tem um valor associado a ela. Esse valor é a probabilidade de essa palavra aparecer em um documento sobre esse tópico. Por exemplo, dado um documento sobre esportes (Tópico 3), a palavra jogo tem uma probabilidade de 2% de aparecer neste documento.

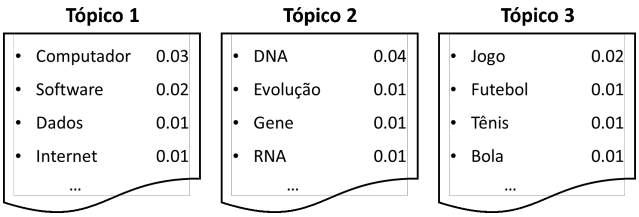

Figura 4.6: Exemplo de tópicos

A maneira mais intuitiva de entender a lógica do LDA é observando seu processo gerador, ou seja, um processo imaginário assumido pelo modelo como responsável pela criação dos documentos na coleção. Uma notação de placa do processo gerador do LDA é ilustrada na Figura 4.7. Em notação de placa, nós sombreados representam variáveis observáveis, enquanto os outros nós representam variáveis latentes. A notação de placa simplifica a representação de grandes modelos gráficos probabilísticos, colocando estruturas replicadas em retângulos numerados, onde o número representa replicações (por exemplo, retângulos numerados com M e N referem-se às estruturas replicadas de M e N, respectivamente, enquanto K se refere ao número de tópicos).

Em LDA, os tópicos são definidos como uma distribuição multinomial sobre palavras do vocabulário (distribuição φ), documentos são considerados como uma misturas de tópicos (distribuição θ) e palavras são derivadas, uma a uma, selecionando um tópico Z de θ_W de ϕ_z, onde o índice Z indica um tópico.

Esse processo generativo reflete o fato de que os documentos contêm vários tópicos. Por exemplo, um documento pode pertencer aos tópicos esportes e política. Em outras palavras, cada documento tem tópicos em diferentes proporções (definidos pela distribuição θ). Apenas palavras dentro de documentos são variáveis observáveis e todas as anteriores (α e β) são definidas como distribuições de Dirichlet (Dir). O principal objetivo do algoritmo é inferir as variáveis ocultas (ou latentes),

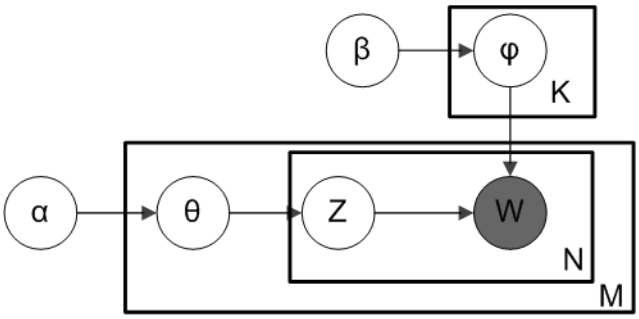

Figura 4.7: Modelo gráfico LDA em notação de placa

como proporções de tópicos de documentos e distribuições de palavras por tópico. O processo gerador de LDA é detalhado na Figura 4.8.

1. Para cada tópico z

    a) Aplique uma distribuição multinomial sobre todos os termos, $\varphi_z \sim \mathrm{Dir}(\beta)$

2. Para cada documento d do corpus

    a) Aplique uma distribuição multinomial para os tópicos de destino, $\theta_d \sim \mathrm{Dir}(\alpha)$

    b) Para cada palavra w no documento d

        i. Obtenha um tópico da distribuição escolhida, $z_{w,d} \sim \mathrm{Multinomial}(\theta_d)$

        ii. Obtenha um termo do tópico escolhido, $w \sim \varphi_{z_{w,d}}$

Figura 4.8: Processo gerador do LDA

É importante mencionar que os modelos de tópico não têm nenhuma informação prévia sobre os tópicos em uma coleção e que os documentos não têm nenhuma anotação ou palavra-chave para definir quais assuntos eles abordam. A distribuição tópica é gerada inferindo-se a estrutura latente que, com maior probabilidade, é responsável por gerar a coleção de documentos observada. Existem várias técnicas para realizar essa inferência, incluindo Inferência Variacional [73], Propagação de Expectativas [74] e Monte Carlo via Cadeias de Markov (MCMC) [75]. Esta última sendo a mais utilizada em modelos de tópicos probabilísticos.

4.7 Regressão Colaborativa de Tópicos – CTR

Regressão Colaborativa de Tópicos (CTR) [3] [2] é um método que combina a filtragem colaborativa com modelagem probabilística de tópicos. É um modelo probabilístico que se integra perfeitamente ao método LDA [73], descrito na seção acima, e a fatoração de matriz probabilística (PMF) [76]. CTR inclui uma variável latente ε_j que compensa as proporções do tópico θ_j ao modelar as classificações do usuário. À medida que mais usuários classificam os artigos, temos uma ideia

[3] Do Inglês, Collaborative Topic Regression

melhor do que é esse deslocamento. A variável *offset* pode explicar, por exemplo, que o artigo *A* é mais interessante para pesquisadores de aprendizado de máquina do que para pesquisadores de análise de redes sociais. O quanto da previsão depende do conteúdo e de quanto ela depende de outros usuários, depende de quantos usuários avaliaram o artigo. A Figura 4.9 mostra o modelo de CTR.

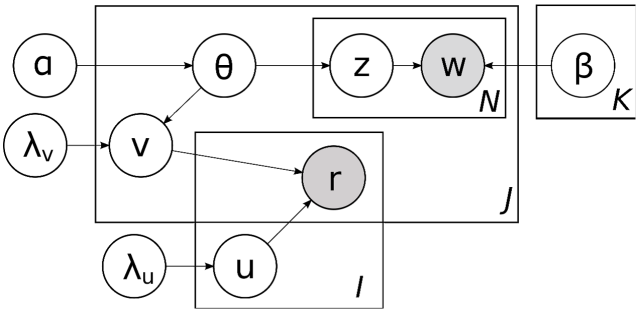

Figura 4.9: Modelo gráfico CTR em notação de placa

O processo gerador do CTR é detalhado na Figura 4.10 a seguir. Este processo assume que existem K tópicos $\beta = \beta_{1:K}$.

1. Para cada usuário i, extraia o vetor latente $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_k)$
2. Para cada item j,
    a) Extraia as proporções dos tópicos $\theta_j \sim Dirichlet(\alpha)$
    b) Extraia o *offset* latente do item $e_j \sim \mathcal{N}(0, \lambda_v^{-1} I_k)$ e defina o vetor latente do item como $v_j = \epsilon_j + \theta_j$.
    c) Para cada palavra w_{jn},
        i. Obtenha a atribuição do tópico $z_{j,n} \sim Mult(\theta)$
        ii. Obtenha a palavra $w_{j,n} \sim Mult(\beta_{z_{j,n}})$
3. Para cada par usuário-item (i, j) obtenha a classificação $r_{i,j} \sim \mathcal{N}(u_i^T v_j, c_{i,j}^{-1})$

Figura 4.10: Processo gerador do CTR

A propriedade chave desse método está em como o vetor latente do item v_j é gerado. Note que $v_j = \varepsilon_j + \theta_j$, quando $e_j \sim N(0, \lambda_v^{-1} I_k)$, é equivalente a $v_j \sim N(\theta_j, \lambda_v^{-1} I_k)$, onde assumimos que o vetor latente de item v_j está próximo das proporções do tópico θ_j, mas pode divergir dele se for necessário. Note que o valor esperado de r_j é uma função linear de θ_j,

$$E[r_{i,j} u_i, \theta_j, \varepsilon_j] = u_i^T (\theta_j + \varepsilon_j)$$

Capítulo 5

RECOMENDAÇÃO DE TAGS

Neste trabalho, a recomendação de *tags* é feita de duas maneiras distintas. Usando um modelo de vetores de palavras e uma representação em redes complexas, a relevância das palavras é calculada usando métricas de caracterização de redes complexas e as palavras mais relevantes são selecionadas como as *tags* recomendadas.

A outra maneira proposta para recomendação de *tags* também utilizam as métricas das redes complexas. Entretanto, as métricas não são usadas diretamente na recomendação. Estas métricas são usadas como dados de entrada para técnicas de aprendizado de máquinas tais como Máquinas de Vetor Suporte, Regressão Logística e *Random Forest* e a recomendação é então realizada.

Este capítulo descreve as etapas de cada uma das metodologias de recomendação de *tags* propostas. Os resultados obtidos serão apresentados e discutidos.

5.1 Etapas da Metodologia Proposta

A metodologia proposta na recomendação de *tags* funciona em etapas distintas, mostradas na Figura 5.1. As etapas de **(i)** filtragem e tratamento dos dados, **(ii)** construção dos vetores de palavras, **(iii)** representação da rede complexa e **(iv)** cálculo das métricas estão presentes nas duas metodologias propostas. Usando os resultados destas etapas, a recomendação de *tags* pode ser feito diretamente ou utilizando técnicas de aprendizado de máquinas. A seguir são detalhadas e discutidas cada uma destas etapas.

5.1.1 Normalização

A fase de normalização segue as etapas tradicionais do pré-processamento de texto em Processamento de Linguagem Natural, que inclui:

- Conversão para minúsculo. A maioria dos algoritmos de mineração de texto são sensíveis

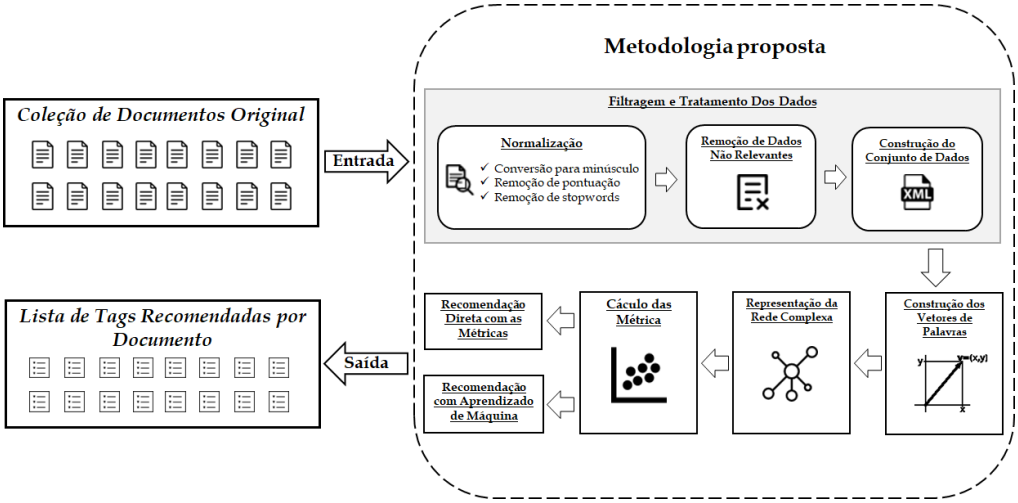

Figura 5.1: Visão geral da metodologia de recomendação de *tags* proposta neste trabalho.

a maiúsculas e minúsculas. Portanto, todas as palavras devem ser minúsculas para serem consideradas as mesmas.

- Pontuação, acentuação e remoção de caracteres especiais. Pontuação e acentuação não são relevantes para o processo de recomendação de tags.
- Remoção de *stopwords*. Uma palavra é considerada uma *stopword* quando é considerada irrelevante para o problema. A remoção dessas palavras é um passo importante no pré-processamento de dados textuais uma vez que a frequência de tais palavras geralmente é alta e pode enviesar o processo de aprendizagem.

5.1.2 Remoção de Dados Não relevantes

O corpus utilizado nesse trabalho é composto de milhares de textos, sendo que cada texto tem associado um conjunto de *tags*. Mas nem todas as *tags* associadas a um determinado texto pertencem ao mesmo, ou seja, o trabalho proposto recomenda como *tags* apenas palavras que pertencem ao texto e as *tags* associadas a um determinado texto que não pertencem ao mesmo são excluídas nessa etapa. Isto é feito com o objetivo de não comprometer o desempenho dos algoritmos de predição. Em alguns casos todas as *tags* associadas a um determinado texto não pertencem ao mesmo. Nesses casos o texto em questão foi removido do conjunto de dados final.

5.1.3 Construção de Vetores de Palavras

Todos os dados, exceto as *tags*, após serem normalizados e as informações irrelevantes removidas, são processados pelo programa *word2vec*, descrito na Subseção (4.1). Este programa cria uma representação vetorial do vocabulário do corpus.

Word2vec é um grupo de modelos relacionados que são usados para produzir vetores de palavras. Esses modelos são redes neurais de duas camadas que são treinadas para reconstruir contextos

linguísticos de palavras. O *word2vec* toma como entrada um grande corpus de texto e produz um espaço vetorial, tipicamente de várias centenas de dimensões, onde para cada palavra do vocabulário é atribuído um vetor correspondente no espaço. Os vetores de palavras estão posicionados no espaço vetorial de tal maneira que as palavras que compartilham contextos comuns no corpus estão localizadas nas proximidades uma das outras.

5.1.4 Representação da Rede Complexa

Uma rede complexa corresponde a um grafo, que é representado por um conjunto de vértices ligados por arestas que conjuntamente formam uma rede. Esta rede ou grafo permite representar relações [54]. Com o objetivo de explorar estas relações e suas propriedades, um grafo $G = (V, A)$ foi criado utilizando o conjunto de palavras do vocabulário do corpus, denominado V, onde cada palavra representa um vértice. Uma aresta A entre duas palavras é criada somente se o cosseno entre os vetores que as representam for maior ou igual a 0,5. Quanto maior for o cosseno entre dois vetores de palavras maior é a similaridade entre elas. Ao criarmos uma arestas quando o cosseno for maior ou igual a 0,5 estamos conectando palavras que compartilham um contexto semelhante no *corpus*.

5.1.5 Cálculo das Métricas

Nessa etapa, para cada texto T da coleção, o procedimento abaixo é executado:

1. Seja G a rede complexa construída e G' o subgrafo de G referente ao texto T. O subgrafo G' é selecionado utilizando o vocabulário V' do texto T.

$$G = (V, A)$$

2. As métricas de grafo listadas na Subseção (4.2.1) são calculadas para o subgrafo G'.
3. A métrica *Random* é calculada utilizando o conjunto V'.
4. A métrica TF-IDF é calculada utilizando o texto T.

5.1.6 Recomendação de Tags

A fase final no processo de recomendação automática de *tags* é a criação de uma matriz de palavras por métricas. Esta matriz pode ser diretamente usada para recomendar as *tags* ou, alternativamente, serve de entrada para os algoritmos de predição que são utilizados nesse trabalho.

Para cada texto T do corpus é criada uma matriz M' onde cada linha dessa matriz contém uma palavra $P \in V'$, onde V' é o vocabulário do texto T, além das métricas calculadas anteriormente. Esta matriz obtida é usada diretamente para a recomendação de *tags*.

Uma nova coluna é adicionada à matriz M', possuindo o valor 1 quando a palavra P é uma palavra-chave do texto T, ou 0, caso contrário. A matriz M' é calculada para todos os textos do corpus e, no final, essas matrizes são concatenadas em uma única matriz M. É essa matriz que servirá de entrada para os algoritmos *Máquinas de Vetor Suporte (SMV)*, *Regressão Logística* e *Random Forest*.

5.2 Experimentos e Resultados

Nesta seção são descritos os procedimentos realizados para o desenvolvimento deste trabalho.

5.2.1 Aquisição dos Dados

O conjunto de dados utilizado neste trabalho foi obtido a partir do site de avaliação de filmes IMDb, na língua inglesa. A motivação para a escolha deste conjunto de dados está nas palavras-chave associadas pelos usuários do site aos filmes. Estas palavras-chave serão utilizadas posteriormente para validar o trabalho realizado.

Inicialmente foi obtido um arquivo com 40.110 identificadores de filmes pertencentes ao site. Estes identificadores serviram de entrada para um programa que fez o *download* das informações de cada filme, tais como o título e a descrição, e os dados foram então gravados em um arquivo texto para serem processados posteriormente.

5.2.2 Filtragem e Tratamento dos Dados

Após a aquisição dos 40.110 filmes do site IMDb, os dados obtidos passaram por um processo de filtragem e tratamento com o objetivo de normalizar e remover os dados não relevantes. Cada etapa deste processo é descrita abaixo.

Normalização

O processo de normalização consiste em converter todas as palavras para minúsculo e remover toda a pontuação, com o objetivo de padronizar o conjunto de dados. A normalização do conjunto de dados também efetua a remoção de *stopWords*, onde são removidos todos os termos que não possuem valores semânticos significativos.

Remoção de Dados Não Relevantes

Este trabalho tem por objetivo desenvolver um processo automatizado de recomendação de palavras-chave em textos. O trabalho proposto recomenda apenas palavras extraídas do texto analisado. Sendo assim as palavras-chave selecionadas pelos usuários do site IMDb, que não pertencem ao texto descritivo do filme, foram removidas. Em alguns casos todas as palavras-chave selecionadas

pelos usuários não pertencem a descrição do filme. Nesses casos, o filme em questão foi removido do conjunto de dados final.

Construção do Conjunto de Dados

Após a normalização e a remoção dos filmes que não possuem palavras-chave selecionadas do texto descritivo, um arquivo XML foi construído com os dados restantes. Este arquivo contém 29.096 filmes e seus dados. O conjunto de dados contido nesse arquivo será utilizado como entrada em todos os experimentos descritos nesse trabalho.

Abaixo uma descrição das informações contidas no arquivo XML, por filme:

1. **code**: Código do filme no site IMDb. Este código é a chave utilizada para fazer o *download* dos dados do filme;
2. **title**: Título do filme;
3. **plot_outline**: Descrição curta do filme;
4. **plot:** Descrição detalhada do filme;
5. **keywords:** Palavras-chave selecionadas pelos usuários do site IMDb, extraídas do texto descritivo do filme.

A Figura 5.2 mostra um exemplo de um filme extraído do arquivo XML final após normalização e tratamento.

```
1  <movie>
2    <title>toy story</title>
3    <plot_outline>cowboy doll profoundly threatened jealous new spaceman
4    figure supplants top toy boy room</plot_outline>
5    <plot>little boy named andy loves room playing toys especially doll
6    named woody toys andy come life woody believes life toy good...</plot>
7    <keywords>toy cowboy jealousy boy birthday spaceman moving fight
8    mission neighbor escape threat</keywords>
9  </movie>
```

Figura 5.2: Exemplo de filme após normalização e tratamento

5.2.3 Construção de Vetores de Palavras

Todos os dados, exceto as palavras-chave, após serem normalizados e as informações irrelevantes removidas, foram processados pelo programa *word2vec*, para criar uma representação vetorial do vocabulário IMDb.

O corpus utilizado para este trabalho consiste nos campos *title*, *plot_outline* e *plot* de todos os filmes concatenados em uma única linha de texto. Os vetores calculados para o vocabulário IMDb possuem dimensão 200 e o modelo utilizado foi o CBOW inferindo uma palavra a partir do seu contexto.

5.2.4 Representação da Rede Complexa

O conjunto de palavras do vocabulário IMDb foi utilizado para criar um grafo. Neste vocabulário, cada palavra representa um vértice e uma aresta foi criada somente se o cosseno entre os vetores que as representam for maior ou igual a 0,5.

5.2.5 Aprendizado de máquina não supervisionado

Neste experimento, para cada filme F da coleção foi executado o procedimento abaixo:

1. Seja G a rede complexa construída em 5.2.4 e G' o subgrafo de G referente ao filme F. O subgrafo G' foi selecionado utilizando-se o vocabulário do filme F que é composto do conjunto V' de palavras dos campos $title$, $plot_outline$ e $plot$.

$$G = (V, A)$$

$$G' = (V', A')$$

$$V' \subseteq V$$

$$A' \subseteq A$$

2. As métricas de grafo listadas na Seção 4.2.1 foram calculadas para o subgrafo G'.
3. A métrica *Random* foi calculado utilizando o conjunto V'.
4. A métrica TF-IDF foi calculada utilizando os campos *title*, *plot_outline* e *plot* do filme F.
5. Para cada métrica calculada nos itens 2,3 e 4 acima, foram selecionadas as 10 palavras mais bem classificadas e o resultado comparado com as palavras-chave reais. A partir dessa comparação foi calculado a precisão de cada métrica em relação as palavras-chave reais e os resultados são mostrados na Figura 5.3.

Analisando a Figura 5.3, observa-se que utilizar as métricas individualmente para recomendar *tags* é uma metodologia que não apresentou bons resultados. Nenhuma das métricas de grafo superou a representação TF-IDF, que obteve uma precisão de 30.64 e foi o melhor resultado entre todas as métricas.

Com o objetivo de melhorar o resultado obtido, estas métricas em conjuntos foram usadas como entrada para as técnicas de aprendizado de máquinas na tarefa de recomendação de *tags*.

5.2.6 Aprendizado de Máquina

Os resultados obtidos em três experimentos utilizando técnicas diferentes de aprendizado de máquina são apresentados. Cada experimento processa os mesmos dados, mas utiliza algoritmos

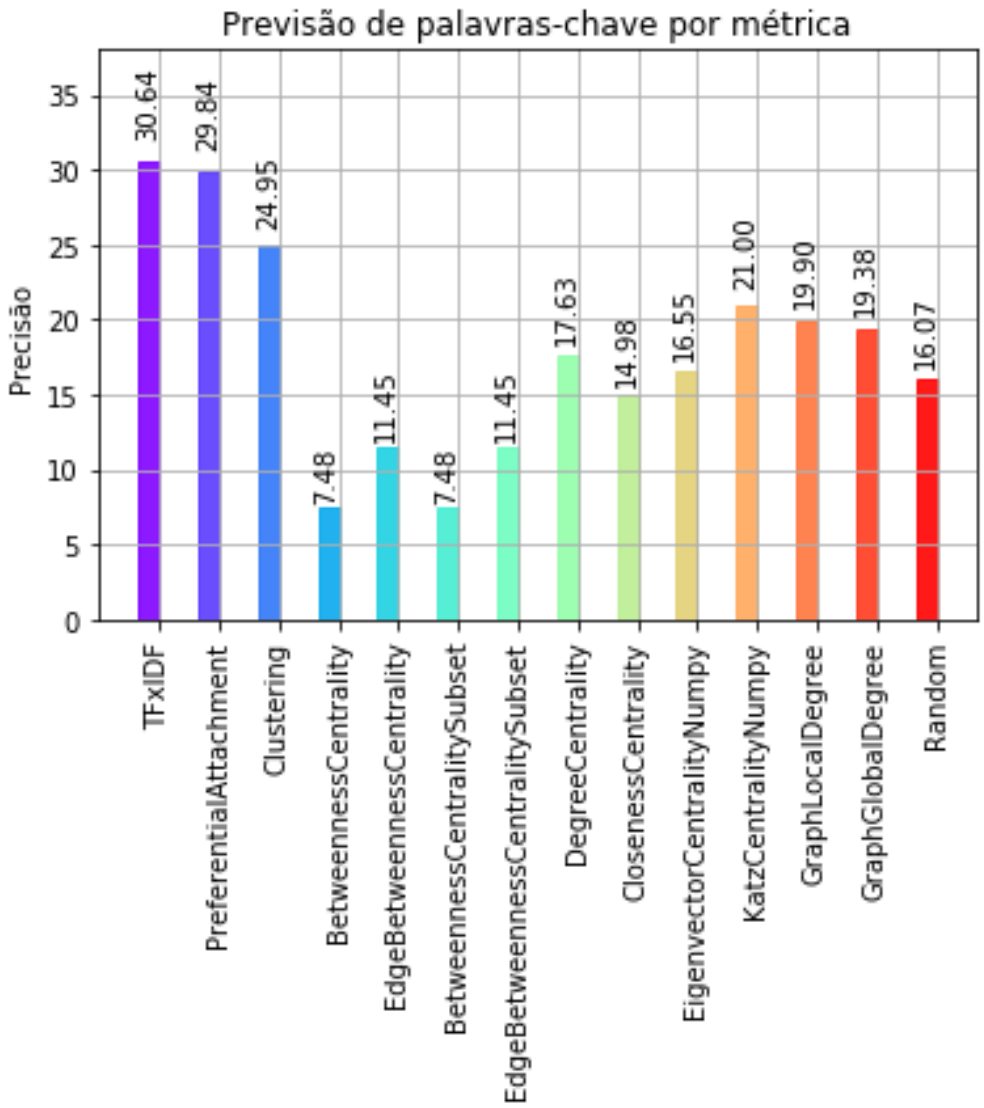

Figura 5.3: Resultado obtido com métricas individualmente

diferentes para processá-los. Os algoritmos utilizados nos experimentos são: *Máquinas de Vetor Suporte (SVM)*, *Random Forest* e *Regressão Logística*. Os dados processados pelos algoritmos foram formatados como uma matriz de palavras por resultados de métricas e sua construção é descrita a seguir.

5.2.7 Resultados Obtidos

Usando o mesmo grafo criado anteriormente, uma matriz M de palavras por métricas foi criada. Para cada filme F da coleção foi executado o procedimento abaixo:

1. Seja G a rede complexa construída anteriormente e G' o subgrafo de G referente ao filme F. O subgrafo G' foi selecionado utilizando-se o vocabulário do filme F que é composto do

conjunto V' de palavras dos campos $title$, $plot_outline$ e $plot$.

$$G = (V, A)$$

$$G' = (V', A')$$

$$V' \subseteq V$$

$$A' \subseteq A$$

2. As métricas de grafo foram calculadas para o subgrafo G'.

3. Para cada palavra $P \in V'$ foram calculadas as métricas de grafo listadas abaixo. São elas:

   (a) *katz Centrality*

   (b) *Eigenvector Centrality*

   (c) *Preferential Attachment*

   (d) *Clustering*

   (e) *Degree Centrality*

   (f) *Closeness*

   (g) *Betweenness*

   (h) *Graph Degree*

4. A métrica TF-IDF foi calculada utilizando o vocabulário V' do filme F.

5. Para as métricas dos itens 2 e 3 acima foi criada uma nova linha na matriz M, com a palavra P e os resultados das métricas.

6. Além das métricas acima, a matriz possui mais uma coluna, *IsKeyword*, que possui o valor 1 quando a palavra P é uma palavra-chave do filme F, ou 0, caso contrário.

Ao final, a matriz M resultante possui 2.104.337 linhas por 11 colunas. A Figura 5.4 mostra um exemplo extraído da matriz M referente ao filme *Toy Story* no qual

TI - *TF-IDF*
KC - *katz Centrality*
EC - *Eigenvector Centrality*
PA - *Preferential Attachment*
CL - *Clustering*
DC - *Degree Centrality*
CO - *Closeness*
BE - *Betweenness*
GD - *Graph Degree*
IK - *Is Keyword*

Para validar este experimento foi utilizado o método chamado *K-fold cross validation* que é um método de validação de modelos de predição que divide o conjunto de dados em K subconjuntos. O modelo então é executado K vezes. Cada vez, um dos subconjuntos k é usado como o conjunto

Word	TI	KC	EC	FA	CL	DC	CO	BE	GD	IK
buy	1.70678	0.08230	3.27E-14	1	0.08222	0.00014	0.03474	0.00360	17	0
accidentaly	3.98781	0.07550	6.85E-19	0	0.00000	0.00000	0.00000	0.00000	0	0
hots	3.42354	0.07550	3.43E-18	0	0.00000	0.00000	0.00000	0.00000	0	0
young	0.35139	0.14281	5.97E-12	11	0.05032	0.00072	0.03998	0.00709	88	0
party	1.18095	0.07550	4.21E-11	0	0.37082	0.00006	0.03085	0.00000	7	0
jealous	3.63315	0.08153	6.28E-13	1	0.09541	0.00018	0.03570	0.00360	22	0
cowboy	6.55855	0.07550	4.19E-10	0	0.00000	0.00001	0.03020	0.00000	1	1
favorite	7.95700	0.07550	-9.49E-19	0	0.55612	0.00002	0.02285	0.00000	2	0
woody	31.80536	0.07550	1.33E-17	0	0.00000	0.00002	0.02334	0.00000	3	0
takes	0.77955	0.08989	8.07E-14	2	0.23800	0.00012	0.03521	0.00185	15	0
becoming	1.53094	0.08799	2.79E-15	2	0.22075	0.00002	0.03190	0.00010	3	0
journey	1.20462	0.07550	1.55E-13	0	0.07888	0.00008	0.03491	0.00000	10	0
moving	1.74066	0.07983	4.01E-18	1	0.61199	0.00002	0.02873	0.00010	2	1
planet	1.43963	0.07950	2.27E-14	1	0.06005	0.00032	0.03543	0.00010	39	0
trying	1.02497	0.09548	7.06E-14	18	0.14275	0.00031	0.03526	0.00360	38	0

Figura 5.4: Amostra do filme Toy Story

de teste e os outros $k-1$ subconjuntos são concatenados para formar um conjunto de treinamento. A vantagem deste método é que ele é menos sensível a como os dados são divididos. Todo dado pertence a um conjunto de testes exatamente uma vez, e pertence a um conjunto de treinamento $k-1$ vezes. Para este experimento a matriz M foi dividida em cinco *folds*, na proporção de 80% para treino e 20% para teste.

Existe um viés no conjunto de dados utilizado. A maioria das palavras não são palavras-chave. Existem 1.998.458 palavras na matriz M que não são palavras-chave e 105.874 palavras na matriz M que são, de modo que aproximadamente 5% das palavras da matriz M, são palavras-chave de filmes, como mostra a Figura 5.5.

Analisar um conjunto de dados não-balanceado produz geralmente resultados ruins, pois o modelo tem dificuldade de identificar corretamente os resultados positivos e acaba produzindo um número muito grande de falsos negativos. Para lidar com este problema, um dos métodos existentes é o *downsampling*. O objetivo deste método é criar um conjunto de dados balanceado, combinando o número de amostras na classe minoritária com uma amostra aleatória da classe majoritária.

Neste trabalho a classe minoritária é a classe de palavras-chave e a classe majoritária é a de palavras que não são palavras-chave, como podemos ver na Figura 5.5. Utilizando o método de *downsampling* foi criado um conjunto de dados com as 105.874 palavras da classe minoritária mais 105.874 palavras, escolhidas aleatoriamente, da classe majoritária. O resultado foi uma nova matriz M' com 211.748 palavras e suas respectivas métricas. A matriz M' foi então processada pelos métodos *SVM*, *Random Forest* e *Regressão Logística* e os resultados estão apresentados na Tabela 5.1. Esta tabela também apresenta os resultados obtidos ao selecionar palavras-chave aleatoriamente e o desvio padrão do método.

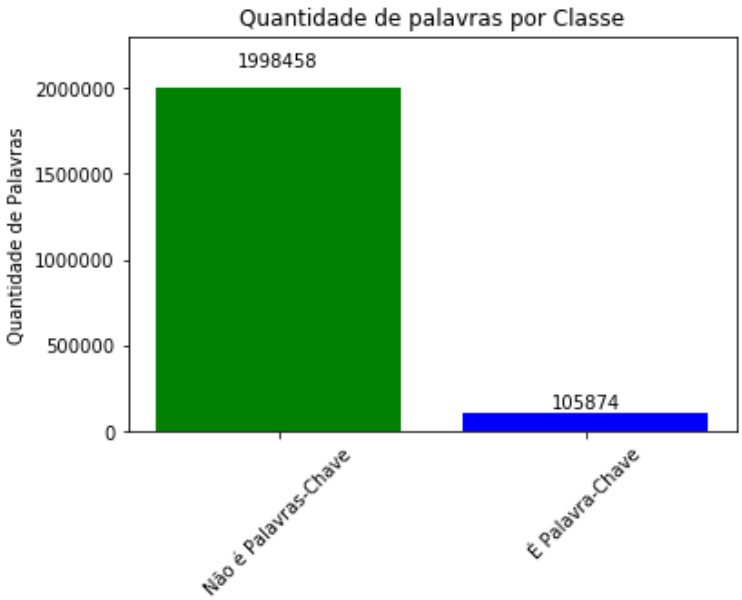

Figura 5.5: Viés no conjunto de dados

	Precisão	**Aleatório**	**Desvio Padrão**
Random Forest	0.855136	0.499089	0.000761
SVM	0.703920	0.500425	0.004594
Regressão Logística	0.647334	0.501034	0.002006

Tabela 5.1: Resultados obtidos pelos métodos *SVM*, *Random Forest* e *Regressão Logística*.

É possível ver, analisando a Tabela 5.1, a superioridade do método *Random Forest* sobre os outros métodos. *Random Forest* obteve um ganho superior a 21% em relação ao segundo lugar, o método *SVM* e superior 32% em relação ao terceiro colocado, o método *Logistic Regression*.

O desempenho obtido pelos métodos utilizados nesse trabalho é comparado na Figura 5.6. Como podemos ver, o método *Random Forest* é bastante superior aos métodos SVM e Regressão Logística, obtendo uma precisão de aproximadamente 85% contra uma precisão aproximada de 70% e 65% para os métodos SVM e Regressão Logística, respectivamente. Os *baselines Random* e TF-xIDF tem desempenho bastante fraco e ficam praticamente empatados com precisão aproximada de 50%.

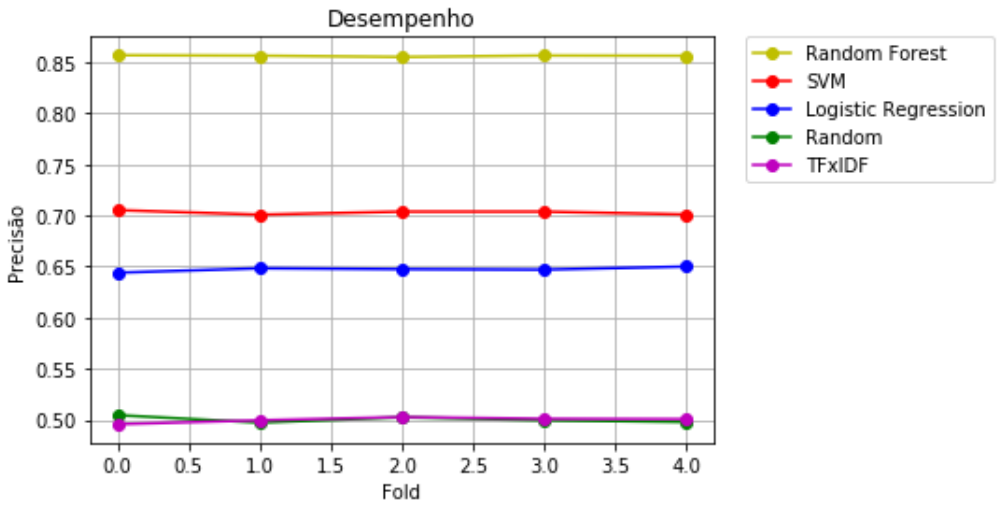

Figura 5.6: Desempenho dos métodos

Capítulo 6

RECOMENDAÇÃO DE ARTIGOS CIENTÍFICOS

A metodologia proposta para expansão do conjunto de dados utiliza vetores de palavras e propriedades de redes complexas para selecionar palavras para expandir este conjunto com o objetivo de melhorar os resultados dos métodos LDA e CTR. Estes métodos serão utilizados aqui para a recomendação de artigos científicos.

O processo proposto neste trabalho consiste em uma série de etapas que devem ser completadas, tal como a expansão do conjunto de dados, filtragem e o tratamento dos mesmos, antes que esse conjunto seja disponibilizado para processamento. Estas etapas serão descritas a seguir.

6.1 Etapas da Metodologia Proposta

A metodologia proposta na recomendação de artigos científicos funciona em seis etapas distintas, como mostradas na Figura 6.1: **(i)** representação da rede complexa, **(ii)** cálculo das métricas, **(iii)** expansão dos documentos a partir da rede complexa criada anteriormente, **(iv)** separação do conjunto de dados a ser processado, **(v)** seleção dos tópicos utilizando o método LDA e **(vi)** recomendação de artigos científicos utilizando o método CTR.

6.1.1 Conjunto de Dados

O conjunto de dados utilizado nessa etapa do trabalho são usuários e suas bibliotecas de artigos obtidos do site *CiteULike* [1]. No site *CiteULike*, usuários registrados criam bibliotecas de referência pessoais. Cada artigo geralmente tem um título e um resumo. Dois conjuntos de dados, *citeulike-t* e *citeulike-a*, fornecem estas informações, respectivamente. Outras informações sobre os artigos, como os autores, publicações e palavras-chave, não foram utilizadas neste trabalho.

[1] http://www.citeulike.org/faq/data.adp

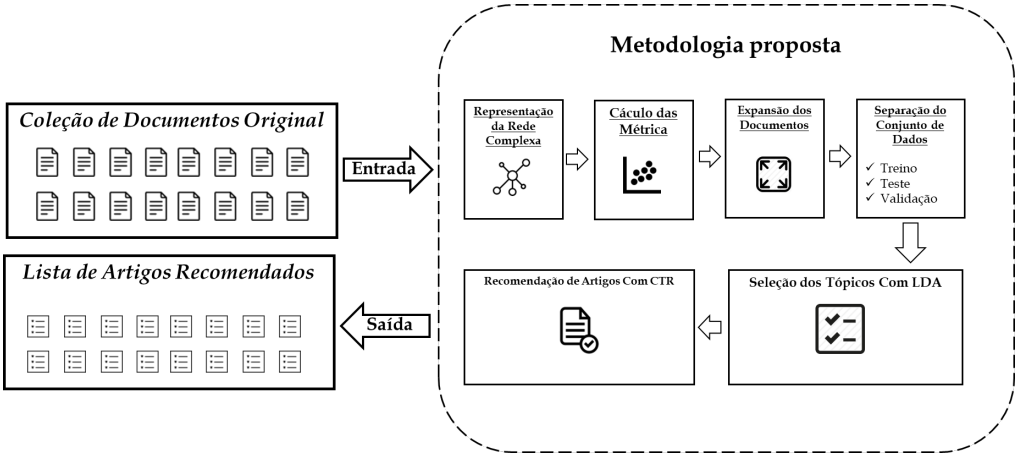

Figura 6.1: Visão geral da metodologia de recomendação de artigos científicos

Do conjunto de dados *citeulike-a* original, foram removidos *stop words* e usuários com menos de 10 artigos em sua biblioteca de referências pessoais. Como resultado foi obtido um conjunto de dados com 5.551 usuários, 16.980 artigos com 204.986 pares de itens-usuário observados.

Do conjunto de dados *citeulike-t* original, foram removidos *stop words* e usuários com menos de 3 artigos em sua biblioteca de referências pessoais. Como resultado foi obtido um conjunto de dados com 7.497 usuários, 25.975 artigos com 134.860 pares de itens-usuário observados.

Nos conjuntos de dados de destino, para cada artigo, seu título e resumo foram concatenados para formar uma única linha de texto.

6.1.2 Vetores de Palavras

Nesta etapa do trabalho os vetores de palavras utilizados são os vetores fornecidos pelo *Google* e gerados pelo programa *Word2vec* a partir de palavras e frases do conjunto de dados do *Google News* [2] (cerca de 100 bilhões de palavras). O modelo contém vetores de 300 dimensões para 3 milhões de palavras e frases. No capítulo 5 os vetores de palavras foram gerados pelo *word2vec* a partir do conjunto de dados IMDb. Para este experimento optou-se por utilizar os vetores de palavras fornecidos pelo *Google* uma vez que o tamanho do conjunto de dados utilizado para treinamento influencia a qualidade dos vetores gerados. Somados os conjuntos *citeulike-a* e *citeulike-t* possuem menos de 8 milhões de palavras contra cerca de 100 bilhões de palavras utilizadas para treinar o *Word2vec* pelo *Google*, o que justifica a escolha dos vetores do *Google*.

6.1.3 Representação da Rede Complexa

Assim como no Capítulo 5, uma rede complexa será criada para esta etapa do trabalho, mas desta vez os vértices do grafo $G = (V, A)$ são criados utilizando o vocabulário do conjunto de

[2] https://code.google.com/archive/p/word2vec/

dados *CiteULike*, denominado V, onde cada palavra representa um vértice. Uma aresta entre duas palavras é criada somente se o cosseno entre os vetores que as representam estas duas palavras for maior ou igual a 0,5.

6.1.4 Cálculo das Métricas

Nessa etapa, para cada texto T da coleção, o procedimento abaixo é executado

1. Seja G a rede complexa construída e G' o subgrafo de *G* referente ao texto T. O subgrafo G' selecionado utilizando-se o vocabulário do texto T.

$$G = (V, A)$$

$$G' = (V', A')$$

$$V' \subseteq V$$

$$A' \subseteq A$$

2. As métricas de grafo são calculadas para o subgrafo G'. Para cada palavra $P \in V'$ são calculadas as métricas de grafo listadas abaixo. São elas:

   (a) *Betweenness Centrality*

   (b) *Closeness Centrality*

   (c) *Clustering*

   (d) *Degree Centrality*

   (e) *Degree*

   (f) *Eigenvector Centrality*

   (g) *Preferential Attachment*

6.1.5 Expansão dos Documentos

Os documentos são expandidos adicionando-lhes novas palavras. As expansões foram feitas em quatro versões diferentes, 25%, 50%, 75% e 100% de expansão. Para expandir os documentos, o processo gerador descrito na figura 6.2 é executado para cada métrica listada na seção 6.1.4.

Observe que o processo gerador mostrado na Figura 6.2 seleciona somente novas palavras para serem adicionadas ao documento. O conjunto de dados resultante que será a entrada dos métodos LDA e CTR. Este conjunto é dividido em três conjunto de dados distintos; treino, teste e validação, como descrito a seguir.

1. Para cada documento T, extraia seu subgrafo $G' = (V', A')$

a) Para cada aresta $A' \in G'$

a.1) Para cada aresta A'', conectada à aresta A', calcule o peso $w = \mathrm{W}(A'') * M(A'')$. Onde $\mathrm{W}(A'')$ é o peso da aresta A'' e $M(A'')$ é o valor da métrica calculada para o termo $A'' \notin V'$, em relação ao grafo G.

b) Extraia as N palavras mais bem classificadas, de acordo com seu peso w calculados no passo (a.1) acima. N é número de palavras necessárias para atingir o grau de expansão solicitado.

c) Adicione a final do documento T a lista de N palavras selecionada acima.

Figura 6.2: Processo gerador para o documento expandido

6.1.6 Separação do Conjunto de Dados

Para a tarefa de recomendação, são selecionados aleatoriamente P itens da coleção do usuário, para cada usuário, para formar o conjunto de treinamento e usar o restante dos itens como o conjunto de testes e validação. Para o treinamento, consideramos tanto a configuração esparsa, de $P = 1$ até $P = 5$, quanto a configuração densa $P = 10$. Dizer que $P = 1$ significa que o conjunto de treino tem 1 artigo para cada usuário, $P = 2$ significa dois artigos para treino e assim por diante. Para o experimento, cada usuário é apresentado, hipoteticamente, com M artigos ordenados por sua classificação e avaliados com base em quais desses artigos pertencem ou não a coleção do usuário. Para cada valor de P, a avaliação é realizada 5 vezes com diferentes divisões aleatórias.

6.1.7 Experimentos e Resultados

Neste trabalho, a recomendação de artigos científicos usando CTR será comparado a um conjunto de artigos e classificações feitas pelos usuários. Hipoteticamente, os M artigos com melhor classificação pelo CTR são apresentados para cada usuário. Verifica-se então quais destes artigos apresentados estão contidos na biblioteca de cada usuário. Duas métricas possíveis para comparação são precisão e revocação. No entanto, como foi dito anteriormente, as classificações zero são incertas. Eles podem indicar que um usuário não gosta de um artigo ou não sabe sobre ele. Isso dificulta o cálculo da precisão. Mas, uma vez que as classificações existentes são verdadeiras positivas, focamos na revocação. A revocação considera apenas os M artigos positivamente classificados. Para cada usuário, a definição de revocação é

$$Revocacao@M = \frac{\textit{Total de artigos avaliados pelo usuário e que estão entre os M mais bem avaliados}}{\textit{Total de artigos avaliados pelo usuário}}$$

A revocação final de todo o experimento será a revocação média de todos os usuários.

A parametrização do CTR segue os valores proposto em [2], onde é descrito que o CTR pode alcançar um bom desempenho quando $a = 1, b = 0.01, \lambda_u = 0.1$, onde a e b são parâmetros de ajuste satisfazendo $a > b > 0$. Além disso, o parâmetro de precisão λ_v equilibra como o vetor latente v_j diverge das proporções de tópico θ_j. Variamos o parâmetro no seguinte intervalo $\lambda_v \in 10, 100, 1000, 10000$.

O primeiro experimento realizado comparou os resultados dos conjuntos de dados expandidos quando $P = 1$ e $P = 10$ e $M \in \{25, 50, 75, 100, 125, 150, 175, 200\}$. Hipoteticamente, os M artigos com melhor classificação pelo CTR são apresentados para cada usuário. Verifica-se então quais destes artigos apresentados estão contidos na biblioteca de cada usuário.

A Figura 6.3 mostra os resultados obtidos entre as métricas utilizadas quando P, a quantidade de artigos usadas para treinar o modelo, é igual a 1 ou 10. Nas figuras, o eixo x representa o número de artigos apresentados para cada usuário e o eixo y representa o valor do revocação.

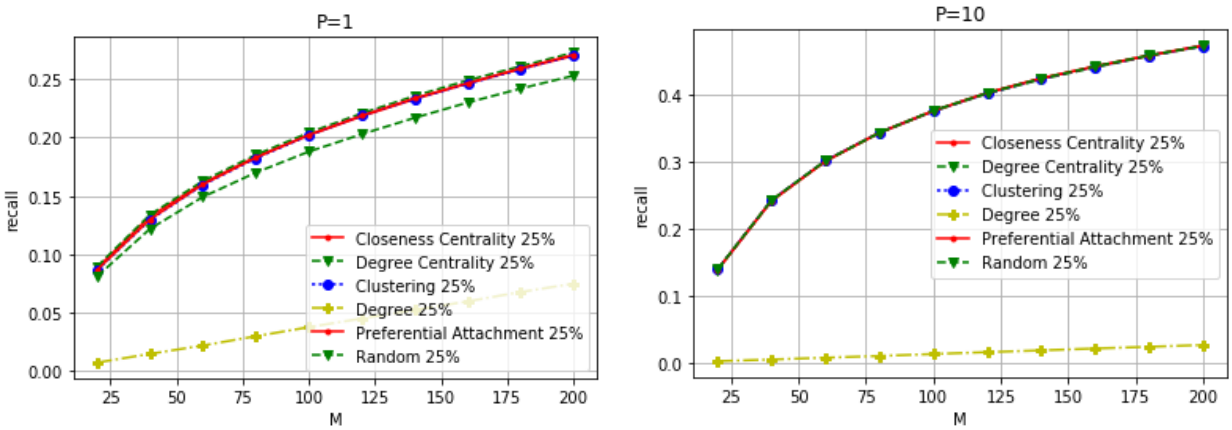

Figura 6.3: Comparação entre métricas quando P = 1 e P = 10

Pode-se concluir, a partir dos resultados obtidos, que:

1. Expandir o conjunto de dados em mais de 25% não altera de forma significativa o resultado do experimento, como pode-se ver na Tabela 6.1.
2. Verificou-se também que a métrica que obteve o melhor resultado foi *degree centrality*, seguida por *closseness centrality*.
3. Não há ganho significativo de desempenho quando $P = 10$ em relação ao conjunto regular, veja Figura 6.4.

Analisando estes resultados percebeu-se que poderia haver um ganho em relação ao conjunto de dados regular quando $P = 1$, veja a Figura 6.4. Ou seja, quando há apenas um artigo para treinar o modelo CTR. Esta situação é chamada de *cold start* é e definida na próxima seção.

A partir das conclusões obtidas com o experimento inicial, decidiu-se que:

1. Os experimentos seriam repedidos com $P \in \{1, 2, 3, 4, 5\}$ para comparar os resultados em relação ao conjunto de dados regular quando $P = 1$ e $M \in \{50, 100, 150, 250, 300\}$. O objetivo é melhorar o desempenho do algoritmo que executa os experimentos.

2. Os experimentos seriam repedidos apenas para o conjunto de dados expandido em 25% com métrica *degree centrality*, uma vez que foi esta métrica que obteve melhor performance.

	Revocaçao Média em P=1		**Revocaçao Média em P=10**	
Expansao	**Degree Centrality**	**Closseness Centrality**	**Degree Centrality**	**Closseness Centrality**
25%	0.199390779	0.201529983	0.360484799	0.359872674
50%	0.200949484	0.199926851	0.360813943	0.360604816
75%	0.201049092	0.201872140	0.360515609	0.359933033
100%	0.200099311	0.198121156	0.360295279	0.358754245
Média	0.200372166	0.200362532	0.360527407	0.359791192

Tabela 6.1: Comparação entre métricas quando P = 1 e P = 10

6.1.8 *Cold Start*

O problema de *cold start* consiste no processo de recomendação para um usuário que tem pouco ou nenhum histórico passado. Para simular essa condição, neste experimento, o histórico atual do usuário foi dividido de um modo que, de 1 a 5 itens foram usados para treinamento e o restante do histórico foi usado para teste e validação.

O objetivo deste experimento é comparar os resultados obtidos de um conjunto de dados expandido, usando vetores de palavras e propriedades de rede complexas para expandi-lo, com os resultados obtidos do mesmo conjunto de dados, sem expansão, como descrito na seção 6.1.5. A partir desta comparação concluir como um conjunto de dados expandido pode ser usado para melhorar o desempenho do modelo CTR para uso no problema de *cold start*.

O conjunto de dados usado neste experimento é o *citeulike-a* expandido em 25%. Este conjunto de dados foi expandido usando as palavras mais bem classificadas do vocabulário do conjunto de dados de acordo com a métrica *degree centrality*, sobre o grafo criado com os vetores *Word2Vec* do Google.

A Figura 6.4 compara os resultados obtidos utilizando o conjunto de dados descrito acima, expandido em 25%, quando o número de artigos para treinamento do modelo CTR varia de $P = 1$ a $P = 5$, com os resultados obtidos com o conjunto de dados regular quando $P = 1$ e $P = 10$. A Tabela 6.2 mostra a revocação média, por conjunto de dados e o número de artigos de treinamento.

A Figura 6.5 mostra o ganho percentual da revocação de cada conjunto de dados expandido, de $P = 1$ a $P = 5$, em relação ao conjunto de dados regular com $P = 1$. O ganho médio da revocação é 5,24061674785.

A Figura 6.6 abaixo mostra o ganho percentual da revocação de cada conjunto de dados expandido, de $P = 1$ a $P = 5$, em relação ao conjunto de dados regular com $P = 10$. O ganho médio da revocação é 3,38952647925.

Quando o conjunto de dados é expandido usando 50%, 75% e 100%, os resultados não são diferentes. A Figura 6.7 apresenta os resultados do percentual de expansão utilizado. Como é

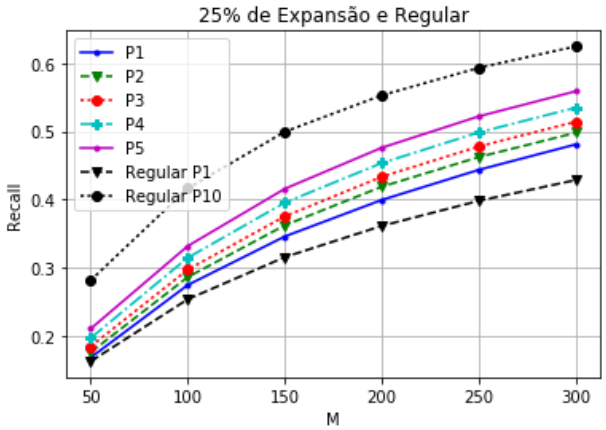

Figura 6.4: Degree centrality e Regular

Conjunto de Dados	Revocaçao Média
Degree Centrality 25% - P1	0.352164407
Degree Centrality 25% - P2	0.367329642
Degree Centrality 25% - P3	0.380499840
Degree Centrality 25% - P4	0.399044512
Degree Centrality 25% - P5	0.419254378
Regular - P1	0.320048071
Regular - P10	0.494832919

Tabela 6.2: Revocação Média por Conjunto de Dados

Figura 6.5: Ganho Percentual em Relação ao Conjunto Regular quando P = 1

possível ver, quase não há diferenças entre os conjuntos de dados expandidos. Observe que o mesmo acontece quando $P = 2$, $P = 3$, $P = 4$, $P = 5$. A Tabela 6.3 demonstra em números essa observação.

A coluna *Relação Média de Revocação* da Tabela 6.3 é a relação da coluna *Revocação Média*, do conjunto de dados corrente, com a coluna Revocação Média do conjunto de dados regular, quando $P = 1$. A coluna % fornece, em porcentagem, a razão da revocação média, da linha especificada,

Figura 6.6: Ganho Percentual em Relação ao Conjunto Regular quando P = 10

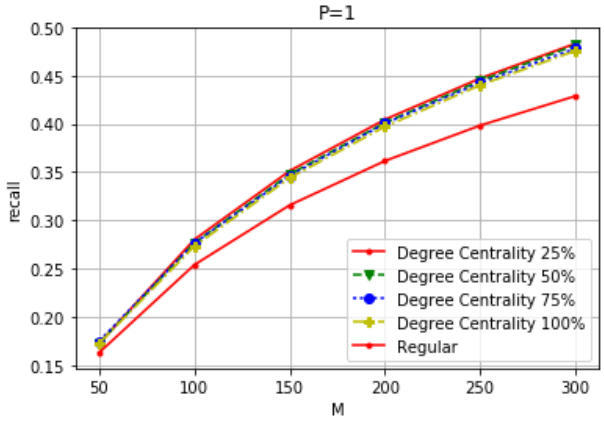

Figura 6.7: Comparação de Percentual de Expansão

relacionada à linha correspondente do conjunto de dados expandido de 25%. A maioria dos valores está na terceira casa e apenas alguns estão na segunda casa decimal. Portanto, podemos concluir que a expansão do conjunto de dados em mais de 25% não melhora significativamente os resultados, como podemos ver na Figura 6.8, que representa a relação de ganho do conjunto de dados expandido em relação ao conjunto regular, veja a coluna *Relação Regular* da Tabela 6.3.

Todos os gráficos referentes às tabelas anteriores podem ser vistos na Figura 6.9. Podemos concluir, analisando os gráficos e as tabelas deste capítulo, que a técnica aqui proposta pode melhorar o desempenho do modelo CTR para o problema de *cold start*. Em relação ao conjunto de dados regular, com $P = 1$, o conjunto de dados expandido obteve um ganho de 10%.

Como esperado, quando P aumenta, o desempenho também aumenta, com um ganho médio de cerca de 5% em relação ao P anterior. No entanto, quando $P = 10$, o desempenho aumenta um pouco mais lentamente em relação ao conjunto de dados regular, como é possível ver na Tabela 6.3. Nesta situação, o ganho médio é de cerca de 3%. Isso demonstra os limites do método proposto, que vai diminuindo sua contribuição para a performance do modelo CTR à medida que mais dados são disponibilizados para treino. Outra conclusão importante é que expandir o conjunto de dados em mais que 25% não acrescenta melhoria significativa no desempenho da metodologia proposta.

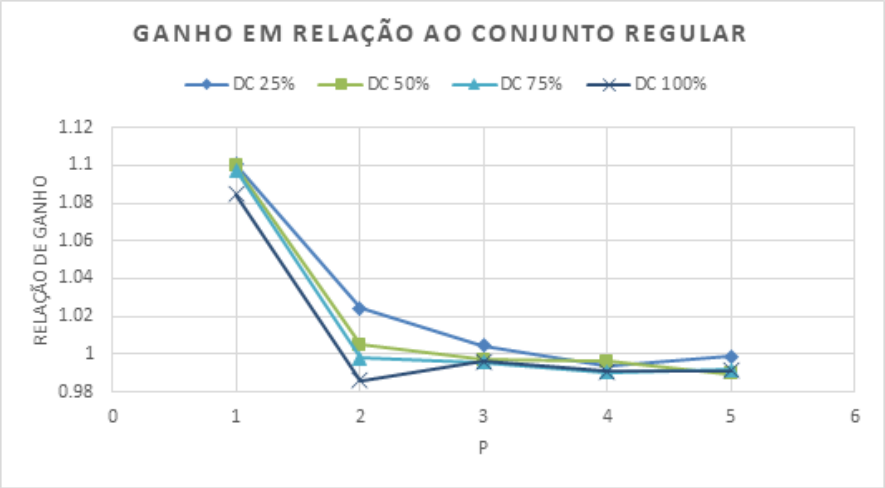

Figura 6.8: Comparação de Percentual de Expansão: ganho obtido em relação ao conjunto regular

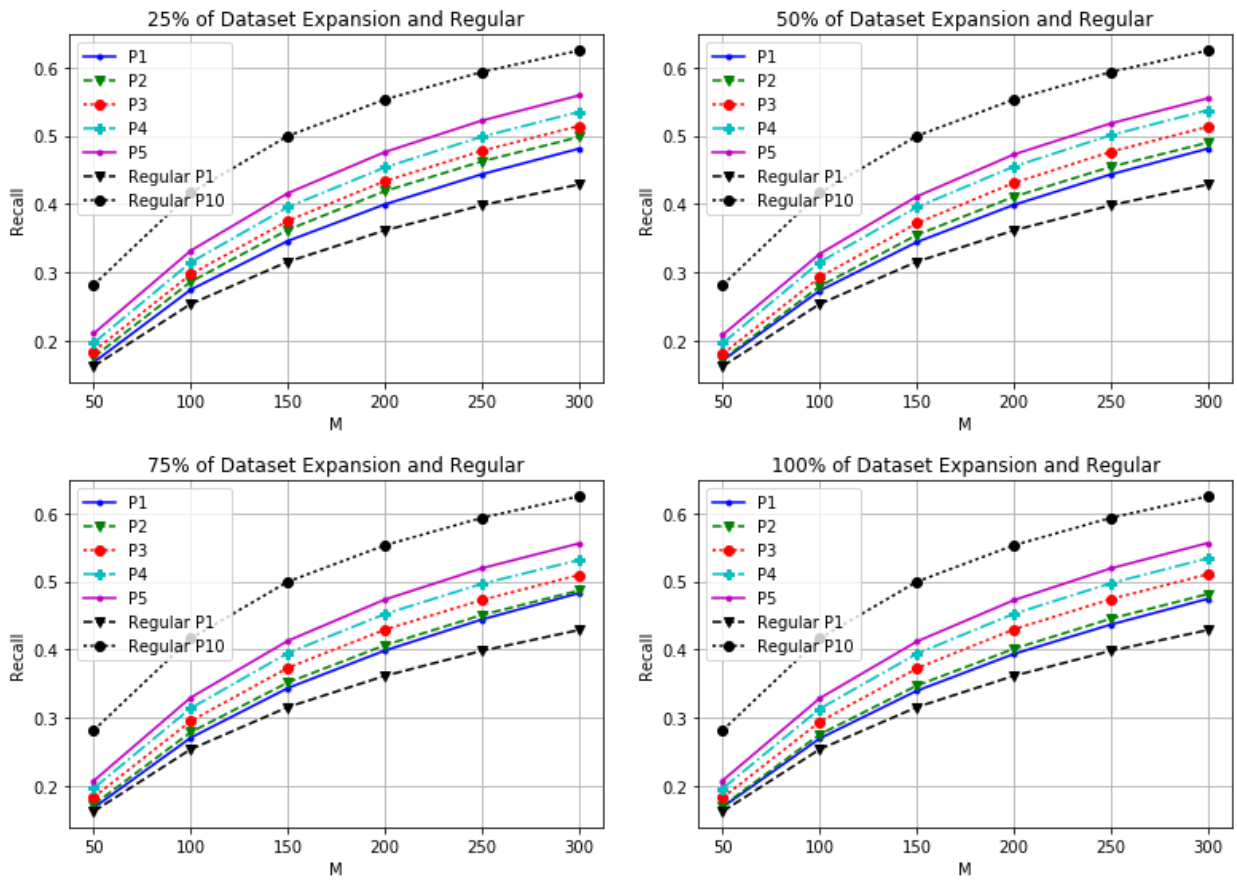

Figura 6.9: Comparação entre Percentuais de Expansão

6.1.9 Expansão em 5%

Após observarmos que expandir o conjunto de dados em mais de 25% não acrescenta melhorias significativas, foi realizado um experimento onde o conjunto de dados foi expandindo em 5%,

afim de verificarmos de uma expansão inferior a 25% poderia adicionar um ganho ao processo. O resultado do experimento está mostrado Figura 6.10.

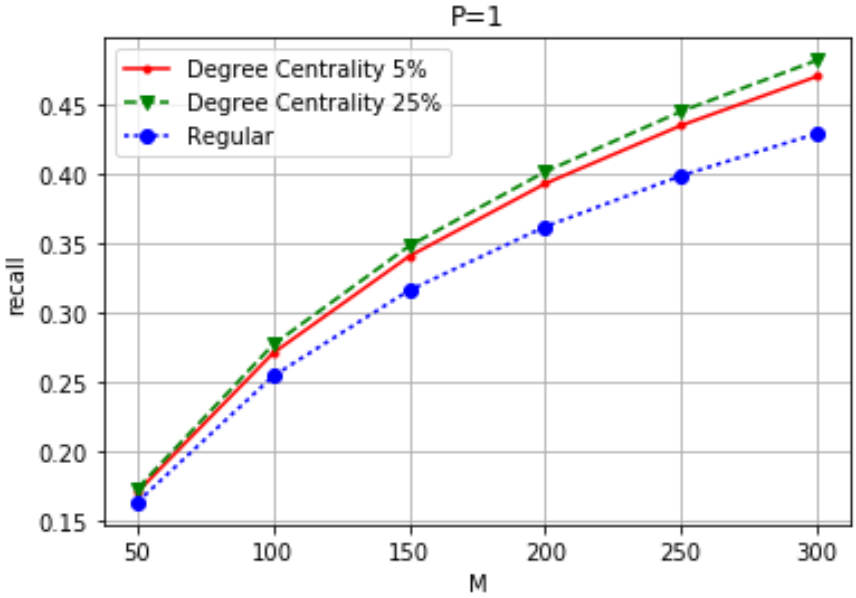

Figura 6.10: Expansão de 5%

Expandir o conjunto de dados em menos de 25% não acrescenta eficiência ao processo como é possível observar na Figura 6.10. Desta forma, não será mais considerado em futuros experimentos expansões diferentes de 25%.

6.1.10 Comparação com Outros Métodos

Nesta etapa foi comparado os resultados obtidos com o modelo CTR e o conjunto de dados expandido em 25% com os métodos:

1. **SLIM** - Supersparse Linear Integer Model [77]
2. **BPR** - Bayesian Personalized Ranking [78]
3. **MostPopular** - Gera uma classificação de artigos mais populares entre os usuários.
4. **Random** - Gera aleatoriamente uma lista de artigos.

Foi utilizado a biblioteca *MyMediaLite* (MLT) [79], que possui os modelos acima implementados, para executar esta etapa do experimento. Como entrada para os métodos acima, utilizou-se os mesmos conjuntos de dados de treino e teste utilizados no experimento com o CTR e as métricas foram calculadas utilizando as mesmas rotinas utilizadas para calcular as métricas na saída do CTR nos experimentos anteriores. O objetivo com esta abordagem é homogeneizar as entradas

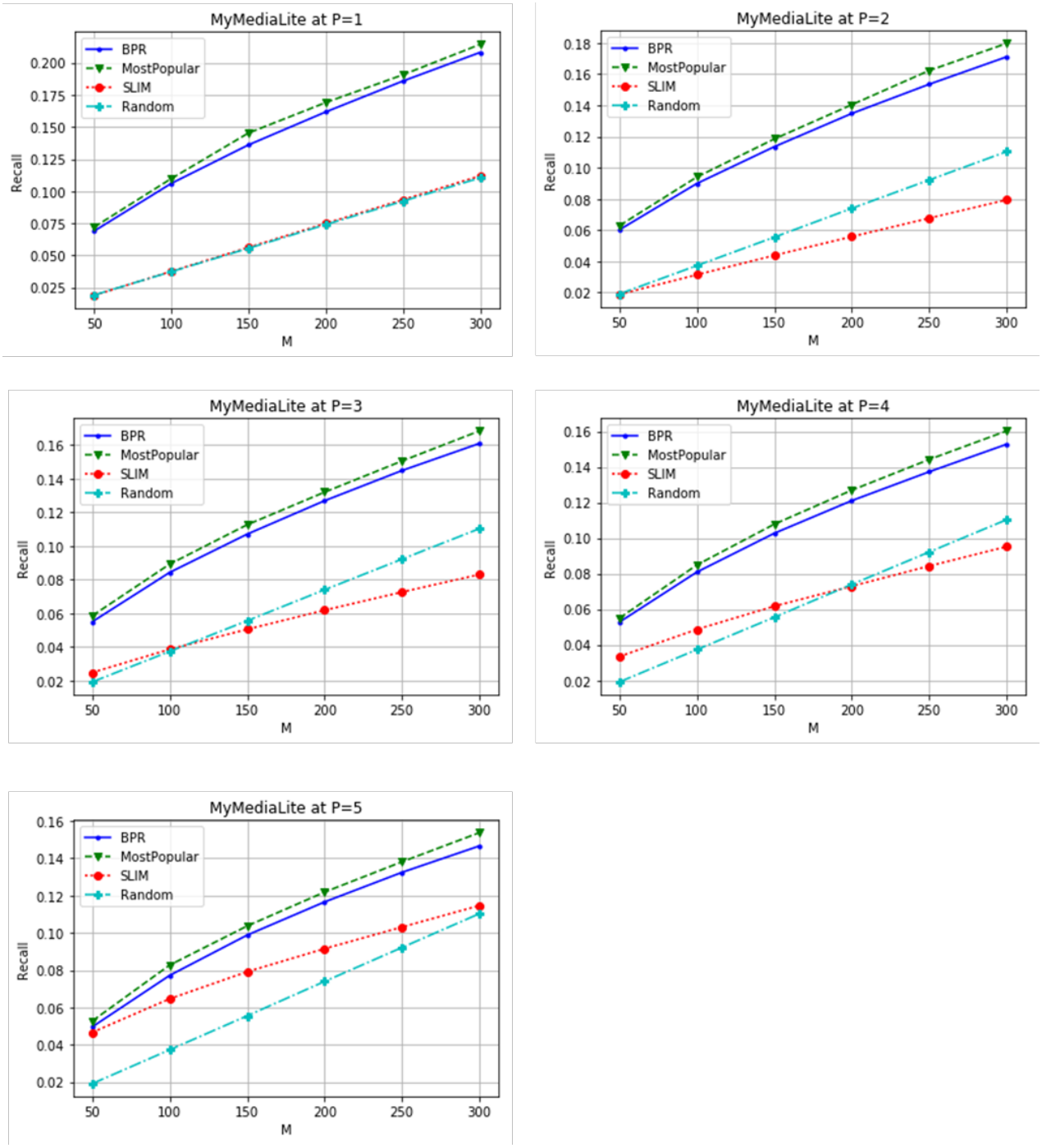

Figura 6.11: Modelos MyMediaLite

de dados e o cálculo das métricas e assim garantir que as diferenças de resultados são apenas relacionadas as metodologias de recomendação utilizadas.

A Figura 6.11 compara os resultados obtidos com os modelos do MLT, de $P = 1$ até $P = 5$ utilizando o conjunto de dados regular.

Na Figura 6.12, os resultados obtidos anteriormente foram comparados com o resultado obtido com o modelo CTR, quando $P = 1$, utilizando o conjunto de dados regular. Foi utilizado $P = 1$

apenas pois com um único artigo para treino é que a metodologia proposta, ou seja, expandir o conjunto de dados visando melhorar a performance do modelo CTR, obteve melhor resultado.

Uma vez que nenhum método da biblioteca MLT obteve performance superior ao CTR, utilizando o conjunto de dados regular, não foi necessário comparar os resultados com o conjunto de dados expandido. É importante observar que entre todos os métodos da biblioteca MLT apenas o SLIM não obteve sua melhor performance em $P = 1$. Isto demonstra o potencial de alguns métodos em relação ao problema de *cold start*.

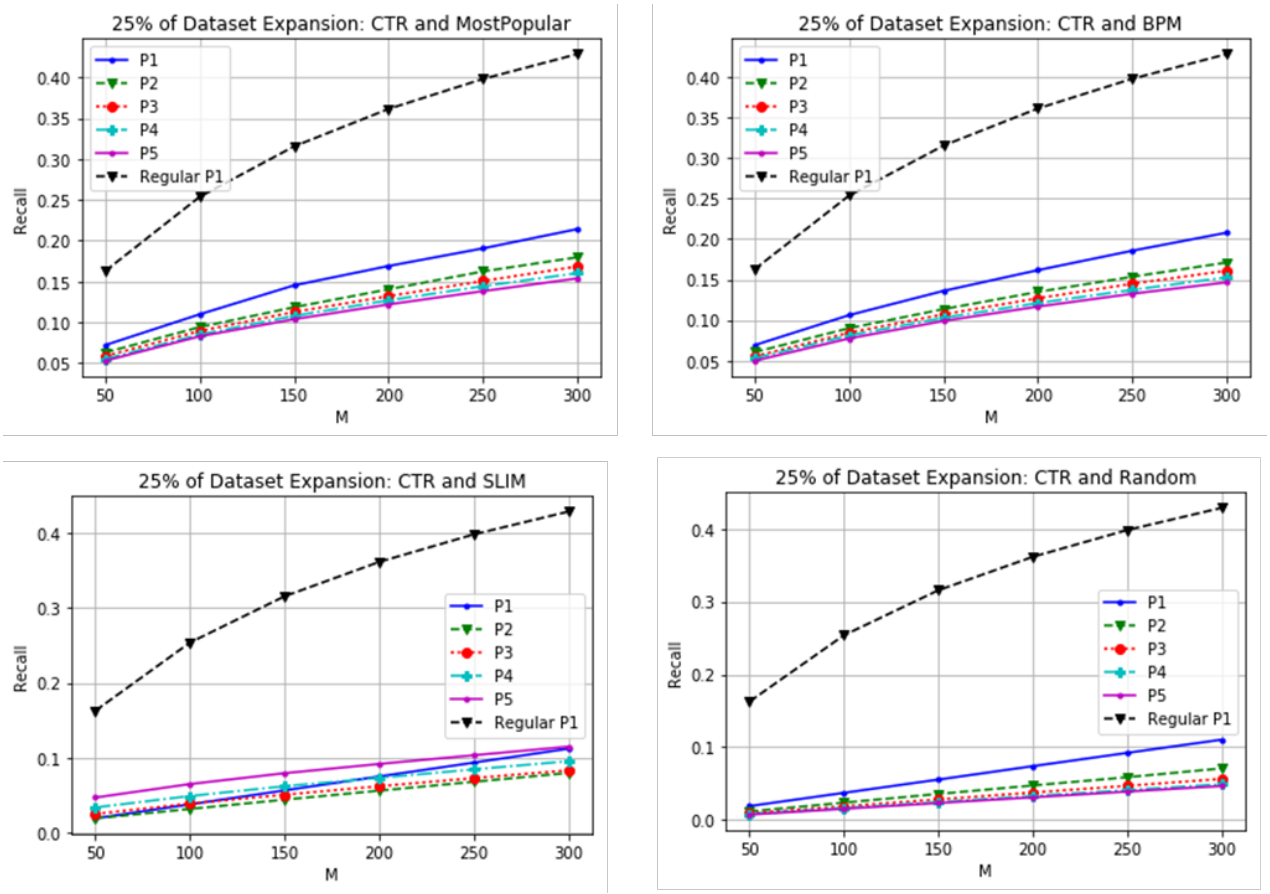

Figura 6.12: Comparação CTR e MyMediaLite

Conjunto de Dados	**Revocaçao Media**	**Relaçao Média de Revocaçao**		**Relaçao Regular**
Degree Centrality 25% - P1	0.352164407	1.100348476		1.100348476
Degree Centrality 25% - P2	0.367329642	1.147732718		1.024357219
Degree Centrality 25% - P3	0.380499840	1.188883404		1.004390240
Degree Centrality 25% - P4	0.399044512	1.246826799		0.993792928
Degree Centrality 25% - P5	0.419254378	1.309973146		0.998595353
Regular - P1	0.320048071			
Regular - P10	0.494832919			

Conjunto de Dados	**Revocaçao Média**	**Relaçao Média de Revocaçao**	**%**	**Relaçao Regular**
Degree Centrality 50% - P1	0.351969184	1.099738496	0.000554	1.099738496
Degree Centrality 50% - P2	0.360361072	1.125959207	0.018971	1.004924251
Degree Centrality 50% - P3	0.377707418	1.180158394	0.007339	0.997019193
Degree Centrality 50% - P4	0.400027918	1.249899482	-0.002464	0.996242034
Degree Centrality 50% - P5	0.415390197	1.297899393	0.009217	0.989391505

Conjunto de Dados	**Revocaçao Média**	**Relaçao Média de Revocaçao**	**%**	**Relaçao Regular**
Degree Centrality 75% - P1	0.351264819	1.097537686	0.002554	1.097537686
Degree Centrality 75% - P2	0.357824435	1.118033407	0.025877	0.997850435
Degree Centrality 75% - P3	0.377224865	1.178650643	0.008607	0.995745418
Degree Centrality 75% - P4	0.397622002	1.242382124	0.003565	0.990250265
Degree Centrality 75% - P5	0.416437049	1.301170313	0.006720	0.991884935

Conjunto de Dados	**Revocaçao Média**	**Relaçao Média de Revocaçao**	**%**	**Relaçao Regular**
Degree Centrality 100% - P1	0.347174760	1.084758170	0.014169	1.084758170
Degree Centrality 100% - P2	0.353562367	1.104716444	0.037479	0.985964978
Degree Centrality 100% - P3	0.377288516	1.178849523	0.008440	0.995913436
Degree Centrality 100% - P4	0.397900282	1.243251619	0.002867	0.990943303
Degree Centrality 100% - P5	0.416108389	1.300143407	0.007504	0.991102122

Conjunto de Dados	**Revocaçao Média**	**Relaçao Média de Revocaçao**	**%**
Regular - P1	0.320048071	1	0.091197
Regular - P2	0.358595259	1.120441870	0.023778
Regular - P3	0.378836656	1.183686736	0.004371
Regular - P4	0.401536880	1.254614280	-0.006246
Regular - P5	0.419844111	1.311815784	-0.001407

Tabela 6.3: Comparação de Percentual de Expansão

Capítulo 7

CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, abordamos o problema de recomendação de *tags* e recomendação de artigos científicos e seus desafios. Uma revisão de literatura foi realizada, onde apresentamos os trabalhos mais representativos no campo de sistemas de recomendação e discutimos suas limitações e similaridades em relação a este trabalho. Conhecendo os trabalhos existentes propusemos um arcabouço baseado em vetores de palavras e redes complexas para criar sistemas de recomendação e mostramos sua robustez, eficácia e limitações.

7.1 Recomendação de *Tags*

Os métodos de recomendação de *tags* foram avaliados utilizando o conjunto de dados obtido do *site* de avaliação de filmes IMDb. Foram dois experimentos distintos, sendo o primeiro não supervisionado, recomendando *tags* utilizando diretamente as métricas de redes complexas e o segundo método, supervisionado, utilizando metodologias de aprendizado de máquina. Os resultados mostraram que combinar as métricas de rede é mais eficiente do que usá-las separadamente para recomendar *tags* nos filmes do conjunto de dados IMDb. Para os métodos não supervisionados, o melhor resultado foi obtido pela métrica TF-IDF, que não é métrica de rede complexa, e que obteve uma precisão de aproximadamente 31%, o que evidencia as limitações deste arcabouço, veja Figura 5.3. Este resultado é muito inferior ao que foi obtido usando o algoritmo *Random Forest* que foi de aproximadamente 85% no cenário supervisionado.

Apesar dos resultados encontrados utilizando aprendizado de máquina mostrarem todo o potencial da abordagem desenvolvida é necessário refazer os experimentos com outros conjuntos de dados e comparar os resultados com um *baseline* forte para sabermos o quão eficiente é a metodologia proposta.

7.2 Recomendação de Artigos Científicos

Para a recomendação de artigos científicos o método proposto de expansão de documentos foi avaliado utilizando o conjunto de dados disponibilizado pelo site CiteULike, que consiste de seus usuários e suas bibliotecas de artigos científicos. Podemos concluir, a partir dos experimentos realizados, que a técnica aqui proposta pode melhorar o desempenho do modelo CTR para o problema de *cold start*. Em relação ao conjunto de dados regular, com $P = 1$, o conjunto de dados expandido obteve um ganho de 10%, o que pode ser considerado um bom resultado. Mas, em relação a $P = 10$, não houve ganho significativo o que pode demonstrar uma limitação da metodologia proposta.

Outra conclusão importante é que expandir o conjunto de dados em mais de 25% não acrescenta melhoria significativa no desempenho da metodologia proposta. Talvez, melhorar o aproveitamento das métricas de rede complexa para expansão do conjunto de dados, possa causar uma melhor influência, uma vez que o algoritmo proposto para expansão se baseia no grau do vértice. Observa-se que isto teve uma influência muito grande na escolha das palavras selecionadas para expandir o conjunto de dados.

7.2.1 Trabalhos Futuros

Como trabalhos futuros, as seguintes tarefas podem ser realizadas: **(1)** Realizar novos experimentos utilizando o conjunto de dados MovieLens 20M. **(2)** Selecionar um baseline de recomendação de *tags* para comparação. **(3)** Criar outras redes complexas que utilizam valores de *threshold* para o cosseno diferentes de 0,5. **(4)** Melhorar o aproveitamento das métricas de rede complexa para expansão do conjunto de dados.

1. Novos experimentos utilizando o conjunto de dados *MovieLens* 20M. Repetir o experimento utilizando um novo conjunto de dados para poder determinar qual é influência do mesmo sobre o resultado obtido. As atividades que compõem esta etapa são as mesmas descritas no Capítulo 5. De pose do grafo os experimentos serão executados novamente, mas agora com o novo conjunto dados, e os resultados comparados com os resultados obtidos com o conjunto de dados do IMDb.

2. Selecionar um *baseline* forte. Para podermos determinar o quão eficiente é a metodologia desenvolvida nesse trabalho precisamos compará-la com um método sabidamente eficiente para de recomendação de *tags*. Nesta etapa as atividades serão:

   - Pesquisa e desenvolvimento de um *baseline* forte para comparação;
   - Criação de um experimento que utiliza este método para processar os conjuntos de dados IMDb e *MovieLens*;
   - Comparar os resultados com o experimento anterior.

3. Experimentar com redes complexas criadas com *threshold* diferentes de 0,5. Em todos os experimentos realizados neste trabalho foi utilizado uma rede complexa criada a partir

de vetores de palavras. Nessa rede complexa uma aresta foi criada entre duas palavras se, e somente se, o cosseno entre elas era maior ou igual a 0,5. Este valor foi encolhido intuitivamente e portanto não teve nenhum subsídio teórico ou experimental. Sendo assim é necessário repetir os experimentos aqui realizados utilizando outras redes complexas criadas com valores de *thresholds* diferentes de 0,5 e assim podermos determinar qual o valor ótimo para este parâmetro.

4. Melhor aproveitamento das métricas de redes complexas. Melhorar o aproveitamento das métricas de rede complexa para expansão do conjunto de dados, uma vez que o algoritmo proposto para expansão baseasse fortemente no grau do vértice; o que teve uma influência muito grande na escolha das palavras selecionadas para expandir o conjunto de dados.

Referências Bibliográficas

[1] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. Expert Syst. Appl., 39(11):10059–10072, September 2012.

[2] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, KDD, pages 448–456. ACM, 2011.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 2005.

[4] Kerstin Bischoff, Claudiu S. Firan, Wolfgang Nejdl, and Raluca Paiu. Can all tags be used for search? In CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management, pages 193–202, New York, NY, USA, 2008. ACM.

[5] X. Li, L. Guo, and Y.E. Zhao. Tag-based social interest discovery. 2008.

[6] Flavio Figueiredo, Henrique Pinto, Fabiano Bélem, Jussara M. Almeida, Marcos André Gonçalves, David Fernandes, and Edleno Silva de Moura. Assessing the quality of textual features in social media. Inf. Process. Manage., 49(1):222–247, 2013.

[7] Yang Song, Lu Zhang, and C. Lee Giles. Automatic tag recommendation algorithms for social recommender systems. TWEB, 5(1):4, 2011.

[8] Sérgio D. Canuto, Fabiano Muniz Belém, Jussara M. Almeida, and Marcos André Gonçalves. A comparative study of learning-to-rank techniques for tag recommendation. JIDM, 4(3):453–468, 2013.

[9] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. CoRR, abs/1006.5278, 2010.

[10] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In THE ADAPTIVE WEB: METHODS AND STRATEGIES OF WEB PERSONALIZATION. VOLUME 4321 OF LECTURE NOTES IN COMPUTER SCIENCE, pages 325–341. Springer-Verlag, 2007.

[11] Robin Burke. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4):331–370, 2002.

[12] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. Knowledge-Based Systems, 46:109 – 132, 2013.

[13] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. Machine Learning, 29:131–163, 1997.

[14] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification. Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2001.

[15] C. Bishop. Pattern Recognition and Machine Learning. Springer Verlag, 2006.

[16] K. Shyong, Tony Lam, Dan Frankowski, and John Riedl. Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. ETRICS 2006, LNCS 3995:14–29, 2006.

[17] Vijay S. Iyengar and Tong Zhang. Empirical study of recommender systems using linear classifiers. In David Wai-Lok Cheung, Graham J. Williams, and Qing Li, editors, PAKDD, volume 2035 of Lecture Notes in Computer Science, pages 16–27. Springer, 2001.

[18] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 22(1):5–53, 2004.

[19] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. 1998.

[20] Zhi-Dan Zhao and Ming sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. International Workshop on Knowledge Discovery and Data Mining, 0:478–481, 2010.

[21] G.D. Linden, B. Smith, and J. York. Amazon.com recommendations. item-to-item collaborative filtering. IEEE Internet Computing, 7(1):76–80, 2003.

[22] X. Su and T.M. Khoshgoftaar. A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009:4, 2009.

[23] SongJie Gong, HongWu Ye, and HengSong Tan. Combining memory-based and model-based collaborative filtering in recommender system. In PACCS, pages 690–693. IEEE Computer Society, 2009.

[24] J. Ben Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Vol. 4321, pages 291–324. Springer-Verlag, 2006.

[25] Robin Burke. Hybrid Web Recommender Systems, pages 377–408. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[26] David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In Proceedings of the 18th international conference on World wide web, WWW '09, pages 111–120, New York, NY, USA, 2009. ACM.

[27] G. Begelman, P. Keller, and F. Smadja. Automated Tag Clustering: Improving search and exploration in the tag space. WWW2006, May, pages 22–26, 2006.

[28] Andreas Hotho, Robert Jhke, Christoph Schmitz, and Gerd Stumme. Folkrank: A ranking algorithm for folksonomies. In Proc. FGIR 2006, page to appear, 2006.

[29] Kaipeng Liu, Binxing Fang, and Weizhe Zhang. Speak the same language with your friends: augmenting tag recommenders with social relations. In Mark H. Chignell and Elaine G. Toms, editors, HT, pages 45–50. ACM, 2010.

[30] R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In Knowledge Discovery in Databases: PKDD 2007, Lecture Notes in Computer Science. Springer, 2007.

[31] Meiqun Hu, Ee-Peng Lim, and Jing Jiang. A probabilistic approach to personalized tag recommendation. In Ahmed K. Elmagarmid and Divyakant Agrawal, editors, SocialCom/PASSAT, pages 33–40. IEEE Computer Society, 2010.

[32] Wei Feng and Jianyong Wang. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '12, pages 1276–1284, New York, NY, USA, 2012. ACM.

[33] Pasquale Lops, Marco de Gemmis, Giovanni Semeraro, Cataldo Musto, and Fedelucio Narducci. Content-based and collaborative techniques for tag recommendation: an empirical evaluation. J. Intell. Inf. Syst., 40(1):41–61, 2013.

[34] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, pages 540–547. ACM, 2009.

[35] Dawei Yin, Shengbo Guo, Boris Chidlovskii, Brian D. Davison, Cédric Archambeau, and Guillaume Bouchard. Connecting comments and tags: improved modeling of social tagging systems. In Stefano Leonardi, Alessandro Panconesi, Paolo Ferragina, and Aristides Gionis, editors, WSDM, pages 547–556. ACM, 2013.

[36] Manel Hmimida and Rushed Kanawati. A graph-based meta-approach for tag recommendation. In Hocine Cherifi, Sabrina Gaito, Walter Quattrociocchi, and Alessandra Sala, editors, COMPLEX NETWORKS, volume 693 of Studies in Computational Intelligence, pages 309–320. Springer, 2016.

[37] Walter Carrer-Neto, Maria Luisa Hernández-Alcaraz, Rafael Valencia-García, and Francisco García Sánchez. Social knowledge-based recommender system. application to the movies domain. Expert Syst. Appl., 39(12):10990–11000, 2012.

[38] Seok Kee Lee, Yoon Ho Cho, and Soung Hie Kim. Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. Inf. Sci., 180:2142–2155, June 2010.

[39] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. Tv program recommendation for multiple viewers based on user profile merging. User Model. User-Adapt. Interact., 16(1):63–82, 2006.

[40] Jesús Serrano-Guerrero, Enrique Herrera-Viedma, José Angel Olivas, Andres Cerezo, and Francisco P. Romero. A google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0. Inf. Sci., 181(9):1503–1516, 2011.

[41] Carlos Porcel, Álvaro Tejeda-Lorente, M. A. Martínez, and Enrique Herrera-Viedma. A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. Inf. Sci., 184(1):1–19, 2012.

[42] Edward Rolando Núṇẽz Valdéz, Juan Manuel Cueva Lovelle, Oscar Sanjuán Martínez, Vicente García-Díaz, Patricia Ordóṇẽz de Pablos, and Carlos Enrique Montenegro Marín. Implicit feedback techniques on recommender systems applied to electronic books. Computers in Human Behavior, 28(4):1186–1193, 2012.

[43] Osmar R. Zaïane. Building a recommender agent for e-learning systems. In ICCE, pages 55–59, 2002.

[44] Jose Jesus Castro-Schez, Raúl Miguel, David Vallejo, and Lorenzo Manuel López-López. A highly adaptive recommender system based on fuzzy logic for b2c e-commerce portals. Expert Syst. Appl., 38(3):2441–2454, 2011.

[45] Kevin McNally, Michael P. O'Mahony, Maurice Coyle, Peter Briggs, and Barry Smyth. A case study of collaboration and reputation in social web search. ACM TIST, 3(1):4, 2011.

[46] Massimiliano Zanin, Pedro Cano, Javier M. Buldú, and Oscar Celma. Complex networks in recommendation systems. In CEA'08: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications, pages 120–124, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[47] Anísio Lacerda, Adriano Veloso, and Nivio Ziviani. Exploratory and interactive daily deals recommendation. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, RecSys, pages 439–442. ACM, 2013.

[48] Adriano C. M. Pereira, Arlei Silva, Wagner Meira Jr., and Virgílio A. F. Almeida. Seller's credibility in electronic markets: a complex network based approach. In Katsumi Tanaka, Xiaofang Zhou, and Adam Jatowt, editors, WICOW, pages 59–66. ACM, 2009.

[49] C.D. Manning and H. Schütze. Foundations of statistical natural language processing. MIT Press, 1999.

[50] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[51] Albert-Laszlo Barabasi. Linked: How Everything is Connected to Everything Else and What it Means for Business, Science and Everyday Life. 2003.

[52] Aurlien Gron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc., 1st edition, 2017.

[53] ABM Nasiruzzaman and Hemanshu Pota. Critical node identification of smart power system using complex network framework based centrality approach. pages 1–6, 08 2011.

[54] M. Newman. The structure and function of complex networks. 2003.

[55] M.E.J. Newman. Clustering and preferential attachment in growing networks. Physical Review E, 64(2):025102, 2001.

[56] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations, 2002.

[57] J. Saramäki, M. Kivelä, J.P. Onnela, K. Kaski, and J. Kertesz. Generalizations of the clustering coefficient to weighted complex networks. Physical Review E, 75(2):027105, 2007.

[58] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. Social Networks, 30(2):136 – 145, 2008.

[59] Linton C. Freeman. Centrality in social networks: Conceptual clarification. Social Networks, 1(3):215–239, 1979.

[60] Phillip Bonacich. Power and centrality: A family of measures. American Journal of Sociology, 92(5):1170–1182, 1987.

[61] Mark Newman. Networks: An Introduction. Oxford University Press, Inc., New York, NY, USA, 2010.

[62] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[63] G. Salton and M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA, 1986.

[64] Gareth James. An introduction to statistical learning: with applications in R. Springer, New York, NY, 2013.

[65] Bogumil Kaminski, Michal Jakubczyk, and Przemyslaw Szufel. A framework for sensitivity analysis of decision trees. CEJOR, 26(1):135–159, 2018.

[66] Thomas G. Dietterich. Ensemble methods in machine learning. Lecture Notes in Computer Science, 1857:1–??, 2000. It is a good classic article reviewing ensemble methods. He shows intutively why ensembles is a good idea: Statistical, computational, representational.

[67] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In International Conference on Machine Learning, pages 148–156, 1996.

[68] L. Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.

[69] Leo Breiman. Random forests. Machine Learning, 45:5–32, 2001.

[70] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. Journal of the ACM, 41(1):67–95, 1994.

[71] Gang Wang, Jinxing Hao, Jian Ma, and Hongbing Jiang. A comparative assessment of ensemble learning for credit scoring. Expert Syst. Appl., 38(1):223–230, 2011.

[72] B. Efron. Bootstrap methods: another look at the jackknife. Ann. Statistics, 7:1–26, 1979.

[73] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. Journal of Machine Learning Research, 3:2003, 2003.

[74] T. Minka and J. Lafferty. Expectation-Propagation for the generative aspect model. In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI), San Francisco, CA, 2002. Morgan Kaufmann.

[75] T. L. Griffiths and M. Steyvers. Finding scientific topics. Proceedings of the National Academy of Science, 101:5228–5235, 2004.

[76] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In Advances in Neural Information Processing Systems, volume 20, 2008.

[77] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. Machine Learning, 102(3):349–391, Mar 2016.

[78] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, 2009.

[79] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In Proceedings of the fifth ACM conference on Recommender systems, pages 305–308. ACM, 2011.