Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Flecha: Um sistema de recomendação de questões de concurso público

Renato Rangel Costa Cruz Bomfim

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Ladeira

Brasília
2017

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Flecha: Um sistema de recomendação de questões de concurso público

Renato Rangel Costa Cruz Bomfim

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof. Dr. Leticia Lopes Leite
CIC/UnB

Prof. Dr. Edison Ishikawa
CIC/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
CIC/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 4 de outubro de 2017

Dedicatória

Dedico este trabalho de graduação a minha Mãe pelos 28 anos de incansável dedicação à minha formação.

Agradecimentos

Gostaria de agradecer ao professor Marcelo Ladeira pela liberdade na escolha do tema do projeto e também por todos os ensinamentos que vão muito além da esfera acadêmica.

Faço um reconhecimento a Priscila Rangel, minha irmã, pelas revisões textuais da vida.

Resumo

Esse estudo apresenta um sistema de recomendação de questões sobre Noções de Informática e Direito Administrativo que foram aplicadas em concursos públicos, com base na maior incidência dos temas nas provas. A ferramenta pode ser utilizada por candidatos para otimizar o tempo de estudo, permitindo escolher para resolver os exercícios cujos temas foram mais abordados em concursos.

O sistema criado é capaz de extrair questões oriundas de meios não estruturados como PDFs, classificar automaticamente as questões e fazer recomendações de tal forma que a quantidade recomendada é proporcional ao que foi cobrado anteriormente.

Para fazer a classificação foram induzidos classificadores com o SVM nas implementações SVC e LinearSVC e realizados experimentos com diferentes parâmetros. Também foram testados diferentes tipos de pré-processamento.

Já na recomendação foi proposto um sistema que clusteriza as questões em grupos que serão recomendados proporcionalmente ao tamanho de cada *cluster*. Foram realizados experimentos com diferentes números de *clusters*, a eficácia de uma função de decaimento e o comportamento da qualidade da recomendação quando se aumenta o número de questões recomendadas.

Na classificação, os melhores resultados obtidos foram com o LinearSVC. A recomendação obteve os melhores resultados sem a utilização do decaimento e com um número pequeno de *clusters*.

Palavras-chave: Sistema de recomendação, mineração de texto, aprendizado de máquina

Abstract

This research presents a recommendation system for Brazilian civil service exams on Information Technology and Administrative Law. Based on the higher incidence of subjects in the tests, the tool can be used by candidates to optimize the study time, allowing to choose exercises whose subjects were most approached in the last exams.

The system was created to extract questions from unstructured media such as PDFs, automatically classify questions and make recommendations such that the recommended amount is proportional to what was previously seen on past exams.

In order to do the classification, it was induced SVM classifiers with the SVC and LinearSVC implementations and experiments were performed with different parameters. Different types of preprocessing have also been tested.

In the recommendation, the exam question were clusterized into similar groups and recommended in proportion to the size of each cluster. Experiments were performed with different numbers of clusters, the effectiveness of the decay function and the behavior of the recommendation when increasing the number of recommended questions.

In the classification, the best results were obtained with LinearSVC. The recommendation obtained the best results without the use of decay function and with a small number of clusters.

Keywords: Recommender system, text mining, machine learning

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BOW *Bag-of-Words.*

CESPE Centro de Seleção e Promoção de Eventos.

CRISP-DM *Cross-Industry Standard Process for Data Mining.*

DTM *Document-Term Matrix.*

PDF *Portable Document Format.*

RBF *Radial Basis Function.*

SVM *Support Vector Machine.*

TF-IDF *Term Frequency – Inverse Document Frequency.*

WWW *World Wide Web.*

Capítulo 1

Introdução

Com a obrigatoriedade da realização de exames para acessar cargos públicos, os concursos ganharam notoriedade no Brasil. Essas provas abrangem geralmente uma extensa carga de conteúdo e exigem dos candidatos muitas horas de dedicação aos estudos.

Uma das alternativas consideradas na hora de se preparar para as provas é a estratégia de resolver exercícios cobrados em concursos anteriores. Como existe um grande número dessas questões, é importante saber quais são as mais relevantes.

Os sistemas de recomendação são importantes ferramentas que as empresas utilizam para melhorar a experiência dos usuários. Empresas como a *Netflix* utilizam sistemas de recomendação para sugerir quais filmes os usuários provavelmente vão se interessar. De acordo com [1], os sistemas de recomendação são essenciais para que seus clientes encontrem filmes de acordo com o seu perfil e obtenham uma percepção de entretenimento mais personalizada. Também em [1], é descrita a importância desse tipo de sistema que sugere 80% do que é assistido na plataforma.

Os sistemas de recomendação também são utilizados no contexto do comércio eletrônico. Por exemplo, a *Amazon* utiliza esse tipo de sistema para personalizar suas ofertas, e em [2], fala como suas recomendações são personalizadas de acordo com o interesse de cada consumidor. Por exemplo, um engenheiro de software receberia anúncios sobre programação e uma mãe de recém-nascido, sobre brinquedos.

Esse estudo apresenta uma proposta que permite, com base em mineração de textos, criar um sistema de recomendação de questões de concurso público que auxilie na preparação de candidatos a carreiras públicas. Para essa tarefa foi necessário extrair questões da banca de seleção Centro de Seleção e Promoção de Eventos (CESPE), automatizar o processo de classificação de questões e criar um sistema de recomendação. As questões foram divididas em três categorias: direito administrativo, noções de informática e geral.

Após a classificação hierárquica em tipos e subtipos, as questões são clusterizadas. Cada grupo, ou *cluster*, será ordenado de acordo com uma heurística de relevância e

recomendado dentro de uma lista de questões de acordo com a incidência dos temas em concursos anteriores. O produto final do sistema de recomendação é uma lista de questões que aborda os temas mais vistos nos concursos anteriores.

1.1 Objetivo

Objetivo geral

Essa pesquisa foca na criação de um sistema capaz de utilizar técnicas de mineração de dados para se recomendar listas de questões para estudo visando preparação para concursos públicos com base em padrões estatísticos obtidos ao analisar concursos anteriores similares.

Objetivos específicos

- elaborar um sistema de extração automático de questões de concurso público;
- criar um sistema de classificação hierárquica que permita classificar automaticamente questões de diferentes categorias;
- criar um sistema recomendação.

1.2 Estrutura do Documento

O trabalho está estruturado da seguinte forma: o Capítulo 2 descreve a literatura básica e o estado da arte sobre sistemas de recomendação e classificação automática textual; o Capítulo 3 expõe a metodologia empregada na realização do estudo; o Capítulo 4 descreve o experimento realizado; o Capítulo 5 apresenta os resultados obtidos no experimento; o Capítulo 6 relata as conclusões obtidas do trabalho e descreve temas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta alguns conceitos essenciais de classificação de texto e sistemas de recomendação.

2.1 Classificação Automática de Texto

De acordo com [3], dado um conjunto de documentos $D = \{d_1, d_2, ..., d_n\}$ e as suas categorias associadas $c(d_i) \in \{c_1, c_2, ..., c_l\}$, classificação textual é o problema de estimar qual é a categoria $c(d)$ de um novo documento d. O processo de classificação será especificado nos parágrafos seguintes.

A etapa inicial, coleta de dados, é a fase em que se buscam os dados necessários para se realizar a classificação. Para essa etapa, foi necessário a utilização de um *web crawler*. Em [4], define *crawling* como o processo de armazenar sites da *World Wide Web* (WWW) seguindo os links encontrados em cada página.

Pré-processamento é a etapa em que se preparam os dados a partir dos quais serão induzidos modelos. São descritas, a seguir, as principais técnicas realizadas no pré-processamento.

Para se transformar os dados em um formato adequado aos algoritmos de classificação, é necessário transformar o texto em um vetor em que as colunas representam a ocorrência ou não de uma determinada palavra do *corpus*. Esse tipo de técnica é conhecida como *Bag-of-Words* (BOW), por que não leva em conta a ordem ou a relação entre as palavras, apenas a ocorrência ou não delas.

O processo responsável por dividir um texto em várias palavras é o *tokenization*, em [5], o autor o define como um processo de divisão de um texto em *tokens*. Que começa com a remoção dos respectivos sinais de pontuação e substituindo tabs e outras marcações não-textuais por um espaço simples. Depois os espaços simples determinam o começo e

o fim de cada *token*. O conjunto de diferentes palavras obtido depois de juntar todos os documentos do texto é conhecido como dicionário de palavras dos documentos.

De acordo com [6], os textos têm uma grande variedade de palavras, e portanto, a BOW criada é muito variável e pode apresentar péssimo resultado na classificação. Para resolver esse problema é necessário reduzir a variabilidade deles com algumas técnicas de pré-processamento.

Uma prática muito comum na mineração de textos é a normalização. Ela consiste na padronização das palavras para se facilitar a similaridade entre si. Todas as palavras são convertidas para maiúsculo ou minúsculo, são removidos os hifens e os sinais de acentuação.

A remoção de *stopwords* consiste na retirada de palavras que não trazem valor semântico ao texto. Em [7], diz que os *stopwords* só têm finalidade sintática mas não refletem o assunto do texto.

Em [8], caracteriza o *stemming* como um procedimento computacional que reduz todas as palavras com o mesmo radical para uma forma comum, frequentemente retirando de cada palavra seus sufixos e flexões. Por exemplo: as palavras "cachorro" e "cachorrinho" seriam consideradas diferentes caso não passassem pelo *stemming*. Após o uso da ferramenta, as palavras ficam no radical "cachorr" e, portanto, iguais.

De acordo com [9], o *Term Frequency – Inverse Document Frequency* (TF-IDF) é uma métrica estatística que dá uma pontuação a cada palavra de acordo com sua importância dentro do *corpus*. Para cada documento do *corpus*, a frequência de ocorrência de cada termo é calculado. Então esse valor é multiplicado pelo logaritmo do inverso da frequência nos documentos. O resultado final é uma BOW onde as colunas contém os valores do TF-IDF para cada termo do *corpus*.

2.1.1 Classificação

SVM

Segundo [3], *Support Vector Machine* (SVM) é uma técnica de aprendizado supervisionado que, dado um conjunto de treinamento de dados categorizados em uma das duas possíveis classes, cria um modelo capaz de relacionar um novo exemplo a uma das categorias previamente conhecidas. O SVM é capaz de encontrar um hiperplano entre dados das classes maximizando a distância entre as margens. Na Figura 2.1 [10], a linha preta contínua representa o hiperplano que procura maximizar a distância entre as margens representadas pelas duas linhas pontilhadas na imagem.

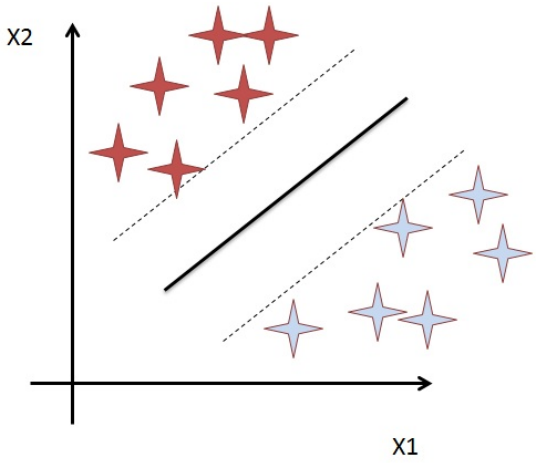


Figura 2.1: Hiperplano que maximiza as margens entre as categorias.

Tabela 2.1: Matriz de confusão.

	Especialista diz sim	**Especialista diz não**
Sistema diz sim	Verdadeiro Positivo	Falso Positivo
Sistema diz não	Falso Negativo	Verdadeiro Negativo

2.1.2 Avaliação

De acordo com [11], as medidas que avaliam eficácia mais utilizadas em recuperação de informação são definidos em termos da matriz de confusão, Tabela 2.1:

$$\text{Precisão} = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Positivo}$$

$$\text{Revocação} = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Negativo}$$

$$\textit{F-measure} = (1 + \beta^2) \times \frac{precisão \times revocação}{(\beta^2 \times precisão) + revocação}$$

A revocação é a proporção dos membros de uma classe que o sistema conseguiu classificar corretamente. A precisão é a proporção dos documentos atribuídos a uma classe pelo sistema que são realmente membros dessa classe. *F-measure* é a média harmônica ponderada entre a revocação e a precisão. Neste trabalho foi escolhido o $\beta = 1$ para que a revocação e a precisão tem igual importância dentro do *F-measure*. Um sistema ideal tem revocação, precisão e *F-measure* igual a 1.

O *cross-validation*, segundo [12], é a partição aleatória de um conjunto de dados D em k subconjuntos totalmente disjuntos $D_1, D_2, ..., D_k$ com aproximadamente igual tamanho. Então o classificador é treinado e testado k vezes. Cada vez $t \in \{1, 2, 3, ..., k\}$ é treinado

em $D \backslash D_t$ e testado em D_t. A estimativa obtida pelo *cross-validation* é a soma dos valores obtidas em cada teste dividido pelo total de testes. A vantagem desse método é que toda a base de dados é testado pelo menos uma vez e treinado $k-1$ vezes, portanto diminuindo a variância do resultados encontrados.

2.2 Recomendação

Basicamente, segundo [13], os sistemas de recomendação podem ser divididos em dois grupos. Os com filtragem colaborativa e os baseados em conteúdo. Neste trabalho abordaremos os sistemas de recomendação baseados em conteúdo.

2.2.1 Recomendação baseada em conteúdo

O sistema de recomendação baseado em conteúdo é definido por [14] como um tipo de sistema de recomendação que analisa as descrições de cada item para achar os itens que são de interesse para um determinado usuário. Esse tipo de técnica se diferencia das técnicas de filtragem colaborativa por que estes não utilizam informações do item a ser recomendado, apenas a similaridade entre os usuários que gostam de determinado item. Enquanto aquele usa as informações intrínsecas do item para criar a sua recomendação.

2.2.2 Clusterização hierárquica

De acordo com [15], métodos de clusterização dividem um conjunto de objetos em subconjuntos, conhecidos como *clusters* de tal forma que objetos dentro de qualquer *clusters* sejam similares entre si e diferentes dos demais *clusters*. Alguns métodos são hierárquicos e constroem *clusters* de *clusters*.

Já [16], fala que a clusterização hierárquica utiliza a similaridade entre itens para propor uma hierarquia entre as categorias dos itens. A similaridade é calculada a partir das propriedades de cada item. Quando se extrai uma hierarquia de um texto em linguagem natural, a adjacência entre os termos, e também, as relações sintáticas entre os termos são capazes de exprimir um poder descritivo para se induzir a hierarquia semântica de conceitos relacionados com estes termos.

2.2.3 Similaridade entre texto

Segundo [17], textos podem ser similares nas formas léxica ou semântica. As palavras são similares lexicalmente se elas tem uma sequência de caracteres similar. Palavras são similares semanticamente se elas tem o mesmo significado ou são usadas da mesma

forma. Nesse trabalho foram utilizadas apenas técnicas de similaridade léxica. Para calcular a similaridade são utilizados algoritmos baseados nas medidas de similaridade ou dissimilaridade entre dois textos.

A similaridade do cosseno é uma medida de similaridade que calcula o cosseno do ângulo formado entre duas linhas de um *Document-Term Matrix* (DTM). Essa medida varia entre 0 a 1, ou seja, quanto maior a similaridade entre dois textos, mais perto de 1 é a pontuação atribuída.

De acordo com [18], a similaridade Jaccard é a proporção de termos compartilhados em dois textos em relação ao total de termos únicos nos textos.

Outro importante conceito utilizado no trabalho é o de medoide. Em [19], o define como o elemento mais representativo de um conjunto de dados em que a dissimilaridade é mínima em relação aos demais elementos do conjunto.

2.3 Trabalhos Correlatos

O uso de técnicas de recomendação para ajudar professores a procurar e selecionar questões de uma base de dados é proposto em [20]. A ferramenta usa uma abordagem híbrida, de *feature-augmentation* e recomendação. O sistema de recomendação usa técnicas de recomendação baseadas no conteúdo e de conhecimento recorrendo ao uso de uma nova função heurística. O sistema coleta *feedbacks* tanto explicitamente como implicitamente para melhorar recomendações futuras.

Já em [21], se propõe um sistema de recomendação baseado em conteúdo que utiliza aspectos semânticos para representar assuntos em materiais educacionais. O sistema utiliza uma abordagem híbrida incorporando tanto aspectos léxicos como semânticos para a recomendação. A avaliação do sistema foi em um ambiente controlado após a sua prototipação.

Em [13], foi proposto um sistema de recomendação baseado em conteúdo que classifica documentos textuais escritos em croata. O sistema sugere materiais de ensino para estudantes de pós-graduação com base nos seus interesses e o que já foi estudado por cada estudante. Neste trabalho foram descritas técnicas de redução da dimensionalidade, remoção de *stopwords* e criação de uma DTM.

Capítulo 3

Metodologia

A metodologia de mineração de dados utilizada neste trabalho é o *Cross-Industry Standard Process for Data Mining* (CRISP-DM), proposto em [22]. Trata-se de um ciclo dividido em seis fases: entendimento do negócio, entendimento dos dados, preparação dos dados, modelagem, validação e entrega. A Figura 3.1 [22] mostra visualmente o processo.

3.1 CRISP-DM

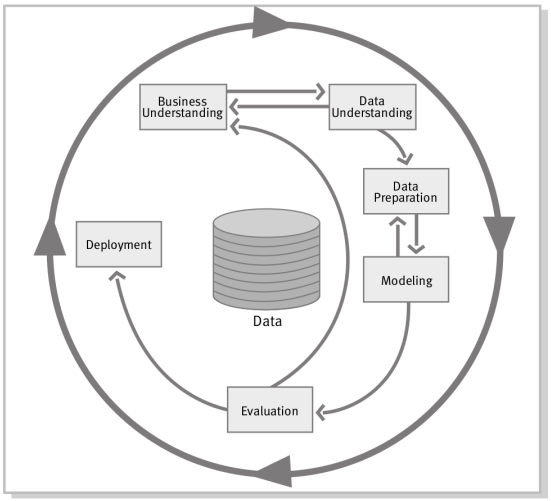


Figura 3.1: Fases do CRISP-DM.

O entendimento do negócio é a fase inicial do projeto, em que se procura entender os objetivos e demandas em relação às necessidades do cliente. Depois, o aprendizado adquirido será convertido em uma definição de problema de mineração de dados e, também, num plano preliminar desenvolvido para atingir as metas.

A fase de entendimento dos dados se inicia com a coleta de informações e prossegue com atividades que permitem a familiarização com os dados, tais como fazer análises descritivas, identificar problemas de consistência e detectar subconjuntos de informações para gerar hipóteses.

A fase da preparação engloba as atividades necessárias para construir a base de dados que vai alimentar as ferramentas de modelagem. As tarefas incluem integração de dados de diferentes fontes, seleção de atributos, transformação e limpeza de dados para que possam ser utilizadas pelas ferramentas de modelagem.

A modelagem é a fase do processo de mineração em que vários modelos são induzidos a partir dos dados já preparados. É nesta fase também que os parâmetros são calibrados para otimizar a performance do modelo. Há diversas técnicas para o mesmo tipo de problema de mineração, algumas com padrões específicos para o formato dos dados. Portanto, muitas vezes se faz necessário voltar à fase de preparação.

O objetivo da fase de validação é fazer uma cuidadosa avaliação do modelo já criado e revisar os passos utilizados no desenvolvimento para chegar aos objetivos do negócio.

A fase final, a entrega, é o momento em que se colocam os modelos criados no ambiente de produção.

3.2 Visão geral

O sistema proposto neste trabalho, conforme mostrado na Figura 3.2, é dividido em duas grandes etapas: a mineração de dados e a recomendação.

A mineração de dados engloba cinco tarefas: coleta de dados, pré-processamento, seleção de atributos, indução de classificadores hierárquicos e a avaliação dos modelos induzidos. A coleta é responsável por extrair do formato original, todas as informações relacionadas ao problema, e reuni-las em um banco de dados. O pré-processamento é a tarefa onde se transforma a base de dados do formato entregue pela coleta em um formato adequado para os algoritmos de classificação. A seleção de atributos identifica os itens que melhor representam a base de dados. A indução de classificadores é responsável pela criação dos modelos de classificação. Avaliação dos modelos é o momento em que se mede a qualidade dos classificadores criados na etapa anterior.

Já a recomendação envolve as seis seguintes tarefas: pré-processamento, seleção de atributos, clusterização hierárquica, heurística dos medoides, função de decaimento e avaliação da recomendação. O pré-processamento e a seleção de atributos são equivalentes aos descritos na etapa anterior. A clusterização hierárquica é responsável por dividir os subtipos das questões de concurso em *clusters*. A heurística dos medoides dá uma pontuação para cada questão de acordo com a similaridade em relação às demais. A função

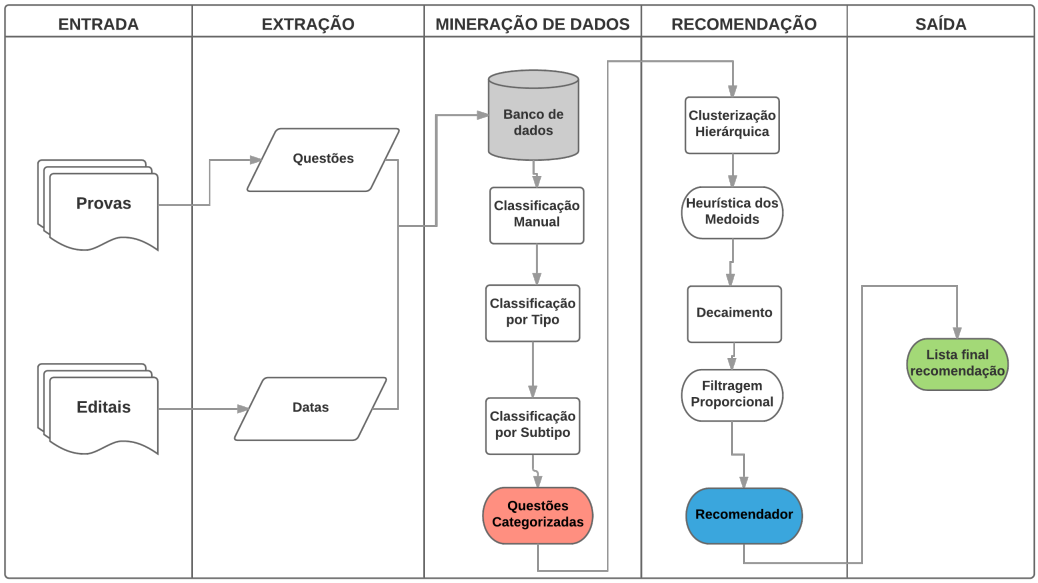


Figura 3.2: Fluxograma do sistema de recomendação Flecha.

de decaimento é responsável por diminuir a pontuação recebida da heurística dos medoides de tal maneira que uma questão mais antiga tenha menos pontos. A avaliação da recomendação é responsável por medir a efetividade do recomendador.

3.3 Entendimento do negócio

A preparação para a carreira pública requer que os candidatos estudem diversos conteúdos, relacionados a diversas áreas do conhecimento, em um curto intervalo de tempo. Tudo isso faz com que os concursandos necessitem de ferramentas que o auxiliem e agilizem a preparação para o exame.

Nos editais estão descritos os tópicos que serão cobrados no concurso. A partir da inspeção desses documentos foi possível identificar 11 sub-tópicos para "Noções de Informática" e 12 sub-tópicos para "Direito Administrativo" utilizados na classificação manual.

Para cumprir os objetivos desse projeto foi proposto um método para se avaliar a qualidade das recomendações. Esta avaliação é a similaridade entre as recomendações propostas com o que realmente foi cobrado em concurso.

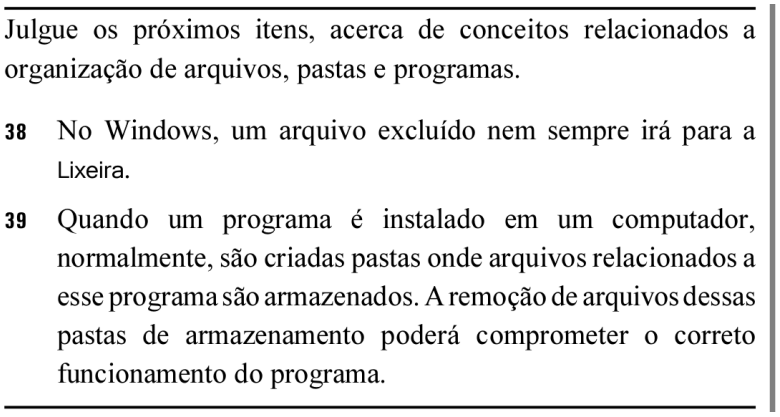
3.3.1 Flecha

Mesmo com a base de dados dividida em tipos e subtipos, o número de questões é ainda muito grande para que uma pessoa possa se preparar de maneira eficiente. Para resolver

esse problema, este estudo propõe o Flecha, um sistema de recomendação que indica quais são as questões mais relevantes para que o candidato estude inicialmente, com base na ocorrência do assunto delas em concursos anteriores.

3.4 Entendimento dos dados

Os dados são compostos de 19324 PDFs de concursos entre março de 2003 e fevereiro de 2014. Foram obtidas 266779 questões em 3554 provas de 542 concursos.



Julgue os próximos itens, acerca de conceitos relacionados a organização de arquivos, pastas e programas.

38 No Windows, um arquivo excluído nem sempre irá para a Lixeira.

39 Quando um programa é instalado em um computador, normalmente, são criadas pastas onde arquivos relacionados a esse programa são armazenados. A remoção de arquivos dessas pastas de armazenamento poderá comprometer o correto funcionamento do programa.

Figura 3.3: Amostra de uma questão no seu formato original.

As questões em seu formato original, o PDF, são compostas por estruturas que compreendem o comando e o corpo. Um comando de questão pode ter vários corpos. Cada junção do comando e do corpo representa uma questão. Na Figura 3.3 [23] por exemplo, a frase "Julgue os próximos itens, acerca de conceitos relacionados a organização de arquivos, pastas e programas." é o comando da questão. Já as frases "38 No Windows, um arquivo excluído nem sempre irá para a Lixeira." e "39 Quando um programa é instalado em um computador, normalmente, são criadas pastas onde arquivos relacionados a esse programa são armazenados. A remoção de arquivos dessas pastas de armazenamento poderá comprometer o correto funcionamento do programa." são dois corpos.

Capítulo 4

Solução Proposta: Flecha

Este capítulo trata das tarefas empregadas na confecção do sistema de recomendação.

4.1 Etapa de mineração de dados

Para a realização dessa pesquisa foram seguidos os passos descritos a seguir.

O trabalho teve início com realização de download de material relativo a concursos do site do CESPE[1] utilizando a ferramenta HTTRACK[2]. Em seguida, foi realizada a seleção e a extração dos PDFs que continham questões, sendo necessário criar um algoritmo para fazer a extração automática do conteúdo e o posterior armazenamento em um banco de dados.

A etapa de pré-processamento consistiu na preparação da base de dados para que as informações estivessem em um formato adequado aos algoritmos de classificação. Nessa etapa, foram removidos todos os sinais de acentuação e pontuação e foram removidas também as *stopwords*, palavras que não trazem nenhum valor informativo para o texto. Ainda nessa fase, as palavras foram colocadas em minúsculo. Foi utilizado o conjunto de *stopwords* em português da biblioteca NLTK [24].

Foi feito também, o *stemming* das palavras que consiste na remoção dos sufixos e inflexões com o intuito de diminuir a quantidade de texto na base de dados. Na última etapa do pré-processamento, os dados foram representados em uma tabela de atributo-valor, em que as linhas representam os documentos e as colunas, a ocorrência ou não do respectivo termo. A implementação de *stemming* utilizada neste trabalho foi o *Snowball* da biblioteca NLTK [24].

O método utilizado neste estudo para calcular o peso do respectivo atributo foi o TF-IDF.

[1] `http://www.cespe.unb.br/concursos/`, acesso em 01/02/2014
[2] `http://www.httrack.com/`, acesso em 01/02/2014

Tabela 4.1: Classificação manual por subtipo de noções de informática.

Subtipo	**Quantidade**
Banco de dados	50
Sistema operacional Windows	50
Algoritmo e estrutura de dados	59
Hardware e periféricos	60
Sistema operacional Unix	70
Segurança da informação	75
Correio eletrônico	83
Governança de TI	86
Protocolos de comunicação	100
Internet e intranet	100
Editor eletrônico	392

Tabela 4.2: Classificação manual por subtipo de direito administrativo.

Subtipo	**Quantidade**
Processo administrativo	50
Bens públicos	56
Controle da administração	66
Regime jurídico	70
Contratos administrativos	73
Serviços públicos	74
Organização da administração pública	74
Agentes públicos	74
Improbidade administrativa	84
Atos administrativos	91
Licitação	99
Poderes da administração	133

A principal seleção de atributos realizada no trabalho foi a remoção das palavras com baixa ocorrência, e também, das que ocorrem em mais de 80% da base de dados.

Para a indução dos classificadores hierárquicos, foi realizada a classificação manual de 25520 das questões por tipo, em que 7276 são de noções de informática, 5421 são de direito administrativo e o restante é composto por questões de diversas áreas do conhecimento aos quais foram atribuídas a classe "geral". Foram escolhidas essas duas classes por que são bem diferentes entre si e existe uma grande quantidade de concursos abordando esses temas. Já o total de questões por subtipo classificadas manualmente está apresentado na Tabela 4.1 e Tabela 4.2.

Foram criados classificadores hierárquicos com diferentes implementações do SVM no *scikit-learn* [25]. O SVC [26] e LinearSVC [27] foram os algoritmos utilizados na confecção dos modelos preditivos.

A mineração de dados foi avaliada em relação ao *F-measure* utilizando a técnica de *cross-validation* com *k-fold*. Por terem mais unidades classificadas manualmente, as classes por tipo tiveram testes para $k = 10$. Os experimentos por subtipo utilizaram $k = 3$.

Os resultados obtidos foram descritos no Capítulo 5 e analisados no Capítulo 6.

4.2 Etapa de recomendação

O pré-processamento e a seleção de atributos da clusterização foram realizados da mesma forma que na Seção 4.1.

Para a realização da clusterização hierárquica foi utilizada a ferramenta *Scipy*, [28] com os algoritmos *linkage* e *fcluster*. O primeiro gera uma clusterização hierárquica. O segundo forma um conjunto de *clusters* não-hierárquicos de acordo com o número de *clusters* desejados a partir da estrutura retornada do *linkage*.

Na etapa da heurística dos medoides, em cada *cluster* gerado anteriormente é atribuído uma pontuação para as questões de acordo com a distância relativa às demais no *cluster*. Para que não sejam apresentados valores muito diferentes em cada utilização do sistema é realizada a normalização dos valores encontrados da seguinte forma: a questão com menor distância entre as demais recebe uma pontuação de 100 e a com maior recebe o valor de 0. As outras questões recebem um valor proporcional à sua distância.

Após esta etapa, a função de decaimento diminui a pontuação proporcionalmente a idade de cada questão de acordo com a seguinte equação:

$$D(s, i) = s - i * t$$

em que $D(s, i)$ é o valor final de pontuação após o decaimento, s é a pontuação normalizada da distância relativa às demais questões, i é a idade da questão em semanas e t é a taxa de decaimento. A taxa de decaimento é a quantidade de pontos que será decaído em função do tempo (em semanas).

A lista final de recomendações é composta por questões representantes de cada *cluster* proporcionalmente ao tamanho do *cluster* em relação ao total. Elas são ordenadas de forma decrescente em relação as pontuações obtidas.

Por último, foi realizada a avaliação da recomendação da seguinte forma: para cada concurso realizado foi criada uma recomendação com todas as questões anteriores e atribuído um valor de similaridade entre as recomendações e as questões do concurso. Então a média das similaridades entre as questões do concurso e a recomendação e o valor máximo de similaridade são armazenados e analisados no Capítulo 6.

Capítulo 5

Resultados

5.1 Mineração de dados

Para a seleção dos algoritmos de classificação hierárquica utilizados no sistema de recomendação foram realizados testes com diferentes tipos de preprocessamento e implementações de SVM.

Na classificação por tipo e subtipo, foram realizados dois experimentos para definir a melhor combinação de classificadores. Para a classificação por tipo, o *cross-validation* foi realizado com 10 *k-fold* e por subtipo foi com 3 *k-fold* e o resultado apresentado nos gráficos é a média dos valores de *F-measure* obtidos em cada *fold* pelo total de atributos utilizados na indução dos classificadores.

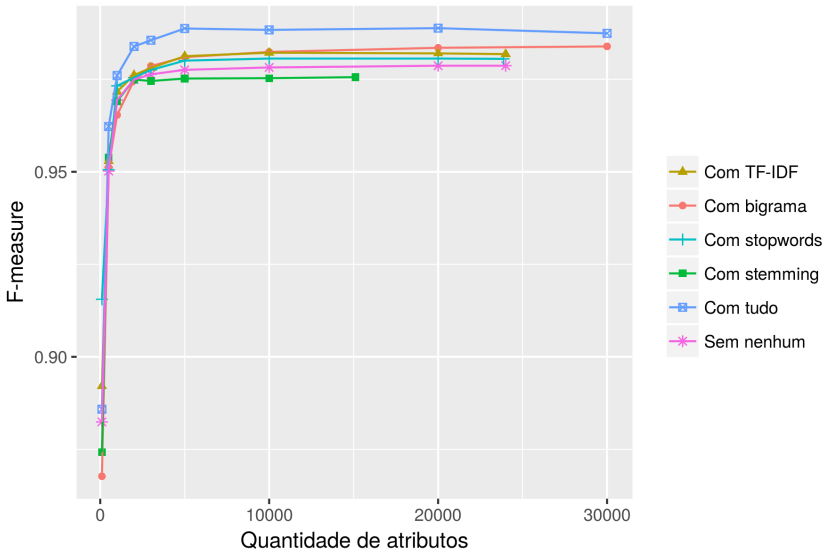


Figura 5.1: Resultado da classificação por tipo.

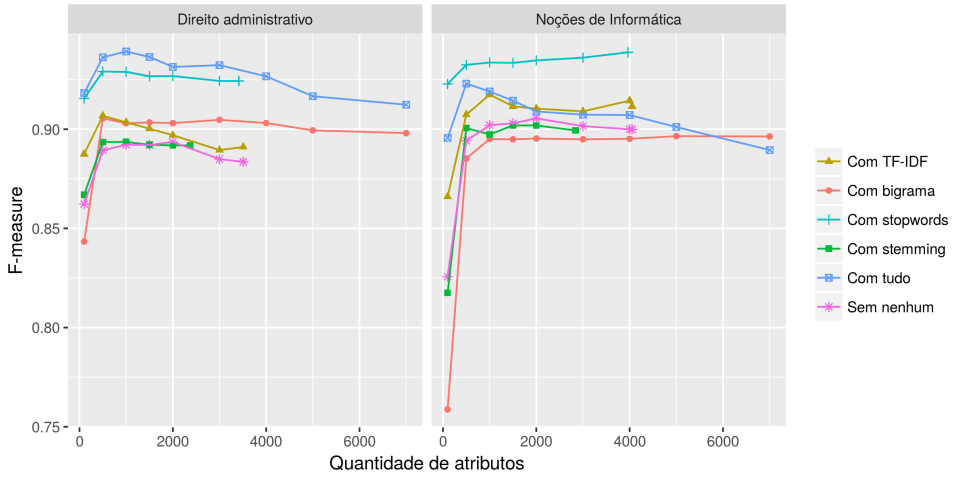


Figura 5.2: Resultado da classificação por subtipo.

O primeiro experimento foi realizado com as seguintes combinações de preprocessamento:

- apenas com *stopwords*
- apenas com *stemmer*
- apenas com TF-IDF
- apenas com bigrama
- sem nenhum pré-processamento
- com tudo

O resultado da classificação por tipo se encontra na Figura 5.1 e a por subtipo na Figura 5.2. Pode-se ver que a abordagem escolhida que reune todas as técnicas obteve melhor desempenho entre a classificação por tipo e subtipo.

Já o segundo experimento foi realizado com as seguintes implementações e parâmetros do SVM:

- SVC
  - *kernel*: linear com o parâmetro C para 0.1 e 1
  - *kernel*: RBF com o parâmetro C para 0.1 e 1
- LinearSVC
  - parâmetro C para 0.1 e 1

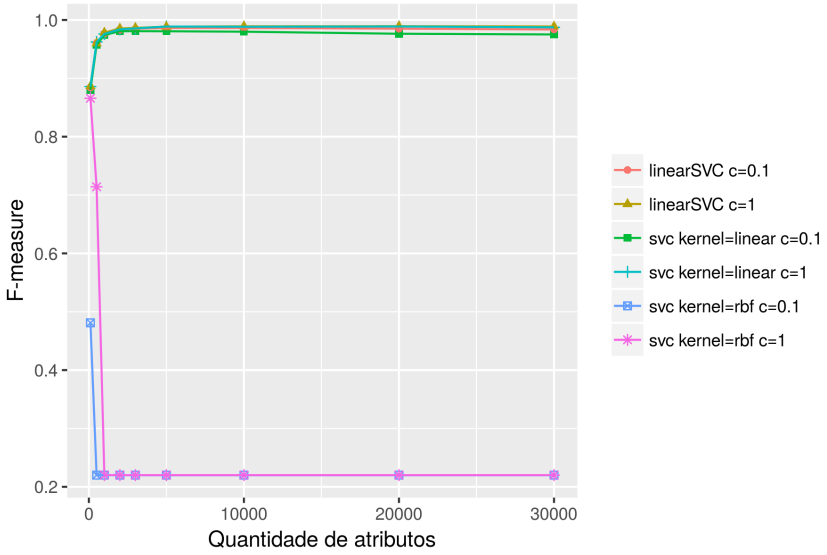


Figura 5.3: Resultado da classificação por tipo com diferentes implementações de SVM.

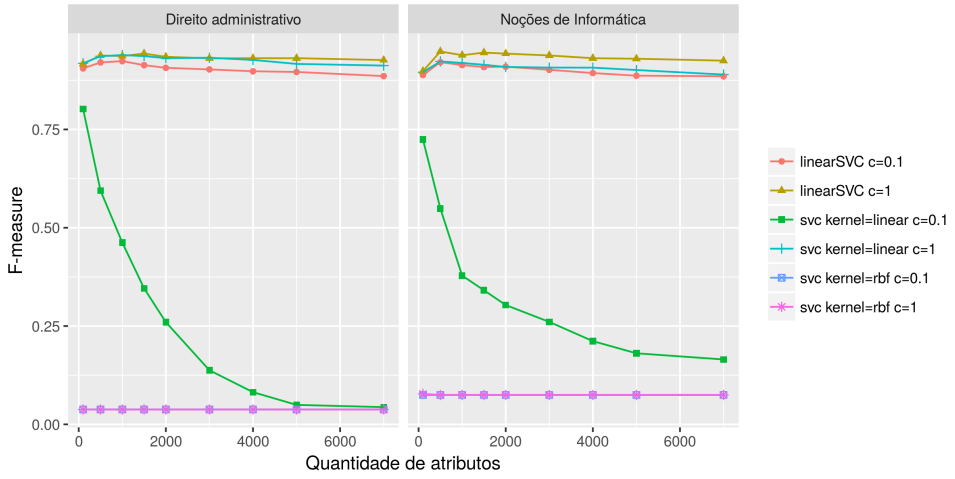


Figura 5.4: Resultado da classificação por subtipo com diferentes implementações de SVM.

O resultado da classificação por tipo se encontra na Figura 5.3 e a por subtipo na Figura 5.4. Pode-se observar que o *kernel* RBF não apresentou bons resultados na classificação textual. Tanto o SVC com o *kernel*: linear quanto o LinearSVC apresentaram bons resultados.

A partir dos resultados obtidos nesta fase foi definido o preprocessamento padrão que compõe todas as etapas (*stopwords*, *stemmer*, TF-IDF, bigrama) e o algoritmo o LinearSVC com parâmetro C de 1. Essa combinação foi utilizada para preparar os dados para a etapa de recomendação.

Tabela 5.1: Número de questões encontradas de noções de informática.

Subtipo	**Quantidade**
Algoritmo e estrutura de dados	2551
Banco de dados	1193
Correio eletrônico	539
Editor eletrônico	4666
Governança de TI	2143
Hardware e periféricos	2432
Internet e intranet	1772
Protocolos de comunicação	2586
Segurança da informação	2738
Sistema operacional Windows	941
Sistema operacional Unix	1004

Tabela 5.2: Número de questões encontradas de direito administrativo.

Subtipo	**Quantidade**
Agentes públicos	1146
Atos administrativos	1932
Bens públicos	712
Contratos administrativos	842
Controle da administração	1216
Improbidade administrativa	788
Licitação	2581
Organização da administração pública	1183
Poderes da administração	1805
Processo administrativo	592
Regime jurídico	1379
Serviços públicos	1792

5.2 Recomendação

Primeiramente foram aplicados os algoritmos de classificação por tipo em toda a base. Foi encontrado o total de 16004 questões sobre noções de informática e de 22566 para direito administrativo.

A Tabela 5.1 e a Tabela 5.2 apresenta a divisão em subtipo, respectivamente, de noções de informática e direito administrativo após a execução dos algoritmos de classificação por subtipo.

Na avaliação da recomendação foram realizados experimentos para se testar a necessidade e qualidade do decaimento, o comportamento da similaridade encontrada com diferentes números de *cluster*, os efeitos do aumento do número de questões recomendadas e, por último, a comparação do melhor modelo de recomendação encontrado com a recomendação aleatória.

A recomendação aleatória utilizada como *baseline* nesse trabalho é medida a partir da seleção aleatória de n questões de um subtipo.

5.2.1 Decaimento

Para testar a utilidade do decaimento no processo de recomendação de questões, foi comparado o desempenho das recomendações para o algoritmo com e sem decaimento. A Figura 5.5 apresenta o resultado com a média das similaridades encontradas na avaliação. A Figura 5.6 apresenta os valores máximos de similaridade encontrados.

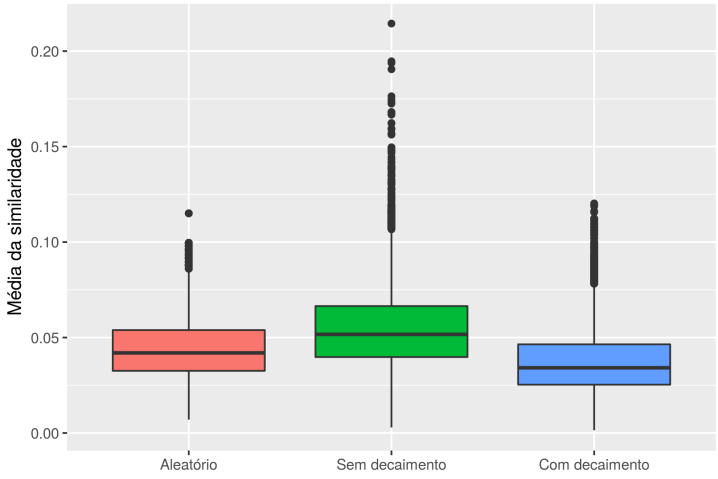


Figura 5.5: Comparação do decaimento.

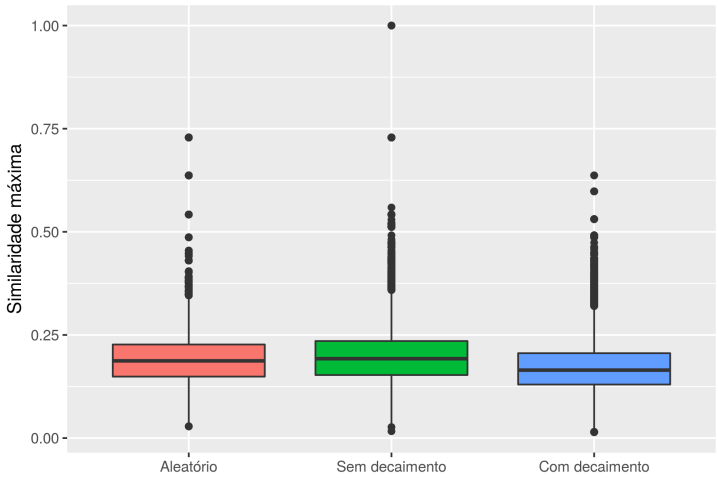


Figura 5.6: Comparação do decaimento.

É possível observar que o uso do decaimento piorou a performance da recomendação.

A média obtida com o decaimento foi de 0,037 enquanto sem o decaimento foi de 0,054. O uso do decaimento apresentou um resultado pior até do que a recomendação aleatória que obteve 0,044. Quando não se utilizou o decaimento obteve-se uma leve melhora em relação à recomendação aleatória.

O valor da máxima similaridade encontrada utilizando o decaimento teve a média de 0,173. Este valor é menor do que o obtido sem o decaimento, 0,199 e também do aleatório, 0,192.

5.2.2 Número de *clusters*

Foi testado também se o número de *clusters* influencia o desempenho do recomendador. Para realizar esse experimento foram feito testes com diferentes números de *clusters*: 4, 10, 20 e 100. As Figuras 5.7 a 5.8 apresentam os resultados.

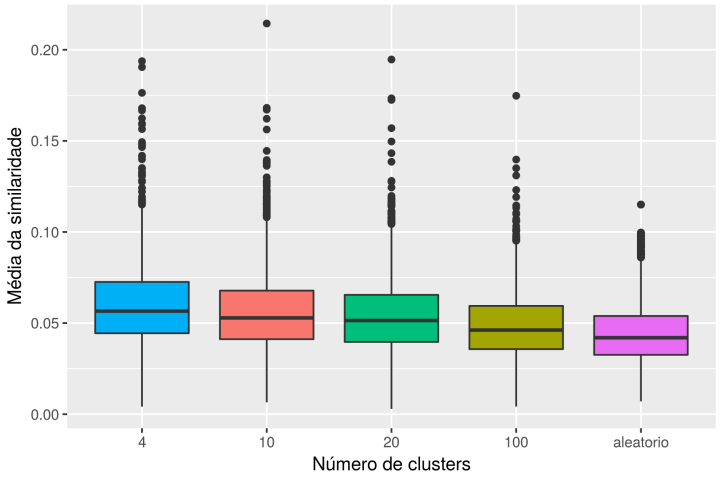


Figura 5.7: Comparação do número de *clusters*.

A partir dos resultados é possível notar que a recomendação com um número menor de *clusters* apresentou melhores resultados para a média da similaridade. O experimento com maior similaridade média foi o com 4 *clusters* que obteve 0,060. O pior resultado foi o com 100 *clusters* que obteve 0,049. Todos os experimentos obtiveram resultados melhor do que a recomendação aleatória que foi de 0,044.

Já na similaridade máxima, observa-se uma leve melhora na similaridade para um número maior de *clusters*. Os resultados variaram de 0,196 para 4 *clusters* até 0.206 para 100 *clusters*. Todos os experimentos obtiveram resultados melhores do que a recomendação aleatória.

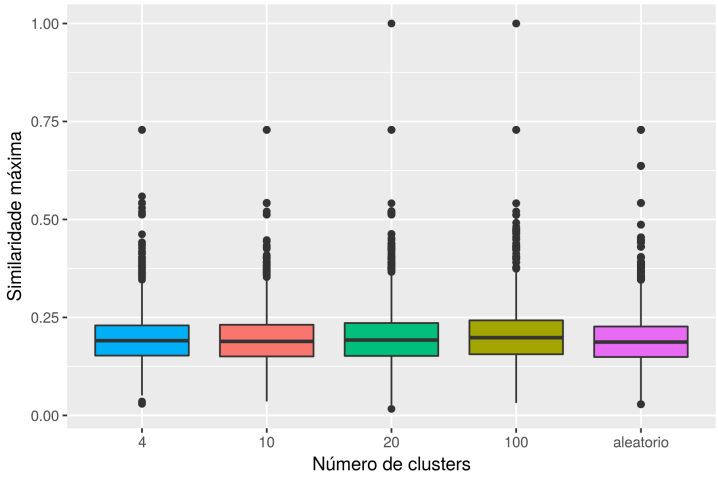


Figura 5.8: Comparação do número de *clusters*.

5.2.3 Quantidade de questões recomendadas

Foi testado também qual o comportamento do desempenho do recomendador quando se aumenta a quantidade de recomendação. Para realizar esse experimento foram realizados testes com 25, 50, 75 e 100 questões. As Figuras 5.9 a 5.10 apresentam os resultados.

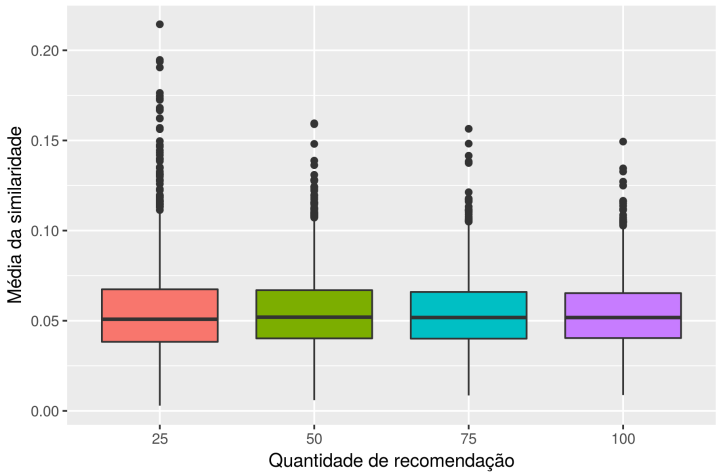


Figura 5.9: Comparação da quantidade de recomendação.

Uma maior quantidade de recomendação não apresentou uma diferença significativa na média de similaridade obtida. Para a recomendação de 25 e 50 questões obteve-se 0.055 e para 75 e 100, 0,054.

Já na similaridade máxima obteve-se uma melhora para uma quantidade maior de recomendação. O menor foi de 0.157 para 25 questões e o maior foi de 0.230 para 100 questões.

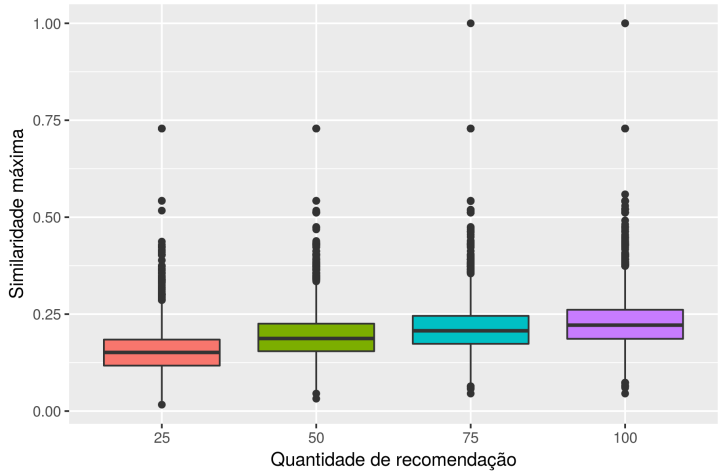


Figura 5.10: Comparação da quantidade de recomendação.

5.2.4 Melhor recomendação pelo tempo

Na análise final do trabalho, foram comparados os parâmetros que obtiveram o melhor desempenho nos experimentos com a recomendação aleatória nas mesmas características. O número de *clusters* é 4, a quantidade da recomendação é de 25 questões. As Figuras 5.11 a 5.12 apresentam os resultados.

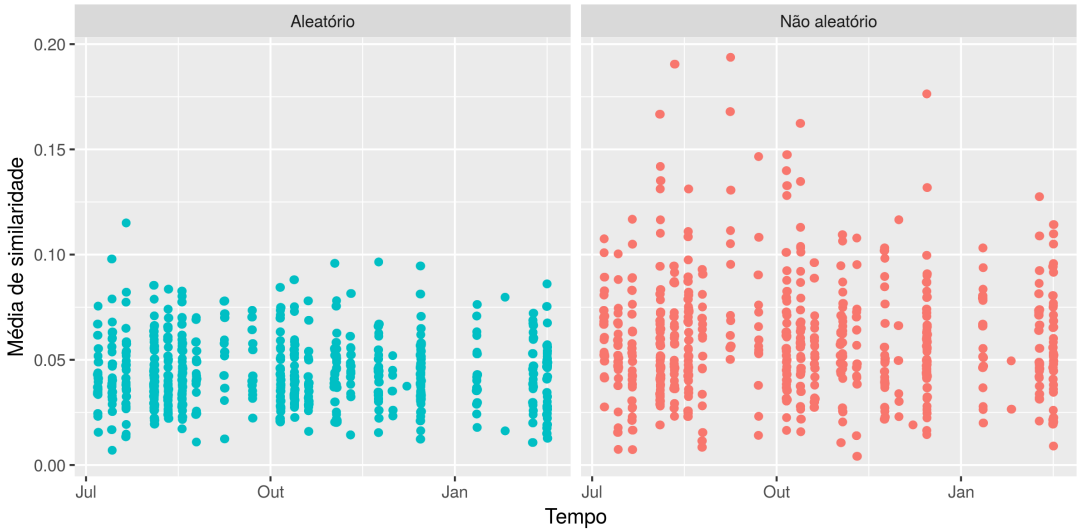


Figura 5.11: Qualidade da recomendação pelo tempo.

É possível perceber uma leve superioridade do sistema proposto com o *baseline*.

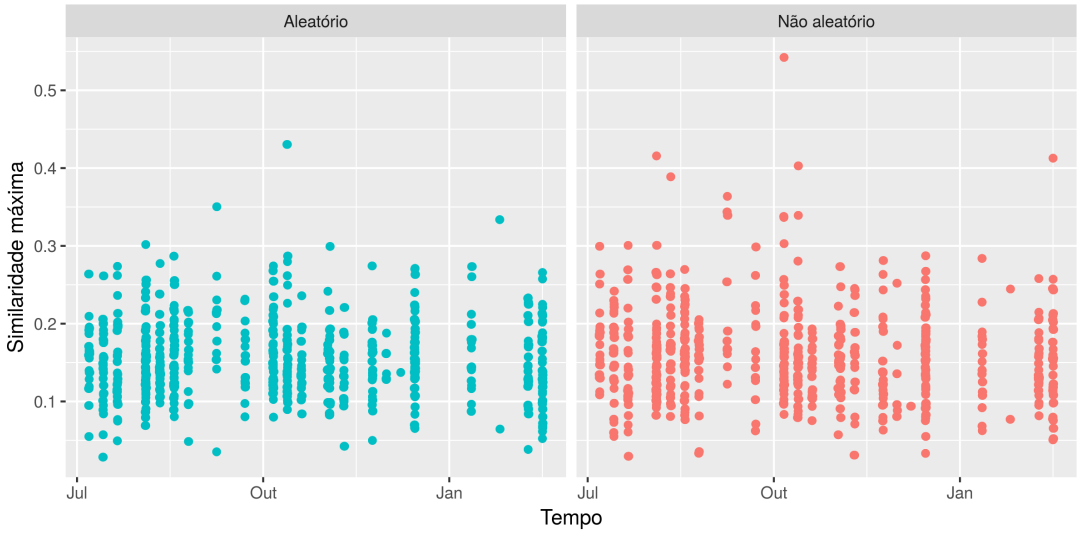


Figura 5.12: Qualidade da recomendação pelo tempo.

Capítulo 6

Conclusão

O objetivo desse trabalho foi de criar um sistema de recomendação de questões de concursos públicos baseado em questões já cobradas. Para esse objetivo foi apresentada uma metodologia de extração, classificação e recomendação de questões.

A partir dos resultados encontrados foi possível chegar a conclusão de que a etapa de mineração de dados apresentou bons resultados, em comparação ao que se acha na literatura [29], tanto na classificação por tipo como na de subtipo. Já os resultados da recomendação foram levemente superiores aos de uma recomendação aleatória.

Em geral, o pré-processamento ajudou na melhoria da performance do classificador. O uso das técnicas de remoção de *stopwords*, TF-IDF, bigrama tanto individualmente como combinados mostraram uma curva de resultados superiores a não utilização de nenhum pré-processamento. O *stemming* utilizado no trabalho não apresentou uma melhora no resultado da classificação.

O algoritmo com o melhor desempenho foi o LinearSVC para o $C = 1$. A implementação do SVM, SVC, utilizando o *kernel* RBF apresentou os piores desempenhos.

A função de decaimento proposta no trabalho apresentou uma piora na performance da recomendação quando comparada com a recomendação aleatória ou a não-utilização do decaimento.

O número de *clusters* se mostrou um parâmetro influente na média da recomendação. Entretanto, não se mostrou um fator que influencia a similaridade máxima encontrada.

O aumento da quantidade da recomendação mostrou que não é um fator que varia a média da similaridade. Essa observação se deve porque com o aumento de número de questões entram tanto questões com melhor similaridade quanto questões menos similares, portanto a média da similaridade não varia consideravelmente.

6.1 Principais contribuições

Foi desenvolvido um sistema capaz de extrair e armazenar questões de concursos vindas de diversas formatações de PDF. Esse sistema pode ser adaptado para outras tarefas que exigem a extração de textos em formatos variáveis dentro de um PDF.

Foi criada uma base de dados de concursos públicos com mais de 200.000 questões que podem ser utilizados para as mais diversas análises de textos em linguagem natural.

O sistema de recomendação criado pode ser adaptado para ajudar estudantes em diferentes contextos de estudo onde a quantidade de questões é muito grande e deseja-se aprender o conteúdo de forma uniforme dentre os diversos tópicos.

6.2 Trabalhos Futuros

Utilizar outros algoritmos que a literatura fala que têm um bom resultado em classificação textual como o *random forest*, as redes neurais e o *naive bayes*. Utilizar uma combinação de classificadores para melhorar a performance da classificação.

Na avaliação da recomendação e na heurística dos medoides ao invés de somente usar técnicas de similaridade léxica, utilizar similaridade semântica entre sentenças.

Levar em consideração o desempenho do aluno para oferecer questões compatíveis com o nível de cada aluno.

Como o sistema se propõe a fornecer uma lista de questões diversificada entre o que cai nos concursos, criar uma avaliação que meça o grau de diversidade que o recomendador está propondo.

Referências

[1] Gomez-Uribe, Carlos A e Neil Hunt: *The netflix recommender system: Algorithms, business value, and innovation.* ACM Transactions on Management Information Systems (TMIS), 6(4):13, 2016. 1

[2] Linden, Greg, Brent Smith e Jeremy York: *Amazon.com recommendations: Item-to-item collaborative filtering.* IEEE Internet computing, 7(1):76–80, 2003. 1

[3] Dhillon, Inderjit S, Subramanyam Mallela e Rahul Kumar: *A divisive information-theoretic feature clustering algorithm for text classification.* Journal of Machine Learning Research, 3(Mar):1265–1287, 2003. 3, 4

[4] Singh, Aameek, Mudhakar Srivatsa, Ling Liu e Todd Miller: *Apoidea: A decentralized peer-to-peer architecture for crawling the world wide web.* Em *Workshop on Distributed Information Retrieval*, páginas 126–142. Springer, 2003. 3

[5] Hotho, Andreas, Andreas Nürnberger e Gerhard Paaß: *A brief survey of text mining.* LDV Forum - GLDV Journal for Computational Linguistics and Language Technology, 20(1):19–62, maio 2005, ISSN 0175-1336. `http://www.kde.cs.uni-kassel.de/hotho/pub/2005/hotho05TextMining.pdf`. 3

[6] Sriram, Bharath, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu e Murat Demirbas: *Short text classification in twitter to improve information filtering.* Em *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, páginas 841–842, New York, NY, USA, 2010. ACM, ISBN 978-1-4503-0153-4. `http://doi.acm.org/10.1145/1835449.1835643`. 4

[7] El-Khair, Ibrahim Abu: *Effects of stop words elimination for arabic information retrieval: a comparative study.* arXiv, 2017. [Online; acesso 18-setembro-2017]. 4

[8] LOVINS, J. B.: *Development of stemming algorithm.* Mechanical Translation and Computation Linguistics, 11(1):23–31, 1968. `http://ci.nii.ac.jp/naid/10026812451/en/`, [Online; acesso 18-setembro-2017]. 4

[9] Blei, David M, Andrew Y Ng e Michael I Jordan: *Latent dirichlet allocation.* Journal of Machine Learning Research, 3(Jan):993–1022, 2003. 4

[10] Shouval, Roni, 2012. `https://commons.wikimedia.org/wiki/File:%D7%9E%D7%9B%D7%95%D7%A0%D7%AA_%D7%95%D7%95%D7%A7%D7%98%D7%A8%D7%99%D7%9D_%D7%AA%D7%95%D7%9E%D7%9B%D7%99%D7%9D_%D7%93%D7%95%D7%92%D7%9E%D7%90.jpg`, File: `LambdaPlaques.jpg, Online; acesso 18-setembro-2017.` 4

[11] Sebastiani, Fabrizio: *Machine learning in automated text categorization.* ACM Comput. Surv., 34(1):1-47, março 2002, ISSN 0360-0300. http://doi.acm.org/10.1145/505282.505283. 5

[12] Kohavi, Ron: *A study of cross-validation and bootstrap for accuracy estimation and model selection.* Em *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 2 de *IJCAI'95*, páginas 1137-1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc., ISBN 1-55860-363-8. http://dl.acm.org/citation.cfm?id=1643031.1643047. 5

[13] Ćavar, Ivana, Zvonko Kavran, Natalija Jolić, Neven Anđelović, Ivan Cvitić e Marko Gović: *Content-based recommender system for textual documents written in croatian.* Em *DATA ANALYTICS 2013, The Second International Conference on Data Analytics*, 2013. 6, 7

[14] Pazzani, Michael J. e Daniel Billsus: *The adaptive web.* páginas 325-341. Springer-Verlag, Berlin, Heidelberg, 2007, ISBN 978-3-540-72078-2. http://dl.acm.org/citation.cfm?id=1768197.1768209. 6

[15] Berendt, Bettina, Andreas Hotho, Dunja Mladenic, Maarten Van Someren, Myra Spiliopoulou e Gerd Stumme: *A roadmap for web mining: From web to semantic web.* Em *Web Mining: From Web to Semantic Web*, páginas 1-22. Springer, 2004. 6

[16] Maedche, Alexander e Steffen Staab: *Ontology learning for the semantic web.* IEEE Intelligent systems, 16(2):72-79, 2001. 6

[17] Gomaa, Wael H. e Aly A. Fahmy: *A survey of text similarity approaches.* International Journal of Computer Applications, 68(13):13-18, April 2013. 6

[18] Niwattanakul, Suphakit, Jatsada Singthongchai, Ekkachai Naenudorn e Supachanun Wanapu: *Using of jaccard coefficient for keywords similarity.* Em *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2013. 7

[19] Estivill-Castrol, Vladimir e Alan T. Murray: *Discovering associations in spatial data -- An efficient medoid based approach*, páginas 110-121. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, ISBN 978-3-540-69768-8. https://doi.org/10.1007/3-540-64383-4_10. 7

[20] Hage, Hicham e Esma Aïmeur: *Exam question recommender system.* Em *Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*, páginas 249-257. IOS Press, 2005. 7

[21] Góis, Marcos de Meira: *Melhorias para um sistema de recomendação baseado em conhecimento a partir da representação semântica de conteúdos*. 2015. Dissertação de Mestrado. 7

[22] Chapman, Pete, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer e Rudiger Wirth: *CRISP-DM 1.0 step-by-step data mining guide*. Relatório Técnico, The CRISP-DM consortium, 2000. http://www.crisp-dm.org/CRISPWP-0800.pdf, [Online; acesso 23-Março-2017]. 8

[23] Centro de Seleção e Promoção de Eventos (CESPE): *Caderno de questões - conhecimentos básicos para o cargo 10*, 2011. http://www.cespe.unb.br/concursos/MPE_PI2011/arquivos/MPEPI11_CB3_01.pdf, [Online; acesso 23-Março-2017]. 11

[24] Loper, Edward e Steven Bird: *Nltk: The natural language toolkit*. Em *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, páginas 63-70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. https://doi.org/10.3115/1118108.1118117. 12

[25] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12:2825-2830, 2011. 13

[26] Chang, Chih Chung e Chih Jen Lin: *LIBSVM: A library for support vector machines*. ACM Transactions on Intelligent Systems and Technology, 2:1-27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 13

[27] Fan, Rong En, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang e Chih Jen Lin: *LIBLINEAR: A library for large linear classification*. Journal of Machine Learning Research, 9:1871-1874, 2008. 13

[28] Jones, Eric, Travis Oliphant, Pearu Peterson *et al.*: *SciPy: Open source scientific tools for Python*, 2001-. http://www.scipy.org/, [Online; acesso 23-Março-2017]. 14

[29] Joachims, Thorsten: *A support vector method for multivariate performance measures*. Em *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, páginas 377-384, New York, NY, USA, 2005. ACM, ISBN 1-59593-180-5. http://doi.acm.org/10.1145/1102351.1102399. 24