Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Matemática Aplicada e Computacional

Felipe Gusmão Contratres

Emprego de análise de sentimentos em um sistema de recomendação para o caso de *cold-start*

São Paulo
Fevereiro de 2017

Emprego de análise de sentimentos em um sistema de recomendação para o caso de *cold-start*

Monografia final da disciplina
MAP2040 - Trabalho de Formatura.

Orientação: Profa. Dra. Solange N. Alves de Souza

São Paulo
Fevereiro de 2017

Agradecimentos

Acima de tudo agradeço a Deus por todas as conquitas que tem me permitido.

Agradeço imensamente à minha orientadora Profa. Solange por todo acompanhamento e dedicação a mim e a este trabalho. Sem este suporte não teria sido possível alcançar o que aqui foi alcançado.

Ao colega Alan Prando, agradeço por todas as discussões e ensinamento sobre *Big Data*, Sistemas de Recomenção e Aprendizado de Máquina durante a Iniciação Científica e durante todo o desenvolvimento deste trabalho.

Aos membros da banca examinadora, Profa. Edit e Prof. Jorge, por aceitarem o convite para participar.

Agradeço à Fundação de Apoio a Pesquisa do estado de São Paulo (Fapesp) pelo suporte dado a este trabalho através projeto #2014/04851-8.

Resumo

Sistemas de Recomendação tem o objetivo de auxiliar os usuários a encontrar informações, produtos ou serviços que sejam relevantes para as suas necessidades. É comum a utilização desta classe de sistemas em *e-commerces* com o objetivo de auxiliar os seus consumidores a encontrar os produtos que desejam. Na literatura são evidenciados problemas relacionados à escassez de dados sobre os usuários, principalmente quando trata-se de um recém cadastrado: este caso é conhecido como *cold-start*. Uma oportunidade encontrada para obter informações sobre estes usuários e possibilitar a recomendação de produtos que lhes sejam relevantes é a utilização de textos publicados por eles em suas redes sociais. Porém, um problema encontrado com a utilização destes textos é o sentimento (positivo ou negativo) contido neles, pois recomendações podem ser feitas baseadas em publicações com conteúdo negativo sobre determinado assunto e obtendo, portanto, produtos que não sejam aderentes às necessidades do consumidor. Este trabalho apresenta a proposta de um Sistema de Recomendação que utiliza as publicações feitas pelos usuários nas redes sociais Facebook e Twitter, aplicando técnicas de análise de sentimentos e aprendizado de máquina para tentar compreender suas preferências e amenizar o problema de *cold-start*. Desta forma, mesmo que não existam avaliações explícitas feitas pelos usuários - como por exemplo atribuir uma nota de 0 a 5 para um determinado produto, ou avaliações implícitas - como visualizar ou comprar determinado produto, pretende-se obter recomendações personalizadas de produtos.

Palavras-chave: sistema de recomendação; redes sociais; análise de sentimento; recomendação social; mineração de dados em redes sociais; aprendizado de máquina; recomendação baseada em conteúdo.

Abstract

Recommendation Systems is intended to assist users in finding information, products or services that are relevant to their needs. It is common to use this class of systems in e-commerces in order to help their consumers find the products they want. The literature discusses problems related to data sparsity about users, especially when it is a newly registered: this case is known as cold-start. An opportunity found to obtain information about these new users and to enable the recommendation of products that are relevant to them is the use of texts published by them in their social networks. However, a problem encountered with the use of these texts is the sentimental content (positive or negative) present in them, processing recommendations based on publications with negative content on a certain subject and obtaining, therefore, products that do not fit the needs of the consumer. This paper presents a Recommendation System proposal that uses the publications made by users on social networks Facebook and Twitter, applying techniques of sentimental analysis and machine learning to try to understand their preferences and reduce the cold-start problem. Thus, even if there are no explicit ratings made by users - such as assigning a score of 0 to 5 for a particular product, or implicit ratings - such as viewing or purchasing a particular product, is desired to obtain personalized product recommendations.

Keywords: recommender system; social network; sentimental analysis; social recommendation; social data mining; machine learning; content-based recommendation.

Sumário

Lista de Abreviaturas

SR	Sistema de Recomendação
IA	Inteligência Artificial
PLN	Processamento de Linguagem Natural
KDD	Knowledge Discovery from Data
SVM	Support Vector Machine
API	Application Programming Interface
TF-IDF	Term Frequency-Inverse Document Frequency
VSM	Vector Space Model
BOW	Bag of Words
AM	Aprendizado de Máquina
RMSE	Root Mean Square Error
MRR	Mean Reciprocal Rank

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

A quantidade e a variedade de dados gerados diariamente cresce a cada dia. Segundo um estudo do IDC (International Data Coroporation), a quantidade de dados e de informações existentes no mundo em 2009 era de aproximadamente 800 EB[1], enquanto que em 2011 esta quantidade superou 1.8 ZB[2]. A estimativa é que para 2020 a proporção da expansão seja de 40%[19].

Atualmente, dados e informações são gerados pelos mais variados dispositivos. Casas inteligentes, dispositivos conectados, smartphones, tablets, wearables geram diariamente uma quantidade enorme de dados em suas operações [19, 12]. Segundo uma estimativa feita em 2015 pelo Gartner, em 2016 aproximadamente 6,4 bilhões de dispositivos estariam conectados à internet (30% a mais do que em 2015)[40]. Além da quantidade maior de dispositivos conectados à internet, a quantidade de pessoas conectadas à internet cresce. Era estimado que em 2016 aproximadamente 45% da população mundial estivesse conectada à internet, ou seja aproximadamente 3,4 bilhões de pessoas[19].

Além dos dispositivos conectados à internet, a utilização de redes sociais também contribui para o enorme aumento da quantidade de dados gerados diariamente em todo o mundo. Os dados gerados por estes serviços, além de caracterizado pela grande quantidade de bytes, também é caracterizado pela heterogeneidade[14]. Em redes sociais muitos dos dados trafegados são, além de textos, imagens, vídeos, etc[2].

Dessa quantidade imensa de dados emerge a oportunidade de Empresas ganharem diferenciais competitivos em seus mercados de atuação. Porém, para conseguir lidar com a quantidade de dados gerada diariamente e obter o maior retorno possível deles, é importante ter a capacidade de processá-los de uma forma muito rápida, se possível em real-time[19].

Segundo Jamiy et al apud Sheikh (2015), estes três aspectos citados anteriomente - Volume, Variedade e Velocidade, são os chamados 3Vs que caracterizam o que é chamado de

[1] 1 exabyte é equivalente à 1.048.576 terabytes
[2] 1 zetabyte é equivalente à 1.024 exabytes

Big Data, que podem ser descritos como[43, 19]:

- **Volume**: quantidade de dados muito grande, gerando a necessidade de novas formas para armazená-los, processá-los e gerenciá-los.
- **Variedade**: grande variedade do tipo de dado, como descrito anteriormente, e que consistem de dados estruturados (comum nos mais variados Sistemas de Informações), semi-estruturados (ex. dados capturados de sensores) e não-estruturados (ex. fotos, vídeos, textos).
- **Velocidade**: que representa tanto a velocidade na qual o dado precisa ser processado, quanto em que é criado.

  Estas características levam a necessidade de tecnologias capazes lidar com o armezanamento e processamento de um grande volume de dados de forma rápida e com custo que seja viável para os propósitos.

Atualmente pesquisadores destacam ainda outros dois Vs para caracterizar dados como Big Data [28]:

- **Veracidade**: que evidencia a necessidade de que tais dados tenham a confiabilidade, ou seja que estes dados estejam corretos. A importância desta característica é evidente uma vez que processar dados sem confiabilidade, gera, também, informações sem confiabilidade. A vericidade está relacionada à qualidade de dados.
- **Valor**: representa o valor que é possível ser obtido através dos dados, ou seja, representa o valor da informação. O emprego de dados de baixa qualidade pode levar a geração de informação sem valor, ou que pode conduzir a tomada de decisão incorreta.

Apesar do que pode ser alcançado em termos de oportunidade, buscar informações em meio a esta enorme montanha de dados é uma tarefa difícil e complexa[18]. Para auxiliar os usuários dos mais variados sistemas a encontrar a informação, produto ou serviço que deseja, uma importante classe de sistemas tem sido desenvolvida e aprimorada nos últimos anos: Sistemas de Recomendação (SR).

Segundo Bobadilla et al (2013), SR fazem uso de diferentes fontes de informações para prover aos usuários previsões e recomendações de itens[7]. Para Jain et al (2015) esta classe de sistemas abrange ferramentas e técnicas com o objetivo de fazer recomendações úteis de itens que possam interessar para um determinado usuário[18]. SR são largamente utilizadas por E-commerces [26, 50].

Um problema enfrentado por SR é a escassez de dados de novos usuários ou novos produtos para o E-commerce. Este problema é descrito na literatura como *cold-start* e *data sparsity*, respectivamente [18, 50, 10]. Quando um usuário é recém cadastrado no sistema,

poucas informações estão disponíveis sobre ele, dificultando que uma recomendação personalizada e aderente às suas preferências seja feita. Algumas propostas [10] utilizam dados demográficos fornecidos pelo usuário no momento do cadastro para efetuarem alguma recomendação. Contudo, neste caso, não se pode dizer que sigam a ideia da recomendação, pois se baseiam em dados gerais, ex. pessoas do sexo feminino, ou da faixa etária X, para prover uma recomendação. Não se pode assumir que determinada garota vai gostar de uma recomendação de um vestido rosa, baseado na informação de que a maioria das garotas gostam de rosa. A recomendação deve ser específica, ou seja, deve considerar os gostos individuais, e não os de uma categoria.

Tirar proveito do contexto Big Data para mitigar problemas como o de cold-start é um caminho natural para esta classe de sistemas [50]. Segundo Baldominos et al (2015), com o advento do Big Data, muitas pesquisas foram feitas abordando aspectos computacionais para lidar com grandes volumes e velocidade no tratamento de dados, possibilitando tirar proveito disto em SR[4].

Zhang e Pennacchiotti (2013) propõem uma abordagem utilizando dados de redes sociais para encontrar classes de produtos que o usuário potencialmente se interessaria[50]. Prando (2016), propõe uma abordagem que além de obter as classes de produtos de interesse dos usuários através dos dados sociais, também obtém os produtos de interesse utilizando técnicas de mineração de texto nas postagens dos usuários [35].

Um aspecto não abordado por Prando (2016) é compreender a opinião e o sentimento no conteúdo dos textos escritos pelos usuários nas redes sociais. Assim, em sua proposta tanto textos com teor negativo sobre um determinado produto quanto textos com teor positivo, são empregados e usados para gerar uma recomendação, porém sem diferenciar o que é ruim do que é bom.

1.2 Objetivos

Este trabalho tem como objetivo empregar técnicas de análise de sentimentos em dados extraídos de redes sociais para melhorar recomendações de um SR voltado para amenizar o problema cold-start. Assim, este trabalho estende o sistema de recomendação proposto por Prando (2016), visando melhorar o desempenho das recomendações geradas pela avaliação do sentimento contido o dado social que é extraído do Facebook e Twitter.

1.3 Método do trabalho

O desenvolvimento do trabalho será dividido da seguinte forma:

1.3.1 Levantamento bibliográfico e análise

O início do trabalho consiste do levantamento das referências bibliográficas sobre os assuntos relacionados à pesquisa. Estes assuntos incluem Sistemas de Recomendação, Redes Sociais, Aprendizado de Máquina, Análise Textual, Análise de Sentimentos e Big Data.

Serão analisadas as técnicas de análise de sentimentos com o objetivo de encontrar as que mais se adequam a resolução do problema proposto. Estas serão então detalhadas no trabalho com o objetivo de justificar as técnicas que serão utilizadas.

1.3.2 Escolha de algoritmos

Os algoritmos de Análise de Sentimentos foram avaliados sob os aspectos pertinentes ao objetivo do trabalho para a escolha dos que melhor se adequam à resolução do problema.

Para que a fazer a escolha, foram analisados e detalhados os possíveis algoritmos que podiam resolver o problema proposto. Considerou-se como determinante a existência da implementação do algoritmo no ambiente de implementação empregado no SR, pois não é objetivo deste trabalho implementar algoritmos de análise de sentimento, mas sim utilizá-los.

1.3.3 Avaliação das recomendações

Para avaliar os resultados obtidos a principal métrica a ser utilizada será o RMSE, assim como feito em Prando (2016). Isto possibilitará a comparação do resultado aqui obtido com o obtido por Prando (2016).

1.3.4 Obtenção de dados sociais

Como no trabalho de Prando (2016), as redes empregadas para a obtenção de dados dos usuários são: Facebook e Twitter.

Para a obtenção dos dados sociais dos usuários, foi desenvolvida uma funcionalidade que obtém os dados através das APIs disponibilizadas pelas redes sociais. Esta funcionalidade solicita ao usuário a autorização para acessar os dados que serão necessários para a recomendação (dados demográficos, postagens, “curtidas”).

1.3.5 Análise de resultados

Esta é a etapa em que discute-se a precisão das avaliações feitas com as técnicas propostas utilizando os dados minerados das redes sociais. Foram avaliados também aspectos como a escassez de dados do usuário nas redes sociais no momento da avaliação.

Neste etapa procede-se a análise comparativa entre as avaliações feitas com e sem o emprego das técnicas de análises de sentimentos, ou seja são comparados os resultados prévios obtidos por Prando (2016) com os obtidos neste trabalho.

Capítulo 2

Fundamentação Teórica

2.1 Introdução

Neste capítulo são apresentados os principais conceitos e problemáticas de sistemas de recomendação, inteligência artificial, mineração de dados e processamento de linguagem natural. A discussão sobre processamento de linguagem natural e mineração de dados é então complementada com o estudo sobre as principais técnicas que serão utilizadas no trabalho: mineração de textos, similaridade de cossenos e classificadores. Por fim, são explorados alguns trabalhos correlatos com grande relevância para a proposta deste trabalho.

2.2 Sistemas de Recomendação

Segundo Bobadilla et al (2013), Sistemas de Recomendação (SR) fazem uso de diferentes fontes de informações para prover aos usuários previsões e recomendações de itens [7]. SR são muito utilizados em E-commerces [26, 50]. Jamiy et al (2015) defende que a principal tarefa de SR aplicados ao comércio eletrônico é prever a relevância ou o potencial de compra de um determinado produto para um usuário específico, para estabelecer de forma ordenada uma lista de produtos como sugestão [19].

Algumas abordagens e técnicas que são utilizadas para os processos de recomendação feitos em SR são filtragem colaborativa, filtragem demográfica, recomendação baseada em conteúdo e híbrida[7]. Algumas outras técnicas são as de recomendação sensível ao contexto, baseada em utilidade, conteúdo, conhecimento, social [19, 10, 4][citar modarch alan].

A etapa do processo de recomendação que utiliza as técnicas citadas é denominada de filtragem. A tabela 2.1 contém uma breve descrição das principais técnicas.

Além das técnicas de filtragem, os métodos de recomendação são divididos em duas classificações: a) *memory-based* e b) *model-based* [7]. Bobadilla et al (2013) evidencia as seguintes definições para estas duas classificações [7]:

- **Memory-based**: tem como insumo os dados do passado dos usuários e em geral utiliza medida de similaridade entre itens e usuários para processar as recomendações.

Tabela 2.1: *Técnicas de filtragem aplicadas à sistemas de recomendação. Fonte: elaborado pelo autor*

Filtragem	**Descrição**	**Referência**
Colaborativa	utiliza a informação sobre um grupo de usuários e de itens a suas respectivas relações para processar a recomendação para o usuário final. É utilizado para recomendar produtos cuja essência é cultural, ou seja, cuja relação entre usuários e itens pode ser levada em consideração como filmes, restaurantes, artigos.	[19, 18]
Demográfica	utiliza informações demográficas como idade, sexo, gênero, educação e etc. para obter as recomendações desejadas. Esta abordagem se baseia no princípio de que usuários com certos atributos em comum também terão preferências em comum.	[18, 7]
Baseada em conteúdo	faz recomendações para os usuários baseado em suas ações do passado. Também utiliza o conteúdo dos itens e dos usuários para obter as recomendações.	[7, 19]
Sensível ao contexto	baseia-se em informações de contexto do usuário, como por exemplo, tempo, localização, sensores *wireless* e etc. para obter as recomendações. Estas informações podem ser obtidas explicitamente ou implicitamente.	[7]
Baseada em conhecimento	utiliza o conhecimento sobre usuários e produtos para inferir sobre as preferências do usuários e obter as recomendações.	[7]
Híbrida	utiliza uma ou mais das demais técnicas para chegar à recomendação para o usuário. Esta é a modalidade que é mais utilizada pois permite que os pontos fracos de uma determinada técnicas sejam fortalecidos por outras técnicas.	[7]

- **Model-based**: utiliza as informações contidas no SR para criar modelos que geram as recomendações. Alguns exemplos de técnicas que são utilizadas em SR que utilizam métodos model-based são classificadores Bayesianos, algoritmos bio-inspirados (como algoritmos genéticos e redes de imunidade artificiais), redes neurais entre outros.

Tanto para as abordagens de filtragem quanto para a escolha do método de recomendação existem algoritmos específicos que são aplicados para a resolução do problema. A escolha entre elas é feita levando em consideração a combinação do seguintes aspectos [7]:

1. os tipos de dados disponíveis para o SR utilizar no processo de recomendação,
2. O algoritmos de filtragem a ser utilizado (e.g. filtragem colaborativa, recomendação baseada em conteúdo),
3. Escolha entre a utilização direta dos dados ou a aplicação de modelos nestes dados,
4. As técnicas que serão utilizadas: abordagens probabilísticas, redes Baysianas, algoritmos de agrupamentos e classificação, decomposição e redução de dimensionalidade para diminuir o nível de dispersão dos dados e etc,
5. A escassez e dispersão dos dados e a necessidade de escalabilidade do sistema,
6. O objetivo que se deseja alcançar com as recomendações (por exemplo, alavancar a venda de produtos com maior rentabilidade),
7. A precisão desejada que deseja-se alcançar como resultado.

2.2.1 Formalização do problema de recomendação

O problema de recomendação pode ser reduzido ao problema de estimar avaliação que um determinado usuário faria para um determinado produto, considerando as informações disponíveis que existem de ambos. O sistema então tem como objetivo oferecer como recomendação os itens que foram considerados de maior importância para o usuário [1].

Segundo Adomavicius e Tuzhilin (2005) o problema de recomendação pode ser mais formalmente formulado da seguinte forma[1]:

Sejam U o conjunto de todos os usuários e K o conjunto de todos os itens de um determinado sistema. Definimos $util(u, k)$ como uma função utilidade, que mede o quanto um determinado item $k \in K$ é útil para um determinado usuário $u \in U$: $util : U \times K \mapsto R$.

O conjunto U é composto dos usuários do sistema. Cada usuário é unicamente representado. Adomavicius e Tuzhilin (2005) expõe que este conjunto pode conter muitos elementos.

Os elementos do conjunto K variam de acordo com o contexto no qual o sistema de recomendação é aplicado. Por exemplo, caso a recomendação seja de produtos a serem comprados, o conjunto K conterá todos os produtos disponíveis elegíveis a serem recomendados. Caso o sistema tenha como objetivo recomendar novas conexões de amizades, o conjunto

K conterá outros usuários do sistema. Adomavicius e Tuzhilin (2005) explicita que este conjunto também pode conter muitos elementos.

O conjunto R, que é contradomínio da função utilidade, pode ser qualquer conjunto completamente ordenado que represente a utilidade de um determinado item para um determinado usuário, como por exemplo um subconjunto finito dos números inteiros.

Então o objetivo é maximizar a utilidade para o usuário, ou seja, procuramos $k_u \in K$ de modo que a função $util(u, k_u)$ tenha o maior valor possível:

$$k_u = \arg\max_k util(u, k), \quad \forall u \in U \tag{2.1}$$

Em sistemas de recomendação a função utilidade é em geral caracterizada por avaliações, que indicam o quanto um determinado usuário tem interesse em determinado item. A descrição desta função pode variar dependendo do objetivo de cada aplicação.

Segundo Adomavicius e Tuzhilin (2005), o problema central da recomendação é que a função *util* em geral não é conhecida para todo o espaço $U \times K$, mas sim em apenas um subconjunto deste espaço. O caso onde, fixado determinado $u \in U$ e/ou $k \in K$, o subconjunto $\{u\} \times K$ e/ou $U \times \{k\}$ não tem nenhum valor definido para a função *util*, enfrentamos um problema conhecido na literatura como *cold-start*[1], que será aprofundado na próxima seção.

É necessário então que sejam aplicados métodos e técnicas para obter os valores de *util* no subconjunto desconhecido. Para um determinado $\bar{k} \in K$ onde a função $util(u, \bar{k})$ é desconhecida, precisamos estimar a avaliação do usuário $u \in U$ para o item $\bar{k}$ baseando-se em $util(u, \hat{k})$ que seja conhecido e $\hat{k} \in K$ é similar a $\bar{k}$. Por exemplo, um sistema de recomendação de músicas sugeriria a um determinado usuário músicas que tenham características parecidas com músicas que ele previamente já havia avaliado positivamente no passado (gênero, batidas por minuto (BPM), ano, país de origem e etc).

2.2.2 Desafio para os sistemas de recomendação: cold-start

Muitos desafios são enfrentados para que boas recomendações sejam feitas. Jain et al (2015) destaca alguns: falta de dados, mudança da preferência dos usuários, super-especialização, privacidade, *cold-start*, entre outros [18]. Para os propósitos deste trabalho, será aprofundado o problema denominado como *cold-start*.

Segundo Dang et al (2014), o problema de *cold-start* ocorre quando não é possível fazer uma recomendação confiável devido à falta inicial de avaliações de um determinado usuário ou item quando são recém inseridos no sistema [10]. Jain et al (2015) expõe que este problema tem duas categorias: novos itens e novos usuários [18]. Bobadilla et al (2013), traz ainda uma terceira categoria: nova comunidade [7].

Quando um novo item entra no sistema, não existe nenhuma avaliação sua feita por qualquer usuário, dificultando que este item seja recomendado. Neste caso, como o item

[1] Em tradução livre: arranque a frio

raramente será recomendado, poucas são as chances de que ele seja avaliado pelos usuários, fazendo assim que entre em um círculo vicioso de modo que um conjunto de itens é deixado de fora das possíveis recomendações do sistema [7].

O caso onde um novo usuário é cadastrado no sistema, não há qualquer informação sobre ele como, no caso de um E-commerce, avaliações feitas em produtos, histórico de compras, padrão de navegação ou coisas semelhantes. Esta segunda categoria, segundo Bobadilla et al (2013), é a mais importante para o sistemas de recomendação que já estão em operação atualmente.

A categoria nova comunidade do problema de *cold-start* ocorre quando um novo sistema de recomendação passa a ser utilizado e não há informações sobre os usuários e nem sobre os itens. Sistemas de recomendação que iniciam a sua operação sem conter dados dos usuários previamente se enquadram nesta categoria [7].

2.3 Inteligência Artificial

Existem oito definições de Inteligência Artificial (IA) que são dominantes e definem as estratégias para o seu estudo. Estas definições são agrupadas em quatro categorias que definem tais estratégias, as quais são seguidas por pessoas diferentes e utilizam métodos diferentes[33].

Segundo Haugeland (1985) IA é "O novo e interessante esforço para fazer computadores pensarem (...) máquinas com mentes, no sentido total e literal."[16] Para Bellman (1978) IA endereça a "[Automatização de] atividades que associamos ao pensando humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado..."[5] Estas duas definições, segundo Novig e Russell (2013), tratam de processos de pensamento e raciocínio e medem o sucesso em termos de fidelidade ao raciocínio humano[33].

Já para Charniak e McDermott (1985) IA é "O estudo das faculdades mentais pelo uso de modelos computacionais."[8] De forma semelhante Winston (1992) define como "O estudo das computações que tornam possível perceber, raciocinar e agir."[48] Estas duas definições, ressaltam Norvig e Russell (2013), tratam também de processos de pensamento e raciocínio, porém medem o sucesso comparando-o a um conceito ideal de inteligência, que é chamado de racionalidade: "um sistema é racional se 'faz a coisa certa', dado o que sabe."[33]

Outras duas definições tratam IA no âmbito do comportamento. Para Kurzweil (1990) é "A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas"[23] e segundo Rich (1991) é "O estudo de como computadores podem fazer tarefas que hoje são melhor desempenhadas pelas pessoas."[39] Estas duas definições medem o sucesso comparando-se ao desempenho humano[33].

Para Poole et al (1998) "Inteligência Computacional é o estudo do projeto de agentes inteligentes" [34] e para Nilsson (1998) "IA... está relacionada a um desempenho inteligente de artefatos."[32] Estas duas definições tratam o problema de IA como de comportamento,

porém buscam o sucesso pela racionalidade[33].

Para avaliar se uma máquina de fato é inteligente, Turing (1950) propôs um teste chamado de Jogo da Imitação[45], que é chamado atualmente de Teste de Turing [30]. Um computador passará pelo teste se um interrogador humano não conseguir distinguir se as respostas dadas pela máquina são de um humano ou não[33]. A proposta de Turing era que o interrogador e os interrogados (um computador e uma pessoa) estivessem em salas separadas de modo que não fosse possível comunicar-se diretamente. Se o interrogador não conseguisse decidir se as respostas eram dadas pelo computador ou pela pessoa, então pode-se supor que a máquina é inteligente[30].

Para que um computador consiga passar no teste de Turing é necessário que ele tenha muitas capacidades[33]:

- **processamento de linguagem natural** para que seja possível a comunicação entre o interrogador e ele, de modo que se consiga intepretar a pergunta feita e comunicar de forma clara a resposta;
- **representação de conhecimento** de modo que seja possível armazenar o que sabe e ouve;
- **raciocínio automatizado** para que consiga, utilizando o conhecimento armazenado e tirando novas conclusões, responder às questões propostas pelo interrogador;
- **aprendizado de máquina** de tal modo que seja possível adaptar-se às circunstâncias e, aprendendo com os padrões do que aconteceu no passado, extrapolá-los.

Essas quatro disciplinas citadas anteriomente, somadas a **visão computacional** e **robótica**, compõem a maior parte dos estudos de IA. Para os propósitos deste trabalho serão aprofundados os assuntos aprendizagem de máquina e processamento de linguagem natural.

Atualmente muitas aplicações utilizam IA para executar tarefas e melhorar as experiências dos usuários: aplicativos são a cada dia aprimorados com mais recursos inteligentes; *websites* utilizam técnicas e métodos de IA para aumentar o seu retorno e lucratividade; serviços oferecidos *online* incorporam recursos automatizados que poupam muito tempo de seus usuários. Alguns exemplos de aplicações que evidenciam o que IA é capaz de fazer em seu estado atual: veículos robóticos, reconhecimento de fala, tradução automática, combate à *spam*, recomendações inteligentes, robótica[33].

2.3.1 Aprendizado de máquina

Aprendizagem é uma das áreas desafiadoras que são estudadas no campo da IA. Esta é uma tarefa de grande importância para os sistemas inteligentes, pois é um dos componentes mais importantes do comportamento inteligente [30]. Segundo Norvig e Russell (2013) pode-se considerar que um sistema estará aprendendo se conseguir melhorar o seu desempenho em suas tarefas após fazer observações sobre o mundo [33].

Uma definição feita por Simon (1983) para aprendizagem é:

> "Representa mudanças no sistema que são adaptativas no sentido de que elas permitem que o sistema faça uma mesma tarefa ou tarefas desenhadas para uma mesma população mais eficientemente e mais efetivamente na próxima vez." [42]

A importância da aprendizagem está relacionada ao fato de que é uma tarefa impossível prever todas as situações em que um determinado sistema pode encontrar-se e também as mudanças pelas quais as situações podem passar no decorrer do tempo. Um exemplo disto seria uma aplicação que tem como objetivo prever o mercado de ações no dia seguinte: muitos são os cenários possíveis e mudanças podem ocorrer de súbito[33].

Outro aspecto que evidencia a importância da aprendizagem é que ainda que uma determinada tarefa seja possível de ser feita por humanos, nem sempre é factível que seja codificada em uma solução genérica: identificar se uma pessoa é da família ou não baseado em suas características é uma tarefa possível de ser feita por humanos, mas mesmo os melhores programadores não são capazes de programar um computador para executar esta tarefa, exceto por meios de aprendizagem[33].

Norvig e Russell (2013) classificam a aprendizagem em três principais categorias, cuja distinção é feita pelo tipo de retroalimentação que o sistema recebe. Estas três categorias são: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço.

- A **aprendizagem supervisionada** assume a existência de um professor, alguma forma de adequação ou alguma entrada externa de exemplos de treinamento [30]. Este tipo de aprendizagem tenta induzir regras, dados os exemplos passados para o sistema. Alguns exemplos de aplicações que utilizam aprendizagem supervisionada são:
  - classificação de spam
  - reconhecimento facial
  - classificação de textos
- Já a **aprendizagem não supervisionada** pressupõe padrões de entrada para sistema, mas fornecidos implicitamente [33], ou seja, não é necessária a existência de um professor que dê exemplos para o aprendizado, o sistema reconhece e avalia os conceitos [30]. Um exemplo de aprendizagem não supervisionada são os algoritmos de agrupamento, que tem a capacidade de detectar automaticamente potenciais grupos úteis em uma determinada entrada [33].
- E por fim na **aprendizagem por reforço**, o sistema aprende a partir de uma série de estímulos - positivos ou negativos, que são fornecidos [33]. Sistemas que utilizam este tipo de aprendizagem transformam situações em ações baseados na análise de quais trazem maiores recompensas, também levando em conta os reflexos subsequentes destas

ações (esta característica é chamada de reforço atrasado). Esta forma de aprendizagem é mais geral do que as abordagens supervisionada e não supervisionada [30].

Neste trabalho serão utilizadas técnicas de aprendizagem supervisionada com três objetivos: (i) associar cada um dos dados obtidos através das redes sociais a categorias de produtos, (ii) classificar um determinado dado social como positivo ou negativo, ou seja, fazer a classificação do sentimento contido no texto e (iii) treinar as os modelos que terão a capacidade de fazer as classificações dos objetivos (i) e (ii).

2.3.2 Processamento de Linguagem Natural

Segundo Luger (2006) um dos objetivos da Inteligência Artificial é o processamento e compreensão da linguagem humana [30]. A linguagem é parte fundamental da inteligência humana. Avanços na automação deste processamento têm um impacto enorme sobre a facilidade de utilização e eficácia dos próprios computadores.

Alan Turing (1950) ao propor o seu jogo da imitação (que é conhecido atualmente como Teste de Turing) utilizou a linguagem como meio para identificar se uma máquina de fato passa em seu teste [45]. Há duas razões principais pelas quais é importante que os sistemas atuais tenham a capacidade de processar linguagem natural: se comunicar com os seres humanos e adquirir informações com base na linguagem escrita. Na internet há mais de um trilhão de páginas onde muito conhecimento pode ser adquirido [33].

Para Jurafsky e Martin (2009) processamento de linguagem natural (PLN) são as técnicas computacionais que processam a fala e a escrita humana como linguagem. Esta é uma definição muito ampla que inclui desde uma simples contagem de palavras até uma extremamente complexa como tradução de fala em tempo real . Muitos esforços são feitos para que as tecnologias necessárias para esta compreensão evoluam, mas muitos avanços ainda são necessários [21].

O que distingue os sistemas PLN dos demais sistemas de processamento de dados é a necessidade da utilização de conhecimento sobre a linguagem. Mesmo programas extremamente simples, como por exemplo um que faça a contagem de palavras contidas em uma frase tem a necessidade de ter conhecimento sobre o que significa ser uma palavra, o que o torna um sistema PLN, mesmo que simples [21].

Atualmente muitas das aplicações que fazem parte do dia-a-dia de grande parte das pessoas utilizam técnicas de processamento de linguagem natural. Aplicativos desenvolvidos para smartphones a cada dia trazem mais facilidades e automações de tarefas relacionadas à linguagem. Em smartphones com sistemas operacionais Android ou iOS estão implementados recursos de correções de escrita, de reconhecimento de voz. Programas como o Skype fazem tradução de fala em tempo real.

> "O Skype Translator usa aprendizado de máquina. Por isso, quanto mais você usa a ferramenta, melhor ela fica." (https://www.skype.com/pt-br/features/skype-translator, acessado em 16/07/2016)

Ambiguidade é um dos grandes problemas enfrentados por PLN [31]. Existem várias formas de se definir e interpretar cada componente de uma sentença de tal forma que o seu significado seja alterado. Segundo Jurafsky e Martin (2009) grade parte das tarefas de sistemas PLN podem ser vistas como problemas de ambiguidade em algum nível [21]. Por exemplo a sentença "Sentado na varanda, José avistou seu amigo." traz o seguinte questionamento: quem estava sentado na varanda, José ou seu amigo?

Os desafios que são enfrentados pelos estudos de PLN, além do citado no parágrafo anterior, são inúmeros. Conhecimentos profundos sobre fonética, fonologia, morfologia, sintaxe, semântica são necessários para que um programa de computador consiga reconhecer a fala de um ser humano ou até mesmo um texto escrito. Alguns dos problemas que são objetivos de PLN: correção de escrita, verificação de gramática, recuperação de informação, tradução automatizada [21]. Outro tema muito estudado atualmente com aplicações em áreas como sistemas de recomendação [38, 24] e análise de opinião pública [29] é análise de sentimentos.

Para a resolução destes problemas encontrados pelos sistemas de PLN muitas técnicas são empregadas. Métodos estatísticos podem ser empregados [21, 31] e também técnicas de mineração de dados e aprendizado de máquina [30, 33].

2.4 Mineração de Dados

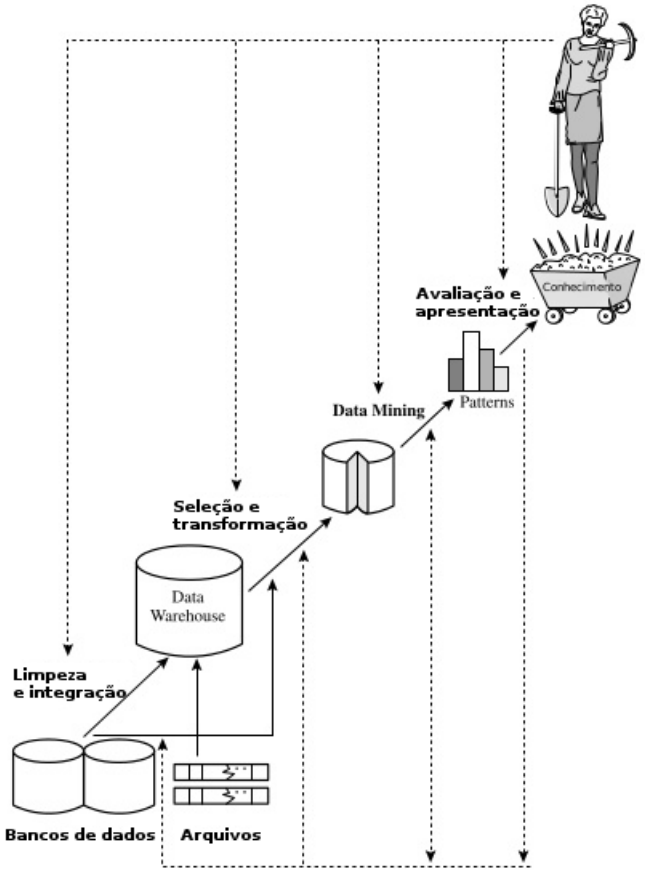
Atualmente muitos dados são gerados diariamente e a necessidade de analisá-los é eminente. Segundo Han et al (2012), mineração é o processo de descoberta de padrões interessantes e conhecimento de uma grande quantidade de dados. As fontes de dados podem ser bancos de dados, internet, outros repositórios ou até mesmo dados que são inseridos no sistema dinamicamente [15].

Mineração de dados é uma importante etapa da descoberta de conhecimento nos dados (em ingles *Knowledge Discovery from Data* - KDD) [15] e além disto desempenha um papel essencial em outras áreas do conhecimento. Outras etapas do processo de descoberta e conhecimento de dados são: sanitização de dados, integração de dados, seleção de dados, trasformação de dados, avaliação de padrões e apresentação do conhecimento. A figura 2.1 ilustra a jornada da descoberta de conhecimento e evidencia onde a mineração de dados se encaixa neste contexto.

Han et al (2012) destaca algumas das funcionalidades de mineração de dados: caracterização e discriminação; mineração de padrões frequentes, associações e correlações; classificação e regressão; análise de agrupamentos; e análise de valores extremos [15]. Tarefas executadas por estas funcionalidades podem ser classificadas como descritivas ou preditivas. As tarefas descritivas verificam e evidenciam as propriedades dos dados que estão sendo analisados, enquanto que as tarefas preditivas fazem induções nos dados de modo a obter previsões baseadas neles.

- **Caracterização e discriminação**: caracterização de dados é a sumarização das prin-

Figura 2.1: *Mineração de dados como um passo da descoberta de conhecimento de dados*



***Fonte**: Han et al (2012)*

cipais características de uma determinada classe de dados. Discriminação de dados é a comparação entre as características gerais de uma determinada classe de dados e outras classes de dados [15].

- **Mineração de padrões, associações e correlações**: esta funcionalidade da mineração de dados está relacionada a descoberta de associações interessantes existentes nos dados [15].

- **Classificação e regressão**: é o processo de encontrar um modelo quee descreva e distingua classes e conceitos dos dados [15]. A funcionalidade de classificação será aprofundada nas próximas seções para o propósito deste trabalho.

- **Análise de agrupamentos**: busca nos dados relações, de modo que seja possível agrupá-los de uma forma automática. Este processo muitas vezes é utilizado como um passo inicial antes de utilizar-se um processo de classificação [15].

- **Análise de valores extremos**: valores que não estão de acordo com o comportamento

geral apresentado pelos dados observados são considerados valores extremos e, em geral, são descartados por muitos métodos. Porém em algumas aplicações estes eventos raros são extremamente importantes (por exemplo, detecção de fraudes). Esta funcionalidade estuda justamente a ocorrência destes eventos raros [15].

A área de minineração de dados incorporou técnicas de diversas outras áreas como estatística, aprendizado de máquina, reconhecimento de padrões, bancos de dados, computação de alta performance entre outros [15].

Um exemplo que evidencia o quanto estas demais disciplinas são importantes para a área de mineração de dados é o problema de classificação citado anteriomente [15]. Segundo Han et al (2012), o problema de aprendizagem supervisionada pode ser enxergado como um problema de classificação, onde a supervisão no aprendizado vem de exemplos rotulados nos dados de treinamento dos algoritmos de mineração de dados. Já o problema de aprendizagem não supervisionada pode ser interpretado como um problema de agrupamento, de modo que os padrões são reconhecidos automaticamente nos dados [15].

As diferenças encontradas entre as áreas de mineração de dados e aprendizado de máquina são, muitas vezes, o foco: para as tarefas de classificação e agrupamento, a área de aprendizado de máquina tem um grande esforço para melhorar a acuracidade dos modelos. A área de mineração de dados, além da preocupação com a acuracidade também tem uma grande preocupação com a eficiência e escalabilidade dos métodos aplicados [15].

2.5 Análise de Sentimentos

Segundo Singh e Dubey (2014), análise de sentimentos é uma combinação de processamento natural de linguagem e mineração de texto que utiliza técnicas de aprendizado de máquina para classificar textos como positivos ou negativos [44]. Guellil e Boukhalfa (2015) destaca que análise de sentimentos é uma tarefa de PLN para identificar o conteúdo subjetivo (contendo sentimentos) e classificá-lo como positivo, negativo ou neutro. Bhatia et al (2015) destacam que as técnicas de análises de sentimentos tem o objetivo de detectar e extrair informaçõe subjetivas a partir de documentos de texto e executar esta tarefa utilizando-se dados não estruturados é um problema que desencadeia uma série de desafios [6].

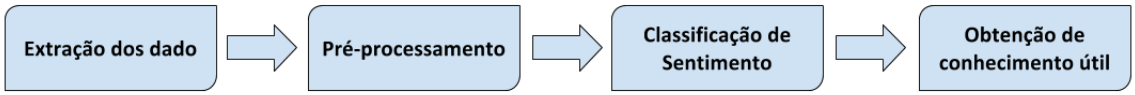
O problema de identificar o sentimento contido em uma determinada frase ou documento pode ser abordado como um problema de classificação [14, 27] e uma das abordagens possíveis para a sua solução é a utilização de métodos de aprendizado de máquina e mineração de dados como Máquinas de Vetores de Suporte (em inglês Support Vector Machines - SVM), classificadores Baysianos [6, 14]. Segundo Colace et al (2013), a classificação de sentimentos é a principal tarefa da análise de sentimentos [9]. Guellil e Boukhalfa (2015) destacam três categoria para esta classificação [14]:

- **A nível de documento** que tem como objetivo classificar como positivo, negativo ou neutro um documento completo,

- **A nível de sentença** cujo objetivo é classificar o sentimento de cada sentença de um determinado documento,
- **A nível de aspecto** que indentifica textos e sentenção que contenham determinado aspecto que deseja-se buscar e após isto o classifica como positivo, negativo ou neutro.

Um processo global de análise de sentimentos, ilustrado na 2.2 consiste em quatro passos: extração dos dados, pré-processamento, classificação de sentimento e identificação do conhecimento útil [14].

Figura 2.2: *Fluxo global de um processo de análise de sentimentos*



***Fonte**: elaborado pelo autor*

A primeira das etapas da análise de sentimento é a extração dos dados, onde os dados são obtidos das fontes de interesse. Segundo Guellil e Boukhalfa (2015) a obtenção dos dados se dá de três formas diferentes [14]:

1. a partir de um *corpus* já existente (*existing corpus*) que utiliza-se de textos já existentes anteriormente,
2. a partir de um *corpus* manual (*manual corpus*) extraído diretamente das redes sociais,
3. a partir de um *corpus* automático (*automatic corpus*) que é extraído automaticamente das redes sociais utilizando-se as APIs (Application Programming Interfaces) disponíveis.

Na etapa de pré-processamento são feitos tratamentos nos dados para possibilitar a etapa de classificação. Algumas das técnicas aplicadas nesta etapa são: quebra das sentenças em palavras, tradução, *lemmatization* ou *stemming* que têm como objetivo reduzir as palavras para uma forma mais primitiva, *stop word removal* cujo objetivo é tirar palavras comuns a língua (e.g. 'e', 'ou', 'para', 'de') [14]. As técnicas pertencentes a esta etapa que serão utilizadas no presente trabalho serão aprofundadas nas próximas seções.

Na terceira etapa a classificação do sentimento da sentença ou documento é efetivamente realizada. Segundo Wan (2012), Guellil e Boukhalfa (2015) e Singh e Dubey (2014) esta tarefa pode ser feita utilizando-se aprendizado de máquina [47, 14, 44].

Guellil e Boukhalfa (2015) destacam que a maioria dos trabalhos que endereçam a terceira etapa com aprendizagem supervisionada utilizam SVM como classificador (método é determinístico). Outras duas técnicas com recorrência de utilização são redes Bayesianas e

máxima entropia (ambos os métodos são probabilisticos). Os trabalhos que utilizam aprendizagem não supervisionada são em sua grande maioria baseados em dicionários [14].

A quarta e última etapa do processo de análise de sentimentos é a obtenção do conhecimento útil gerado a partir do processo. Durante este passo a classificação obtida na terceira etapa do processo é utilizada de forma a obter ganhos com a informação [14].

2.6 Técnicas e Algoritmos utilizados

Esta seção tem como objetivo definir as técnicas e algoritmos que foram utilizados no trabalho proposto: técnicas de mineração de textos, TD-IDF, similaridade de cossenos e classificadores que utilizam aprendizado de máquina.

2.6.1 Mineração de Texto

Segundo Hotho (2005), mineração de texto lida com a análise de textos de forma computacional, utilizando conhecimentos de recuperação de informação (em inglês Information Retrieval), extração de informação e PLN e conecta estas áreas através de técnicas de mineração de dados, aprendizado de máquina e estatística [17].

Definimos como *corpus* o conjunto de documentos que são considerados e como *corpora* o conjunto de *corpus* utilizados [31]. Para que seja possível obter conhecimento através dos textos, é necessário que sejam aplicadas técnicas para préprocessá-los e armazená-los em uma estrutura de dados apropriada para o processamento [17]. Uma abordagem possível é considerar cada documento como um conjunto de palavras, denominada como *Bag of Words* (BOW). Para que seja definida uma importância para cada um dos termos presentes em um documento, técnicas são aplicadas com o objetivo de criar uma representação vetorial. Após isto é possível que sejam obtidos conhecimentos a partir dos textos.

2.6.2 Técnicas de pré-processamento

Duas técnicas da etapa de pré-processamento dos dados e que são utilizadas neste trabalho são Filtragem e de *Stemming*, que removem ou alteram termos do texto com o objetivo de eliminar palavras com baixa relevância e reduzir a dimensão do espaço vetorial que contém os vetores que serão processados posteriomente [17].

Métodos de filtragem removem palavras específicas que não são relevantes para o processamento do texto. Uma técnica de filtragem é a eliminação de palavras como artigos, conjunções e preposições, que são conhecidas na literatura como *stop words* [46, 14, 13, 17]. Esta técnica é chamada de *stop words removal* e será utilizada neste trabalho para a redução da quantidade de termos e eliminação de palavras irrelevantes para o processamento.

O método de *Stemming* tem como objetivo obter a forma mais básica de cada termo do documento, retirando 's' de palavras que estão no plural, 'indo', 'endo', 'ando' de termos

que estejam no gerúndio e etc para que palavras que tenham o mesmo significado sejam representadas de uma única forma [17]. Esta técnica será utilizada no trabalho também na etapa de pré-processamento.

2.6.3 TF-IDF

A técnica TF-IDF (em inglês *Term Frequency/Inverse Document Frequency*) tem como objetivo obter as características de um determinado texto. Com esta técnica é possível determinar a relevância de cada termo de um determinado documento em relação a todos os demais documentos do *corpus*. Segundo Liu (2012) esta técnica tem se mostrado efetiva para a classificação de textos [27].

Formalmente esta técnica pode ser expressa como [1]: Seja N número o total de documentos considerados, $w_i \in W$ um termo que aparece n_i vezes nestes documentos e f_{ij} o número de vezes que o termo w_i aparece no documento $d_j \in D$, onde W é o conjunto de todos os termos presentes em todos os documentos do *corpus* e D é o conjunto de todos os documentos do *corpus*. Definimos então a frequência do termo w_i no documento d_j (ou frequência normalizada) como:

$$TF_{ij} = \frac{f_{ij}}{max_z f_{zj}} \tag{2.2}$$

onde o máximo é calculado para todos os termos $w_z \in W$ que pertençam ao documento $d_j \in D$.

Definimos então a frequência inversa de um determinado termo w_i como:

$$IDF_i = log\frac{N}{n_i} \tag{2.3}$$

Por fim, definimos o TF-IDF do termo w_i no documento d_j como:

$$t_{ij} = TF_{ij} \times IDF_i \tag{2.4}$$

Com esta técnica é possível obter uma medida de relevância de cada termo de um determinado documento em relação a todos os documentos que são considerados. Termos que aparecem recorrentemente penalizam a medida, enquanto que termos pouco frequentes nos outros documentos são considerados relevantes e elevam a medida. Por exemplo, em um conjunto de textos que contenham informações sobre computadores, palavras como "Gabinete", "RAM", "Teclado"aparecerão em muitos documentos fazendo com que o IDF destes termos seja um número pequeno e, por consequência, com que a medida TFIDF destes termos seja também pequena. Termos que aparecem com uma frequência menor nos documentos, que podem ser considerados chave para distinguir o conteúdo de um determinado documento dos outros, terão a medida TF-IDF maior, demonstrando maior relevância do termo.

Utilizando-se esta técnica é possível criar uma representação vetorial dos documentos,

que é conhecida na literatura de VSM (em inglês *Vector Space Model*)[29, 49] e é obtida da seguinte forma [1]: para um determinado documento d_j que contenha k termos, definimos a função $Conteudo(d_j) = (t_{1j}, t_{2j}, ..., t_{kj})$ para obter o vetor que representa este termo. O VSM é obtido aplicando-se a função conteúdo em todos os documentos do *corpus*.

Neste trabalho, o objetivo é utilizar esta técnica para obter uma representação vetorial dos textos contidos em cada dado social dos usuários e também dos produtos que poderão ser sugeridos. Também será utilizada esta técnica para fazer a classificação dos dados sociais com relação ao sentimento, ou seja, para determinar se um determinado dado social é positivo ou negativo.

2.6.4 Similarilidade de cossenos

Em um determinado espaço vetorial é possível obter uma medida similaridade entre dois vetores através do cálculo do cosseno ângulo entre eles. Seguno Lima (2012) em um determinado espaço vetorial com produto interno, o cosseno do ângulo entre dois vetores u, v não-nulos é definimo como [25]:

$$cos(u, v) = \frac{< u, v >}{|u||v|} \tag{2.5}$$

onde $< u, v >$ é o produto interno entre u e v e $|x|$ é a norma de um determinado vetor x.

A partir desta definição de cosseno entre dois vetores quaisquer de um espaço vetorial, é possível obter uma medida de similaridade entre eles. Vetores cujo cosseno do ângulo entre eles seja próximo a 1 têm uma grande similaridade, enquanto que vetores cujo cosseno do ângulo entre eles seja menor ou igual a zero, não têm nenhuma similaridade [1, 24, 37, 18]. A partir disso, é possível definir a seguinte medida de similaridade :

$$simcos(u, v) = max\{0, cos(u, v)\} \tag{2.6}$$

Neste trabalho, o objetivo é utilizar esta medida de similaridade com os vetores do VSM obtido após a aplicação da técnica de TF-IDF para obter produtos que são semelhantes aos dados sociais dos usuários.

2.6.5 Classificadores

A classificação de texto tem o objetivo de associar documentos com classes pré-definidas [17]. Para executar esta tarefa são requeridas técnicas de mineração de dados e aprendizado de máquina. Neste trabalho, a tarefa de classificação é aplicada em dois contextos: associar dados sociais a categorias de produtos e fazer a classificação de sentimentos de um determinado documento.

Os classificadores que serão utilizados são Naive Bayes e SVM, que serão aprofundados nas próximas seções.

Naive Bayes

Classificadores Bayesianos são classificadores estatísticos supervisionados que têm a capacidade de prever a probabilidade de que um determinado item pertença a uma classe, dada a informação de outros itens que pertecem às classes. Estes métodos baseiam-se no Teorema de Bayes para obter a classificação [15]. Segundo Han et al (2012) classificadores como Naive Bayes têm a sua performance comparável com classificadores mais complexos como Árvores de Decisões e Redes Neurais, além de terem uma grande acuracidade, eles demonstram uma grande performance em bancos de dados grandes.

Temos que assumir a hipótese de que o efeito do valor de um atributo em uma determinada classe é independente dos valores dos outros atributos. Esta hipótese é chamada de independência de classe condicional (em inglês Class-Conditional Independence) [15]. Segundo Jarecki, Meder e Nelson (2013), assumir esta hipótese, mesmo que ela não seja completamente válida, não implica que o método perca robustes [20, 11].

O Teorema de Bayes nos traz que:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.7}$$

O método de classificação Naive Bayes pode ser formalizado da seguinte forma [15]:

Seja D um conjunto que contenha os vetores que serão utilizados para treinar o classificador. Cada coordenada do vetor $X = (x_1, x_2, ..., x_n)$, $x \in D$, representa a medida dos atributos $A_1, A_2, ..., A_n$, respectivamente. Suponha que existam m possíveis classes $C_1, C_2, ..., C_m$ para que cada elemento de D esteja associado. Dado um vetor $Y \notin D$, o classificador irá prever a classe cuja probabilidade seja a maior possível, ou seja, escolhemos i tal que $P(C_i|X) > P(C_j|X), i \neq j$. Pelo Teorema de Bayes:

$$P(C_i|x) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{2.8}$$

Como para qualquer classe C_i o valor de $P(X)$ é constante, precisamos maximizar o valor de $P(X|C_i)P(C_i)$. Calculamos $P(C_i)$ a partir dos dados de treinamento: $P(C_i) = |C_{i,D}|/|D|$, onde $|C_{i,D}|$ é a quantidade de vetores que sejam da classe C_i em nosso conjunto de treinamento D e $|D|$ é o número total de vetores de treinamento. Como fizemos a hipótese de independência de classe condicional, podemos calcular:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i) \tag{2.9}$$

onde $P(x_k|C_i) = |C_{k,i,D}|/|C_{i,D}|$ em que $|C_{k,i,D}|$ é o número de vetores em D que estão associados à classe C_i e que tem o valor x_k para o atributo A_k.

O classificador então escolhe a classe C_i para o vetor X se, e somente se

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad para \quad 1 \leq j \leq m, j \neq i \tag{2.10}$$

Este método será utilizado no trabalho para obter uma categoria de produtos a qual um determinado dado social pode ser associado.

SVM - Suport Vector Machine

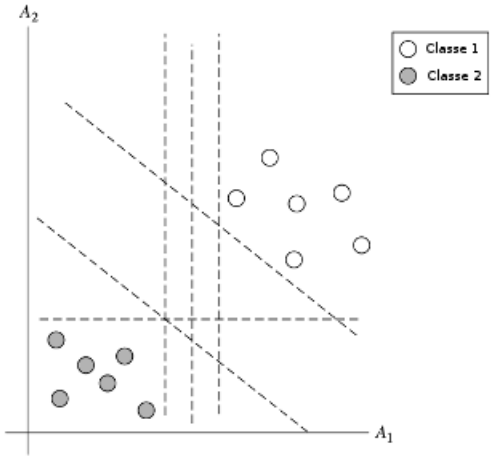
Segundo Han et al (2012), SVM é um método para classificação de dados lineares e não lineares, que utiliza um mapeamento não linear para transformar os dados originais de treinamento em uma dimensão maior, na qual procura um hiperplano que seja um separador ótimo das classes encontradas nos dados [15]. Han et al (2012) destaca que este método é muito menos propenso a ter *overfitting* nos dados do que os outros modelos [15].

Este método será utilizado no trabalho para classificar o sentimento contido em um determinado texto como negativo ou positivo, portanto apenas o problema que classifica entre duas classes será necessário. Segundo Singh e Dubey (2014) este método é muito utilizado para a tarefa de classificação de sentimentos em textos [44] e segundo Guellil e Boukhalfa (2015) é o mais utilizado na literatura para a classificação de sentimentos quando a abordagem é a de aprendizado de máquina supervisionado [14].

Formalmente este método pode ser descrito da seguinte forma:

Seja D o conjunto de treinamento e $(X_i = (x_1, x_2, ..., x_k), y_i) \in D$, onde x_k são atributos do vetor X_i e y_i é classe atribuída a este vetor. No caso que será tratado, $y_i \in \{+1, -1\}$, que pode ser considerado como o sentimento positivo e negativo. Supondo um caso onde existam apenas dois atributos, uma representação destes dados poderia ser feito como na figura 2.3

Figura 2.3: *SVM: exemplo dos dados com apenas dois atributos*



***Fonte**: Han et al (2012)*

A separação feita entre os dados poderia ser escolhida de muitas formas [15]. O que o método SVM tem como objetivo é escolher a melhor forma de separar os dados, ou seja, obtendo o menor erro de classificação possível, baseado nos dados de treinamento. No caso onde apenas dois atributos existem, o método procura uma reta que separa da melhor

forma os dados, em um caso onde existem três atributos, o que é procurado é um plano. Generazilando para os casos de dimensão n, o método procura um hiperplano que melhor faça a separação dos dados [15].

Seja $W = (w_1, w_2, ..., w_n)$ um vetor de pesos, n é o número de atributos utilizados, m o número total de vetores em D e b um número real, o hiperplano procurado pode ser escrito como

$$WX + b = 0 \tag{2.11}$$

O vetor W é ajustado de modo que o hiperplano descrito por 2.11 defina dois "lados":

$$H_1 : WX + b \geq 1 \quad para \quad y_i = +1, \quad e \tag{2.12}$$

$$H_2 : WX + b \leq -1 \quad para \quad y_i = -1 \tag{2.13}$$

A partir de 2.11, 2.12 e 2.13, é possível obter:

$$y_i(WX + b) \geq 1, \quad \forall i \tag{2.14}$$

É necessário então maximizar a distância entre a sepação feita por 2.12 e 2.13. Para fazer isto, é necessário que a norma quadrática de W, $||W||$, seja minimizada, onde $||W|| = \sqrt{w_1x_1 + w_2x_2 + ... + w_nx_n}$, dado que a condição descrita em 2.14 seja satisfeita, ou seja:

$$\begin{aligned} \min_{W} \quad & \frac{1}{2}||W||^2 \\ \text{sujeito a} \quad & -y_i(WX_i + b) + 1 \leq 0, \quad 1 \leq i \leq m \end{aligned} \tag{2.15}$$

Construindo o Langrangeano para este problema de otimização:

$$\mathcal{L}(W, b, \alpha) = \frac{1}{2}||W||^2 - \sum_{i=1}^{m} \alpha_i[y_i(WX_i + b) - 1] \tag{2.16}$$

Encontrando as derivadas do Lagrangeano em relação a W e a b, temos:

$$\nabla_W \mathcal{L}(W, b, \alpha) = W - \sum_{i=1}^{m} \alpha_i y_i X_i = 0 \tag{2.17}$$

$$W = \sum_{i=1}^{m} \alpha_i y_i X_i \tag{2.18}$$

$$\frac{\partial}{\partial b}\mathcal{L}(W, b, \alpha) = \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.19}$$

Aplicando o que foi encontrado em 2.18 na equação 2.16, obtemos que

$$\mathcal{L}(W, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle X_i, X_j \rangle \tag{2.20}$$

Encontramos então o seguinte problema dual:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle X_i, X_j \rangle \\ \text{sujeito a} \quad & \alpha_i \geq 0, \quad 1 \leq i \leq m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{2.21}$$

Resolvendo o problema dual a solução problema original é encontrada e com isto fica definido o hiperplano que tem a capacidade de fazer as classificações desejadas.

2.7 Trabalhos correlatos

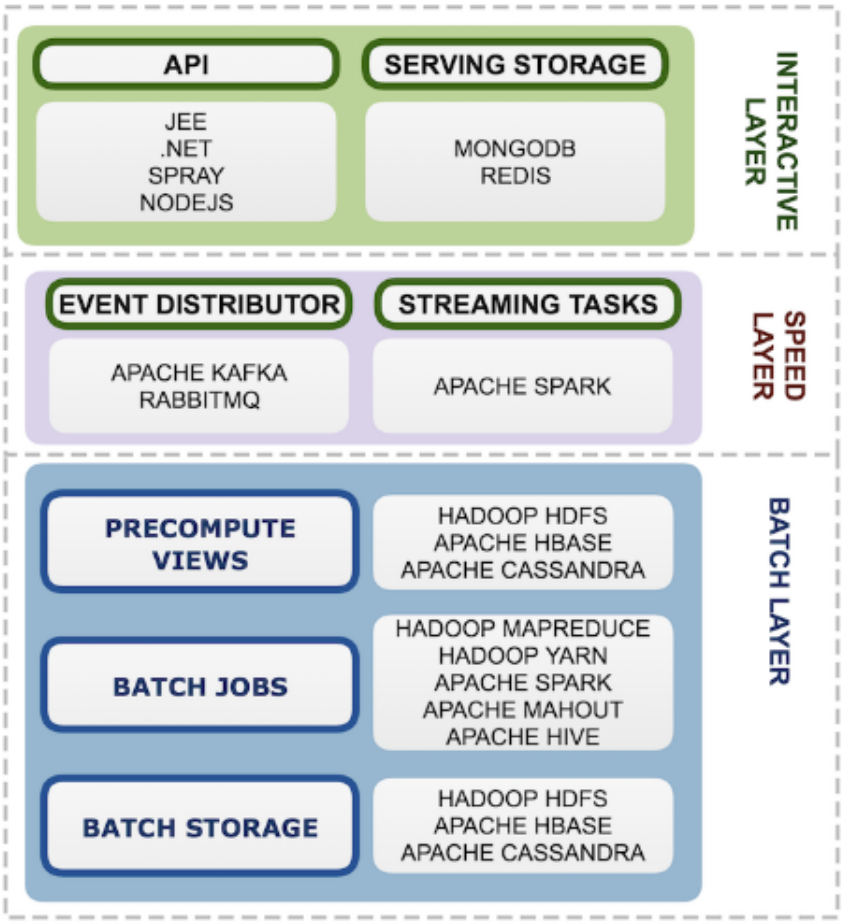
Prando (2016) propôs como possível solução para amenizar o problema de *cold-start* para novos usuários a utilização de dados extraídos das redes sociais Facebook e Twitter, aplicando técnicas de mineração de texto e aprendizado de máquina para obter recomendações. Utilizando as mensagens escritas pelos usuários, o sistema busca encontrar sugestões de produtos para que sejam personalizadas para eles e aderentes as suas preferências. A utilização do conteúdo produzido ou compartilhado pelo usuário caracteriza o processo de recomendação como **baseado em conteúdo** [7, 19].

Prando e Souza (2016) proposuram a seguinte arquitetura SR descrito, que é composto pelas camadas *Interactive Layer*, *Speed Layer* e *Batch Layer* [36]:

- ***Interactive Layer***: responsável por receber as requisições de recomendação feitas pela aplicação que interage com o usuário,
- ***Speed Layer***: responsável por processar a recomendação em tempo real,
- ***Batch Layer***: responsável por executar tarefas cujo tempo de resposta é maior e não são necessárias em tempo real.

A figura 2.4 representa a arquitetura proposta com as tecnologias sugeridas para cada camada.

Figura 2.4: *Arquitetura do SR*



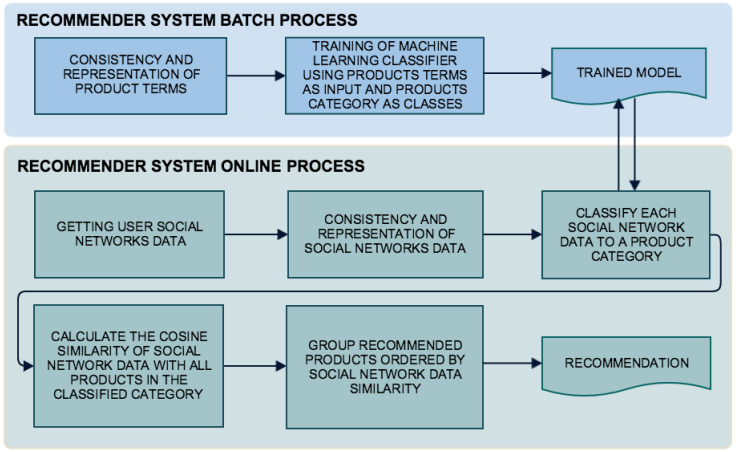
***Fonte**: Prando e Souza (2016)*

O processo de recomendação é iniciado com uma requisição feita para a camada *Interactive Layer*, que recupera as informações das redes sociais e as passa para a camada *Speed Layer*. Esta utiliza modelos criados e treinados na camada *Batch Layer*, os quais ficam disponíveis para utilização. Ao final, as sugestões são apresentadas ao usuário.

Para obter as recomendações a partir dos textos extraídos das redes sociais, o sistema armazena os dados utilizando uma representação conveniente para o processamento. Cada dado social é considerado como um documento e, utilizando as técnicas de *Stemming* e *Stop Word Removal* (Seção 2.6.2), é gerado um *corpus* dos dados sociais do usuário e dos produtos. A partir destes corpus, utilizando a técnica TF-IDF, é feita a representação vetorial dos textos para a construção do VSM (Seção 2.6.3). Por um processo semelhante é obtido o *corpus* dos produtos. Cada vetor deste VSM que representa um dado social que é então submetido a um classificador *Naive Bayes* previamente treinado (Seção 2.6.5, que o associa a uma determinada categoria de produtos. Após a obtenção da categoria de produtos, é calculada a similaridade entre este vetor e todos os vetores associados aos produtos pertencentes à categoria obtida pelo classificador, utilizando a técnica Similaridade de Cossenos (Seção 2.6.4).

O fluxo deste sistema de recomendação pode ser divido em dois, conforme a figura 2.5: (1) treinamento dos algoritmos de aprendizado de máquina que são utilizados pelo SR e (2) processamento da recomendação.

Figura 2.5: *Fluxo de recomendação do sistema proposto por Prando (2016)*



Fonte: *Prando (2016)*

O fluxo (1), no qual é feito o treinamento do modelo de classificação, é composto por duas etapas, com o objetivo de disponibilizar o modelo treinado:

- **Representação vetorial dos produtos**: tem como resultado um VSM em que cada vetor representa um produto através da técnica TF-IDF (Seção 2.6.3) e está associado a uma categoria de produtos;
- **Treinamento do classificador *Naive Bayes***: utilizando o VSM criado no passo anterior, o classificador Naive Bayes (Seção 2.6.5) é treinado para classificar vetores associando-os às categorias de produtos.

O fluxo (2) é o responsável para geração da recomendação personalizada para cada usuário. Esta etapa é composta pelos seguintes procedimentos:

1. **Obtenção dos dados sociais**: através das APIs disponibilizadas pelo Facebook e pelo Twitter, o sistema busca, mediante a autorização do usuário, os dados sociais (textos publicados)
2. **Representação vetorial dos dados sociais**: criação do VSM de dados sociais utilizando um espaço vetorial de mesma dimensão e ordenação do VSM de produtos. Esta etapa utiliza, assim como no fluxo (1), a técnica TF-IDF
3. **Classificação do dado social**: utilizando o classificador disponibilizado no fluxo (1), cada texto publicado pelo usuário é associado a uma categoria de produtos;

4. **Cálculo de similaridade**: para cada vetor do VSM de dados sociais, é calculada a similaridade de cossenos (Seção 2.6.4) com todos os vetores do VSM de produtos cuja categoria pertença à mesma do dado social;

5. **Agrupamento das recomendações ordenador por similaridade**: ordena as recomendações geradas pela sua similaridade para que sejam sugeridos os produtos cuja similaridade calculada sejam as maiores possíveis.

Através das etapas descritas anteriormente o sistema processa a recomendação baseada no conteúdo do usuário e a disponibiliza.

Um aspecto não abordado por Prando (2016) é o sentimento (positivo ou negativo) contido nos dados sociais dos usuários. É possível que um determinado texto publicado pelo usuário tenha conteúdo negativo sobre determinado assunto e o sistema gere uma recomendação relacionada a este assunto. Por exemplo: para um texto como "Eu realmente odeio os Beatles!" o sistema tenderá a sugerir produtos relacionados à banda The Beatles, porém o texto indica que o usuário rejeitará esta recomendação.

A proposta deste trabalho é utilizar as técnicas de análise de sentimentos (Seção 2.5) para evitar que este cenário descrito anteriomente ocorra para que as recomendações geradas sejam mais coerentes para o usuário.

2.8 Conclusão

Este capítulo apresentou os principais aspectos sobre SR, descreveu o problema de *cold-start* encontrado na literatura, apresentou as principais técnicas utilizadas no sistema proposto por Prando (2016) e descreveu as técnicas de Análise de Sentimentos que serão utilizadas nas novas etapas que serão adicionadas no sistema.

Baseando-se no trabalho de Guellil e Boukhalfa (2015), o classificador a ser utilizado para obter os sentimentos dos dados sociais é o SVM, portanto este capítulo também descreveu a sua teoria.

Na validação do SR, assim como em Prando (2016), será utilizada a métrica Root Mean Square Error (RMSE) também usada nas propostas de Zhang (2013) [50] e Amatriain (2013) [3], com o objetivo de avaliar a recomendação e a relevância dos dados recuperados das redes sociais para usuários *cold-start*.

Capítulo 3

Especificações do sistema proposto

3.1 Introdução

Este trabalho tem como proposta utilizar técnicas de Análise de Sentimentos (Seção 2.5) para complementar o sistema de recomendação proposto por Prando (2016), que utiliza técnicas de AM (Aprendizado de Máquina) e mineração de texto para amenizar o problema de *cold-start* (descrito na seção 2.2.2) [35]. Similarmente a Prando (2016), O SR aqui proposto também utiliza os textos publicados pelo usuário e os relacionado a produtos do E-Commerce para gerar recomendações, o que caracteriza a abordagem como baseada em conteúdo. Para avaliar a efetividade da proposta, foi eleborado um experimento que simula a interação do usuário com um E-Commerce no qual cada recomendação gerada é avaliada com uma nota de 1 a 5 (o apêndice A contém a descrição da página web construída).

Apresenta-se neste capítulo os detalhes do SR proposto, que na recomendação considera o sentimento contido no conteúdo publicado pelo usuário.

3.2 Definição dos Vector Space Models

Para que sejam aplicadas as técnicas que são propostas para a resolução do problema, é necessário que os dados que serão processados estejam em uma estrutura adequada para os algoritmos. Esta seção tem como objetivo definir os VSMs que serão utilizados no SR (conforme descrito na seção 2.6.3).

3.2.1 Produtos do E-Commerce

Os dados dos produtos considerados foram obtidos através da utilização de um *Web Crawler* criado com o pacote de código aberto "scrapy" (SCRAPY, 2016) em um grande E-Commerce brasileiro. Foram *crawleados* 12.771 produtos e persistidos em forma de documentos em um banco de dados não relacional (MongoDB). Cada documento é composto pelas seguintes informações:

- Categorias: até 4 níveis, sendo que o primeiro nível representa se é um livro, dvd ou cd;
- Nome: título do produto que é considerado no E-Commerce;
- Endereços web do produto e da imagem: urls nos quais o produto encontrava-se no momento que foi obtido;
- Descrição: conteúdo existente no E-Commerce que descreve o produto.

Diferenciando-se do que foi utilizado por Prando (2016), que considerou em sua base de produtos Eletrodomésticos e Eletrônicos, este trabalho utiliza Livros, DVDs e CDs como base de produtos a serem recomendados. Esta alteração foi feita com o objetivo de captar os aspectos culturais contidos nas publicações feitas pelos usuários e conseguir mais aderência nas sugestões feitas. Além disto, a quantidade de produtos utilizadas por Prando (2016) foi de 1.249 produtos, ou seja, menos de 10% da quantidade utilizada neste trabalho.

Para a representação dos produtos do E-Commerce, é proposta a utilização da técnica TF-IDF. Cada produto do E-commerce será representado como um vetor numérico $p_i \in VSM_p$, $p_i = (w_{1i}, w_{2i}, ..., w_{ki})$, onde w_{ji} é a representatividade do termo j para o produto i em relação a todo o *corpus*, obtido através da aplicação da função $Conteudo(p_i)$, conforme descrito na seção 2.6.3. Os termos considerados para cada produto são os contidos em sua descrição.

Antes da aplicação da função *Conteudo* é necessário que sejam aplicadas as técnicas de pré-processamento para a mineração de textos descritas na seção 2.6.2: Filtragem e de *Stemming*, que removem ou alteram termos do texto com o objetivo de eliminar palavras com baixa relevância e reduzir a dimensão do espaço vetorial que será utilizado (VSM_p) [17]. Desta forma são mantidos apenas os substantivos simples e compostos no *corpus* considerado e todos eles são reduzidos a sua forma mais primitiva utilizando-se as técnicas de *Stemming*. As técnicas de pré-processamento foram aplicadas utilizando-se o pacote de código aberto "nltk" em Python (NLTK PACKAGE, 2015).

Após o pré-processamento, aplica-se a técnica TF-IDF e assim obtem-se o espaço vetorial VSM_p contendo todos os produtos do E-Commerce em um formato adequado para a utilização nos algoritmos de AM. Além da representação vetorial p_i para o produto i é necessário o conhecimento sobre a categoria de produtos c_i a qual i pertence (livro, dvd ou cd) de tal modo que o $Pr_i = (c_i, p_i) \in P_{c_i}$ é a representação completa do produto i dentro do SR proposto e o conjunto de todos os produtos é dado pela equação 3.1 a seguir.

$$P = \bigcup_{c \in C} P_c \tag{3.1}$$

onde C é o conjunto de todas as categorias contidas no E-Commerce.

3.2.2 Dados sociais

Os dados sociais são obtidos através das APIs disponibilizadas pelas redes sociais Facebook e Twitter, mediante a autorização do usuário. Caso o usuário não conceda a autorização, o SR não é capaz de gerar as recomendações baseadas em seus conteúdos publicados e sinaliza de que não foi possível obter recomendações.

A representação vetorial VSM_s dos dados sociais é obtida da mesma forma como foi obtida a dos produtos (descrita na seção 3.2.1). O espaço vetorial considerado para o VSM_s tem a mesma dimensão e também mesma ordenação do espaço vetorial considerado para os produtos. Caso algum termo encontrado nos dados sociais não exista no espaço de produtos, ele é desconsiderado.

3.3 Classificador de Categoria

A representação vetorial dos produtos é utilizada para o treinamento de um classificador Naive Bayes (Seção 2.6.5) que determina a categoria $cp_i = c_j \in C$ de produtos a qual uma publicação i do usuário se assemelha. A escolha pelo classificador Naive Bayes foi feita baseada nos resultados obtidos por Prand (2016). O objetivo desta classificação é reduzir a quantidade de produtos que o SR terá que considerar para fazer a recomendação.

O conjuto total de vetores considerados para o treinamento do classificador foi divido em dois subconjuntos para possibilitar mensuração da acuracidade obtida com o classificador. O primeiro subconjuto (com 80% dos elementos existentes) foi utilizado para o treinamento do classificador, conforme descrição feita na seção 2.6.5. O segundo subconjuto (com os 20% restantes dos elementos) foi usado para verificar a acuracidade do modelo. O resultado obtido foi de $93,11\%$ de acerto para estas novas instâncias.

O modelo do classificador é previamente treinado e fica disponível para que durante o processo de recomendação seja utilizado. Esta funcionalidade foi criada utilizando-se a biblioteca de código aberto MLLIB contida no Apache Spark em linguagem Python.

3.4 Detalhamento do processo de recomendação

Para a obtenção das recomendações, o SR utiliza os textos publicados pelo usuário na representação descrita na seção 3.2 e calcula a similaridade com os produtos do E-Commerce. Para que não sejam necessárias as comparações com todos os produtos, cada dado social é classificado em uma categoria de produtos utilizando-se o classificador Naive Bayes já treinado com os dados dos produtos, conforme descrito na seção 3.3. A representação completa $So_i = (cp_i, s_i) \in S$ do dado social i é então obtida, onde cp_i é a categoria prevista do dado social i e s_i a sua representação vetorial no VSM_s.

Para cada $So_i \in S$ o sistema calcula a similaridade de cossenos (Seção 2.6.4) com todos os produtos $Pr_i \in P_{cp_i}$, obtendo assim uma medida de similaridade entre cada dado social e

os produtos do E-Commerce. São recomendados então os dez produtos cujas similaridades com os dados sociais sejam as maiores possíveis. A figura 3.1, a seguir ilustra este fluxo de recomendação.

3.4.1 Processo de análise de sentimentos

Como descrito na seção 2.5, a identificação do conteúdo subjetivo e classificação de seu sentimento em positivo ou negativo é um problema de Análise de Sentimentos e existem várias técnicas possíveis para abordá-lo.

O problema de identificar o sentimento contido em uma determinada frase pode ser abordado como um problema de classificação [14, 27] e uma das abordagens possíveis para a sua solução é a utilização de classificadores Baysianos e SVM [6, 14]. Segundo Singh e Dubey (2014), como a técnica SVM fornece a maior diferença marginal entre as classes, ou seja, consegue uma melhor distinção entre as classes pois maximiza a distância da separação feita dos dados, ela é a mais apropriada para a tarefa de classificação [44], portanto a proposta deste trabalho é utilizá-la para realizar a classificação do sentimento contido nas publicações dos usuários.

Treinamento do classificador SVM

Para fazer o treinamento do classificador SVM foi utilizada uma base de dados contendo 12.176 publicações reais de usuários nas redes sociais classificadas em positivas ou negativas. A classificação foi feita manualmente por usuários e disponibilizada para esta pesquisa.

O processo para representação vetorial das publicações foi feito da mesma forma como descrito na seção 3.2. Cada publicação i foi representada como $Tr_i = (sentimento_i, t_i) \in T$, onde $sentimento_i \in \{0, 1\}$ - 0 representando o sentimento negativo e 1 representando o sentimento positivo, e t_i é representação vetorial da publicação de treinamento i aplicando-se as técnicas de pré-processamento e TF-IDF (Seções 2.6.2 e 2.6.3).

O conjuto total de vetores considerados para o treinamento do classificador foi divido em dois subconjuntos para possibilitar mensuração da acuracidade obtida com o classificador. O primeiro subconjuto (com 80% dos elementos existentes) foi utilizado para o treinamento do classificador, ou seja, utilizado para a obtenção do vetor de pesos $W = (w_1, w_2, ..., w_n)$, onde n é o número total de termos considerados, conforme descrição feita na seção 2.6.5. O segundo subconjuto (com os 20% restantes dos elementos) foi usado para verificar a acuracidade do modelo. O resultado obtido foi de $80,26\%$ de acerto para estas novas instâncias.

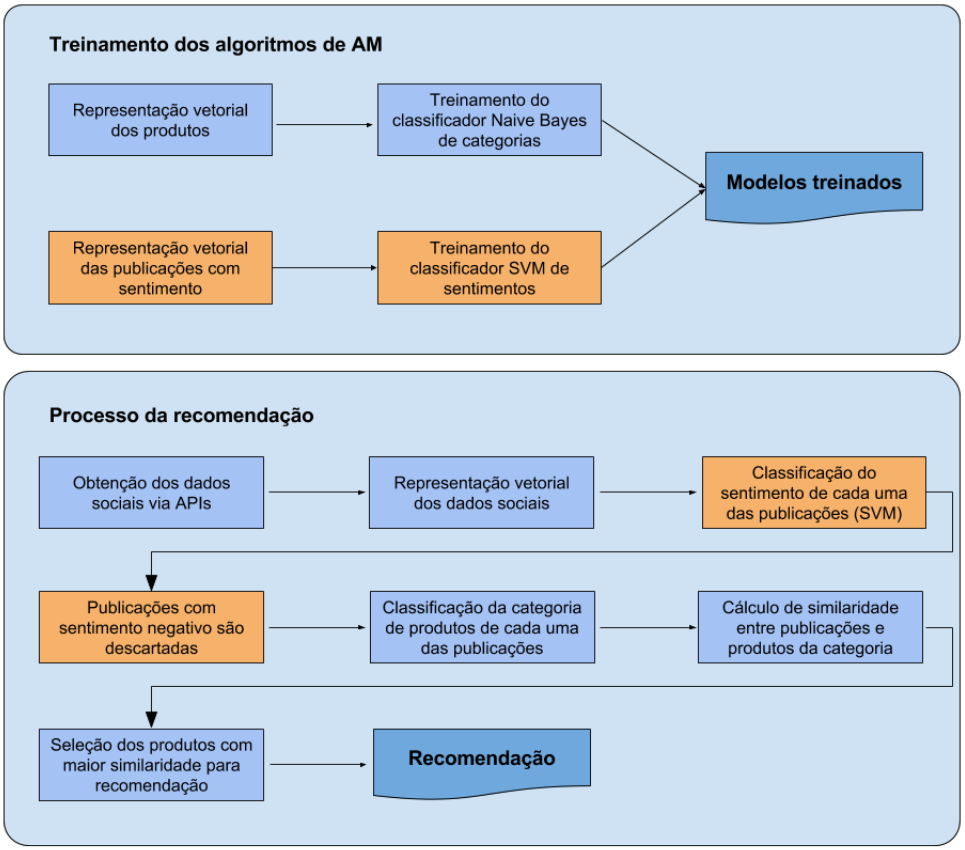
O modelo do classificador SVM foi previamente treinado com os dados citados anteriormente e disponibilizado para utilização no momento da recomendação. Esta funcionalidade foi criada utilizando-se a biblioteca de código aberto MLLIB contida no Apache Spark em linguagem Python.

3.4.2 Novo fluxo proposto

Para avaliar o sentimento expresso nos dados sociais utilizouse-se SVM, sendo descartadas as publicações com sentimento negativo, ou seja, apenas publicações com sentimento positivo são consideradas para o processo de recomendação.

O novo fluxo para a recomendação é o descrito pela figura 3.1:

Figura 3.1: *Fluxo de recomendação proposto*



Fonte: *elaborado pelo autor*

As etapas que foram adicionadas estão destacadas em laranja na figura 3.1. O código-fonte utilizado para o processamento da recomendação está disponível no apêndice B.

Treinamento dos algoritmos de AM

Este novo fluxo de treinamento dos algoritmos de AM descrito foi implementado utilizando-se a linguagem Python e processado na plataformade computação distribuída Apache Spark, utilizando-sea biblioteca MLLIB contida nesta plataforma:

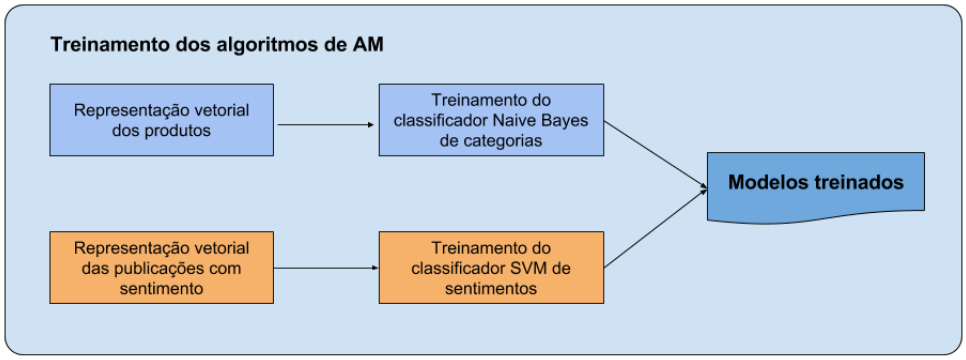
1. É feita a representação vetorial P dos produtos contidos no E-Commerce;
2. Utilizando-se P é executado o treinamento do classificado Naive Bayes para a classificação das categorias de produtos, que é então disponibilizado;

3. As publicações com a classificação dos sentimentos são representadas vetorialmente, gerando assim o conjunto T;

4. A partir de T o classificador SVM é treinado e disponibilizado.

A figura 3.2 representa este fluxo:

Figura 3.2: *Fluxo de treinamento dos classificadores*



Fonte: *elaborado pelo autor*

Processo de recomendação

A nova proposta de fluxo (2) contém os processos descritos pelo SR proposto por Prando (2016) e são adicionados os novos passos para classificação dos sentimentos dos textos dos usuários:

1. São obtidos os dados sociais do Facebook e/ou Twitter através das APIs disponibilizadas pelas redes sociais;

2. É feita então a representação vetorial destes dados sociais;

3. As publicações são classificadas em relação ao sentimento como positivas ou negativas utilizando-se o classificador SVM já disponível - as publicações com o sentimento negativo são descartadas;

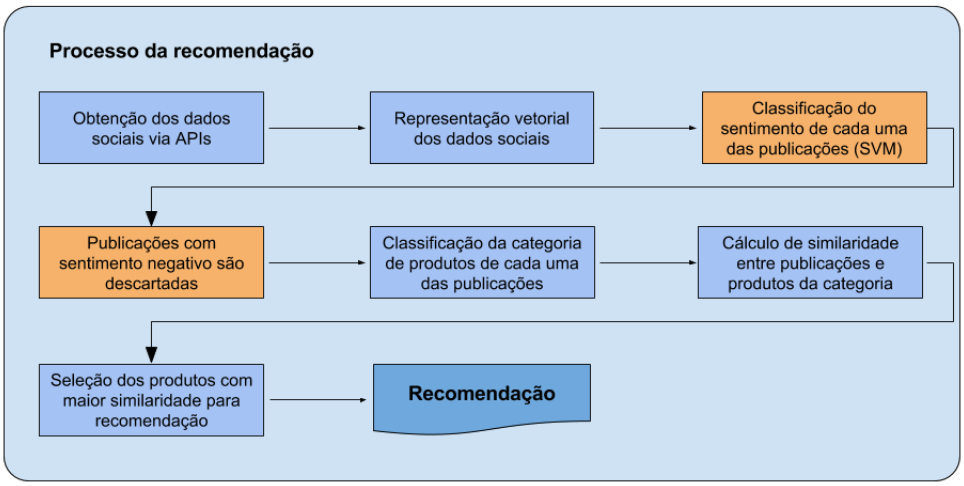
4. Cada publicação positiva do usuário é classificada em uma categoria de produtos utilizando-se o classificador Naive Bayes;

5. É calculada a similaridade entre as publicações e todos os produtos contidos na categoria de produtos na qual o dado social foi enquadrado;

6. São selecionadas as dez recomendações com as maiores similaridades calculadas.

A figura 3.3 representa este fluxo:

Figura 3.3: *Fluxo de recomendação*



***Fonte**: elaborado pelo autor*

3.5 Conclusão

Este capítulo descreveu o SR proposto para tratar o problema de *cold-start*, empregando apenas dados sociais e considerando o sentimento positivo/negativo contido nas publicações dos usuários. Foram descritas as formas utilizadas para a definição dos espaços vetoriais utilizados, bem como a sua utilização para o treinamento dos classificadores e cálculo das similaridades para gerar as recomendações para os usuários.

Capítulo 4

Análise de Resultados

4.1 Introdução

Nos capítulos anteriores foram apresentados os fundamentos teóricos das técnicas utilizadas e os detalhes do SR proposto. Foram descritos os detalhes do procedimento para obter a recomendação da proposta bem como as etapas do fluxo de recomendação proposto.

O objetivo deste capítulo é apresentar o procedimento seguido na obtenção dos resultados, além da análise dos mesmos.

4.2 Experimento para obtenção dos resultados

Um experimento foi elaborado e distribuído de forma aleatória através da rede social Facebook e sem uma prévia explicação do objetivo da aplicação. Houve uma participação integral (todas as fases) do experimento de 64 pessoas. O tamanho da amostra foi considerado para obter-se uma boa significância estatísica na análise dos resultados, considerando uma margem de erro de 10% e um intevalo de confiança de 90% [22].

O experimento aplicado simula a interação entre o usuário em seu primeiro acesso e o E-Commerce. Para a obtenção das recomendações o sistema necessita que o usuário dê autorização para acesso aos dados utilizando as suas credenciais do Facebook e/ou inserindo o nome de acesso utilizado no Twitter. Caso nenhuma das duas concessões sejam feitas, é informada a impossibilidade de processar com as recomendações.

Após as validação dos acessos, são processadas as recomendações e exibidas no máximo dez para o usuário, ordenadas em ordem descrente pela similaridade obtida. Para cada recomendação exibida, o usuário deve atribuir uma nota de 1 a 5, de tal modo que 1 represente pouca afinidade com a recomendação feita e 5 represente muita afinidade. Caso o usuário não atribua nenhuma nota, a recomendação é descartada.

As etapas descritas nos parágrafos anteriores podem ser visualizadas no apêndice A.

4.2.1 Nota do sistema

Para transformar a medida de similaridade obtida em um número comparável com a nota atribuída pelo usuário, aplica-se a equação 4.1, que tem com objetivo transformar $s \in [0, 1]$ em $n_s \in \{1, 2, 3, 4\}$, onde s é similaridade obtida pelo sistema e n_s é nota associada a esta similaridade:

$$notaSistema(p_{ij}, s_{ij}, i) = \left\lfloor \frac{simcos(p_{ij}, s_{ij})}{\max\limits_{p \in P_i, s \in S_i} simcos(p, s)} \times 4 \right\rfloor + 1 \tag{4.1}$$

onde:

- $simcos(u, v)$ é função definida na seção 2.6.4,
- p_{ij} é o produto j recomendado para o usuário i,
- s_{ij} é o dado social relacionado ao produto j recomendado para o usuário i,
- P_i é conjunto de todos os produtos recomendados para o usuário i e
- S_i é conjunto de todos os dados sociais relacionados aos produtos contidos em P_i.

4.3 Medida de erro: Root Mean Square Error (RMSE)

Segundo Shani e Guanawardana (2011), em aplicações que tem como objetivo prever a nota que um determinado usuário daria para um item recomendado, uma métrica muito utilizada para avaliar a acuracidade das notas previstas é o RMSE [41]. Neste trabalho esta métrica foi considerada para a avaliação dos resultados.

Considerando $\hat{n}_{ij} = notaSistema(p_{ij}, s_{ij}, i)$ - equação 4.1, como sendo a nota atribuída pelo sistema para o produto j recomendado para o usuário i, n_{ij} a nota atribuída pelo usuário i para o produto j recomendado, N a quantidade total de usuários participantes e M_i a quantidade de recomendações feitas para o usuário i, o cálculo do RMSE é dado por:

$$RMSE(i, j) = \sqrt{\frac{1}{K} \sum_{i=1}^{N} \sum_{j=1}^{M_i} (\hat{n}_{ij} - n_{ij})^2} \tag{4.2}$$

$$K = \sum_{i=1}^{N} \sum_{j=i}^{M_i} 1 \tag{4.3}$$

O valor do RMSE varia entre 0 e 4, pois como $|\hat{n}_{ij} - n_{ij}| \leq 4$, temos que $0 \leq (\hat{n}_{ij} - n_{ij})^2 \leq 4^2$, e portanto:

$$0 \leq \sqrt{\frac{1}{K} \sum_{i=1}^{N} \sum_{j=1}^{M_i} (\hat{n}_{ij} - n_{ij})^2} \leq \sqrt{\frac{1}{K} \sum_{i=1}^{N} \sum_{j=1}^{M_i} 4^2} = \sqrt{\frac{4^2}{K} \sum_{i=1}^{N} \sum_{j=1}^{M_i} 1} = \sqrt{4^2} = 4$$

Logo, temos:

$$0 \leq \sqrt{\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{M_i}(\hat{n}_{ij} - n_{ij})^2} \leq 4 \tag{4.4}$$

4.4 Visão geral dos resultados

A infraestrutura utilizada, devido a restrições de custos, foi de baixo poder computacional e isto ocasionou problemas nas execuções e um alto nível de desistência dos usuários, que saíram do experimento antes da conclusão do processamento das recomendações.

Os principais dados sobre os acessos ao experimento são:

- 163 usuários interagiram com o experimento inserindo as credenciais do Facebook e/ou do Twitter;
- Para 128 usuários as recomendações foram geradas com sucesso;
- Em 35 casos não foram geradas as recomendações pelas seguintes razões: não concessão de todos os acessos necessários para o Facebook, Twitter inexistente ou concorrência no processamento da recomendação;
- Do total de 128 usuários, 64 todo o processo e avaliaram as recomendações geradas;
- Dos usuários que avaliaram as recomendações, 9 utilizaram apenas o Twitter, 49 utilizaram apenas o Facebook e 6 utilizaram ambas as redes sociais;
- Foram avaliadas no total 581 recomendações.

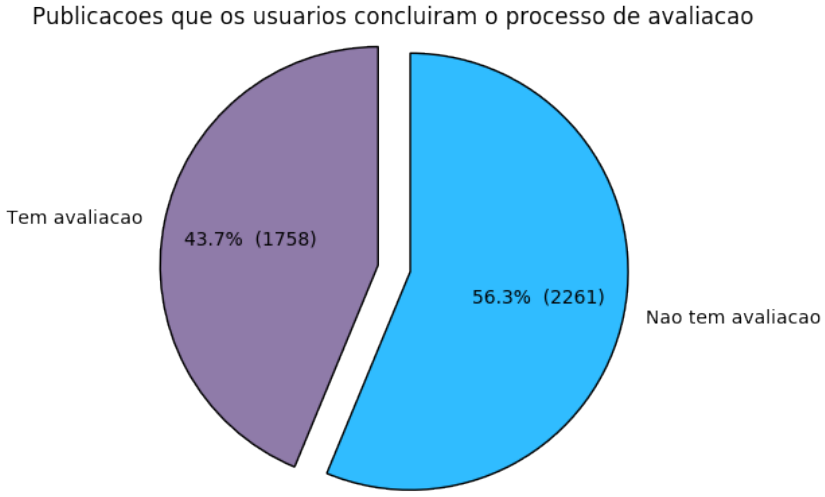
Devido a natureza do conteúdo dos dados sociais, grande parte das similaridades concentrou-se na faixa de 0 a $0,05$. A maior similaridade obtida foi de $0,71$ em apenas um caso. Os casos com maiores valores para a similaridade foram os que identificaram palavras comuns entre os textos e os produtos, por exemplo: a publicação "Faço muita escolha errada nessa vida, mas essa com certeza não foi uma delas" encontrou com o livro "O que a Vida é 50 Anos Depois" para o qual a similaridade foi 0,28. A tabela 4.1 e a figura 4.1 indicam a quantidade de recomendações que foram geradas para cada range de similaridade.

Foram processadas no total 4.019 publicações obtidas através das redes sociais Facebook e Twitter. Deste total, 1.758 publicações são de usuários que finalizaram o processo de avaliação, ou seja 43,7%. Portanto, houve um grande número de publicações processadas, porém devido a limitação computacional, muitas recomendações não foram avaliadas pelos usuários. A figura 4.2 ilustra este resultado.

Figura 4.1: *Quantidade de recomendações por range de similaridade*

***Fonte**: elaborado pelo autor*

Figura 4.2: *Publicações que os usuários concluíram o processo de avaliação*



***Fonte**: elaborado pelo autor*

Assim, para um cenário de *cold-start* e utilizando apenas dados postadosnas redes sociais já citadas, considera-se que o SR proposto apresenta boa assertividade e que pode ser aperfeiçoado com o emprego de outros dados obtidos da rede social, como dados demográficos (ex. idade, sexo, localização). A comparação efetiva como trabalho de Prando (2016) não foi possível, pois os dados e usuários são diferentes.

Tabela 4.1: *Quantidade de recomendações por range de similaridade*

Range de similaridade	Quantidade de recomendações	% do total
0 a 0,01	42	7,2%
0,01 a 0,02	198	34,1%
0,02 a 0,03	97	16,7%
0,03 a 0,04	93	16%
0,04 a 0,05	53	9,1%
0,05 a 0,06	17	2,9%
0,06 a 0,07	20	3,4%
0,07 a 0,08	17	2,9%
0,08 a 0,09	7	1,2%
0,09 a 0,1	9	1,5%
0,1 a 0,11	10	1,7%
0,11 a 0,28	16	2,8%
0,28 a 0,71	1	0,2%
0,71 a 1	1	0,2%

***Fonte**: elaborado pelo autor*

Tabela 4.2: *Diferença em módulo entre as notas.*

Diferença em módulo	Quantidade de recomendações	% do total
0	101	17,4%
1	223	38,4%
2	159	27,4%
3	66	11,4%
4	32	5,5%

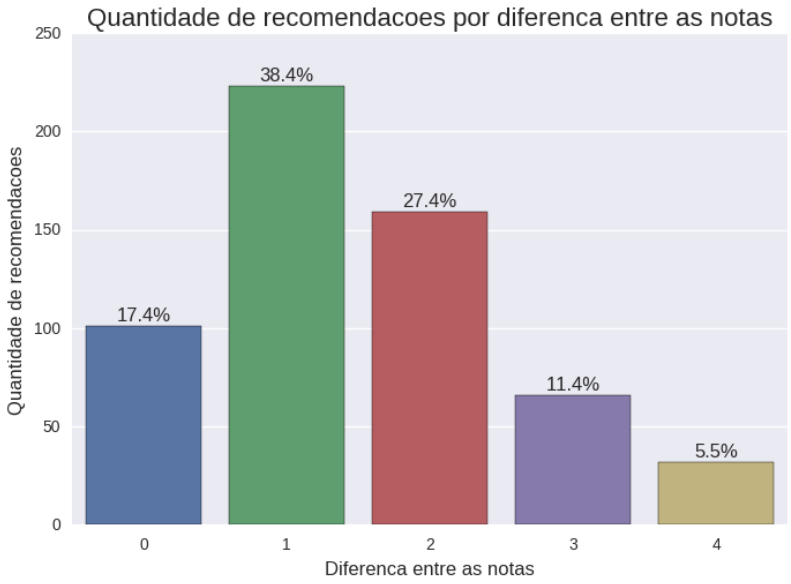
***Fonte**: elaborado pelo autor*

4.5 Recomendações

Conforme descrito na seção 4.2.1, o sistema atribui uma nota para cada recomendação em função de sua similaridade. Como o usuário também atribui uma nota para cada uma das recomendações, a avaliação da taxa de acerto das notas atribuídas pelo sistema em relação às notas atribuídas pelos usuários é uma das formas escolhidas para avaliar a efetividade das técnicas propostas para o processamento das recomendações: o sistema previu corretamente 17,4% das notas. A tabela 4.2 e a figura 4.3 descrevem este resultado e exibem também os percentuais dos erros em módulo acima de 1.

Como descrito na seção 4.3, a métrica RMSE tem grande aplicação na avaliação de sistemas que tem como objetivo prever a nota do usuário, portanto, será uma métrica considerada para a avaliação deste experimento. O cálculo do RMSE para as 581 avaliações feitas pelos usuários resultou em $1,22$, ou seja o sistema prevê notas que desviam em média $|1,22|$ das notas atribuídas pelos usuários. O resultado obito por Prando (2016) foi um RMSE de $1,71$.

Figura 4.3: *Quantidade de recomendações por diferença entre as notas sistema x usuário*



***Fonte**: elaborado pelo autor*

4.6 Análise de sentimentos

Foram processadas 1.758 publicações obtidas através das redes sociais Facebook e Twitter para obter as recomendações dos usuários que finalizaram todo o processo de avaliação. Cada uma destas publicações foi classificada quanto ao sentimento, conforme o novo fluxo para o processo de recomendação descrito na seção 3.4.2. Deste total, 383 foram descartadas do processo pois o classificador de sentimentos SVM (descrito na seção 3.4.1) as classificou como negativas. A figura 4.4 ilustra a quantidade de publicações consideradas positivas e negativas no processo.

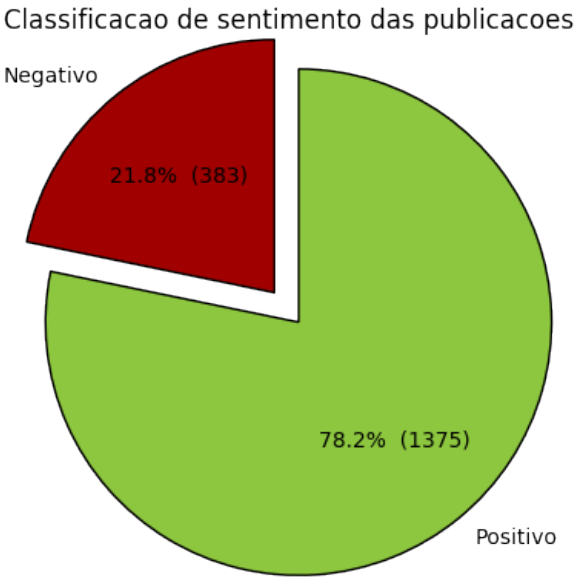
Considerando que a acuracidade do classificador de sentimento SVM, conforme descrito na seção 3.4.1, é de $80,26\%$, aproximadamente 307 publicações foram desconsideradas de forma correta e 76 foram falsos positivos em relação ao sentimento contido na publicação ser negativo, portanto, os falsos positivos são em média 1 por usuário. Como cada usuário participante que finalizou todo o processo tem em média 39 publicações, a desconsideração de 1 falso positivo não prejudicou a geração das recomendações e evitou que fossem geradas sugestões que estariam menos aderentes às suas necessidades.

4.7 Conclusão

Este capítulo apresentou a análise dos resultados obtidos através do experimento descrito no capítulo 3, bem como a formalização da principal métrica utilizada para calcular o erro apresentado.

Foram explorados inicialmente os resultados mais gerais, relativos a utilização do expe-

Figura 4.4: *Classificação de sentimento das publicações*



***Fonte**: elaborado pelo autor*

rimento, publicações obtidas nas redes sociais e as similaridades obtidas no processo. Na sequência foram discutidos os principais resultados observados na recomendação dos produtos e destacou-se que o RMSE calculado foi de $1,22$. O sistema apresenta assertividade de 17% em relação à nota atribuída pelo usuário.

Acredita-se que a proposta apresentada para amenizar o problema de *cold-start* mostrou-se aderente à necessidade e a utilização das técnicas de Análise de Sentimentos mostrou-se de grande importância para melhorar os resultados obtidos.

Capítulo 5

Conclusões

5.1 Resumo

Neste trabalho inicialmente são explorados os conceitos, classificações, técnicas e abordagens sobre Sistemas de Recomendação e a formalização do processo descrito por Adomavicius e Tuzhilin (2005). É então contextualizado o problema presente na literatura chamado de *cold-start*, que é um ofensor para os processos de recomendação. São abordadas as técnicas de aprendizado de máquina, mineração de textos e análise de sentimento e, após isto, são exploradas e formalizadas as técnicas que serão utilizadas para a proposta de resolução do problema.

Na sequência é descrita a proposta feita por Prando (2016) para amezinar o *cold-start* utilizando dados obtidos através das redes sociais Facebook e Twitter, o que caracteriza uma recomendação baseada em conteúdo [35]. Prando (2016) ressalta que a iniciativa baseada em recomendação utilizando redes sociais é recente, com poucos experimentos e pesquisas, principalmente para o contexto de *cold-start*. É evidenciado então o problema que pode ser gerado na proposta feita por Prando (2016) devido ao sentimento contido nas publicações dos usuários.

O trabalho traz então uma proposta de solução para o problema gerado por publicações com o sentimento negativo, descrevendo a abordagem e explorando os aspectos relacionados ao processo de recomendação. É exposto o novo fluxo proposto para processo, evidenciando as diferenças em relação ao fluxo anteriormente proposto por Prando (2016).

Para verificar a efetividade do que foi proposto por este trabalho, um experimento *online* que simula a interação entre um usuário e um E-Commerce no caso de *cold-start* foi elaborado e aplicado, obtendo-se a resposta de 64 pessoas. Este experimento, utilizando dados obtidos através das redes sociais Facebook e/ou Twitter dos usuários, utilizou o fluxo proposto para gerar recomendações que foram avaliadas pelos usuários.

São discutidos na sequência os resultados obtidos pelo experimento aplicado. Como em aplicações que tem como objetivo prever a nota que um determinado usuário daria para um item recomendado, uma métrica muito utilizada para avaliar a acuracidade das notas

previstas é o RMSE [41] esta foi a métrica utilizada para a avaliação. O RMSE obtido neste trabalho foi $1,22$.

Outra forma utilizada para avaliar a efetividade do SR prosposto foi a comparação entre a nota calculada pelo sistema e a nota atribuída pelo usuário. Isso mostrou que o sistema apresenta assertividade de 17%, apesar de ser um valor aparentemente baixo, deve-se considerar que foram empregados apenas textos extraídos de *posts* e *tweets* para a comparação com produtos. Assim, considera-se que o SR proposto apresenta razoável assertividade e para um cenário de *cold-start* se apresenta como uma boa técnica e que pode ser aperfeiçoada com o emprego de outros dados.

Uma terceira forma de avaliar o resultado do experimento seria comparar as recomendações feitas através do fluxo proposto como as recomendações de uma quantidade determinada de produtos mais vendidos do E-Commerce previamente selecionados. Entretanto, para que isto fosse possível, existe a necessidade da utilização de dados contidos em um E-Commerce real e portanto não possível para o escopo deste trabalho.

5.2 Contribuições

Este trabalho trouxe como contribuição:

- Uma abordagem funcional de técnicas de análise de sentimentos para o contexto de Sistemas de Recomendações em situações de *cold-start*;
- Analisou-se, utilizando dados reais, a acurácia de classificadores Naive Bayes para as categorias de produtos ($93,11\%$) e SVM para a classificação dos sentimentos das publicações ($80,26\%$);
- A disponibilização de forma *open source* do código fonte utilizado para o processamento das recomendações e treinamento dos classificadores necessários no experimento utilizado para a avaliação dos resultados;
- Reforçou a importância da utilização de dados obtidos nas redes sociais (não estruturados) para a resolução de problemas decorrentes da escassez de dados (como *cold-start*).

Com emprego de análise de sentimentos e os resultados aqui obtidos, acredita-se que este trabalho, que complementa a proposta de Prando (2016), pode contribuir para a solução do problema de *cold-start*. Além disto, acredita-se que as aplicações das técnicas de Análise de Sentimentos aqui utilizadas também contribuem com a área de Processamento Natural de Linguagem, pois evidenciam a sua utilização com bons resultados na área de Sistemas de Recomendação.

5.3 Pesquisas futuras

Como pesquisas futuras e continuidade da proposta aqui apresentada, sugere-se:

- Avaliação e utilização de técnicas de Aprendizado Profundo (*Deep Learning*) para a construção dos classificadores de categorias e de sentimentos;
- Na escassez de publicações em redes sociais, utilizar dados demográficos do usuário para a aplicação de técnicas de filtragem colaborativa;
- Utilização da rede de conexões presente nas redes sociais para a obtenção das recomendações personalidas com técnicas de filtragem colaborativa;
- Além de textos publicados, utilizar outras informações existentes nas redes sociais como "curtidas", "compartilhamentos" e "*retweets*" dos usuários para captar suas prefências;
- Utilizar outros tipos de mídias (como Fotos, Imagens e Vídeos) para identificar as prefências dos usuários, o que possibilitaria explorar outras redes sociais cujo conteúdo predominante são com estes conteúdos (por exemplo Instagram, Pinterest, Snapchat);

Para mensurar e avaliar os resultados, sugere-se:

- Utilizar outras métricas para a avaliação dos resultados que permitam mensurar a qualidade da recomendação e também o posicionamento na página de cada recomendação feita, como por exemplo *Mean Reciprocal Rank* (MRR);
- Para avaliar a efetividade da Análise de Sentimentos no processo, pode-se dentro da lista de produtos recomendados para o usuário gerar uma determinada quantidade itens utilizando o fluxo com a classificação do sentimento e outra determinada quantidade de itens sem a utilização, permitindo que seja analisado o efeito que se obteve pela aplicação da classificação de sentimento;
- Com o objetivo de comparar a proposta deste trabalho com uma abordagem que faça a recomendação dos itens mais vendidos do E-Commerce, pode-se mesclar sugestões obtidas de ambas as formas e analisar isoladamente os resultados obtidos;

Apêndice A

Experimento

Figura A.1: *Início do experimento*

Figura A.2: *Concessão de acesso às redes sociais*

Figura A.3: *Recomendações geradas pelo SR proposto*

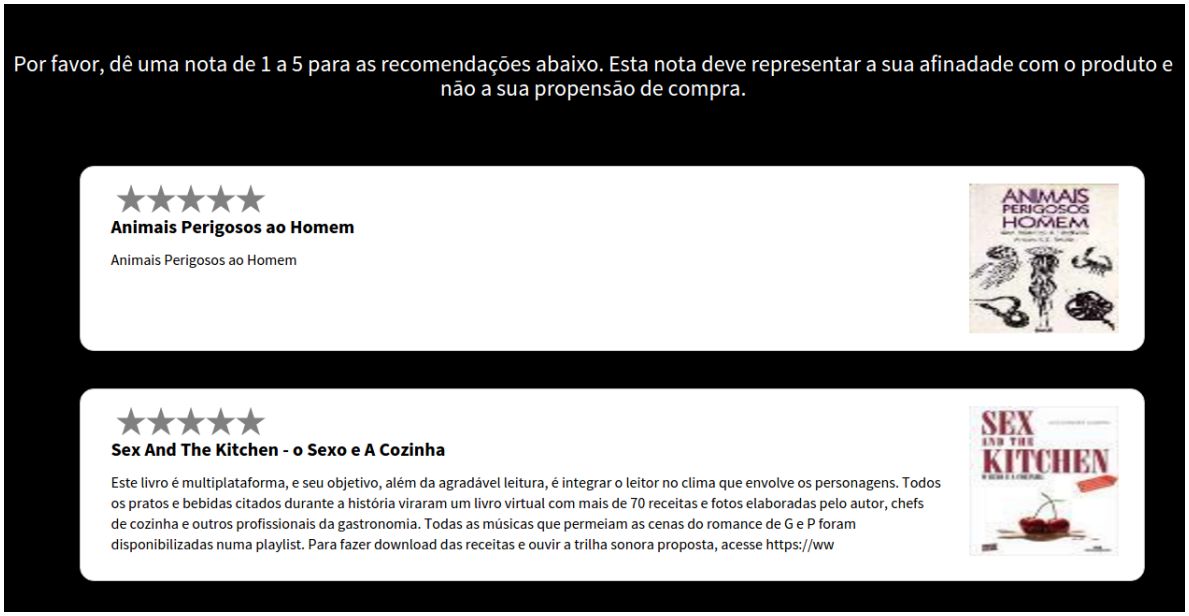
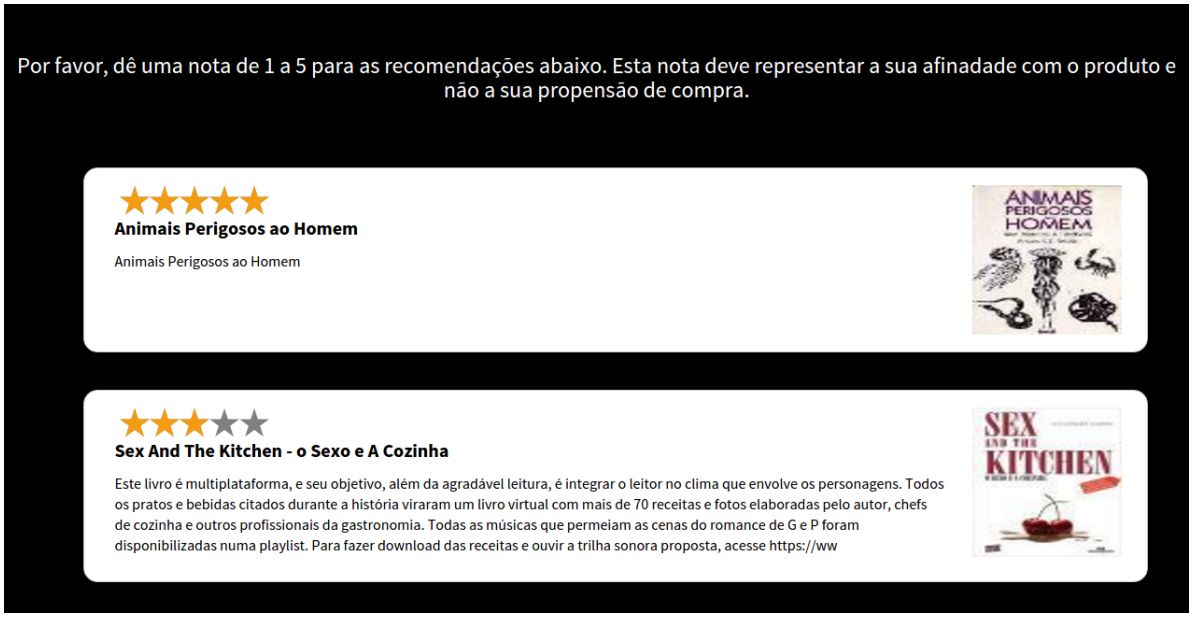


Figura A.4: *Notas atribuídas pelo usuário*



Apêndice B

Códigos-fonte do processo de recomendação

Todo o código-fonte utilizado no experimento está disponível de forma *open source* nos seguintes endereços:

- **Algoritmos de treinamento e recomendação**: https://github.com/felipecontra3/recsys-tcc-ml
- ***Web Crawler* para a obtenção dos produtos**: https://github.com/felipecontra3/recsys-tcc-crawler
- **Página web**: https://github.com/felipecontra3/recsys-tcc-web

Abaixo estão destacados os códigos-fontes utilizados para o treinamento dos algortimos de AM e do processo de recomendação:

Treinamento dos algortimos de AM

```
1  //train_classifier.py
2  import sys, os, math, datetime, shutil, re, unicodedata
3  from timeit import default_timer as timer
4  from nltk.tag import pos_tag
5  from nltk import word_tokenize
6  from nltk.stem.porter import *
7  from nltk.stem import RSLPStemmer
8  from nltk.corpus import stopwords
9  from pymongo import MongoClient
10 from openpyxl import load_workbook
11 from pyspark import SparkContext
12 from pyspark.ml.feature import HashingTF, IDF
13 from pyspark.mllib.regression import LabeledPoint
14 from pyspark.mllib.linalg import SparseVector
15 from pyspark.mllib.classification import SVMWithSGD, SVMModel
```

```
16 from pyspark.sql import Row, SQLContext
17 from pyspark.mllib.classification import NaiveBayes, NaiveBayesModel
18
19 def createMongoDBConnection(host, port, username, password, db):
20   client = MongoClient(host, port)
21   return client[db]
22
23 def removeAccents(s):
24  s = ''.join((c for c in unicodedata.normalize('NFD', s) if
       unicodedata.category(c) != 'Mn'))
25  return re.sub(r'[^\w]', ' ', s)
26
27 def findProductsByCategory(categories):
28    db = createMongoDBConnection(host, port, username, password,
         database)
29    produtos = db.produto_novo
30
31    product_list = []
32    query_filter = {}
33    if categories:
34     query_filter = {"categorias" : {"$in" : categories}}
35
36    for produto in produtos.find(query_filter):
37       keys = produto.keys()
38       description = ''
39       if 'descricaoLonga' in keys:
40          description = removeAccents(description +
              produto['descricaoLonga'])
41       if 'nome' in keys:
42          description = removeAccents(description + produto ['nome'])
43       id = None
44       if '_id' in keys:
45          id = str(produto['_id'])
46
47       category = ''
48       subcategory = ''
49
50       if 'categorias' in keys:
51          categorias_0 = removeAccents(produto['categorias'][0])
52
53          if categorias_0 == "Inicio":
54             category = removeAccents(produto['categorias'][1])
```

```
55              if(len(produto['categorias']) > 2):
56                  subcategory = removeAccents(produto['categorias'][2])
57          else:
58              category = categorias_0
59              if(len(produto['categorias']) > 1):
60                  subcategory = removeAccents(produto['categorias'][1])
61
62      if len(description)>0:
63          product_list.append((id, description, category, subcategory))
64
65  return product_list
66
67 def insertTokensAndCategories(tokens, category,
      categoryAndSubcategory):
68  db = createMongoDBConnection(host, port, username, password,
        database)
69
70  modelCollection = db.model
71  modelCollection.remove({'_type':'token'})
72
73  document_mongo = dict()
74  document_mongo['_type'] = 'token'
75  document_mongo['_datetime'] = datetime.datetime.utcnow()
76  i = 0
77  for t in tokens:
78      document_mongo[t] = i
79      i = i + 1
80
81  modelCollection.insert_one(document_mongo)
82
83  modelCollection.remove({'_type':'category'})
84
85  document_mongo = dict()
86  document_mongo['_type'] = 'category'
87  document_mongo['_datetime'] = datetime.datetime.utcnow()
88  i = 0
89  for c in category:
90      document_mongo[c] = i
91      i = i + 1
92
93  modelCollection.insert_one(document_mongo)
94
```

```
95      modelCollection.remove({'_type':'category and subcategory'})
96
97      document_mongo = dict()
98      document_mongo['_type'] = 'category and subcategory'
99      document_mongo['_datetime'] = datetime.datetime.utcnow()
100     i = 0
101     for c in categoryAndSubcategory:
102        document_mongo[c[0]+","+c[1]] = i
103        i = i + 1
104
105     modelCollection.insert_one(document_mongo)
106
107
108 def main(sc, sqlContext):
109     start = timer()
110
111     stpwrds = stopwords.words('english')
112     tbl_translate = dict.fromkeys(i for i in xrange(sys.maxunicode) if
           unicodedata.category(unichr(i)).startswith('S') or
           unicodedata.category(unichr(i)).startswith('P') or
           unicodedata.category(unichr(i)).startswith('N'))
113
114     print '---Pegando produtos---'
115     start_i = timer()
116     productRDD = sc.parallelize(findProductsByCategory([]))
117     print '####levou %d segundos' % (timer()-start_i)
118
119     print '---Criando corpus---'
120     start_i = timer()
121     corpusRDD = (productRDD.map(lambda s: (s[0],
           word_tokenize(s[1].translate(tbl_translate).lower()), s[2],
           s[3]))
122                     .map(lambda s: (s[0], [PorterStemmer().stem(x) for
                           x in s[1] if x not in stpwrds], s[2], s[3] ))
123                     .map(lambda s: (s[0], [x[0] for x in pos_tag(s[1])
                           if x[1] == 'NN' or x[1] == 'NNP'], s[2], s[3]))
124                     .cache())
125     print '####levou %d segundos' % (timer()-start_i)
126
127     print '---Pegando e persistindo dados de categoria e tokens---'
128     start_i = timer()
129     tokens = corpusRDD.flatMap(lambda x: x[1]).distinct().collect()
```

```
130     numTokens = len(tokens)
131     category = productRDD.map(lambda x: x[2]).distinct().collect()
132     categoryAndSubcategory = productRDD.map(lambda x: (x[2],
            x[3])).distinct().collect()
133     insertTokensAndCategories(tokens, category, categoryAndSubcategory)
134     print '####levou %d segundos' % (timer()-start_i)
135
136     print '---Calculando TF-IDF dos produtos---'
137     start_i = timer()
138     wordsData = corpusRDD.map(lambda s: Row(label=s[0], words=s[1],
            category=s[2], subcategory=s[3]))
139     wordsDataDF = sqlContext.createDataFrame(wordsData)
140     wordsDataForPrediction = corpusRDD.map(lambda s: Row(label=s[0],
            words=s[1], type=s[2]))
141     wordsDataForPredictionDF =
            sqlContext.createDataFrame(wordsDataForPrediction)
142
143     parquet_dir = "recsys-tcc-ml/parquet/wordsDataDF.parquet"
144     if os.path.exists(parquet_dir):
145         shutil.rmtree(parquet_dir)
146
147     wordsDataForPredictionDF.write.parquet(parquet_dir)
148
149     hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures",
            numFeatures=numTokens)
150     idf = IDF(inputCol="rawFeatures", outputCol="features")
151
152     featurizedData = hashingTF.transform(wordsDataDF)
153     idfModel = idf.fit(featurizedData)
154     rescaledData = idfModel.transform(featurizedData)
155
156     VSM = rescaledData.map(lambda t:
            LabeledPoint(category.index(t.category), t.features))
157
158     VSMTrain, VSMTest = VSM.randomSplit([8, 2], seed=0L)
159     print '####levou %d segundos' % (timer()-start_i)
160
161
162     print '--Criando modelo Naive Bayes---'
163     start_i = timer()
164     model = NaiveBayes.train(VSMTrain)
165
```

```
166    nb_dir = "recsys-tcc-ml/models/naivebayes/modelo_categoria"
167    if os.path.exists(nb_dir):
168       shutil.rmtree(nb_dir)
169
170    model.save(sc, nb_dir)
171    print '####levou %d segundos' % (timer()-start_i)
172
173    print '---Testando modelo Naive Bayes---'
174    start_i = timer()
175    prediction = VSMTest.map(lambda p :
          (categoryAndSubcategory[int(model.predict(p.features))],
          categoryAndSubcategory[int(p.label)]))
176    acuraccy = float(prediction.filter(lambda (x, v):
          x[0]==v[0]).count())/float(prediction.count())
177    print 'acuracidade de %f' % acuraccy
178    print '####levou %d segundos' % (timer()-start_i)
179
180    print '---Pegando os posts---'
181
182    start_i = timer()
183    posts = list()
184    wb = load_workbook(filename = 'recsys-tcc-ml/base_sentimentos.xlsx')
185    sheet = wb['Menes']
186    for row in sheet.iter_rows(row_offset=1):
187       post = list()
188       for cell in row:
189          if cell.value is None:
190             break
191          post.append(1 if cell.value == 'Positive' or cell.value ==
             'Neutral' else 0 if cell.value == 'Negative' else
             removeAccents(cell.value))
192
193       if len(post) > 0:
194          posts.append(tuple(post))
195
196    print '####levou %d segundos' % (timer()-start_i)
197
198    print '---Criando corpus---'
199    start_i = timer()
200    postsRDD = sc.parallelize(posts)
201    postCorpusRDD = (postsRDD.map(lambda s: (s[1],
          word_tokenize(s[0].translate(tbl_translate).lower())))
```

```
202                     .map(lambda s: (s[0], [PorterStemmer().stem(x) for
                           x in s[1] if x not in stpwrds]))
203                     .map(lambda s: (s[0], [x[0] for x in pos_tag(s[1])
                           if x[1] == 'NN' or x[1] == 'NNP']))
204                     .cache())
205
206 print '####levou %d segundos' % (timer()-start_i)
207
208 print '---Calculando TF-IDF dos Posts---'
209 start_i = timer()
210 wordsData = postCorpusRDD.map(lambda s: Row(label=s[0], words=s[1]))
211 wordsDataDF = sqlContext.createDataFrame(wordsData)
212
213 hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures",
       numFeatures=numTokens)
214 idf = IDF(inputCol="rawFeatures", outputCol="features")
215
216 featurizedData = hashingTF.transform(wordsDataDF)
217 idfModel = idf.fit(featurizedData)
218 rescaledData = idfModel.transform(featurizedData)
219
220 VSM = rescaledData.map(lambda t: LabeledPoint(t.label, t.features))
221 VSMTrain, VSMTest = VSM.randomSplit([8, 2], seed=0L)
222 print '####levou %d segundos' % (timer()-start_i)
223
224 print '--Criando modelo SVM---'
225 start_i = timer()
226 model = SVMWithSGD.train(VSMTrain, iterations=100)
227
228 svm_dir = "/home/ubuntu/recsys-tcc-ml/models/svm"
229 if os.path.exists(svm_dir):
230     shutil.rmtree(svm_dir)
231
232 model.save(sc, svm_dir)
233
234 print '---Testando modelo SVM---'
235 start_i = timer()
236 prediction = VSMTest.map(lambda p: (p.label,
       model.predict(p.features)))
237 acuraccy = prediction.filter(lambda (v, p): v != p).count() /
       float(prediction.count())
238
```

```
239     print 'acuracidade de %f' % acuraccy
240
241     print '####levou %d segundos' % (timer()-start_i)
242
243     print 'O processo todo levou %d segundos' % (timer()-start)
244
245
246 if __name__ == "__main__":
247     host = 'localhost'
248     port = 27017
249     username = ''
250     password = ''
251     database = 'tcc-recsys-mongo'
252     APP_NAME = 'Recomender System - Treinamento dos Modelos'
253     sc = SparkContext(appName=APP_NAME)
254     sqlContext = SQLContext(sc)
255     main(sc, sqlContext)
256     sc.stop()
```

Processamento da recomendação

```
1  //make_prediction.py
2  import sys, os, math, re, unicodedata
3  from timeit import default_timer as timer
4  from nltk.tag import pos_tag
5  from nltk import word_tokenize
6  from nltk.stem.porter import *
7  from nltk.stem import RSLPStemmer
8  from nltk.corpus import stopwords
9  from pymongo import MongoClient
10 from bson.objectid import ObjectId
11 from pyspark import SparkConf, SparkContext
12 from pyspark.ml.feature import HashingTF, IDF
13 from pyspark.mllib.regression import LabeledPoint
14 from pyspark.mllib.linalg import SparseVector
15 from pyspark.sql import Row, SQLContext
16 from pyspark.sql.functions import col
17 from pyspark.mllib.classification import NaiveBayes, NaiveBayesModel
18 from pyspark.mllib.classification import SVMWithSGD, SVMModel
19
20 def removeAccents(s):
21   s = ''.join((c for c in unicodedata.normalize('NFD', s) if
       unicodedata.category(c) != 'Mn'))
```

```
22   return re.sub(r'[^\w]', ' ', s)
23
24 def createMongoDBConnection(host, port, username, password, db):
25    client = MongoClient(host, port)
26    return client[db]
27
28 def findUserById(userId):
29    db = createMongoDBConnection(host, port, username, password,
         database)
30    return db.users.find_one({'_id': ObjectId(userId)})
31
32 def findPosts(user):
33    posts = []
34
35    if user['facebook']['posts'] is not None:
36       for post in user['facebook']['posts']:
37          if 'message' in post.keys():
38             posts.append((post['id_post'],
                removeAccents(post['message']), u'Post', u'Facebook'))
39
40    if user['twitter'] is not None:
41       for post in user['twitter']:
42          if 'text' in post.keys():
43             posts.append((post['id'], removeAccents(post['text']),
                u'Post', u'Twitter'))
44
45    return posts
46
47
48 def findProductById(prodId):
49    db = createMongoDBConnection(host, port, username, password,
         database)
50    prod = db.produto_novo.find_one({'_id': ObjectId(prodId)})
51    prod['idprod'] = str(prod['_id'])
52    return prod
53
54 def updateUser(user):
55    db = createMongoDBConnection(host, port, username, password,
         database)
56    return db.usuario.save(user)
57
58 def getTokensAndCategories():
```

```
59      db = createMongoDBConnection(host, port, username, password,
            database)
60      model = db.model
61
62      tokens_dict = db.model.find({"_type": "token"}).limit(1).next()
63      del tokens_dict['_type']
64      del tokens_dict['_id']
65      del tokens_dict['_datetime']
66      tokens_list = [None] * (max(tokens_dict.values()) + 1)
67
68      for key, value in tokens_dict.iteritems():
69          tokens_list[value] = key
70
71      categories_dict = db.model.find({"_type":
            "category"}).limit(1).next()
72      del categories_dict['_type']
73      del categories_dict['_id']
74      del categories_dict['_datetime']
75      categories_list = [None] * (max(categories_dict.values()) + 1)
76
77      for key, value in categories_dict.iteritems():
78          categories_list[value] = key
79
80      categories_and_subcategories_dict = db.model.find({"_type":
            "category and subcategory"}).limit(1).next()
81      del categories_and_subcategories_dict['_type']
82      del categories_and_subcategories_dict['_id']
83      del categories_and_subcategories_dict['_datetime']
84      categories_and_subcategories_list = [None] *
            (max(categories_and_subcategories_dict.values()) + 1)
85
86      for key, value in categories_and_subcategories_dict.iteritems():
87          pre_string = key.split(",")
88          categories_and_subcategories_list[value] = (pre_string[0],
                pre_string[1])
89
90      return tokens_list, categories_list,
            categories_and_subcategories_list
91
92  def insertSuggestions(suggestions, iduser, posts):
93      recomendations = dict()
94      recomendations['recomendacoes'] = []
```

```
95      for post in suggestions:
96
97          suggestions_dict = dict()
98
99          suggestions_dict['postId'] = post[0][0]
100         suggestions_dict['products'] = []
101
102         for post_base in posts:
103             #isso nao esta funcionando, verificar o pq
104             if int(post_base[0]) == int(post[0][0]):
105                 suggestions_dict['post'] = post_base
106
107         for product in post:
108             if len(product) > 0:
109                 prod = findProductById(product[1])
110                 if len(prod) > 0:
111                     prod['cosineSimilarity'] = product[2]
112                     suggestions_dict['products'].append(prod)
113
114         recomendations['recomendacoes'].append(suggestions_dict)
115
116     db = createMongoDBConnection(host, port, username, password,
            database)
117     db.users.update_one({"_id": ObjectId(iduser)}, {"$set" :
            recomendations})
118
119     return True
120
121 def cossine(v1, v2):
122     if (v1.dot(v1)*v2.dot(v2)) != 0:
123         return v1.dot(v2)/(v1.dot(v1)*v2.dot(v2))
124     else:
125         return 0
126
127 def main(sc, sqlContext):
128     user = findUserById(iduser)
129     posts = findPosts(user)
130
131     tokens, category, categoryAndSubcategory = getTokensAndCategories()
132     postsRDD = (sc.parallelize(posts).map(lambda s: (s[0],
            word_tokenize(s[1].lower()), s[2], s[3]))
133                 .map(lambda p: (p[0], [x for x in p[1] if x in tokens]
```

```
                    ,p[2], p[3]))
134                .cache())
135    stpwrds = stopwords.words('portuguese')
136    corpusRDD = (postsRDD.map(lambda s: (s[0], [PorterStemmer().stem(x)
           for x in s[1] if x not in stpwrds], s[2], s[3]))
137                    .filter(lambda x: len(x[1]) >= 20 or (x[2] ==
                           u'Post' and len(x[1])>0))
138                    .cache())
139    wordsData = corpusRDD.map(lambda s: Row(label=int(s[0]),
           words=s[1], type=s[2], originalpost=""))
140    wordsDataDF = (sqlContext.createDataFrame(wordsData)
141           .unionAll(sqlContext.read
142                 .parquet("recsys-tcc-ml/parquet/wordsDataDF.parquet")))
143
144
145    numTokens = len(tokens)
146    hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures",
           numFeatures=numTokens)
147    idf = IDF(inputCol="rawFeatures", outputCol="features")
148
149    featurizedData = hashingTF.transform(wordsDataDF)
150
151    idfModel = idf.fit(featurizedData)
152    tfIDF = idfModel.transform(featurizedData).cache()
153
154    postTFIDF = (tfIDF
155                .filter(tfIDF.type==u'Post')
156                .cache())
157
158    NB = NaiveBayesModel.load(sc,
           'recsys-tcc-ml/models/naivebayes/modelo_categoria')
159    SVM = SVMModel.load(sc, "recsys-tcc-ml/models/svm")
160
161    predictions = (postTFIDF
162                    .map(lambda p: (NB.predict(p.features), p[0],
                           SVM.predict(p.features)))
163                    .filter(lambda p: p[2]==1)
164                    .map(lambda p: (p[0], p[1]))
165                    .groupByKey()
166                    .mapValues(list)
167                    .collect())
168    suggestions = []
```

```
169
170     for prediction in predictions:
171         category_to_use = category[int(prediction[0])]
172         tf = tfIDF.filter(tfIDF.type==category_to_use).cache()
173         for post in prediction[1]:
174             postVector = postTFIDF.filter(postTFIDF.label ==
                    post).map(lambda x: x.features).collect()[0]
175             sim = (tf
176                    .map(lambda x: (post, x.label, cossine(x.features,
                           postVector)))
177                    .filter(lambda x: x[2]>=threshold)
178                    .collect())
179             if len(sim) > 0:
180                 suggestions.append(sim)
181
182     if len(suggestions) > 0:
183         insertSuggestions(suggestions, iduser, posts)
184
185 if __name__ == '__main__':
186
187     APP_NAME = 'Recomender System - Calculo de recomendacao'
188     threshold = 0.0002
189     host = 'localhost'
190     port = 27017
191     username = ''
192     password = ''
193     database = 'tcc-recsys-mongo'
194     iduser = sys.argv[1]
195     sc = SparkContext(appName=APP_NAME)
196     sqlContext = SQLContext(sc)
197     main(sc, sqlContext)
198     sc.stop()
```

Referências Bibliográficas

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. 7, 18, 19

[2] Noora Albalooshi, Nikolaos Mavridis, and Nabeel Al-Qirim. A survey on social networks and organization development. In *Proceedings of the 2012 International Conference on Collaboration Technologies and Systems, CTS 2012*, pages 539–545, 2012. 1

[3] Xavier Amatriain. Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013. 26

[4] Alejandro Baldominos, Yago Saez, Esperanza Albacete, and Ignacio Marrero. An Efficient and Scalable Recommender System for the Smart Web. In *Innovations in Information Technology (IIT), 2015 11th International Conference on*, pages 296 – 301, 2015. 3, 5

[5] Richard Bellman. *An Introduction to Artificial Intelligence: Can Computers Think?* Boyd & Fraser, Michigan, ilustrada edition, 1978. 9

[6] Surbhi Bhatia, Manisha Sharma, and Komal Kumar Bhatia. Strategies for Mining Opinions: A Survey. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*, pages 262 – 266, 2015. 15, 30

[7] J Bobadilla, F Ortega, A Hernando, and A Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013. 2, 5, 6, 7, 8, 9, 23

[8] Eugene. Charniak and Drew V. McDermott. *Introduction to artificial intelligence*. Addison-Wesley, 1985. 9

[9] Francesco Colace, Massimo De Santo, and Luca Greco. A Probabilistic Approach to Tweets' Sentiment Classification. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 37–42. IEEE, sep 2013. 15

[10] Thai Thinh Dang, Trong Hai Duong, and Hong Son Nguyen. A Hybrid Framework for Enhancing Correlation to Solve Cold-Start Problem In Recommender Systems. In *Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 1–5, 2014. 2, 3, 5, 8

[11] Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2/3):103–130, 1997. 20

[12] Wei Fan and Albert Bifet. Mining Big Data : Current Status , and Forecast to the Future. *ACM SIGKDD Explorations Newsletter*, 14(2):1–5, 2013. 1

[13] Lokmanyathilak Govindan, Sankar Selvan, and Teng-sheng Moh. A Framework for Fast-Feedback Opinion Mining on Twitter Data Streams. In *Collaboration Technologies and Systems (CTS), 2015 International Conference on*, pages 314–318, 2015. 17

[14] Imene Gupellil and Kamel Boukhalfa. Social big data mining: A survey focused on opinion mining and sentiments analysis. In *12th International Symposium on Programming and Systems, ISPS 2015*, pages 132–141, 2015. 1, 15, 16, 17, 21, 30

[15] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining*. Elsevier, 2012. 13, 14, 15, 20, 21, 22

[16] John Haugeland. *Artificial intelligence : the very idea*. MIT Press, 1985. 9

[17] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A Brief Survey of Text Mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20:19–62, 2005. 17, 18, 19, 28

[18] Sarika Jain, Anjali Grover, Praveen Singh Thakur, and Sourabh Kumar Choudhary. Trends, problems and solutions of recommender system. In *International Conference on Computing, Communication & Automation*, pages 955–958, 2015. 2, 6, 8, 19

[19] F. Jamiy, A. Daif, M. Azouazi, and A. Marzak. The potential and challenges of Big data - Recommendation systems next level application. *IJCSI International Journal of Computer Science*, 11(5), 2015. 1, 2, 5, 6, 23

[20] Jonathan D. Nelson Jana Jarecki, Björn Meder. The Assumption of Class-Conditional Independence in Category Learning. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, pages 2650–2655, 2013. 20

[21] Daniel Jurafsky and James H Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. *Speech and Language Processing An Introduction to Natural Language Processing Computational Linguistics and Speech Recognition*, 21:0–934, 2009. 12, 13

[22] Robert V Krejcie and Daryle W Morgan. Determining Sample Size for Research Activities. *Education and Psychological Measurement*, 39:607–610, 1970. 35

[23] Ray Kurzweil. *The Age of Intelligent Machines*. The MIT Press, 1990. 9

[24] Wei Li and Bo Sun. An improved collaborative filtering recommendation algorithm incorporating opinions analysis. In *Proceedings - 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2015*, volume 2, pages 171–173, 2015. 13, 19

[25] Elon Lages Lima. *Álgebra linear*. IMPA, Rio de Janeiro, 8ed. edition, 2012. 19

[26] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003. 2, 5

[27] Bing Liu. *Sentiment Analysis and Opinion Mining*, volume 5. 2012. 15, 18, 30

[28] Richard K. Lomotey and Ralph Deters. Towards Knowledge Discovery in Big Data. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pages 181–191. IEEE, apr 2014. 2

[29] Yonghe Lu and Jianhua Chen. Public opinion analysis of microblog content. In *ICISA 2014 - 2014 5th International Conference on Information Science and Applications*, pages 1–5, 2014. 13, 19

[30] George F Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, volume 5th. 2005. 10, 11, 12, 13

[31] Christopher D Manning and Hinrich Schütze. Foundations of Natural Language Processing. *Reading*, page 678, 2000. 13, 17

[32] Nils J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, 1st edition, 1998. 9

[33] Peter Norvig and Stuart Russell. *Inteligência Artificial*. São Paulo, 3ed. edition, 2013. 9, 10, 11, 12, 13

[34] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence: A Logical Approach*. Oxford University Press, 1st edition, 1998. 9

[35] Alan Vidotti Prando. *Um Sistema de Recomendação para E-commerce Utilizando Redes Sociais para solução de cold-start*. PhD thesis, 2016. 3, 27, 43

[36] Alan Vidotti Prando and Solange Nice Alves De Souza. A Modular Architecture for Recommender Systems. 2016. 23

[37] Xueming Qian, He Feng, Guoshuai Zhao, Tao Mei, and Senior Member. Personalized Recommendation Combining User Interest and Social Circle. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1–14, 2014. 19

[38] P. Venkata Rajeev and V. Smrithi Rekha. Recommending products to customers using opinion mining of online product reviews and features. In *IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2015*, 2015. 13

[39] Elaine. Rich and Kevin. Knight. *Artificial intelligence*. McGraw-Hill, 1991. 9

[40] Rob van der Meulen. Gartner Says 6.4 Billion Connected "Things"Will Be in Use in 2016, Up 30 Percent From 2015, 2015. 1

[41] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, chapter 8, pages 257–298. 2011. 36, 44

[42] Herbert A. Simon. Why Should Machines Learn? In *Machine Learning*, pages 25–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. 11

[43] Sachchidanand Singh and Nirmala Singh. Big Data analytics. In *2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, pages 1–4, 2012. 2

[44] Vandana Singh and Sanjay Kumar Dubey. Opinion mining and analysis: A literature review. In *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, pages 232–239, 2014. 15, 16, 21, 30

[45] A M Turing. Computing Machinery and Intelligence. *Mind*, 49:433–460, 1950. 10, 12

[46] Jai Prakash Verma, Bankim Patel, and Atul Patel. Big data analysis: Recommendation system with hadoop framework. In *Proceedings - 2015 IEEE International Conference on Computational Intelligence and Communication Technology, CICT 2015*, pages 92–97, 2015. 17

[47] Xiaojun Wan. A Comparative Study of Cross-Lingual Sentiment Classification. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 24–31. IEEE, dec 2012. 16

[48] Patrick Henry. Winston. *Artificial intelligence*. Addison-Wesley Pub. Co, 1992. 9

[49] Yangsen Zhang, Zhenlei Du, and Hongxia Ma. Cross-language Sentiment Classification Based on Support Vector Machine. In *Natural Computation (ICNC), 2015 11th International Conference on*, pages 507–513, 2015. 19

[50] Yongzheng Zhang and Marco Pennacchiotti. Predicting purchase behaviors from social media. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1521–1532, 2013. 2, 3, 5, 26