UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUSTAVO BEVILACQUA

SISTEMA DE RECOMENDAÇÃO DE RECEITAS ALIMENTARES UTILIZANDO FILTRAGEM BASEADA EM CONTEÚDO

CHAPECÓ
2022

GUSTAVO BEVILACQUA

SISTEMA DE RECOMENDAÇÃO DE RECEITAS ALIMENTARES UTILIZANDO FILTRAGEM BASEADA EM CONTEÚDO

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Dr. Denio Duarte

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em: 9/4/2022.

BANCA AVALIADORA

Dr. Denio Duarte – UFFS

Dr. Guilherme Dal Bianco - UFFS

Ma. Andressa Sebben - UFFS

Sistema de Recomendação de Receitas Alimentares Utilizando Filtragem Baseada em Conteúdo

Gustavo Bevilacqua[1]

[1]Universidade Federal da Fronteira Sul - UFFS

gustavocbevilacqua@gmail.com

Abstract. *In recent years, due to the lack of time, the difficulty to maintain a healthy diet is increasing more and more. Because of this, coupled with the fact that the world is experiencing a pandemic moment, psychological problems such as anxiety and depression are becoming more common. In addition, the amount of food recipes available on the internet has been growing at an accelerated pace. Taking this into account, this work propose a study to develop a prototype of a recipe recommendation system that considers the ingredients and their quantities. In this way, the user will be able to find recipes faster and, with this, will be able to develop the act of cooking as a hobby. To achieve this, the prototype uses content-based filtering to generate the recommendations. The experimentation takes place with the system generating five recommendations based on the euclidian distance from the recipe choosen. The experiments have shown that the prototype generated satisfactory recipes for the users.*

Resumo. *Nos últimos anos, por conta da falta de tempo, a dificuldade para manter uma alimentação saudável está aumentando cada vez mais. Devido a isso, aliado ao fato de que o mundo está passando por um momento de pandemia, os problemas psicológicos como a ansiedade e a depressão estão se tornando mais comuns. Além disso, a quantidade de receitas alimentares disponíveis na internet vem crescendo de forma acelerada. Levando isso em conta, o presente trabalho propõe o desenvolvimento de um protótipo de um sistema de recomendação de receitas que considera os ingredientes e suas quantidades. Desta forma, o utilizador pode buscar receitas de forma mais rápida e, com isso, transformar o ato de cozinhar em um hobby. O protótipo utiliza a filtragem baseada em conteúdo para fazer a recomendação. A recomendação é feita por meio de cinco receitas com maior similaridade utilizando a distância euclidiana. Experimentos mostraram que o protótipo recomenda receitas satisfatórias aos usuários.*

1. Introdução

Os sistemas de recomendação consistem em gerar recomendações específicas para cada usuário. Essas recomendações podem ser geradas de forma implícita, por meio do monitoramento das atividades do usuário, ou explícita, como coletando as avaliações do usuário [Bobadilla et al. 2013].

O processamento das informações ocorre por meio de técnicas de *machine learning* e utilizam algoritmos específicos para manipular os dados com o objetivo de encontrar uma medida de similaridade entre dois pontos, que representam as informações obtidas. Além disso, os sistemas de recomendação são altamente utéis e por isso são utilizados em grandes plataformas presentes no dia-a-dia de todos nós, como em recomendações de amizades em redes sociais e recomendações de filmes e séries em aplicativos de *streaming*.

A filtragem baseada em conteúdo seleciona itens considerando o conteúdo de cada um e comparando ao conteúdo dos itens previamente avaliados pelo usuário [Van Meteren and Van Someren 2000]. Em comparação com a filtragem baseada em conteúdo, a filtragem colaborativa tem algumas vantagens em cenários onde cada item possui pouco conteúdo, ou conteúdos de difícil análise [Melville et al. 2002]. Neste caso, considerando o contexto do artigo atual, a filtragem baseada em conteúdo parece ser a mais apropriada.

Observando o crescente aumento no uso dos sistemas de recomendação no contexto atual, assim como a extração e o tratamento de dados em linguagem natural, o presente artigo reúne um conjunto de estudos e o desenvolvimento de um extrator de dados por meio de requisições *HTTP*, que posteriormente são tratados e convertidos para um padrão e então são utilizadas para o desenvolvimento do protótipo de um sistema de recomendação de receitas.

O intuito do desenvolvimento do trabalho em questão é o de responder a seguinte pergunta: É possível extrair e padronizar informações de receitas em linguagem natural para que posteriormente sejam utilizadas no desenvolvimento do protótipo de um sistema de recomendação de receitas alimentares com uma boa acurácia, utilizando *machine learning*?

O desenvolvimento do projeto possibilita a melhora na compreensão dos métodos de extração de dados, ou *web scraping*. Além disso, o tratamento e a padronização dos dados podem ser utilizados como um facilitador no desenvolvimento de projetos futuros, assim como o motivo que levou a escolha do algoritmo para verificar a similaridade entre cada item.

Por meio da experimentação e utilizando o conjunto de dados obtido por meio de técnicas de *web scraping*, foi observado que é possível gerar recomendações de receitas alimentares que atendam aos usuários. Porém, com um número maior de *features* e com um método mais avançado de padronização da quantidade de cada ingrediente, as recomendações geradas poderiam ser melhores.

A estrutura deste trabalho está organizada da seguinte maneira: Na seção 2 é apresentado o referencial teórico, contendo os conceitos básicos para o entendimento do presente trabalho. A seção 3 apresenta trabalhos semelhantes a este. Na seção 4 são

apresentados os experimentos realizados e as configurações escolhidas, e por fim a seção 5 traz a conclusão obtida com o desenvolvimento deste trabalho.

2. Referencial Teórico

Na presente seção, são apresentados os conceitos necessários para o entendimento do projeto desenvolvido neste artigo. Cada subseção faz referência aos principais assuntos ligados ao desenvolvimento da ferramenta proposta neste trabalho, contendo uma breve conceitualização com o intuito de facilitar a compreensão do leitor.

2.1. Web scraping

O *web scraping* é o processo de extrair dados da *web* de forma automática, apenas sendo necessário a programação referente a automação do *software*. Atualmente, é possível converter informações contidas em grandes *websites* para conjuntos de dados organizados, utilizando técnicas de *web scraping* [Zhao 2017]. Conforme cita Jarmul et al. (2017), o *web scraping* é uma técnica utilizada para coletar dados que não estão disponíveis em outros formatos.

Uma das técnicas de *web scraping*, que foi utilizada no presente trabalho, é a de utilizar um *parser HTML*, que extraí os dados das páginas de acordo com as tags *HTML*. Para isso, uma requisição é enviada ao *website* alvo e a resposta da requisição é processada de modo a extrair os dados relevantes para o usuário.

2.2. Aprendizado de Máquina

Os algoritmos de aprendizado de máquina são responsáveis por realizar tarefas desejadas de modo que produzam uma saída esperada sem que haja a necessidade de serem programados explicitamente. Estes algoritmos funcionam por meio de repetição, ou seja, o algoritmo é adaptável, de modo em que as saídas ficam mais precisas a cada repetição feita. Para que a adaptação do algoritmo ocorra, é preciso que o conjunto de dados de entrada seja divido em duas categorias, treino e teste [El Naqa and Murphy 2015].

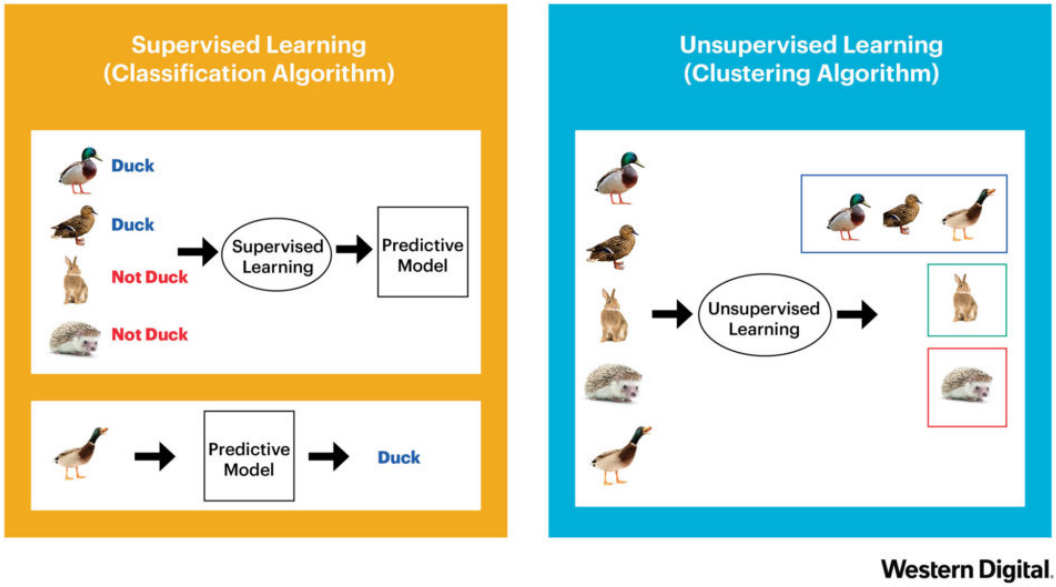
Existem dois tipos principais de algoritmos de aprendizado de máquina: aprendizado supervisionado e aprendizado não supervisionado. Conforme Ayodele (2010) explica, o algoritmo supervisionado realiza o mapeamento dos dados de entrada para uma classe de saída desejada e conhecida, enquanto que o algoritmo não supervisionado não conhece a saída como uma classe.

Conforme descrito anteriormente, os algoritmos de aprendizado supervisionado conhecem as classes de saída para as quais os dados de entrada devem ser classificados. A Figura 1 mostra a diferença entre algoritmos de aprendizado supervisionado e não supervisionado:

Observando a Figura 1, é possível notar que o algoritmo de aprendizado supervisionado, indicado no quadro da esquerda, classifica as imagens dos animais em classes, enquanto que o algoritmo não supervisionado, indicado no quadro da direita, apenas separa cada imagem em grupos.

Diferente do aprendizado supervisionado, o conjunto de entrada no aprendizado não supervisionado porém não possuí os rótulos. O aprendizado não supervisionado pode

Figura 1. Algoritmos de aprendizado de máquina.



Fonte: Zhou (2018)

ser visto como um modo de encontrar padrões em dados que seriam considerados apenas dados ruidosos [Ghahramani 2003].

Conforme explica Ghahramani (2003), apesar das entradas dos algoritmos de aprendizagem de máquina não possuírem rótulos para guiar o aprendizado, é possível criar representações da entrada criando grupos e, assim, prever em qual grupo um próxima entrada possa pertencer.

2.3. KMeans

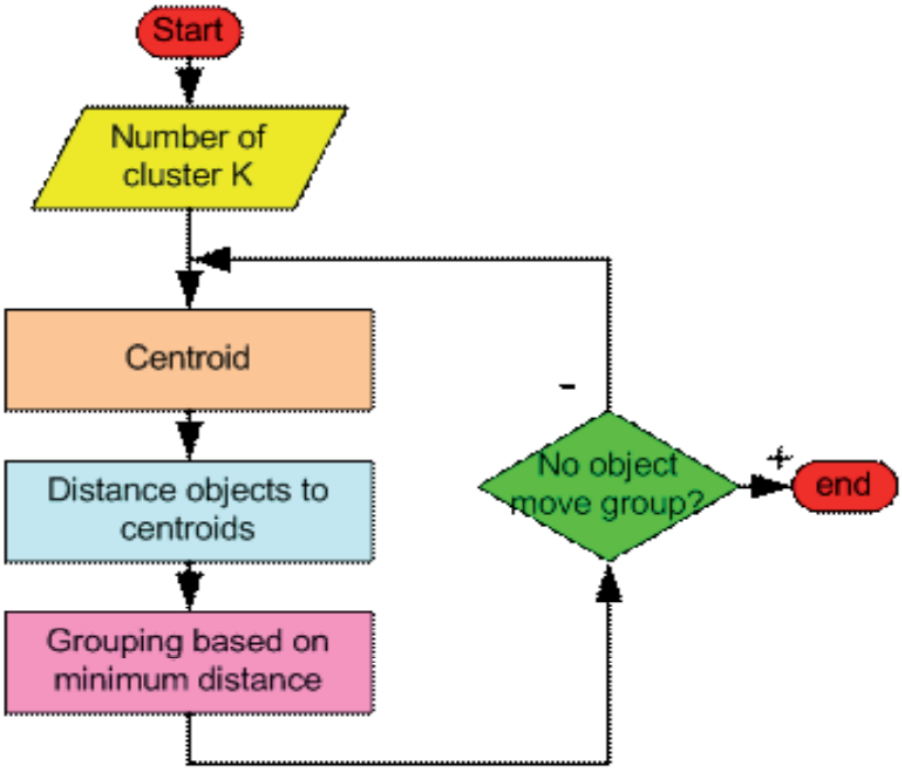
O algoritmo KMeans é um algoritmo de clusterização. De acordo com Flynn et al. (1999), a clusterização é o processo de classificação de padrões de forma não supervisionada em grupos, ou *clusters*. Conforme Ayodele (2010) explica, para a clusterização do KMeans, é preciso primeiramente determinar o número de *clusters* e o centro de cada um deles. Inicialmente é possível assumir o centro em pontos aleatórios contemplando um número k de pontos, e repetindo este processo até que todos os pontos estejam dentro de um *cluster*. Após isso, o algoritmo executará três passos até a conclusão do objetivo na seguinte sequência:

- Determinar o ponto central
- Determinar a distância de cada ponto em relação ao ponto central
- Agrupar os ponto conforme a distância do ponto central

A Figura 2 apresenta o passo a passo do funcionamento do algoritmo KMeans.

A Figura 2 descreve graficamente o modo como o algoritmo encontra o *cluster* de cada objeto. Começando pela definição do número de *clusters* e após isso, repetindo os três passos descritos anteriormente até chegar ao objetivo.

Figura 2. Passo a passo da clusterização por meio do KMeans.



Fonte: Ayodele (2010, p.28)

2.4. Sistemas de recomendação

Um sistema de recomendação tem o objetivo de prover ao usuário recomendações personalizadas [Ricci et al. 2011]. Para que isso seja possível em cenários variados, existem diferentes tipos de sistemas recomendadores, sendo os mais tradicionais: colaborativo, baseado em conteúdo, baseado em demografia, baseado em conhecimento e híbridos [Aggarwal 2016].

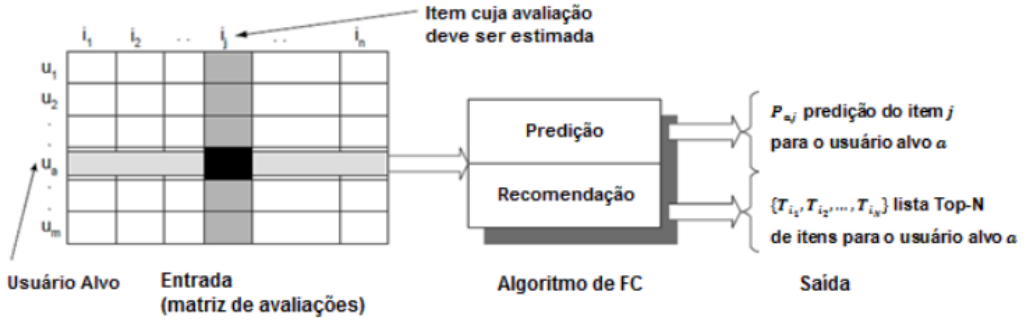
Os sistemas de recomendação são amplamente usados na indústria, pois, de acordo com as informações obtidas durante o uso do produto, é possível que as sugestões dadas pelo sistema tenham uma acurácia maior, assim aumentando o lucro e melhorando a relação com o cliente. Segundo Dias et al. (2008), a implantação de um sistema de recomendação no *checkout* e na navegação, onde o usuário visualiza e escolhe os produtos, de uma determinada loja virtual, geraram uma receita extra de, no mínimo 66% e, em média, 336%.

2.4.1. Filtragem colaborativa

Como o próprio nome sugere, um sistema de recomendação colaborativo utiliza as avaliações realizadas por múltiplos usuários, que possuem gostos similares com o usuário que receberá as recomendações [Aggarwal 2016]. Porém, em um sistema desse tipo, de-

pendendo do conteúdo, existirão muitas opções que não foram avaliadas, o que acaba gerando um esparso conjunto de dados [Sarwar et al. 2001]. A Figura 3 apresenta graficamente o processo que ocorre durante a filtragem colaborativa:

Figura 3. Processo de Filtragem Colaborativa.



Fonte: Medeiros (2013, p.10)

Na matriz de avaliações, exibida à esquerda na Figura 3, as linhas indicam os usuários e as colunas indicam os itens. Já a coluna em cinza, indica o item cuja avaliação será estimada para o usuário alvo.

Conforme Sarwar et al. (2001) explicam, a estimativa para um item pode ser feita de duas formas. Por predição ou por recomendação. A predição estima a avaliação do usuário alvo para determinado item. Já o algoritmo de recomendação, também conhecido como recomendação *Top-N*, seleciona os N itens com maior probabilidade de serem aceitos pelo usuário alvo. Ambos os tipos de algoritmos estão indicados na parte central da Figura 3.

Existem duas abordagens para a filtragem colaborativa. A primeira sendo baseada em modelo, que usa os dados da transação para criar um modelo capaz de gerar recomendações, e a segunda, baseada em memória, que acessa diretamente o banco de dados [Bobadilla et al. 2013]. A Tabela da Figura 4 apresenta um exemplo de Aggarwal (2016), cujas predições foram realizadas por meio da abordagem baseada em memória.

Figura 4. Similaridade entre o usuário alvo e os demais usuários.

Item-Id ⇒ User-Id ⇓	1	2	3	4	5	6	Mean Rating	Cosine$(i,3)$ (user-user)	Pearson$(i,3)$ (user-user)
1	7	6	7	4	5	4	5.5	0.956	0.894
2	6	7	?	4	3	4	4.8	0.981	0.939
3	?	3	3	1	1	?	2	1.0	1.0
4	1	2	2	3	3	4	2.5	0.789	-1.0
5	1	?	1	2	3	3	2	0.645	-0.817

Fonte: Aggarwal (2016, p.34)

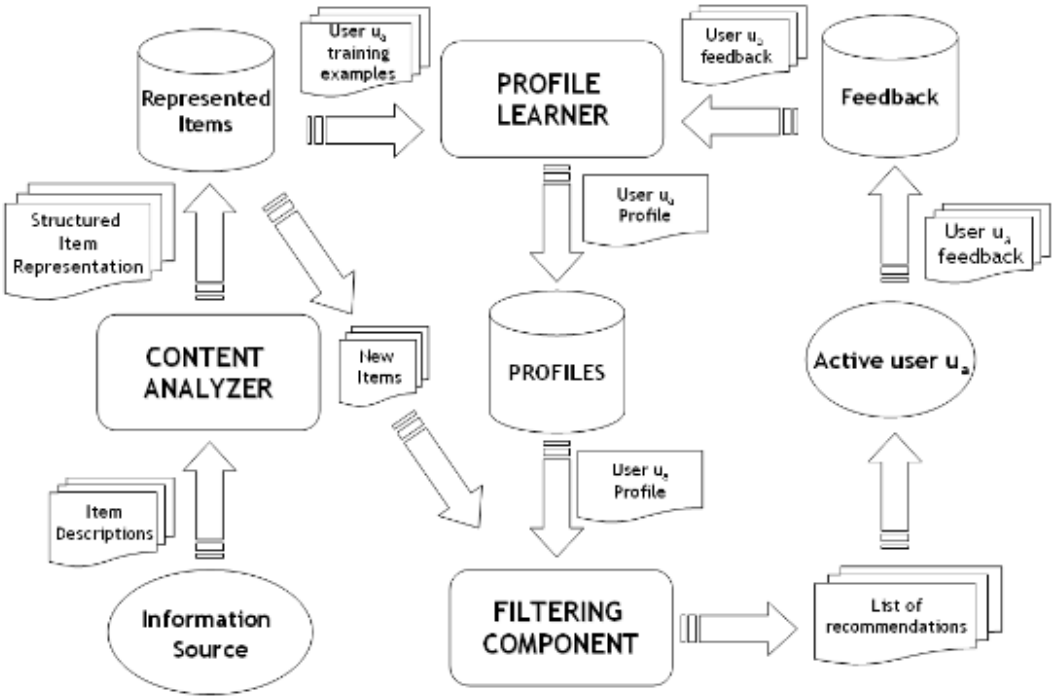
A Figura 4 mostra as avaliações referentes a seis itens, realizadas pelos usuários. Nas três últimas colunas, são mostradas a classificação média de cada usuário, as métricas

de similaridade entre o usuário 3 e o restante dos usuários, utilizando o algoritmo da similaridade por cosseno, que calcula a similaridade de dois vetores de acordo com a ângulo, na primeira coluna, e o algoritmo da similaridade de Pearson, que mede a similaridade entre duas variáveis contínuas, na segunda coluna [Aggarwal 2016].

2.4.2. Filtragem baseada em conteúdo

Na filtragem baseada em conteúdo, as recomendações são feitas de acordo com o conteúdo dos itens anteriormente avaliados positivamente ou negativamente pelo usuário [Aggarwal 2016]. É preciso que o conteúdo, em sua forma original, seja pré-processado de modo que, de acordo com os dados que foram extraídos e limpos, seja possível dizer com mais precisão do que aquele item se trata e, de acordo com o grupo de itens avaliados pelo usuário, cria-se um perfil personalizado [Aggarwal 2016].

Figura 5. Arquitetura de alto nível de um sistema de recomendação baseado em conteúdo.



Fonte: Ricci et al. (2011, p.76)

Conforme a Figura 5, o fluxo do sistema começa com a extração das informações, indicada na imagem como *Information Source*. Tais informações servem como entrada para a etapa da análise de conteúdo, descrita na imagem como *Content Analyzer*. O procedimento dessa etapa é o de extrair os atributos do texto não estruturado, analisar o conteúdo de cada item e gerar uma representação estruturada do mesmo, que será guardada no repositório *Represented Items* [Ricci et al. 2011].

A próxima etapa é a criação de um perfil único do usuário alvo, que ficará salvo no repositório *Profiles*. Esse processo necessita que as reações do usuário alvo sejam armazenadas em um repositório chamado de *Feedback*. Com o perfil criado, é realizada

uma filtragem nos itens não avaliados, e os que apresentam um maior grau de similaridade são recomendados. O *feedback* dos novos itens recomendados é repassado novamente pela etapa do aprendizado de perfil, para que as próximas recomendações tenham um grau de aceitação maior [Ricci et al. 2011].

Quando um novo item é analisado, o mesmo vai para o processo de filtragem, chamado na imagem de *Filtering Component*. O componente de filtragem irá realizar a predição para saber se o item seria de interesse do usuário alvo ou não. Esse processo normalmente ocorre comparando os atributos do item com os atributos encontrados no perfil do usuário, que fica armazenado no repositório *Profiles* [Ricci et al. 2011].

O *feedback* do usuário pode ser coletado implicitamente ou explicitamente. Quando o usuário clica em um item, o sistema pode considerar aquele item como positivo, já que atraiu o usuário, isso seria um modo de *feedback* implícito. Já o *feedback* explícito necessita que o usuário escolha a sua avaliação do item, podendo ser em uma escala númerica ou binária, como com as opções "gostei", que pode ser representada com 1, e "não gostei", que pode ser representada com 0 [Ricci et al. 2011].

2.4.3. Filtragem baseada em demografia

Um exemplo clássico de filtragem baseada em demografia são as recomendações de atrações turísticas quando se está viajando. Esse método de filtragem, quando utilizado de forma singular, tem uma acurácia limitada [Wang et al. 2012]. Devido às limitações, como o fato de as recomendações serem baseadas somente nas informações demográficas do usuário, seu uso acaba sendo muito mais específico.

Na filtragem baseada em demografia, o usuário é classificado de acordo com suas características demográficas e, com isso, torna-se possível fazer recomendações de serviços que se encontram próximos ao usuário alvo [Wang et al. 2012].

Uma das maiores vantagens desse método é a ausência do *cold start*. O *cold start* faz referência ao fato de que, quando um novo usuário, ou um novo item, é adicionado ao sistema, não existem avaliações disponíveis para que se possa realizar recomendações, ou, no caso do novo item, realizar a recomendação do mesmo [Natarajan et al. 2020].

Existem algumas desvantagens, principalmente relacionadas à privacidade e à segurança dos usuários. Outro ponto a ser considerado é o dilema da Estabilidade-Plasticidade, que refere-se a capacidade de um sistema aprender novas informações, porém, sem perder o conhecimento previamente obtido. Isso se deve ao fato de que existe a possibilidade do usuário mudar suas características demográficas com frequência. Devido a esse fato, o sistema pode acabar perdendo todo o conhecimento que foi obtido anteriormente [Fayyaz et al. 2020].

2.4.4. Filtragem baseada em conhecimento

Da mesma forma que a filtragem baseada em conteúdo, a filtragem baseada em conhecimento também utiliza os atributos dos itens para fazer as recomendações. Além disso, o usuário pode especificar explicitamente as características desejadas nos itens a

serem recomendados. Esse tipo de filtragem é mais utilizada quando os produtos alvo são pouco comprados e podem ter diferentes especificações, como casas e carros. Itens desse tipo costumam ter uma combinação de propriedades específicas, ou seja, devido ao seu domínio complexo, torna-se difícil de associar avaliações suficientes para o grande número de combinações possíveis [Aggarwal 2016].

Sistemas com esse modo de filtragem necessitam de interfaces para que o usuário possa selecionar as características desejadas. Os sistemas baseados em conhecimento podem ter suas recomendações baseadas em limitações, em que o usuário especifica os requisitos desejados. Por exemplo, o número mínimo de assentos em um carro, ou baseadas em casos, como um carro com três assentos e que o vendedor esteja na mesma cidade que o usuário [Aggarwal 2016].

A filtragem baseada em conteúdo é similar à filtragem baseada em conhecimento, pois ambas geram recomendações altamente baseadas nos atributos dos itens. A principal diferença entre elas é o fato de que, na filtragem baseada em conteúdo, o sistema aprende com o comportamento passado do usuário, enquanto que, na filtragem baseada em conhecimento, as recomendações são feitas pelas especificações escolhidas pelo usuário [Aggarwal 2016].

2.4.5. Filtragem híbrida

De acordo com Burke (2002), a combinação entre dois ou mais tipos de filtragem constituem um sistema de recomendação híbrido. Tais combinações são escolhidas pensando na minimização das desvantagens de cada modelo para que se obtenha uma melhor performance [Burke 2002]. Ainda, conforme Burke (2002) explica, os sistemas híbridos podem ser separados em 7 categorias, sendo elas: *Weighted*, *Switching*, *Cascade*, *Feature augmentation*, *Feature combination*, *Meta-level* e *Mixed*.

No modelo *weighted*, as predições de vários sistemas de recomendação são computadas considerando-as como variáveis em uma combinação linear [Suriati et al. 2017]. Já no método *switching*, a escolha do tipo de filtragem da recomendação pode mudar de acordo com a necessidade. Burke (2002) exemplifica que um sistema, durante a fase inicial, pode utilizar a filtragem baseada em conhecimento para evitar o problema do *cold start* e, após ter mais avaliações disponíveis, então a filtragem baseada em conteúdo ou a filtragem colaborativa poderiam ser utilizadas.

Burke (2002) define o método *cascade* como um modelo híbrido onde as recomendações feitas anteriormente servem como entrada para um próximo sistema de recomendação, de modo que a saída deste seja refinada em comparação com as anteriores.

O método *feature augmentation*, utiliza o resultado de outro sistema de recomendação como entrada, porém, diferente do método *cascade*, ele as considera como atributos para o próximo sistema [Burke 2002]. Um exemplo de utilização deste método é o sistema *LIBRA* [Mooney and Roy 2000], onde as recomendações de títulos e autores relacionados, geradas pelo site *Amazon.com*, são utilizadas como entrada.

Feature combination é um método onde o objetivo é o de que, antes de aplicar um algoritmo preditivo, é preciso combinar a entrada de várias fontes de dados e for-

mar uma única representação. Normalmente, tais algoritmos preditivos são algoritmos baseados em conhecimento juntamente com informações colaborativas como atributos complementares [Aggarwal 2016].

No método *meta-level*, o modelo que foi aprendido por um sistema recomendador é utilizado como a entrada de outro sistema [Burke 2002]. Bem como Burke (2002) explica, a diferença entre este método e o método *feature combination* está no fato de que, o *meta-level* utiliza o modelo todo como entrada, enquanto que o *feature combination* gera, a partir do modelo aprendido, atributos que são utilizados como entrada para outro sistema.

No procedimento *mixed*, são apresentadas recomendações provenientes de diferentes técnicas de recomendação [Burke 2002]. Um exemplo seria o sistema proposto por Glauber (2013). Tal sistema visa recomendar nomes de uma determinada base de dados. As recomendações do sistema são resultantes de três tipos de filtragem, sendo elas: colaborativa, baseada em conteúdo e baseada em popularidade.

3. Trabalhos Relacionados

Nesta seção são apresentados dois estudos cujos temas se relacionam com o trabalho atual. Para cada um destes, é apresentada uma breve explicação sobre o problema e, além disso, são apresentadas as soluções, os resultados e a relação de cada um com o presente trabalho. O título dos estudos escolhidos será a identificação da subseção que faz referência a ele.

3.1. Recipe Recommendation: Accuracy and Reasoning

O estudo realizado por Freyne et al. (2011) visa analisar como os indivíduos pensam em relação aos alimentos e, em especial, a determinadas receitas. Além disso, também foram feitas análises para identificar a existência de padrões de pensamento em cada usuário.

Primeiramente, os autores focaram apenas em receitas para a identificação de padrões e excluíram os demais assuntos, como planejamento e agenda de refeições. Em seguida, foi utilizada uma coleção de receitas contendo o título, a lista de ingredientes e o modo de preparo. A complexidade da receita foi determinada de acordo com o número de ingredientes e a quantidade de passos para o preparo da mesma. Além disso, cada receita foi inserida manualmente em um dos três tipos definidos pelos autores: cozinha geral, cozinha específica e uma categoria ampla. A Figura 6 apresenta um exemplo dos tipos propostos.

Foram utilizados três algoritmos personalizados: filtragem colaborativa, filtragem baseada em conteúdo e M5P. A filtragem colaborativa realiza a predição de uma receita alvo para o usuário alvo, levando em conta os pesos das avaliações de um número n de vizinhos.

O segundo algoritmo utiliza filtragem baseada em conteúdo. As receitas anteriormente avaliadas têm seus ingredientes separados e, baseado na frequência de cada ingrediente em relação à coleção de receitas, é atribuída uma avaliação. A avaliação dada pelo usuário em cada receita é distribuída igualmente entre os ingredientes da receita e, com isso, é aplicado o algoritmo da filtragem baseada em conteúdo. Este algoritmo prediz

Figura 6. Tipos e valores definidos por Freyne et al. (2011)

General Cuisine	Specific Cuisine	Category
African, American, Asian, European, International, Oceania	African, Australian, Chinese, Eastern European, French, German, Greek, Indian, International, Italian, Japanese, Mexican, Middle Eastern, South East Asian, Southern, Spanish, UK&Ireland	beef, pork, lamb, chicken, veal, fish, vegetables, fruit

Fonte: Freyne et al. (2011)

a avaliação da receita alvo baseando-se nas avaliações que o sistema obteve em relação aos ingredientes presentes na mesma.

O terceiro algoritmo utilizado pelos autores foi o algoritmo M5P. Conforme Freyne et al. (2011) explicam, o M5P é um algoritmo de classificação de árvore binária em que os nós podem ser numéricos ou nominais, porém as folhas podem ser somente numéricas. Em cada nó da árvore é realizado um teste binário de um único atributo, deste modo cada nó folha é uma generalização mais específica contendo um modelo de regressão linear, tal qual realiza a predição da classe.

No estudo escolhido, a avaliação ocorreu com 343 receitas, separando-as em grupos de 35 para que fosse feita uma pesquisa online. Os usuários podiam votar um número ilimitado de vezes, em uma escala de 1 até 5. Após a pesquisa ser finalizada, cada um dos perfis dos usuários tiveram suas avaliações divididas, sendo 90% para treinamento e 10% para o teste.

Para validar os algoritmos da filtragem colaborativa e da filtragem baseada em conteúdo, os autores utilizaram o método de validação cruzada chamado *leave-one-out*. Este método necessita que, para cada um dos itens presentes na coleção de treinamento, crie-se um modelo único. A acurácia deste método é alta, assim como o custo computacional e, por isso, é recomendado que seja usado apenas em pequenas coleções. Para validar o algoritmo M5P, o mesmo foi executado separadamente nas avaliações de cada usuário.

Freyne et al. (2011) selecionaram 20 vizinhos para cada usuário, baseando-se em todo o conjunto de avaliações. Foram feitas 10 iterações para diferentes seleções da coleção de teste. A Figura 7 mostra o erro médio absoluto de cada uma das técnicas utilizadas. O algoritmo com menor erro foi o M5P, enquanto a filtragem colaborativa ficou com um erro de aproximadamente 0.05 acima da filtragem baseada em conteúdo.

Figura 7. Erro médio absoluto de cada algoritmo Freyne et al. (2011)

Content Based Filtering	Collaborative Filtering	Machine Learning (M5P)
1.2083	1.2614	0.9774

Fonte: Freyne et al. (2011)

O trabalho citado nesta seção tem relação com o presente trabalho pelo fato de que, ambos tem como foco a análise de um algoritmo para recomendação de receitas.

Além disso, a ideia deste trabalho é a de realizar a separação dos ingredientes de cada receita para que se possa recomendar novas receitas baseando-se no conteúdo das receitas anteriormente avaliadas positivamente pelo usuário.

3.2. Recipe Recommendation Method by Considering the User's Preference and Ingredient Quantity of Target Recipe

Os autores em [Ueda et al. 2014] explicam que o paladar exigente das pessoas é um dos principais causadores do aumento de problemas de saúde nos últimos anos, isso porque a população tende a buscar uma boa nutrição apenas nos alimentos que são agradáveis ao seu paladar. Levando isso em conta, o estudo de Ueda (2014) tem como objetivo o desenvolvimento de um método de recomendação de receitas considerando as preferências do usuário.

O método desenvolvido separa as receitas em ingredientes e classifica-os levando em conta a frequência de ingestão. Baseando-se na técnica TF-IDF, os autores propuseram a Equação 1.

$$I_k^+ = FF_k * IRF_k \tag{1}$$

Sendo FF_k a frequência de uso do ingrediente representada por $\frac{F_k}{D}$, onde F_k representa a frequência simples do ingrediente durante um determinado período D. O valor de IRF_k faz referência à peculiaridade do ingrediente, obtida pelo logaritmo da divisão do número total de receitas, M, pelo número de receitas que contém o ingrediente, M_k.

Ademais, Ueda (2014) consideraram os ingredientes que estavam presentes em receitas que foram acessadas, porém não foram feitas, como ingredientes que não fazem parte das preferências do usuário, mas levando em conta a frequência de aparição do ingrediente nas receitas acessadas e não feitas em comparação com o total de receitas não realizadas. Além disso, os autores também levaram em conta o impacto do ingrediente na receita como um todo, como por exemplo, 30 gramas de um determinado ingrediente k, em uma receita com um total de 400 gramas, não seria tão impactante como 10 gramas do mesmo ingrediente k, em uma receita com um total de 40 gramas.

Ainda, para analisar a acurácia do método, os autores selecionaram 25 receitas de forma aleatória de um total de 8050 receitas, que foram extraídas de um site de receitas. As receitas selecionadas foram então apresentadas aos usuários para que os mesmos as classificassem em uma escala de 1 até 5. Além disso, no começo do experimento, os usuários inseriram seus históricos de receitas dos últimos 30 dias, dessa forma o sistema obteve as preferências de cada um. A Figura 8 indica o ganho cumulativo com desconto normalizado obtido em cada um dos 4 métodos de avaliação utilizados pelos autores, sendo eles respectivamente: avaliação real do usuário, avaliação de um site popular, avaliação de acordo com o histórico de pesquisa e receitas realizadas, e avaliação do método proposto.

O estudo de [Ueda et al. 2014] tem como similaridade ao presente trabalho, o fato de, como objetivo, desenvolver um sistema de recomendação de receitas. Além disso, a receita foi separada pelos ingredientes de sua composição. A proposta mostrou ser parecida com a do presente trabalho, levando em conta a quantidade de cada ingrediente,

Figura 8. Resultados obtidos Ueda (2014)

	1) correct ranking	2) ranking by popular website	3) browsing and cooking history	4) proposal method
nDCG	1	0.8996	0.9072	0.9381

Fonte: Ueda (2014)

porém também foram considerados os ingredientes que não são preferência do usuário para gerar as recomendações.

4. Experimentos

Esta seção apresenta as etapas realizadas para verificar se a abordagem adotada para a limpeza de dados e a extração de atributos do conjunto de dados gerado possibilita que algoritmos de classificam tenham um bom desempenho na geração de recomendações.

4.1. Extração dos dados

Os dados utilizados foram extraídos de um *site* brasileiro de receitas chamado *Tudo Gostoso*[2]. Para realizar a extração dos dados foi utilizada a biblioteca de análise de dados em documentos *HTML* e *XML*, *Beautiful Soup*[3] e a biblioteca *Requests*[4] para enviar as requisições.

Antes de realizar qualquer extração, decidiu-se separar o processo em etapas, sendo a primeira etapa referente apenas à extração das *urls* referentes às receitas. Cada página do *site* possuia 15 receitas. Desta forma, decidiu-se que seriam usadas as receitas presentes nas primeiras 3.150 páginas, totalizando 47.250 receitas.

Para a extração das *urls* das receitas, foi necessário realizar a análise manual da estrutura do *site* escolhido. Conforme a análise prosseguiu, verificou-se como as *urls* de cada página eram montadas e qual era a classe correspondente ao elemento *HTML* que continha a *url* de acesso de cada receita visível na página.

Após ter concluído a análise inicial, deu-se início ao desenvolvimento da ferramenta responsável por extrair as *urls* de cada receita. Para isso, foi declarada uma variável contendo a *url* base de cada página. Uma estrutura de repetição foi feita de forma a simular o número de cada página, ou seja, começando em 1 e indo até o número 3150. Desta forma, em cada iteração, ocorria a concatenação do número ao final da *url* base e assim era obtido o endereço referente à página.

Cada endereço montado era enviado como argumento para a função *get* da biblioteca *Requests*. A função retornava um objeto de resposta, contendo a propriedade *content*, onde ficava o conteúdo da resposta recebida, em formato *HTML*, e era utilizada ao instanciar a classe *BeautifulSoup*.

A função chamada *findAll*, da classe *BeautifulSoup*, era utilizada passando como argumentos a classe referente ao *link* de cada receita e o tipo do elemento *HTML*. Assim, o retorno da função era uma lista de objetos contendo todos os elementos que possuíam a classe e o tipo especificado. Desta forma, cada objeto presente no retorno da função

[2]https://www.tudogostoso.com.br/

[3]https://www.crummy.com/software/BeautifulSoup/

[4]https://docs.python-requests.org/en/latest/

findAll invocava a função *find*, tendo como argumento o elemento âncora e capturando o atributo *href*, que em seguida era salvo em um arquivo de texto.

Após a finalização desta etapa, foi desenvolvida uma ferramenta para ler o arquivo de texto contendo as *urls*, enviar requisições para cada uma delas e extrair o título e os ingredientes de cada receita. Esta etapa prosseguiu da mesma forma que a etapa anterior, apenas com algumas mudanças nas classes, onde foi necessário analisar a estrutura da página de receitas e verificar quais eram as classes e elementos referentes ao título e aos ingredientes.

A presente etapa ocorreu em duas partes, sendo a primeira com o objetivo de extrair apenas os ingredientes e posteriormente ordená-los, de forma decrescente, por ordem de aparição. O passo a passo da extração está resumido na lista abaixo.

1. Ler o arquivo de texto contendo os *links*;
2. Para cada *link* encontrado enviar a requisição;
3. Extrair os ingredientes da receita pelo elemento *HTML ul (unordered list)* dentro da classe *HTML ingredients-card*;
4. Remover a medida e salvar apenas a descrição do ingrediente em um novo arquivo de texto.

Com a primeira parte concluída e o arquivo de ingredientes completo, foi possível saber quais ingredientes apareciam em, no mínimo, 1% das receitas. A porcentagem de aparições foi escolhida como forma de reduzir o número de cenários e, desta forma, simplificar a conversão das medidas.

Para que todos os ingredientes fossem padronizados, foi necessário convertê-los para o padrão *ASCII* utilizando a biblioteca *Unidecode* [5]. Para realizar o tratamento de palavras no plural, a classe *SequenceMatcher* da biblioteca *difflib*[6] foi utilizada e configurada para reconhecer palavras com similaridade mínima de 75%.

Com as informações dos ingredientes mais populares, foi criada uma planilha com o cabeçalho contendo o título e, em seguida, as próximas colunas continham os ingredientes mais populares, totalizando 57 colunas. Na Figura 9 é possível observar as 9 primeiras colunas do conjunto de dados.

Figura 9. Parte do cabeçalho da planilha referente ao conjunto de dados.

A	B	C	D	E	F	G	H	I
titulo	acucar	leite	sal	farinha trigo	leite condensado	creme leite	margarina	alho

Após a criação do cabeçalho da planilha, ocorreu a última etapa da extração dos dados. Desta vez extraindo o título e, para cada ingrediente, era realizada uma verificação com a biblioteca *difflib* para identificar se alguma das colunas da planilha tinha similaridade de, no mínimo, 75% com o ingrediente lido. Caso o retorno da verificação fosse positivo, então era utilizada uma expressão regular de modo a separar a medida, a quantidade e a descrição do ingrediente. Com essas informações, era realizada a conversão da medida, explicada na subseção abaixo.

[5]https://pypi.org/project/Unidecode/
[6]https://docs.python.org/3/library/difflib.html

4.2. Conversão das medidas

Para a padronização das unidades de medida, foram escolhidas as medidas de mililitro, para ingredientes líquidos, e grama para ingredientes sólidos. Estas medidas foram escolhidas pois, em relação as medidas originais dos ingredientes, eram as que mais mantinham a representação original da quantidade. A separação em ingredientes sólidos e líquidos foi feita de forma manual, apenas observando os ingredientes escolhidos e montando uma lista para cada tipo.

Para o desenvolvimento da ferramenta de conversão de medidas, foi criado um enumerador chamado *Medida*, com as seguintes constantes declaradas: `quilo`, `grama`, `miligrama`, `litro`, `mililitro`, `xicara`, `copo`, `colher_cafe`, `colher_cha`, `colher_sobremesa`, `colher_sopa`, `caixa` e `pitada`.

Após o desenvolvimento do enumerador, foi criada uma classe chamada *Converter*. A classe *Converter* possui listas referentes à cada medida presente no enumerador *Medida*. As listas possuem todas as palavras e siglas que referem-se à determinada medida. Por exemplo, a lista referente à medida quilo, possui palavras como “kg” e “kilo”. Também foi declara uma lista chamada ”unidades”, contendo todas as listas descritas no começo deste parágrafo.

Para questões de separação entre ingredientes sólidos e líquidos, foi criada uma lista chamada `ingredientes_liquidos`, contendo as palavras que faziam referência à ingredientes líquidos, como por exemplo a palavra leite. Esses ingredientes foram separados de forma manual, realizando análises nos ingredientes do cabeçalho da planilha.

Também foram declarados duas variáveis do tipo dicionário. A primeira chamada de `dicionario_conversao_solidos`, e a segunda chamada de `dicionario_conversao_liquidos`. Cada dicionário possui, como chave, os enumeradores *Medida*, referentes ao tipo do ingrediente (sólido ou líquido). O valor referente a cada chave é um dicionário contendo o enumerador da medida padrão, grama ou mililitro, e o valor equivalente para realizar a conversão. Por exemplo, a chave *Medida.Quilo* possuí como valor o dicionário com a chave *Medida.Grama* e valor 1000, indicando que um quilo equivale a mil gramas.

Para obter a medida utilizada no conteúdo do ingrediente extraído da lista de ingredientes da receita, foi criada uma função chamada de *obterMedida*, que possui a descrição da medida como parâmetro. Primeiramente, a função transforma a medida em uma palavra apenas com letras minúsculas e, em seguida, verifica se a lista *unidades* possui alguma sub-lista que possua a medida. Caso a verificação retorne verdadeiro, a função retorna o enumerador *Medida* referente à descrição da medida recebida como parâmetro.

A conversão ocorreu por meio da função chamada de *converter*, que recebia como parâmetro a descrição da medida, a quantidade e o ingrediente. Primeiramente, o parâmetro *medida* era repassado para a função *obterMedida*, que retornava qual era o enumerador *Medida* referente à descrição da medida. Caso não fosse encontrado, então a função retornava o valor 0.

Após verificar qual era a medida, ocorria a checagem para saber se o ingrediente era líquido, apenas conferindo se o mesmo estava presente na lista `ingredientes_liquidos`. Desta forma, era utilizada a variável que possuía o di-

cionário correto. O dicionário era verificado e, desta forma, a proporção correta era obtida. Após isso, a quantidade era transformada em um valor numérico de ponto flutuante, utilizando a classe *Fraction* para reconhecer quantidades que estejam em forma de fração, e guardada em uma variável chamada `valorConvertido`. O retorno da função era a proporção multiplicada pelo valor convertido.

A tabela 1 indica como ficou o resultado da conversão em relação à receita de panqueca americana com cobertura de chocolate.

Panqueca americana com cobertura de chocolate		
Ingrediente	Quantidade convertida	Quantidade original
Açúcar	15 gramas	2 colheres
Leite	125 mililitros	1/2 xícara
Farinha de Trigo	210 gramas	1 xícara
Chocolate	22.5 gramas	3 colheres
Manteiga	7.5 gramas	1 colher

Tabela 1. Comparação entre dados convertidos e dados originais da receita de panqueca americana com cobertura de chocolate.

4.3. Configuração do experimento

Para realizar a *clusterização* foi usada a técnica de redução de dimensionalidade *PCA* com dois componentes, que funciona de modo a simplificar os dados sem que percam sua importância, assim diminuindo e normalizando os valores [Abdi and Williams 2010].

A *clusterização* do projeto desenvolvido neste artigo foi feita com o algoritmo *KMeans*, utilizando a medida de inércia. O algoritmo *KMeans* foi escolhido por tratar-se de um algoritmo relativamente simples de ser implementado. Para auxiliar na definição do número de *clusters*, foi utilizado método do cotovelo, que verifica a porcentagem entre a variação de cada número de *clusters* e define que o melhor número de *clusters* a ser escolhido é aquele que, caso seja adicionado mais um *cluster*, a melhora do modelo não seja significativa [Bholowalia and Kumar 2014]. A Figura 10 mostra os resultados obtidos e, de acordo com análises gráficas e com o auxílio do método do cotovelo, decidiu-se que o melhor número de *clusters* seria 3.

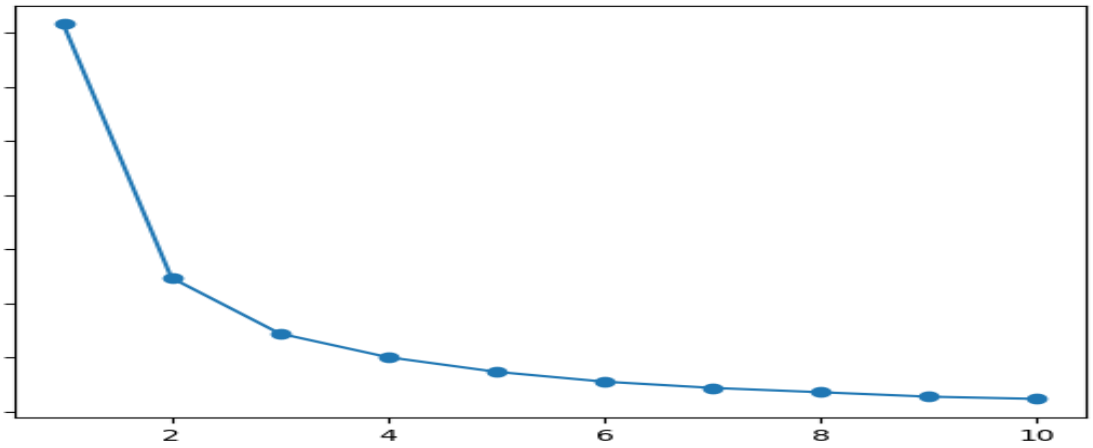
Para realizar as recomendações, o sistema escolhe os pontos próximos medindo a distância euclidiana entre eles e exibe os cinco melhores resultados em nível decrescente de similaridade. O experimento também foi testado utilizando similaridade por cosseno, porém não obteve um bom resultado em comparação com os resultados obtidos utilizando a distância euclidiana como medida.

4.4. Análise dos experimentos

O conjunto de dados criado é composto por 47.261 linhas e 57 colunas, sendo a primeira coluna referente ao título da receita e as demais são referentes aos ingredientes, totalizando um arquivo com tamanho de 6.27 *MegaBytes*.

Após analisar os 3 *clusters* gerados (conforme Figura 11), chegou-se à conclusão que as 3 categorias eram: doces, salgados e temperos.

Figura 10. Resultados obtidos através do método do cotovelo.



Receita	Distância Euclidiana
Brigadeiro	3.75
Banana com brigadeiro	7
Torta de chocolate gelado	7.5
Brigadeiro do asfalto	7.5
Brigadeiro da Mimi	7.7

Tabela 2. Receitas similares.

A Tabela 2 mostra um exemplo de 5 receitas consideradas mais similares à receita de brigadeiro de panela.

Este experimento demonstrou que é possível realizar a extração e padronização de receitas e ingredientes em linguagem natural, porém, para que haja maior precisão nos resultados, o sistema precisa converter as quantidades de cada ingrediente de forma precisa. Além disso, alguns ingredientes são mais específicos que outros, desta forma as recomendações geradas acabam sendo limitadas, pois decidiu-se utilizar apenas os ingredientes mais populares, que apareciam em, no mínimo, 1% das receitas.

Para medir a precisão das recomendações, foram realizados testes envolvendo três pessoas em que foram escolhidas seis receitas, sendo três do *cluster* salgado e três do *cluster* doce. O *cluster* dos temperos não foi considerado pois continha um número pequeno de itens. Foram geradas recomendações para cada receita escolhida. As recomendações geradas eram apresentadas para cada uma das três pessoas, onde as mesmas escolhiam uma entre as cinco receitas recomendadas.

As Figuras 12 e 13 mostram, nas duas primeiras colunas, quais foram as receitas escolhidas para os testes, quais recomendações foram geradas e suas distâncias referentes a receita escolhida. Nas três próximas colunas são mostradas quais foram as escolhas de cada pessoa entre as recomendações geradas.

De acordo com a análise dos registros presentes nos testes, as distâncias euclidianas atenderam satisfatoriamente a recomendação. No caso da receita *Strogonoff de carne moída*, todas as receitas recomendadas possuem, em quantidades similares, os ingredientes: creme de leite, queijo e carne. Já na receita *Pudim de padaria*, as recomendações geradas possuíam os mesmos ingredientes porém as quantidades não eram próximas.

Ao observar os registros do *cluster* salgado, é possível perceber que os ingredientes referentes as proteínas, como carne de frango e carne de gado, não tiveram um grau de reconhecimento elevado, pois muitas vezes os mesmos estão descritos com o tipo de corte da carne, o que acabou gerando problemas no momento do reconhecimento.

Conforme explicado anteriormente, o voto dos participantes um, dois e três estão nas colunas três, quatro e cinco, respectivamente. Analisando os votos de cada uma das pessoas envolvidas, considera-se que a ferramenta desenvolvida possui um desempenho promissor.

5. Conclusão

Esse artigo teve como objetivo apresentar uma abordagem para extrair e padronizar as medidas de receitas obtidas por meio de informações disponíveis na *web*, e a partir disso desenvolver o protótipo de um sistema de recomendação de receitas. A experimentação demonstrou que é possível extrair e padronizar os dados extraídos da *internet*, utilizando as bibliotecas existentes para a linguagem de programação *Python*. A experimentação demonstrou que, no conjunto de dados extraído, obteve-se um melhor resultado utilizando o algoritmo KMeans com três *clusters*. Também possibilitou a verificação das recomendações utilizando a distância euclidiana entre os objetos como medida de similaridade. No estado atual, a ferramenta desenvolvida obteve um resultado razoável em relação às recomendações geradas.

Em trabalhos futuros, para que o sistema gere recomendações mais precisas em receitas com ingredientes mais específicos, o conjunto de dados deve considerar todos os ingredientes, e não só os mais populares. Além disso, é possível desenvolver um sistema capaz de aperfeiçoar as recomendações, criando perfis, separados por grupos, condizentes ao gosto do usuário.

Referências

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.

Bholowalia, P. and Kumar, A. (2014). Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9).

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.

Dias, M. B., Locher, D., Li, M., El-Deredy, W., and Lisboa, P. J. (2008). The value of personalised recommender systems to e-business: a case study. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 291–294.

El Naqa, I. and Murphy, M. J. (2015). What is machine learning? In *machine learning in radiation oncology*, pages 3–11. Springer.

Fayyaz, Z., Ebrahimian, M., Nawara, D., Ibrahim, A., and Kashef, R. (2020). Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *applied sciences*, 10(21):7748.

Freyne, J., Berkovsky, S., and Smith, G. (2011). Recipe recommendation: accuracy and reasoning. In *International conference on user modeling, adaptation, and personalization*, pages 99–110. Springer.

Ghahramani, Z. (2003). Unsupervised learning. In *Summer school on machine learning*, pages 72–112. Springer.

Glauber, R., Loula, A., and Rocha-Junior, J. B. (2013). A mixed hybrid recommender system for given names. *ECML PKDD Discovery Challenge*, page 25.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Jarmul, K. and Lawson, R. (2017). *Python Web Scraping*. Packt Publishing Ltd.

Medeiros, I. (2013). Estudo sobre sistemas de recomendação colaborativos. *Acedido em março*, 2:2020.

Melville, P., Mooney, R. J., Nagarajan, R., et al. (2002). Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, 23:187–192.

Mooney, R. J. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204.

Natarajan, S., Vairavasundaram, S., Natarajan, S., and Gandomi, A. H. (2020). Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data. *Expert Systems with Applications*, 149:113248.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2011). *Recommender Systems Handbook*. Springer.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Suriati, S., Dwiastuti, M., and Tulus, T. (2017). Weighted hybrid technique for recommender system. In *Journal of physics: Conference series*, volume 930, page 012050. IOP Publishing.

Ueda, M., Asanuma, S., Miyawaki, Y., and Nakajima, S. (2014). Recipe recommendation method by considering the users preference and ingredient quantity of target recipe. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 12–14.

Van Meteren, R. and Van Someren, M. (2000). Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pages 47–56.

Wang, Y., Chan, S. C.-F., and Ngai, G. (2012). Applicability of demographic recommender system to tourist attractions: a case study on trip advisor. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 97–101. IEEE.

Zhao, B. (2017). Web scraping. *Encyclopedia of big data*, pages 1–3.

Figura 11. Categorias obtidas.

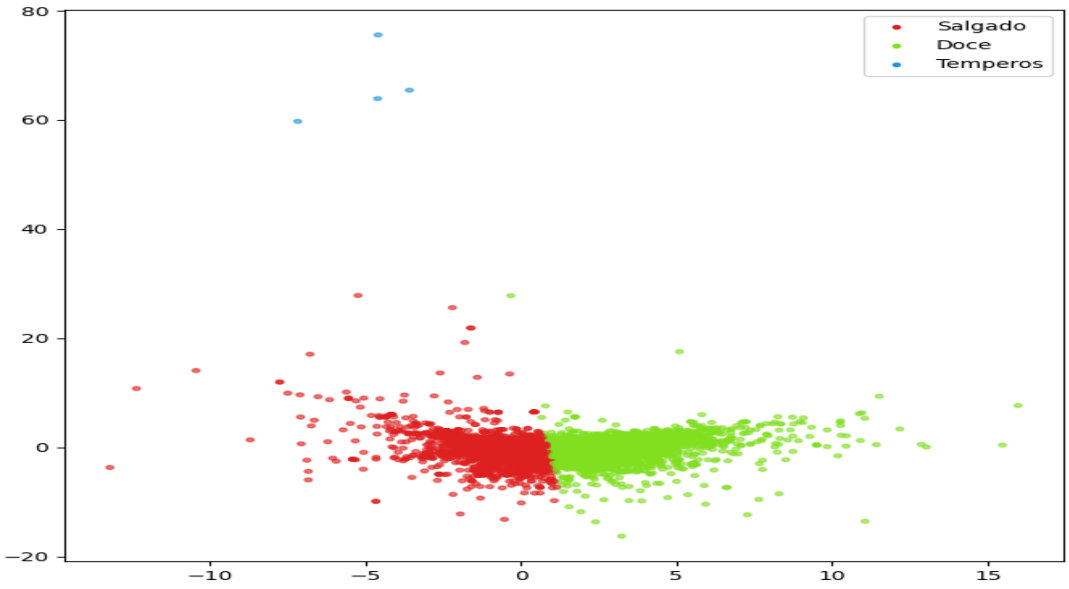


Figura 12. Resultados das recomendações do *cluster* salgado.

Receita	Recomendações	Pessoa 1	Pessoa 2	Pessoa 3	Cluster
Strogonoff de carne moída	1) Frango cremoso delícia - 1.0 2) Aipim cremoso com camarão - 7.5 3) Fricassê de frango saboroso - 7.5 4) Torta de estrogonofe da Cris -15.0 5) Torta romeu e julieta - 15.0	1	1	1	Salgado
Lasanha de pão fácil e gostosa	1) Sanduíche rápido de forno - 175.08 2) Lasanha de pão pullman - 206.83 3) Lasanha ao molho branco de queijo e frango - 231.74 4) Bauru de forno - 234.42 5) Lasanha de massa de pastel - 239.70	3	4	3	Salgado
Coxinha de frango	1) Muffin salgado – 22.5 2) Torta de liquidificador de presunto e queijo – 23.42 3) Panqueca de frango – 23.42 4) Bolinho de churros – 23.71 5) Panqueca verde – 23.94	3	5	1	Salgado

Figura 13. Resultados das recomendações do *cluster* doce.

Receita	Recomendações	Pessoa 1	Pessoa 2	Pessoa 3	Cluster
Bolo de chocolate e canela de liquidificador	1) Bolo morenão – 22.5 2) Pão Caseiro semi-integral fácil – 254.23 3) Biscoito legal – 257.64 4) Filé de peixe e leite de coco – 257.64 5) Bolo de chocolate econômico – 257.64	1	3	1	Doce
Pudim de padaria	1) Pudim de forno – 210.53 2) Bolo de castanha – 373.66 3) Bolo gelado – 420.26 4) Toucinho do céu – 420.86 5) Bolo pudim – 420.86	1	1	1	Doce
Torta holandesa	1) Bolo de pekans – 328.82 2) Mousse de chocolate do Márcio – 358.81 3) Brownie – 364.62 4) Bombons de abóbora com coco – 369.49 5) Brownie perfeito – 371.14	5	3	3	Doce