Luís Gonçalo Ferreira Mendes

Previsão em series temporais aplicada ao e-commerce

Julho 2021

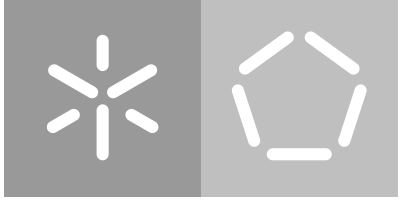
Previsão em séries temporais aplicada ao e-commerce
Luís Gonçalo Ferreira Mendes
Julho 2021

Universidade do Minho
Escola de Engenharia

Luís Gonçalo Ferreira Mendes

Previsão em séries temporais aplicada ao e-commerce

Julho, 2021



Universidade do Minho
Escola de Engenharia

Luís Gonçalo Ferreira Mendes

Previsão em séries temporais aplicada ao e-commerce

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação do(a)
Victor Manuel Rodrigues Alves

Julho, 2021

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

_____, _____

(Localização) (Data)

_____

(Luís Gonçalo Ferreira Mendes)

(Página intencionalmente deixada em branco)

Agradecimentos

Agradeço a todos os que tenham contribuído para a realização desta dissertação. Em primeiro lugar ao meu orientador Victor Manuel Rodrigues Oliveira pelo conhecimento e disponibilidade demonstrada ao longo da elaboração desta dissertação, e não só.

Agradeço ainda ao João Manuel da Silva Sampaio da Kodly Consulting que me acompanhou durante a realização desta dissertação.

À minha família pelo apoio e incentivo, em especial à Estrela Ferreira e Margarida Ferreira pelo incentivo final e pela revisão do documento.

Resumo

Historicamente, as pessoas têm recorrido a amigos próximos ou experts para ajudar na tomada de decisão ou em recomendações sobre assuntos das mais diversas áreas. O crescimento da era digital nas últimas duas décadas, principalmente na web criou um overload de informação. No entanto, a nossa capacidade de avaliar as especificações de cada produto e escolher entre as enumeras alternativas existentes no mercado online, é limitada. Neste sentido, a ciência e tecnologia tem reagido adequadamente através do desenvolvimento de ferramentas para aliviar essa limitação. Os sistemas de recomendação são exemplos dessas ferramentas, que surgiram em meados dos anos 90 e têm tido muito sucesso.

Os sistemas de recomendação funcionam como um atendimento personalizado. Numa situação de atendimento presencial apenas é possível apurar a pretensão do cliente após este a ter descrito. Estes sistemas visam optimizar e indicar a opção mais ajustada de acordo com o perfil do próprio cliente. Uma boa recomendação poderá ir ao encontro das pretensões de um determinado utilizador quando confrontado com uma plataforma e-commerce, o que faz com que o rácio utilizador/compra aumente. Além disso, quantos mais utilizadores satisfeitos com o sistema, mais popular e melhor o sistema se tornará, além de que será criada uma relação de proximidade entre o utilizador e o website.

Existem vários métodos para gerar recomendações personalizadas para utilizadores específicos, mas o estado da arte de sistemas de recomendação são baseados na filtragem colaborativa. Estes sistema são amplamente utilizados por empresas como a Amazon, Netflix entre outras e tem obtido resultados muito significativos sem a necessidade de extrapolar as características dos artigos. Métodos ligados ao Deep Learning levaram a um elevado progresso em vários campos da Inteligência Artificial e, nos anos mais recentes, a um substancial número de propostas para melhorar sistemas de recomendação com Redes Neuronais Artificiais.

Nesta dissertação, realizada na empresa Kodly Consulting, propõem-se a criação de uma solução baseada em redes neuronais recorrentes para geração de recomendações em ambientes e-commerce. Para que o sistema proposto seja facilmente integrado em qualquer aplicação e-commerce, pretende-se criar uma API que disponibiliza um conjunto de serviços úteis e fáceis de integrar.

Palavras-chave: Sistemas de Recomendação, Machine Learning, Redes Neuronais Recorrentes, Deep Learning, e-commerce, API, pipeline, Filtragem Colaborativa.

Abstract

Historically, people have been asking friends or experts for help with decision-making or in getting recommendations on issues from the most diverse areas. The digital growth on the last decades, created a big overload of information. However, our ability to evaluate the specifications of each product and choose from the several alternatives available on the online market is limited. At the same time, science and technology has responded adequately by developing tools to mitigate and to help people in this limitation. Recommendation systems are examples of these tools, which emerged in the mid-1990s and have been very successful.

A recommendation system works like a personalized service. In a face-to-face service situation, it is only possible to determine the customer's claim after she/he has described it. These systems aim to optimize and indicate the most adjusted option according to the client's own profile. A good recommendation can meet the wishes of a certain user when faced with an e-commerce platform, which makes the user/purchase ratio increase. Furthermore, the more users satisfied with the system, the more popular and better the system will become, and a close relationship will be created between the user and the website.

There are several methods to generate personalized recommendations for specific users, but state of the art recommendation systems is based on collaborative filtering. These systems are widely used by companies such as Amazon, Netflix and others and have obtained very significant results without the need to extrapolate the characteristics of the articles. Methods like Deep Learning have led to great progress in various fields of Artificial Intelligence and, in recent years, to a substantial number of proposals to improve recommendation systems with Artificial Neural Networks.

In this dissertation, carried out at Kodly Consulting, we propose the creation of a solution based on recurrent neural networks to generate recommendations. In order to the proposed system be easily integrated into any e-commerce application, it is intended to create an API providing a set of useful and easy-to-integrate services.

Keywords: Recommendation Systems, Machine Learning, Recurrent Neural Networks, Deep Learning, e-commerce, API, Pipeline, Collaborative Filtering.

Índice

Capítulo 1

Introdução

1.1 Contexto e motivação

Em três palavras: Sobrecarga de informação. Como humanos, é natural filtrarmos os inputs que recebemos por algum critério de importância. No entanto, existe um limite sobre a quantidade de informação que se consegue processar de cada vez. Para evitarmos sermos sobrecarregados necessitamos de estratégias para reduzir a complexidade daquilo que vamos assimilando. É por isso importante, especialmente na era do Big Data, termos sistemas com técnicas de heurísticas que facilitam o processo de seleção. Os sistemas de recomendação são sistemas que se especializam no problema de selecionar de um catálogo extenso apenas os itens que são do interesse do utilizador [1, 2].

Os Sistemas de recomendação são geralmente definidos como ferramentas de software e técnicas usadas para proporcionar sugestões para itens que vão ao encontro dos interesses do utilizador [1, 2]. O objectivo dos sistemas de recomendação é ajudar os utilizadores a encontrar apenas informação que se adequa às sua preferências, através da procura numa grande quantidade de informação indiferenciada [1]. Com o rápido desenvolvimento da internet e o surgimento de big data, a quantidade de informação disponível aumentou exponencialmente. É agora cada vez mais difícil obter informações precisas e eficientes em tempo útil. Esse é o principal motivo pelo qual os sistemas de recomendação tem recebido cada vez mais atenção e influenciado a vida das pessoas.

Existem vários métodos para gerar recomendações personalizadas para utilizadores específicos mas o estado da arte de sistemas de recomendação são baseados na filtragem colaborativa. Sistemas de filtragem colaborativa podem ser classificados em duas classes.

- Modelos que se baseiam na vizinhança (onde modelos orientados aos artigos são os mais dominantes.) nos quais conseguem descobrir artigos semelhantes aqueles que obtiveram um feedback positivo do utilizador, e sugeri-lo ao próprio.

- Filtragem colaborativa baseado em Matrix factorization

Estes sistemas têm actualmente um papel importante em quase todas as empresas famosas ligadas à tecnologia. Amazon, Youtube, Netflix, Spotify, IMDb e Tripadvisor são apenas alguns dos exemplos que comportam milhões de utilizadores. A Netflix, por exemplo, chegou a lançar uma competição em que oferecia uma prémio de 1 milhão de dólares à primeira equipa que fosse capaz de melhorar em pelo menos 10% o seu sistema de recomendação.

Face ao exposto, facilmente se depreende que, hoje em dia, sistemas de recomendação são uma componente importante quando interagimos com o mundo, ligados à internet. O objectivo destes sistemas é oferecer uma experiência confortável e intuitiva ao utilizador, ser capaz de sugerir recomendações relevantes e úteis, e ao mesmo tempo evitar o sentimento de que estamos perdidos na web. Motivados por estas características, estes sistemas tem vindo a crescer exponencialmente nos últimos anos. É por isso importante perceber como construir as diferentes camadas de um sistema de recomendação de forma eficiente e útil para os utilizadores.

Uma boa recomendação poderá ir ao encontro às pretensões de um determinado utilizador quando confrontado com uma plataforma e-commerce (ou comercio electrónico) o que faz com que o rácio utilizador/compra aumente.

A escolha deste tema foi motivada pelo grande aumento de popularidade deste tipo de sistemas e ao mesmo tempo pela grande procura que existe no mercado actualmente. Muitas empresas presentes no contexto de e-commerce começam agora a chegar a um elevado nível de maturação e procuram novas formas de aumentar as sua fontes de receitas. Os sistemas de recomendações pelo bons resultados que conseguem apresentar tornaram-se impossíveis de ignorar.

Actualmente o e-commerce é um dos conceitos mais importantes tanto na Internet como na sociedade em geral. Este tipo de comercio permite que os consumidores transaccionem bens e serviços electronicamente sem barreiras de tempo ou distância. O comércio electrónico expandiu-se rapidamente nos últimos anos e prevê-se que continue expandir-se com a mesma taxa de crescimento ou mesmo que haja uma aceleração do crescimento.

A pandemia Covid-19 e os sucessivos confinamentos obrigatórios, impulsionou o crescimento do uso desta ferramenta, onde muitos dos utilizadores, que até aí ofereciam alguma resistência, cederam à evidente utilidade, facilidade de uso e comodidade de um serviço electrónico.

Brevemente as fronteiras entre comércio “convencional” e “electrónico” tenderão a esbater-se, pois cada vez mais negócios deslocam secções inteiras das suas operações para a Internet.

O sucesso do e-commerce tem entre os principais factores, a percepção do consumidor sobre o processo e-commerce. Isto pode ser percepcionado num estudo que indica que 91% dos utilizadores insatisfeitos não voltariam a realizar uma compra electrónica novamente por causa da fraca experiência durante o processo de compra [3]. Isto significa que muitas empresas perdem clientes e não conseguem atingir um crescimento apropriado.

O principal desafio será aplicar os mais recentes avanços no campo das redes neuronais ao contexto do e-commerce.

As redes neuronais recorrentes são um tipo de rede neuronal artificial projectada para reconhecer padrões em sequências de dados, como texto, genomas, caligrafia, palavra falada ou dados de séries numéricas resultantes de sensores, bolsas de valores e agências governamentais [4]. Esses algoritmos consideram tempo e sequência, concedendo uma dimensão temporal.

O levantamento do estado da arte mostra que estes algoritmos são um dos tipos mais poderosos e úteis de rede neuronais, juntamente com o mecanismo de atenção e as redes de memória. As Recurrent Neural Network(RNNs) (Redes Neuronais Recorrentes) são aplicáveis até mesmo a imagens, que podem ser decompostas em uma série de amostras e tratadas como uma sequência.

1.2 Objectivos

O principal objectivo desta tese passa por estudar a aplicação de um sistema de recomendação socorrendo-se dos desenvolvimento nas redes neuronais recorrentes. Este sistema deverá ser facilmente moldado a diferentes contextos de e-commerce, isto é, será criado um pipeline de constante recolha de dados e por conseguinte a criação/actualização de um modelo de dados que se pretende cada vez mais eficaz ao longo do tempo.

1.3 Apresentação da empresa

Este trabalho foi desenvolvido no âmbito da Empresa Kodly Consulting, a operar em Portugal, Reino Unido e Espanha.

A Kodly foi fundada com o objetivo principal de ajudar as empresas a implementar estratégias de transformação digital com sucesso. A empresa usa as melhores tecnologias disponíveis para resolver os problemas do negócio. A empresa defende um "ecossistema" de software empresarial aberto e de primeira linha, que deve ser baseado em microsserviços, API em primeiro lugar, nativo na cloud (nuvem) e sem cabeçalho. As soluções são resilientes, altamente escaláveis, testáveis, fáceis de organizar em torno de modelos de negócios e implantadas de forma independente.

A Kodly especializou-se na Transformação Digital, com forte foco em automação de entrega de software (CI/CD DevOps - Continuous Integration and Continuous Deployment) aplicadas ao e-Commerce e migração de aplicações para a cloud. Assim a Kodly disponibiliza um conjunto de serviços para empresas, nomeadamente:

- "Code factory" (Fábrica de código) - código personalizado de alta qualidade para startUps, ajudando as StartUps a inicializar as suas soluções tecnológicas.
- APIs e microsserviços hospedados na cloud. Os microsserviços estão muito em voga atualmente porque tornam o desenvolvimento, a integração e a manutenção dos aplicativos muito mais fácil.

- Experiências digitais (UX/UI) - A evolução dos “média digitais” destacou o utilizador como fator central no design do produto digital. Ao focar o seu desenvolvimento na criação de conteúdos relevantes para os utilizadores, bem como na sua acessibilidade imediata e facilidade de utilização, as experiências digitais mudaram os paradigmas do pensamento e da produção.

- Automação de integração, testes e entregas (CI / CD - integração contínua / entrega contínua) é um método para entrega frequente de aplicativos aos clientes. Isto torna possível resolver os problemas que a integração de um novo código pode causar para as equipas de operação e desenvolvimento.

- Native and cross-platform mobile Apps (Aplicativos móveis nativos e de plataforma cruzada) - Os aplicativos nativos são caracterizados pela sua capacidade de oferecer uma experiência de utilização otimizada para dispositivos móveis.

A Kodly foi fundada sob o lema de fornecer o melhor ambiente de trabalho possível, incluindo um ideal subjacente de que todos nos devemos esforçar para a melhoria contínua e nunca parar de aprender. A empresa promove autonomia e auto-responsabilidade. Essa mentalidade requer auto-disciplina, resiliência, curiosidade, entusiasmo e boas habilidades de comunicação. Na Kodly Valorizam-se os pontos fortes individuais e espera-se que todos os membros da equipe sejam entusiastas da tecnologia. Os desafios do trabalho e do cliente são levados a sério, e a equipe está focada e preparada para enfrentar os problemas técnicos mais complexos. Em resumo, na Kodly valorizam-se as pessoas que não precisam de supervisão para ”fazer o trabalho”, são capazes de auto-gestão e continuam a entregar valor mesmo quando os objetivos não são atendidos numa ”bandeja de prata”.

1.4 Estrutura da dissertação

Esta dissertação apresenta-se estruturada nos seguintes capítulos:

- Capitulo 1 - Este é o capitulo actual, onde se começa por fazer uma contextualização e onde se descreve o enquadramento deste trabalho. Neste capitulo apresentam-se também as motivações e objectivos desta dissertação.

- Capitulo 2 - Neste capitulo apresentam-se as principais frameworks e linguagens de programação usadas no desenvolvimento da solução apresentada nesta dissertação.

- Capitulo 3 - Neste capitulo apresentam-se os principais conceitos abordados nesta dissertação assim como são apresentadas as tecnologias que são utilizadas durante a realização deste trabalho. Neste capitulo apresentam-se também os principais desafios e problemas encontrados nos sistemas de recomendação e a metodologia aplicada em cada um deles.

- Capitulo 4 - Neste capitulo faz-se o levantamento do estado da arte. Numa primeira parte fala-se sobre abordagens de Deep Learning em sistemas de recomendação e numa segunda parte fala-se sobre a utilização de deep learning em recomendações no contexto do e-commerce.

- Capitulo 5 - Neste capítulo apresenta-se o pipeline concebido nesta dissertação, que se baseia no auto denominado AutoRecRNN, bem com a sua descrição completa.
- Capitulo 6 - Neste capítulo apresenta-se a API desenvolvida neste trabalho, nomeadamente a persistência dos dados, algoritmos usados e serviços disponibilizados.
- Capitulo 7 - Neste capitulo é discutido o caso de estudo usado nesta dissertação, onde se inclui a análise dos dados, os métodos usados e apresentação dos resultados obtidos.
- Capitulo 8 - Finalmente, o último capitulo será dedicado à discussão dos resultados e à apresentação das conclusões que dai advém. No final deste capitulo apresentam-se algumas ideias para trabalho futuro.

A lista de referências, apêndices e um anexo são também incluídos no final do documento.

Capítulo 2

Ambiente de desenvolvimento

2.1 Linguagem de programação: Java

A principal linguagem usada no desenvolvimento deste projecto é Java. Java é uma linguagem de alto nível que derivou da linguagem C e usa o paradigma orientado aos objectos. Esta tecnologia foi lançada pela Sun Microsystems em 1995. Esta empresa foi adquirida pela Oracle em 2010. A Java virtual machine (JVM) está disponível em vários sistemas operativos, tais como MacOs, UNIX ou Windows, o que permite correr uma aplicação em qualquer destes sistemas sem a necessidade de reescrever ou adaptar código para cada um dos sistemas operativos. É muito por causa desta transversalidade que existem cerca de 15 mil milhões de dispositivos que utilizam Java e cerca de 10 milhões de developers espalhados pelo mundo. A escolha desta tecnologia no desenvolvimento deste projecto foi motivada pelas suas características multi-plataforma mas principalmente pela diversidade de bibliotecas e frameworks disponíveis.

2.2 Linguagem de programação: Phyton

Python é uma linguagem de programação de alto nível criada por Guido van Rossum em 1991. A linguagem possui um modelo de desenvolvimento comunitário open source o que possibilitou a criação de grande número de bibliotecas que são muito úteis para ciências de computação e ML (Machine Learning). Atualmente Python é uma das linguagens de programação mais populares para data science, ou ciência de dados.

2.3 Framework: Flask

Flask é uma framework web escrita em Python. É classificada como uma microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível. Não

possui uma cama de abstracção da base de dados mas suporta extensões que pode addicionar features com se fossem implementadas no próprio Flask

2.4 Framework: Spring

Para ajudar no desenvolvimento do projecto foi decidido usar uma framework em Java, Spring. Spring é uma ferramenta open source que fornece suporte de infra-estruturas para desenvolver uma aplicação em linguagem Java. É uma das frameworks mais populares utilizada no desenvolvimento de aplicações enterprise. Entre as principais vantagens desta tecnologia está o facto de na ligação feita à base de dados ser criado um conjunto de métodos para a transação simples com a API. Para o uso do HTTP request, Spring cria um método, com um endpoint HTTP, fazendo com que nos deixemos de preocupar com o Servlet API

2.5 Geração de documentação: Swagger

Swagger é uma linguagem de descrição de interface para descrever APIs REST que utilizam JSON. Swagger é utilizado em conjunto com outras ferramentas de software de open source para especificar, construir, documentar e utilizar serviços REST. Este ferramenta inclui ainda criação de documentação automática, geração de código ao qual inclui casos de teste.

2.6 Ferramenta para automação do processo de compilação: Maven

Maven é um software de gestão e criação automática que gere e constrói um projecto com grande facilidade e foi desenvolvida pela Apache Software Fundation. O processo de construção de um projecto maven é baseado no conceito POM Projecto Object Model. O ficheiro POM é usado para descrever o projecto, definir todas as dependências, o processo de compilação e os plugins necessários. As bibliotecas Java e os plug-ins são descarregados dinamicamente de repositórios globais tais como o Maven 2 Central Repository e são armazenados localmente para serem usados como dependências do projecto.

2.7 Base de dados não relacional: MongoDB

MongoDB é uma base de dados orientada a documentos, open source e multiplataforma, escrito em linguagem C++. É considerada uma base de dados não relacional e usa documentos semelhantes a esquemas JSON. Foi desenvolvido pela MongoDB Inc e publicado sob uma combinação de GNU, Affero General Public License e licença Apache. As suas características permitem que as aplicações modelem informação de modo muito mais natural, pois os dados podem ser aninhados em hierarquias complexas e continuarem a serem indexáveis e fáceis de procurar. O desenvolvimento da MongoDB começou em

Outubro de 2007 pela 10gen, atual MongoDB Inc, e a sua primeira versão pública foi lançada em Fevereiro de 2009.

2.8 Biblioteca para o treino de redes neuronais: TensorFlow

TensorFlow é uma biblioteca de código aberto para Machine Learning. É um sistema para a criação de treino de redes neuronais que são utilizadas detectar padrões e correlações.

2.9 Biblioteca de alto nível para a utilização do TensorFlow:Keras

Keras é uma biblioteca para a aplicação de redes neuronais, open source escrita em Python que nos permite utilizar o TensorFlow de uma forma simples e intuitiva.

2.10 Ambiente de virtualização: Docker

Docker é uma implementação de virtualização de containers que permite a independência entre o código desenvolvido e a plataforma onde é executado. Para alem disso esta ferramenta permite-nos facilmente escalar cada serviço mediante as necessidades.

Capítulo 3

Conceitos e tecnologias

3.1 Tecnologias

3.1.1 Redes Neuronais

Uma rede neuronal pode significar uma rede neuronal biológica, como o nosso cérebro, ou uma rede neuronal artificial simulada num computador. Um único neurónio pode não ser muito poderoso, mas quando ligados uns aos outros os nossos neurónios formam sistema muito poderoso e com uma enorme capacidade de aprendizagem, dedução, reação e adaptação.

As redes neuronais artificias são inspiradas nas redes neuronais biológicas. Nas redes neuronais, o sistema é composto por um grande número de "neurónios" ligados entre si mas cada um capaz de processar informações sozinho. Os vários neurónios da rede conseguem processar grandes quantidades de informações em simultâneo. A informação pode ser armazenada a curto prazo nos próprios "neurónios" ou pode ser armazenada a longo prazo nas ligações entre os neurónios ao quais podemos associar pesos. Assim, abre a possibilidade ao processamento paralelo.

Nas redes neuronais, a saída de um neurónio pode ser usada como a entrada de outros neurónios, e por sua vez, as saídas destes podem ser as entradas de outros neurónios e assim sucessivamente, formando assim uma rede que pode ter várias camadas. As redes neuronais estiveram muito em voga e foram usadas na área de inteligência artificial nos anos 60. Atualmente, as redes neuronais estão novamente em destaque muito devido ao Deep-Learning que está a ser usada para alcançar melhorias significativas em muitas áreas, como o processamento de linguagem natural e de imagens [5].

3.1.2 Deep-Learning

Deep-Learning (DL) é uma abordagem de machine learning (ML) que recorre ao uso de redes neuronais artificias e faz uso de uma hierarquia de representações mais abstratas para promover a generalização.

Deep-learning recorre ao uso de redes neuronais compostas por várias camadas (ver Figura) por meio das quais os dados são transformados. É por isso, muito escalável, ou seja, quando o volume de dados aumenta a performance mantem-se [6]. Segundo Dargan et al., atualmente deep-learning é considerada a melhor escolha para trabalhar com arquiteturas complexas em dados de alta dimensão [6]. A denominação de profundo (deep) deve-se ao facto do número de níveis da rede neural aumentar com o tempo [6].

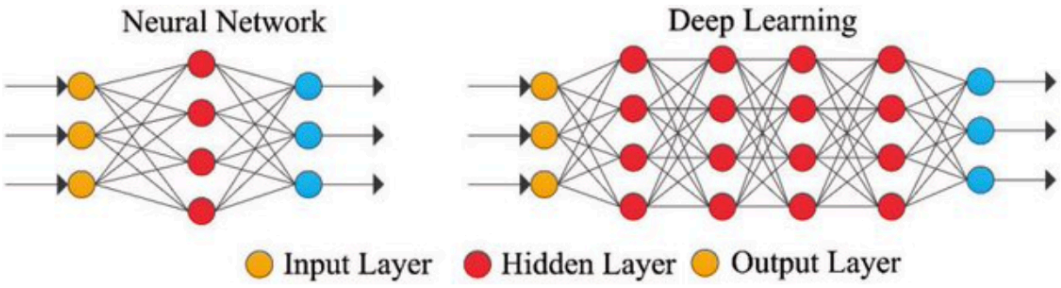


Figura 3.1: Representação de DL (extraído de [7])

A figura 3.1 representa uma rede neuronal simples à esquerda e uma rede com várias camadas (DL) à direita.

3.1.3 Machine learning

Machine learning (ML), ou aprendizagem máquina, é uma subseção da Inteligência Artificial (IA) que transmite ao sistema os benefícios de aprender automaticamente a partir dos conceitos e do conhecimento sem ser programado explicitamente. Começa com observações como as experiências diretas para se preparar para os recursos e padrões nos dados e produzir melhores resultados e decisões no futuro [6].

3.1.4 Redes Neuronais Recorrentes

Os computadores podem realizar computação numérica melhor do que os humanos. No entanto, é desejável que eles também sejam capazes de resolver problemas complexos de percepção. Por esta razão sistemas como Redes Neuronais Recorrentes que foram inspirados no sistema biológico neuronal, têm sido amplamente estudados ao longo dos anos. Uma rede neuronal recorrente (RNN) é uma classe de redes neuronais artificiais onde as conexões entre os nós formam um grafo direccionado ao longo de uma sequência temporal. Isso permite que ele exiba um comportamento dinâmico temporal. Derivado de redes neuronais feedforward, RNNs podem usar o seu estado interno (memória) para processar sequências de comprimento variável de entradas. Isso permite que sejam aplicáveis a problemas como reconhecimento de escrita manual ou reconhecimento da fala.

A Figura 3.2 evidência as diferenças entre as redes neuronais. A imagem da esquerda representa uma rede neuronal recorrente onde o output de uma camada pode ser adicionado como input na mesma

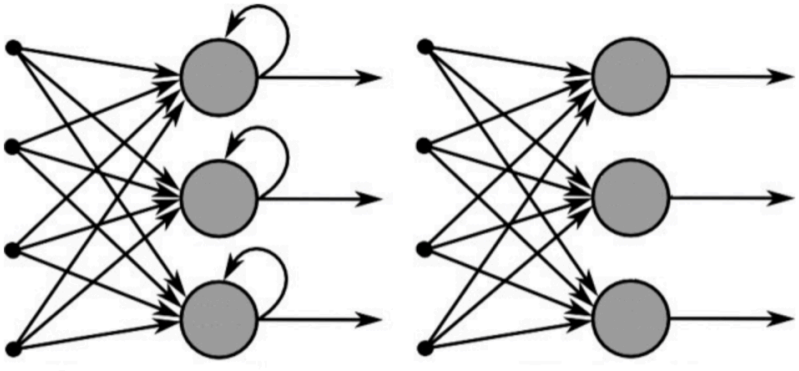


Figura 3.2: Rede Neuronal Recorrente versus Rede feedforward ([7])

camada (loop representado na figura), a imagem da direita representa uma rede neuronal feedforward onde esse loop não ocorre.

3.2 Conceitos

3.2.1 Collaborative filtering (Filtragem colaborativa)

Como já foi referido anteriormente, os sistemas colaborativos são muito utilizados em sistemas de renome como é o caso da Amazon e da Netflix. A principal razão para a sua utilização deve-se ao facto de conseguir resultados bastante significativos sem a necessidade extrapolar as características dos artigos.

Um pessoa quando pretende ver um filme procurar informação junto de pessoas/amigos que partilham o mesmo gosto pelo cinema. É com base neste princípio que este sistema funciona. As recomendações feitas por pessoas que partilham os mesmos gostos são sempre melhores que outras recomendações de outro tipo. É por isso que este sistema é o mais popular de entre todos os sistemas de recomendação.

Um sistema de recomendação deste tipo corresponde a um paradigma que procura reconhecer as preferências do utilizador, tecendo recomendações com base na informação adquirida a partir do utilizador e da comunidade, gerando recomendações com base nessa informação. Esta abordagem utiliza o histórico de actividade e oferece uma apta solução de recomendação para os cenários onde normalmente ocorre sobreposição dos hábitos de consumo.

Esta estratégia de filtragem colaborativa realiza previsões independentes do domínio, já que parte do conteúdo pode não ser facilmente caracterizado. Assim, esta técnica assenta na construção de uma base de dados utlizador-item de modo a retratar a relação de preferência, calculando semelhanças entre perfis de utilizadores, gerando assim as consequentes recomendações dentro dos grupos que se criam e que são apelidados de vizinhanças [8].

O mais comum algoritmo deste tipo representa um cliente como um vetor de itens de dimensão N, onde N é o número de itens diferentes, presentes num catálogo. O conteúdo do vector apresenta um valor positivo para os itens comprados (subentendendo-se que um utilizador apenas compra aquilo que gosta) ou para as avaliações positivas, será apresentado um valor negativo para aqueles que forem avaliados

dessa forma [9]. Este algoritmo gera por isso recomendações baseadas nos clientes que apresentam os interesses mais semelhantes entre si. Para isto, tem que existir um critério de semelhança entre o utilizador A e B. O método mais comum utiliza a distancia cosine que mede o coseno do ângulo entre dois vectores:

$$semelhanca(A,B) = cos(A,B) = \frac{A \bullet B}{||A|| * ||B||}$$

O algoritmo consegue também criar recomendações a partir dos itens dos clientes, usando vários métodos. A técnica mais comum é atribuir um rank a cada item de acordo como número de clientes semelhantes que o compraram. No entanto, este algoritmo consegue ser computacionalmente caro tendo em Θ (MN) o seu pior caso, com M a ser o número de clientes e N o número de produtos existentes no catalogo. Apesar disso, e uma vez que o vetor pertencente a um cliente é por norma pequeno, a performance do algoritmo tende-se a aproximar de Θ (M+N). Este factor permite analisar todos os clientes em Θ (M), deixando de ser um problema. Os utilizadores colaboram, portanto, entre si num modelo simbiótico de auto-ajuda.

3.2.1.1 Memory-based collaborative filtering

Este modelo, como os autores de [10] descrevem, tenta encontrar utilizadores similares ao utilizador activo e usa as suas preferências para prever os ratings para o utilizador ativo. São várias as vantagens de utilizar este tipo de filtragem: é escalável para uma grande quantidade de dados, é relativamente simples de implementar, é fácil de atualizar a base de dados a cada nova entrada e fornece informação sobre como o sistema de recomendação funciona. No entanto, não é um sistema sem limitações. Este modelo é muito lento pois utiliza a base de dados completa cada vez que tem que fazer uma previsão. Além disso, não é capaz de fazer previsões precisas e confiáveis se um utilizador ativo não tiver itens em comum com outros utilizadores.

3.2.1.2 Model-based collaborative filtering

É um modelo machine learning ou data mining que encontra padrões de classificação complexos em dados de treino, sendo posteriormente capaz de fazer previsões ou recomendações baseados no modelo aprendido [11].

Existem várias vantagens num modelo baseado em algoritmos de filtragem colaborativa: Aumenta o desempenho, é escalável para qualquer dataset e evita facilmente o overfitting. No entanto algumas limitações são características deste modelo: O problema da esparsidade torna o modelo incapaz de gerar recomendações razoáveis para utilizadores que não deram qualquer avaliação de produtos.

Métodos como modelos Bayesianos, modelos de clusterização, e redes de dependências tem sido explorados para resolver as deficiências dos algoritmos memory-based. O que é mais comum em algoritmos de classificação é a utilização de aprendizagem não supervisionada desenhada para dividir objetos

em diferentes categorias. Algoritmos de clusterização podem ser usados como modelos de filtragem colaborativa se os ratings dos utilizadores forem atribuídos por categorias, modelos de regressão, métodos Single Value Decomposition, Naive Bayes, Matrix Factorization que encontram fatores comuns que podem ser a razão subjacente para atribuição que um utilizador faz a determinado rating. Existe ainda Matrix Factorization probabilístico que estende MF numa framework probabilística. São modelos model-based CF que fazem recomendações.

3.2.2 Content-based filtering

Sistemas de recomendação baseados em conteúdo utilizam os perfis do utilizador ou a descrição dos artigos para recomendar. Por forma a fornecer informação relevante para o utilizador, o perfil do utilizador tem que ser criado usando web mining ou usando métodos de recolha de informação com os atributos e features dos itens. Sistemas baseados em conteúdo filtram itens com base na similaridade do conteúdo que o utilizador está interessado.

Existem várias vantagens na construção de um sistema deste tipo. É capaz de gerar recomendações através de ratings exclusivos, que são usados pelo utilizador activo para construir o seu próprio perfil. Fornece transparência para os utilizadores activos com a possibilidade de explanar como o sistema funciona, e é adequado para recomendar itens que ainda não foram avaliados por qualquer utilizador. Este traria uma grande vantagem a um novo utilizador. No entanto várias limitações são características deste sistema, como por exemplo: recomendar o mesmo tipo de produtos pois trata-se de um modelo que sofre de sobre-especialização. É mais difícil adquirir feedback dos utilizadores num CBF porque os utilizadores não avaliam os itens e por isso não é possível determinar se a recomendação é correta. Sofre ainda do problema de ser difícil gerar atributos para os itens. Um sistema de recomendação baseado em conteúdo compara os itens já adquiridos pelo utilizador ou procurados pelo mesmo e recomenda itens semelhantes. Um sistema de recomendação baseado em conteúdo puro apenas considera as escolhas do utilizador e um sistema deste tipo deveria considerar as escolhas feitas por amigos, para por exemplo, aumentar o nível de satisfação do utilizador. No entanto este sistema poderá ir ao encontro do que um utilizador procura. São utilizados vários algoritmos neste tipo de problema sendo estes os mais conhecidos: Decision Trees e Naive Bayes Classifier.

O paradigma Content-based oferece auxílio no processo de identificação de novos itens que poderão coincidir com o perfil de utilização de utilizadores, de tal modo que esta abordagem apenas espelha um conjunto de recomendações semelhantes a outro tipo de conteúdo já consumidos pelo utilizador, não acrescentando qualquer variedade nas recomendações sugeridas.

Esta técnica traduz-se num algoritmo dependente do seu próprio domínio dando total prioridade à análise dos atributos que compõem os itens com o fim de gerar previsões. As recomendações tem por base o uso das features extraídas do conteúdo de itens com o qual o utilizador terá de alguma forma interagido no passado. Desde modo, itens com elevada relação de semelhança com produtos comprados ou avaliados positivamente serão recomendados ao utilizador.

São utilizados diferentes tipos de de modelos para identificar as semelhanças entre artigos com o objectivo de produzir recomendações com elevado interesse para o utilizador. Com o objectivo de modelar a relação entre diferentes artigos dentro de um contexto, a nível algorítmico, podem ser utilizados modelos de espaço vectorial (Term Frequency Inverse Document Frequency) ou probabilísticos, tais como Naive Bayes Classifier, árvores de decisão e até redes neuronais, beneficiando assim da sua aprendizagem através de análise estatística ou de técnicas de machine learning

Para este tipo de recomendação ter sucesso é necessário que os produtos estejam bem estruturados para serem fáceis de analisar. Outros grandes obstáculos a este paradigma são a falta de informação sobre os utilizadores e a sua sobre-especialização. Este último ocorre quando estamos perante um catalogo de produtos variados, mas por exemplo, o utilizador apenas tinha comprado um produto e deste modo as recomendações estariam focadas em encontrar o produto mais aproximado da sua única compra, ignorando o restante catalogo da loja.

3.2.3 Sistemas híbridos

Esta técnica combina diferentes estratégias de recomendação e são tentativas de assegurar uma melhor optimização ao nível do sistema, evitando fatores limitados associados à utilização de algoritmos de recomendação mais puros. Esta abordagem tem assim que colmatar as falhas de determinado algoritmo com o uso de outro. A mistura é feita através da implementação independente das múltiplas técnicas, havendo apenas interferência na fase de cruzamento de resultados. Existem diversas técnicas de combinação de resultados, sendo as mais populares:

- Weighted - onde os resultados dos diferentes sistemas são combinados num só.
- Switching - onde o sistema adequa a técnica de recomendação ao contexto inserido.
- Mixed - quando o utilizador recebe os resultados de todos os sistemas presentes.
- Feature combination - quando dados de fonte de vários sistemas são combinados e direcionados para um único sistema de recomendação.
- Cascade - quando um sistema de recomendação melhora os resultados de outro sistema.
- Feature augmentation - no caso em que o resultado do sistema de recomendação é utilizado apenas como fonte para o algoritmo de outro sistema

3.3 Problema e seus desafios

Existem vários problemas que podem ser apresentados nos sistemas de recomendação. Os dois mais comuns são o Cold Start e Data sparcity:

- Cold Start - Este problema é fortemente tratado na literatura [12] e está relacionado com recomendações para novos utilizadores ou novos artigos. No caso de novos utilizadores o sistema não tem qualquer informação sobre as preferências do mesmo e por isso falha quando tenta recomendar algo para esse novo utilizador. No caso da entrada de novos artigos não temos qualquer avaliação dada pelos utilizadores, não permitindo saber a quais desses deve ser recomendado determinado artigo. Para aliviar este problema os autores do artigo [13] usaram um modelo probabilístico para extrair as latent features dos items.

  Há ainda a considerar o problema que ocorre cada vez que se forma um novo grupo de utilizadores, não tendo deste modo ratings suficientes para fazer recomendações para esses utilizadores.

- Data sparcity - Este problema ocorre pelo facto dos utilizadores normalmente avaliarem um número muito reduzido de itens, principalmente quando estamos na presença de catálogos muito extensos. Este fator resulta numa matriz user-item com informação insuficiente para identificar utilizadores ou itens similares, criando um impacto negativo na qualidade das recomendações. Data sparsity é predominante em sistemas de recomendação colaborativos que dependem do Feedback dos utilizadores mais "próximos"para gerar recomendações.

3.4 Principais desafios em técnicas de filtragem colaborativa

Problemas com este tipo de sistemas ocorrem quando um utilizador tem um gosto muito próprio, não permitindo desse modo encontrar outros utilizadores que partilhem os mesmos gostos. Por outro lado, a abordagem baseada em conteúdo pode ser criada com base nos perfis de usuário e itens. Os itens podem ser recomendados com base nas opções anteriores não existindo a preocupação com os gostos mais específicos de determinado utilizador. Apesar de ser um sistema muito popular, outros problemas poderão ser enfrentados aquando da sua implementação :

- Cold start - Este problema ocorre tanto para artigos como utilizadores. Quando o utilizador faz login no sistema, este não tem qualquer informação sobre os novos utilizadores, por isso não o podemos comparar com os outros utilizadores ficando assim impossibilitado de gerar uma recomendação. Quando um artigo é adicionado, como não há qualquer feedback por parte de outro utilizador, não pode ser recomendado. Por exemplo, para um novo utilizador no facebook é difícil fazer uma recomendação uma vez que ainda não tem amigos no sistema.

- Elevado custo para encontrar o melhor vizinho - É necessário encontrar os melhores vizinhos para um novo utilizador numa filtragem colaborativa. Esperamos encontrar um outro utilizador que seja muito próximo. É necessário considerar todos os ratings atribuídos a um artigo. Todas essas considerações tem um custo elevado.

- Problema da ovelha negra - Muitas vezes é fácil encontrar utilizadores que partilham os mesmos likes e dislikes mas pode haver utilizadores que não partilham um número de likes/dislikes suficientes. As estes chamamos ovelha negra por não ser possível dar recomendações significativas ou relevantes.
- Escassez de dados - Utilizadores não conseguem avaliar todos os artigos. Existe por isso um larga quantidade de artigos aos quais nunca é atribuído um rating.
- Escalabilidade - Algoritmos de filtragem colaborativa podem necessitar procurar milhões de utilizadores.
- Qualidade da recomendações - As recomendações devem ser boas, caso contrário os utilizadores poderão perder confiança no site onde estão ou onde fizeram uma compra.

3.5 Principal desafio em técnicas de filtragem baseado em conteúdo

Sobrespecialização ocorre quando não há variedade nas recomendações. O utilizador vai apenas receber artigos que são muito semelhantes aos que foram previamente comprados ou avaliados. Este problema é especialmente importante e deve ser considerado uma vez que em grande parte das vezes, um utilizador não vai comprar um item muito parecido ao que tinha adquirido anteriormente. Este pode também ser conhecido como serendipity problem.

Capítulo 4

Estado da arte

4.1 A abordagem Deep learning em sistemas de recomendação

A utilização de técnicas Deep learning tem sido a escolha de eleição dos investigadores que trabalham com Sistemas de recomendação. Com o crescimento do interesse em técnicas associadas a machine learning resultou numa grande dificuldade ao identificar o estado de arte no momento.

Nos últimos anos, deep learning tornou-se dominante no domínio dos sistemas de recomendação. Novos métodos foram propostos para uma variedade de configurações e de tarefas feita pelo algoritmo, incluindo as top-n recomendações baseadas no perfil longo prazo do utilizador ou para recomendações baseadas na sessão. Dado o crescente interesse em machine learning em geral, seria de esperar um progresso substancial que resulta desses trabalhos. No entanto, indicação existente em outras áreas de machine learning que obtiveram sucesso, medidas com o valor de accuracy sobre os modelos existentes, nem sempre tiveram o impacto esperado.

Lin [14], analisou as duas mais recentes abordagens no campo da aquisição de informação que foram publicados em conferencias reconhecidas. A sua análise revela que estes novos métodos não superam significativamente as baselines atuais quando estes são cuidadosamente parametrizados.

No contexto dos sistemas de recomendação, uma análise mais aprofundada mostra que mesmo os métodos neuronais mais recentes para recomendações baseadas na sessão podem, na maioria dos casos, ser superados por métodos simples como por exemplo, técnicas relacionadas com o vizinho mais próximo. Vários factores contribuem para este fenómeno, como:

- Baselines fracas.
- O estabelecimento de métodos frágeis como novas baselines
- Dificuldade em reproduzir os resultados de artigos publicados.

4.1.1 Abordagens deep learning mais recentes

Dacrema et al. [15] analisaram os 18 melhores algoritmos, que foram apresentados nas conferencias mais importantes sobre este tema. Dos algoritmos escolhidos apenas 7 foram reproduzidos com pouco esforço. No entanto, 6 dos algoritmos reproduzidos pelo autores foram superados com a aplicação de algumas heurísticas relativamente simples baseadas no vizinho mais próximo ou baseadas em grafos. O restante superou as baselines mas não conseguiu superar um método linear não neuronal bem parametrizado.

Os autores [15] definiram as seguintes baselines para compararem os métodos:

TopPopular: um método não personalizado que recomenda o item mais popular. A Popularidade é medida pelo número de avaliações implícitas ou explícitas.

ItemKNN: uma abordagem tradicional, baseada no vizinho mais próximo (KNN) e semelhanças item-item. Foi usada a semelhança cosine s_{ij} :

$[s_{ij} = \frac{r_i \bullet r_j}{|r_i| \cdot |r_j| + h}]$ com os vectores $r_i, r_j \in R^{|U|}$

a representarem os ratings implícitos de um utilizador para os itens i e j, respectivamente, e $|U|$ o número de utilizadores.

UserKNN:Um método baseado na vizinhança utilizando semelhanças utilizador-utilizador. Os hiperparâmetros são os mesmos que foram usados para ItemKNN.

ItemKNN-CBF: Um método que usa uma abordagem baseada em conteúdo com a semelhança a ser calculada usando atributos do item.

$$s_{ij} = \frac{f_i \bullet f_j}{\|f_i\| * \|f_j\| + h}$$

onde os vectores

$$f_i, f_j$$

descrevem as features dos itens i e j, respectivamente.

ItemKNN-CFCBF: Um algoritmo híbrido CF + CFB baseado nas similaridades item-item. A similaridade é calculada após serem concatenados os vectores das features e dos ratings. Os hiperparâmetros são os mesmos usados para ItemKNN, com a adição de um parâmetro w que pesa o conteúdo das features com os ratings que lhes foram atribuídos.

$$s_{ij} \sum_{v} p_{jv} p_{vi}$$

4.1.2 Collaborative Memory Network (CMN)

Apresentado na SIGIR '18, combina redes de memória e mecanismos com factores latentes e modelos de vizinhança. Para avaliar esta abordagem os autores compararam com diferentes Matrix factorization e com abordagens de recomendações neuronais tal como foi o caso do algoritmo ItemKNN. Os 3 datasets utilizados para avaliar: Epinions, CiteULike-a e Pinterest. Os hiperparâmetros óptimos para os métodos

propostos são apresentados mas não há informação sobre como ocorreu o tuning das baselines. Hit rate e NDCG (Normalized Discounted Cumulative Gain) foram as medidas de desempenho utilizando o processo leave-one-out. Os resultados mostraram que CMNs superam todas as baselines em qualquer das métricas utilizadas.

4.1.3 Metapath based Context for RECommendation(MCRec)

Hu et al., apresentaram na KDD (Knowledge Discovery and Data Mining) e o mesmo tira vantagem da informação auxiliar como é o género de filme para as top-n recomendações. Do ponto de vista técnico, os autores propõem uma técnica de amostragem prioritária para selecionar instâncias de caminhos de qualidade mais alta e propõem um novo mecanismo de co-atenção para melhorar as representações do contexto, utilizadores e itens baseados em meta-caminhos. Os autores apresentam 4 variantes do método apresentado contra uma variedade de modelos com diferente complexidade em 3 datasets de dimensão reduzida, (MovieLens100k, LastFm e Yelp). A avaliação é feita com a criação de 80/20 treino-teste aleatórios e avaliando-o 10 vezes. O processo de avaliação pode ser reproduzido [16]. O tradicional método ItemKNN quando parametrizado correctamente consegue superar MCRec em qualquer das métricas utilizadas.

4.1.4 Collaborative Variational Autoencoder (CVAE)

Apresentado na KDD '18, [17] é uma técnica híbrida que considera tanto o conteúdo como os ratings. O modelo aprende as deep latent representations entre itens e utilizadores tanto para o conteúdo como para os ratings. Estes métodos são avaliados com dois datasets relativamente pequenos, CiteULike (135k e 205k interações). Para estes dois datasets, uma versão sparse and dense foi testada. As baselines em [17] incluem 3 modelos deep learning tal como um Collaborative Topic Regression (CTR). Os parâmetros para cada método são tuned baseados no set de validação. Recall em listas de diferentes tamanhos (50 até 300) foi a medida de avaliação usada. Amostras aleatórias de divisão treino-teste são aplicadas e medidas 5 vezes.

Dacrema et al. [15] comparou os resultados, para listas pequenas com cerca de 50 de comprimento, a maioria dos algoritmos CF puros superaram o CVAE com o dataset escolhido (CiteULike). Em listas com comprimento superior, o método híbrido ItemKNN-CFCBF foi o que apresentou melhores resultados. Resultados semelhantes foram obtidos quando utilizado o dataset escasso CiteULike-t. Em geral, em lista com comprimento 50, ItemKNN-CFCBF superou CVAE em todas as configurações testadas. Apenas em listas com comprimentos superiores (100 ou mais), CVAE foi capaz de superar em ambos os datasets. No global, CVAE apenas superou as baselines em certas configurações, sendo elas comparavelmente longas e poucos comuns.

4.1.5 Collaborative Deep Learning (CDL)

[18] O acima abordado, CVAE, considera o muitas vezes citado CDL, apresentado na KDD'15 (Knowledge Discovery and Data Mining), como uma da suas baselines e os autores usam o mesmo processo de avaliação e CiteULike datasets. CDL é um modelo feedforward probabilístico para aprendizagem conjunta de stacked denoising autoencoders (SDAE) e filtragem colaborativa. Aplica técnicas de deep learning para em conjunto aprender a deep representation do conteúdo da informação e da informação colaborativa. A avaliação do CDL mostrou que é favorável, em particular, quando comparado com o amplamente referenciado método CTR (Collaborative topic modeling) [19], especialmente em situações em que estamos na presença de sparse data. Quando comparados os resultados para CVAE e CDL, [15] conclui que este novo CVAE é melhor que CDL em todos os aspectos e configurações, mostrando que houve progresso. No entanto, nenhum dos métodos consegue ser superior na maioria dos casos a qualquer das baselines mais simples.

4.1.6 Neural Collaborative Filtering NCF

Apresentado na WWW'17 (International Conference on World Wide Web), [20], generaliza Matrix Factorization substituindo o produto interno por uma arquitetura neuronal que consegue aprender uma função arbitrária através dos dados. O método híbrido proposto (NeuMF) foi avaliado com dois datasets (MovieLens1M e Pinterest), contendo respetivamente 1 milhão e 1.5 milhões de interações. Os seus resultados mostram que NeuMF é favorável sobre, por exemplo, modelos que usam Matrix Factorization quando usam hit rate e NDCG (Normalized Discounted Cumulative Gain) para avaliar o modelos com listas de tamanhos diferentes até 10.

Avaliados os resultados da experiência foi possível concluir que o dataset Pinterest em duas das baselines personalizadas foram superiores ao NeuMF em todas a métricas. Para o MovieLens, NeuMF superou claramente as baselines definidas.

4.1.7 Spectral Collaborative Filtering (SpectralCF)

Apresentado na RecSys'18, foi desenhado especificamente para endereçar o problema do cold-start e é baseado em conceitos de Teoria espectral de grafos. As recomendação são baseadas em grafos bipartidos de relacionamento item-utilizador e numa nova operação de convolução, que é usado para fazer recomendações diretamente no domínio espectral. Este método foi avaliado com 3 datasets públicos (MovieLens1M, HetRec e Amazon Instant Video) e comparados com uma variedade de métodos.

A avaliação foi baseada em set's de treino-teste aleatório com uma divisão 80/20 utilizando Recall e Mean Average Precision (MAP) com diferentes cutoffs.

Os autores, Lin obtiveram os seguintes resultados: para os HetRec e Amazon Instant Videos datasets, todas as baselines superaram SpectralCF em todas as avaliações, no entanto para o dataset MovieLens, SpectralCF é superior às baselines definidas por larga margem.

4.1.8 Variational Autoenconders for Collaborative Filtering (Multi-VAE)

Apresentado na WWW '18 pelos autores Liang et al. , Multi-VAE é um método de filtragem colaborativa para feedback implícito baseado em auto-encoders variacionais. Os autores introduzem um modelo generativo com probabilidade multinomial. Propõem um parâmetro de regularização diferente para o learning objective, e usa inferência bayesiana para estimar os parâmetros. Este método foi avaliado em 3 datasets binarizados que continham originalmente ratings de filmes ou número de vezes que uma música foi tocada. As baselines nesta experiência incluem métodos matrix factorization de 2008, um modelo linear de 2011 e um método mais recente que usa métodos com base neuronal. O método proposto leva a uma accuracy nos resultados que é aproximadamente 3% superior às baselines determinadas a nível da Recall e NDCG. Os autores em [14], descobriram que o método proposto supera consistentemente as melhores baselines definidas pelos mesmos. Os resultados obtidos foram 10% a 20 % superiores. Este foi o único método examinado na literatura que, sendo mais complexo, foi superior por larga margem.

4.1.9 A mais-valia de métodos neuronais mais complexos

Segundo os autores [14] e a sua análise feita aos melhores métodos propostos nos últimos anos que utilizam algum tipo de rede neuronal, não fica claro a vantagem de usar métodos computacionalmente complexos. Concluem ainda que o nível de progresso atingido no campo dos métodos neuronais neste contexto não mostra qualquer vantagem clara nesta abordagem.

4.2 Utilização de deep learning em recomendações no contexto e-commerce

4.2.1 Content based system

A seguir é feita uma breve apresentação de alguns dos artigos mais relevantes no contexto e-commerce.

Zheng et al. usou as reviews para aprender quais as propriedades e o comportamento do utilizador utilizando uma rede chamada Deep Cooperative Neural Network. O modelo usa também uma shared layer para ligar as features de cada item com o comportamento dos utilizadores. O modelo é comparado com 5 baselines, nomeadamente, Matrix factorization, LDA, Collaborative tópicos refression , Hidden Factor as Topic e Collaborative deep learning, utilizando 3 datasets reais: Yelp reviews, Amazon reviews and Beer reviews. Este modelo superou todas as baselines para todos os datasets.

4.2.2 Collaborative filtering

[23] Foi um dos primeiros trabalhos que ofereceu uma framework que incorporava características deep learning em modelos baseados em filtragem colaborativa tal como é o caso de matrix factorization. O

modelo é comparado com diversas abordagens colaborativas que usam matrix factorization e mostra aumentos de performance em 4 datasets reais: MovieLens-100k, MovieLens-1M, Book-Crossing e Advertising dataset

[24] Desenvolveram um auto-encoder capaz de lidar com a aprendizagem não supervisionada de features e de gerar perfis de utilizador a partir do rating utilizador-artigo para aprimorar a abordagem colaborativa. Os resultados da adição de deep learning às abordagens convencionais mostram, estatisticamente, um aumento significativo da previsão do rating do utilizador utilizando o dataset yelp.com. Foi utilizado no contexto e-commerce.

[25] Mostra que uma filtragem colaborativa pode ser convertida num problema de previsão sequencial e por isso redes neuronais recorrentes podem ser muito úteis. Neste caso em particular, as LSTM's foram aplicadas a um problema de filtragem colaborativa e depois comparadas com o k-nearest neighbor e com Matrix factorization em dois datasets: MovieLens e Netflix. A comparação do desempenho mostra a eficácia em usar modelos deep learning quando comparado com o estado de arte de modelos para filtragem colaborativa.

Para recomendações baseadas na sessão do utilizador existem diversos trabalhos que tem usado redes recorrentes para aumentarem a eficácia de uma recomendação. Hidasi et al. usou apenas RNN para recomendação de sessões curtas ao invés de sessões longas. Eles mostraram que matrix factorization não são adequadas para fazer recomendações baseadas na sessão do utilizador. A avaliação experimental em e-commerce clickstream data e em dataset de serviços semelhantes ao YouTube mostram grande vantagem do uso de RNN em relação às abordagens utilizando matrix factorization. [27] usa features do artigo como imagens ou texto para melhorar as recomendações RNN baseados na sessão. Eles introduzem o conceito de RNN paralela que modela aspetos do artigos nomeadamente o texto e imagem. Quando comparado com uma simples RNN com item-kNN, as features com base RNN tem uma melhoria no desempenho em datasets semelhantes ao serviço de video do YouTube.

[28] Mostrou que a combinação de RNN com kNN é muito eficaz melhorando a accuracy numa aplicação e-commerce. A avaliação da performance é feita usando datasets que são públicos como é o caso competição do TMall entre outras playlists da last.fm, artofmixing.org e 8tracks.com.

[29] Propôs um modelo que incorporasse o dwell time (tempo que um utilizador utiliza a analisar determinado artigo) para melhorar a accuracy das recomendações baseadas na sessão.

[30] Desenvolveu um modelo baseado em redes neuronais convolucionais para incorporar os informação meta-data de utilizadores ou artigos para a abordagem com matrix factorization. O modelo é avaliado com os datasets movie-lens e Amazon review e comparado com o estado da arte de filtragem colaborativa.

[31] Propôs um denoising Auto-encoders baseado numa filtragem colaborativa. O modelo é apresentado como uma framework genérica para todas as abordagens colaborativas oferecendo flexibilidade no tuning do modelo. Este modelo quando comparado supera as ItemPop, ItemCF, Matrix factorization, BPR e FISM nos datasets MovieLens, Yelp e Netflix.

[32] Utiliza redes neuronais para melhorar a short-term prediction convertendo um problema de filtragem colaborativa num problema de previsão de sequência.

4.2.3 Hybrid system

Os sistemas de recomendação híbridos combinam diferentes estratégias de recomendação de forma a tornarem benéfica a sua combinação. A maioria dos estudos dizem que é usado maioritariamente um sistema colaborativo a que se junta outro sistema, sendo determinado o peso que cada um dos métodos terá. O cold-start e data sparsity são os problemas mais comuns e abordados em grande parte dos estudos enquanto que filmes e dataset de filmes são também muitos usados pelos autores desses estudos. A maioria desses métodos são avaliados quando comparados com métodos similares utilizando a accuracy. Encontrar avaliações mais orientadas ao utilizador continua a ser um desafio. Alem disso, novos desafios vão sendo identificados, tais como a variação do contexto do utilizador, a evolução dos gostos do utilizador ou a previsão de recomendações cruzadas. Sistemas híbridos apresentam uma boa base para responder explorando novas oportunidades como são os casos das recomendações baseadas no contexto, envolvendo algoritmos híbridos paralelos ou o processamento de datasets de grande dimensão.

Capítulo 5

AutoRecRNN - Sistema de recomendação

Neste capitulo apresenta-se o pipeline do sistema de recomendação desenvolvido neste trabalho. O sistema de recomendação denomina-se de AutoRecRNN.

Um Pipeline define um conjunto de etapas que devem ser executadas para a obtenção de um objectivo, deste caso uma recomendação. Uma das grandes vantagens do uso de pipelines, é a sua possível automatização permitindo assim uma melhoria continua do resultados obtidos à medida que a qualidade de informação aumenta.

Um Pipeline ML define num conjunto de etapas para treinar o modelo de ML. O modelo é iterativo e os passos são repetidos para que se consiga melhorar a precisão do modelo para obter um algoritmo de sucesso. O pipeline ML opera permitindo que uma sequência de dados seja transformada e correlacionada num modelo que pode ser testado e avaliado para arquivar um resultado (negativo ou positivo).

Neste trabalho conceptualiza-se um pipeline que engloba o típico pipeline ML, complementando-se com uma API REST que facilita a obtenção de recomendações e fortalece o sistema em termos de segurança. O pipeline proposto está representado na Figura 5.1.

Como se pode ver na figura 5.1, o pipeline de recomendações é constituído por 4 passos principais, aquisição, processamento, treino e um último onde são geradas as recomendações. A figura apresenta, em paralelo e com a mesma cor da respetiva etapa, as ferramentas usadas no desenvolvimento de cada uma das etapas do pipeline.

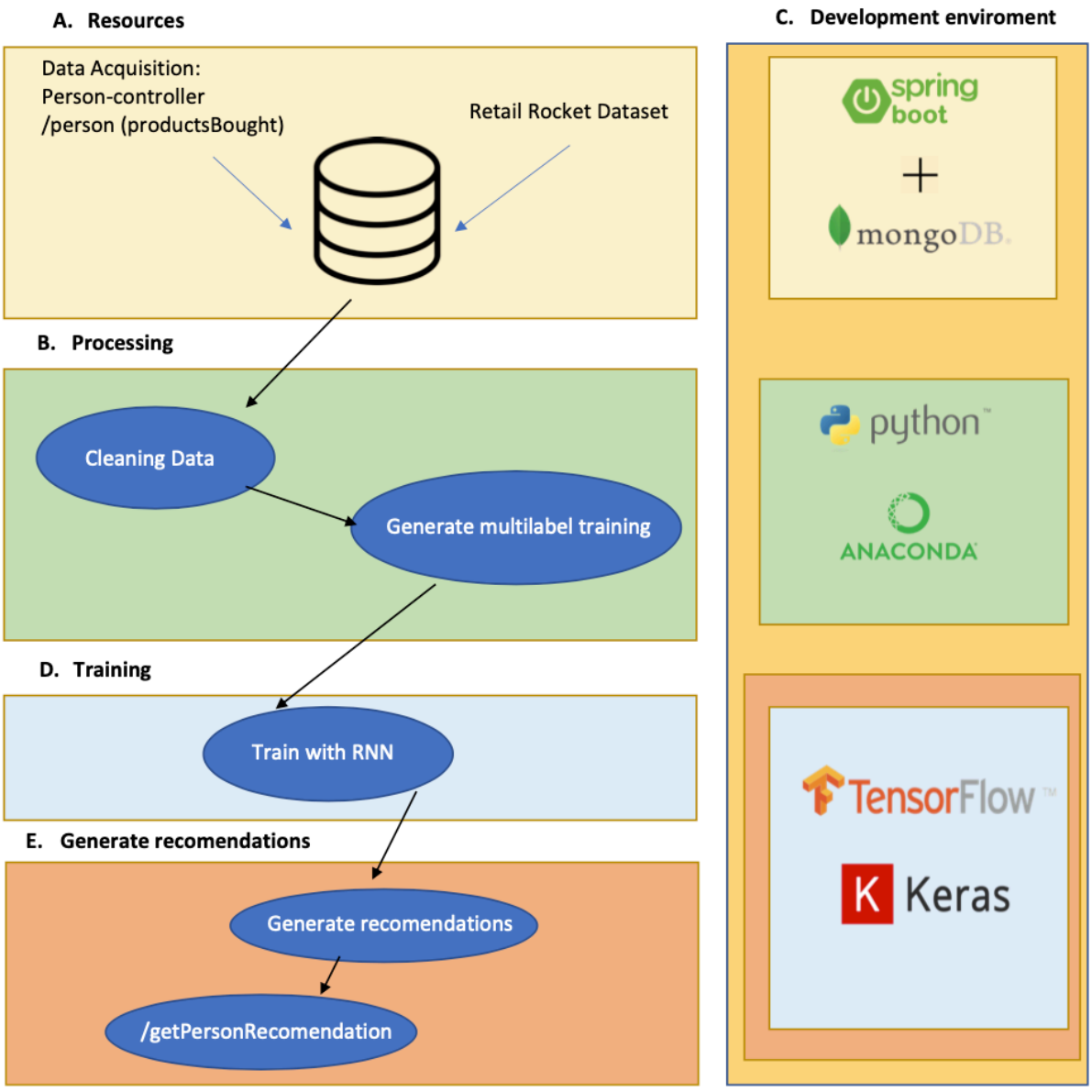


Figura 5.1: Pipeline completo.

Este sistema apresentado foi conceptualizado para ser um sistema recomendação muito simples de integrar com outras aplicações. Por forma a simplificar os processo de geração de recomendações foi desenvolvido uma API REST que devolve, em formato JSON uma série de recomendações. Quando um cliente estiver disposto a usar este sistema de recomendação apenas tem que usar a API Rest disponível para introduzir os dados de compras dos cliente. A partir desses dados e com o processo de treino que ocorrerá será possível obter recomendações personalizadas para cada um dos seus cliente mediante um histórico de compras.

5.1 Aquisição da informação

Quanto à aquisição de informação, representado por A na Figura 5.1, o sistema foi pensado para ser o mais genérico possível. Assim o pipeline de aquisição de informação poderá ligar-se então a qualquer base

de dados desde que esta contenha informação sobre as compras dos utilizadores. Esta informação será depois codificada para se treinar o modelo de dados. No entanto, e por causa de lei de protecção de dados não foi usada nenhuma base de dados onde continha informação do cliente. A alternativa encontrada foi procurar um dataset que tivesse as informações enunciadas acima (compras dos utilizadores ao longo do tempo). Para o caso de estudo foi então utilizado o dataset Retail Rocket Dataset onde contem dados de compras dum retail localizado no Reino Unido.

5.2 Processamento

O processamento de dados, representada por B na Figura 5.1, é representado por duas sub-etapas.

- Limpeza dos dados (Cleaning) - Este processo conceptualmente passa por retirar todos os outliers que influenciariam negativamente o processo de treino.
- Codificação dos dados para treino sob a forma de um problema multi-label (Generate multilabel training) - Este processo de codificação dos dados culminará com um dataset pronto a ser usado para treinar o modelo de dados. Para este processo temos que tomar em atenção que tratou de um problema multilabel. Este tipo de problema está detalhado na secção 7.2.1.

5.3 Treino

Na etapa de treino, representada por D. Trainning na Figura 5.1, o algoritmo de aprendizagem é alimentado com dados de entradas e suas respectivas saídas desejadas são avaliadas. O algoritmo de aprendizagem usado foi o algoritmo de Redes Neuronais Recorrentes (Train with RNN).

Esta etapa é fundamental para preparar a máquina e para aprimorar as habilidades de previsão das recomendações.

A etapa de treino pode ser considerada como a etapa principal (à qual se deve o nome) de machine learning.

5.4 Geração de recomendações

A etapa de Geração de recomendações, representada por E. Generate Recommendations na Figura 5.1, geram-se e avaliam-se as recomendações. Esta etapa permite testar se, após o treino, a máquina está suficientemente capacitada. Com base na configuração definida anteriormente, faz-se a avaliação dos resultados obtidos mediante de dados que a máquina não tinha, ou seja, que a maquina foi capaz de deduzir. Sendo assim, a ideia é que a avaliação seja uma representação de como a máquina se comportará num contexto real. Esta etapa é constituída por duas partes principais:

- Generate Recommendations - gera as recomendações baseado no modelo treinado.

- Get Person Recommendation - selecciona as recomendações especificas para um utilizador.

5.5 Ambiente de desenvolvimento

O ambiente de desenvolvimento, representado por C. Development environment na Figura 5.1, apresenta as frameworks usadas no desenvolvimento de cada uma das etapas do pipeline enumeradas anteriormente. As frameworks usadas em cada etapa está representada pela mesma cor que a etapa correspondente. As frameworks e a forma como elas se integram está detalhada no capitulo 6.

Capítulo 6

Desenvolvimento de API

Neste projecto foi desenvolvida uma API (Application Programming Interface) que permite simplificar o processo de geração de recomendações e dá uma maior segurança ao sistema desenvolvido.

Uma API pode ser definida como um conjunto de definições e protocolos que pode ser usado no desenvolvimento e na integração de aplicações. A API pode ser vista como uma espécie de contrato entre o fornecedor de serviços e o utilizador que recorre ao uso desses serviços. A API define uma interface, através da qual os programadores (ou outras aplicações) interagem com os dados. Uma API bem pensada e bem conseguida deve ser fácil de usar. A boa ou má experiência que os programadores têm ao usar uma API é a métrica mais importante para medir a qualidade das APIs.

REST (REpresentational State Transfer) é um estilo de arquitetura para sistemas distribuídos que foi apresentado pela primeira vez por Roy Fielding em 2000 [33]. Esta arquitetura baseia-se num conjunto de princípios, nomeadamente:

- É uma arquitetura Cliente-servidor, ou seja, separa a interface com o utilizador da forma como os dados estão armazenados. Assim, aumenta a portabilidade e escalabilidade.
- Cada pedido do cliente para o servidor deve conter todas as informações necessárias para entender ao pedido.
- Mantém uma Interface uniforme.

Uma API REST significa que a API está em conformidade com as restrições da arquitetura REST. Uma API organizada de acordo com o estilo de arquitetura REST implica que os serviços são identificados pelo nome do recurso manipulado, e os métodos de acesso têm uma semântica uniforme para todos os recursos, sendo associado HTTP aos verbos [33]:

- POST, valida os parâmetros e cria o objeto de recurso e objetos de outros recursos associados não acessíveis diretamente através do API;
- GET, obtém um ou vários objetos de recurso, dependendo dos parâmetros;
- PUT, atualiza um objeto de recurso;
- DELETE, exclui um objeto de recurso.

A API aqui apresentada foi desenvolvida em Spring usando a linguagem Java ao qual liga um micro-serviço escrito em Python onde contem toda a lógica desde o treino, gestão dos processos ligados ao treino do modelo de dados até às recomendações propriamente ditas. Dentro da aplicação Spring estão ainda presentes alguns algoritmos que geram recomendações tanto baseadas em conteúdo como colaborativas. Estes foram devolvidos para a aplicação ser capaz de gerar recomendações mesmo quando um modelo de dados ainda não se encontra presente no micro-serviço.

Esta API não só é adaptável a diversos modelos machine learning que utilizem os mesmo campos para gerar recomendações, como prevê ainda a possibilidade de re-treinar o modelo de dados ao longo do tempo.

A API é composta por dois serviços que seguem os princípios da arquitectura micro-serviços. Esta API apresenta ainda endpoints que apresentam recomendações baseadas em algoritmos colaborativos tal como algoritmos que se baseiam no conteúdo. Para a persistência dos dados foi utilizada uma base de dados MongoDB.

6.1 Persistência dos dados

Como já foi referido anteriormente, os dados são armazenados numa Base de dados MongoDB. MongoDB é uma base de dados não relacional (ou NoSQL), open-source, muito flexível e escalável. É uma base de dados orientada ao documento, ou seja, usa documentos JSON (JavaScript Object Notation) para armazenar os dados. Cada documento é constituído por uma chave (key) e o respetivo valor.

A base de dados é constituída por duas coleções: utilizadores e produtos. A constituição da coleção utilizadores pode ser vista na Figura 6.1. Nesta coleção, além de se armazenar a informação do utilizador (nome e email) são também identificados dos produtos comprado por este.

A constituição da coleção produto está representada na Figura 6.2. Nesta coleção são armazenadas as informações sobres as características dos produtos.

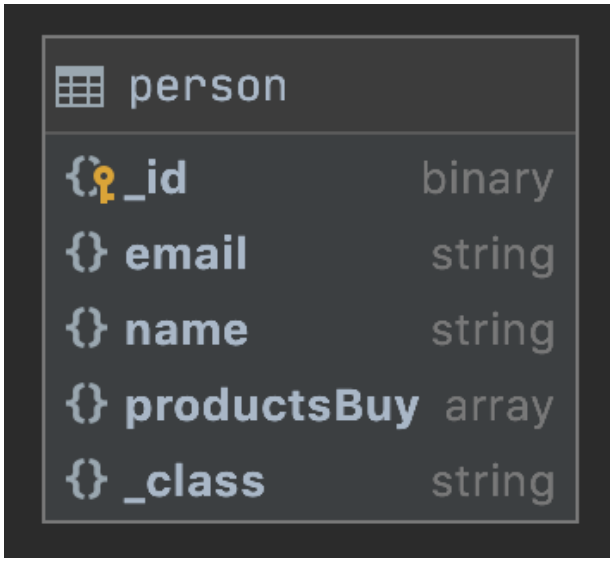


Figura 6.1: Coleção utilizadores

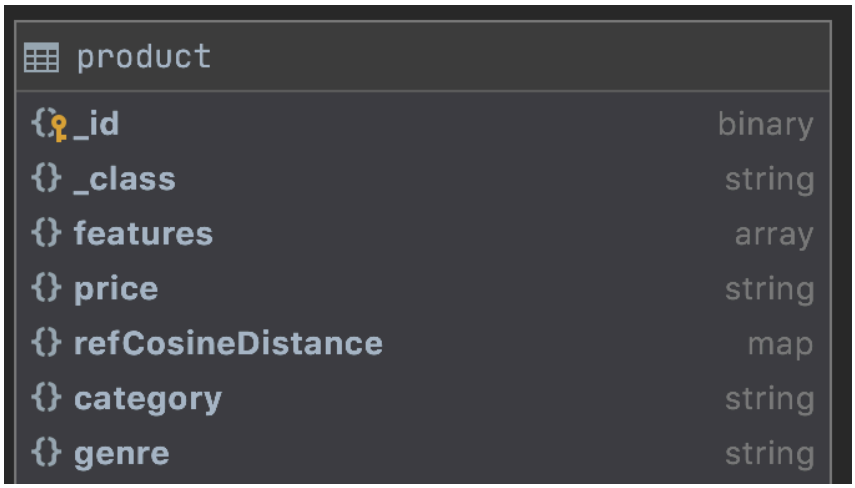


Figura 6.2: Coleção Produtos

6.2 Algoritmos de treino

6.2.1 Treino no modelo colaborativo

O processo de treino no modelo colaborativo passa por calcular a semelhança entre cada utilizador, como representado na Figura 6.3. Para isso foi utilizada similaridade de cosseno (Cosine similarity). Antes de calcular a similaridade foi codificada a relação utilizador-produto comprado utilizando a codificação hot-encoding. Este processo é depois aproveitada quando alguém pede uma recomendação ao sistema. Perante um N de recomendações pedidas, serão devolvidas a Y produtos comprados pelo utilizadores mais ”próximos”do utilizador em questão.

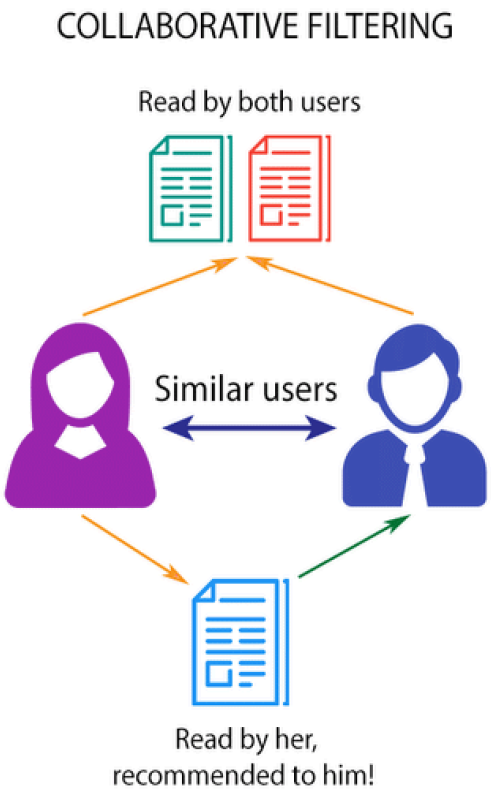


Figura 6.3: Filtragem colaborativa

6.2.2 Treino no modelo baseado em conteúdo

A forma como o treino se desenvolve no modelo baseado em conteúdo é muito semelhante ao apresentado anteriormente. Neste modelo a codificação é feita produto-features e com o treino obtemos relação de similaridade entre todos os produtos pela lista de features apresentada. Como se pode ver na Figura 6.4, quando é pedida a recomendação para um determinado produto, o processo de geração de recomendação passa pela procura dos produtos mais similares ao apresentado.

6.3 Endpoints fornecidos pela API

Os Endpoints (ou ponto de extremidade) são disponibilizados pela API e servem de pontos de comunicação entre os recursos internos e o cliente. Os Endpoints especificam onde as APIs podem aceder a recursos e ajudam a garantir o funcionamento adequado do software incorporado. O desempenho de uma API depende de sua capacidade de comunicar com sucesso com os endpoints da API.

Assim, a API aqui desenvolvida disponibiliza um conjunto de Endpoints que serão descritos em seguida.

O endpoint representado na Figura 6.5 permite-os introduzir dados dos utilizadores no sistema. Juntamente com cada utilizador é fornecido ao sistema os dados de todas as compras até então (6.6). Esta informação será utilizada para obtenção de recomendações.

O endpoint representado na Figura 6.7 foi pensado para a inserção de produtos ligados à moda, como

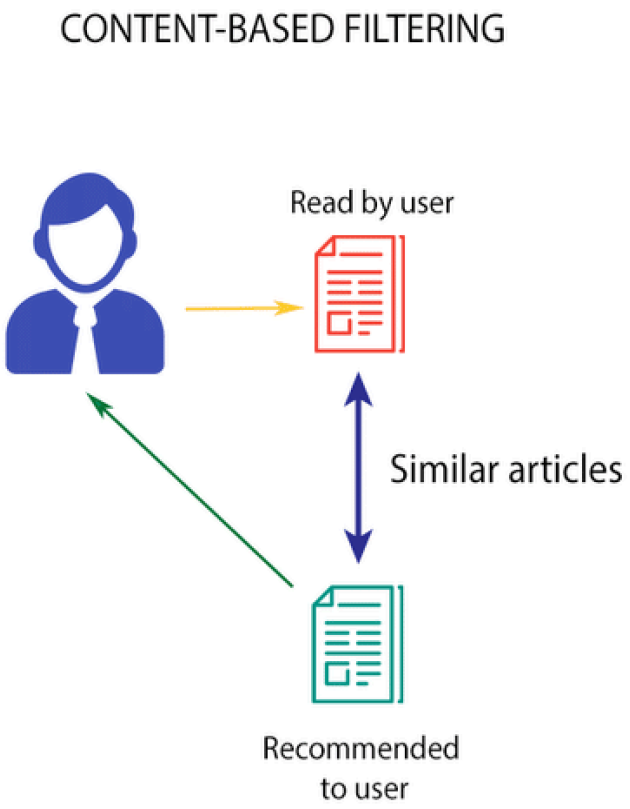


Figura 6.4: Modelo baseado em conteúdo

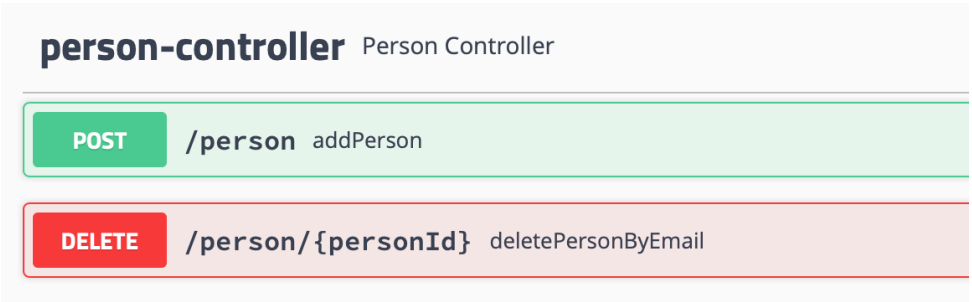


Figura 6.5: Endpoints para a adição de informação sobre o s utilizadores

```
public class PersonDto {
    private String name;
    private String email;
    private String country;
    private String invoiceNo;
    private String invoiceData;
    private List<String> productsBought;
```

Figura 6.6: Dto para a inserção de utilizadores no sistema

podemos observar pela informação ligada à composição do interior e exterior de cada peça (ver Figura 6.8). A recolha desta informação tem em vista uma recomendação baseada no conteúdo.

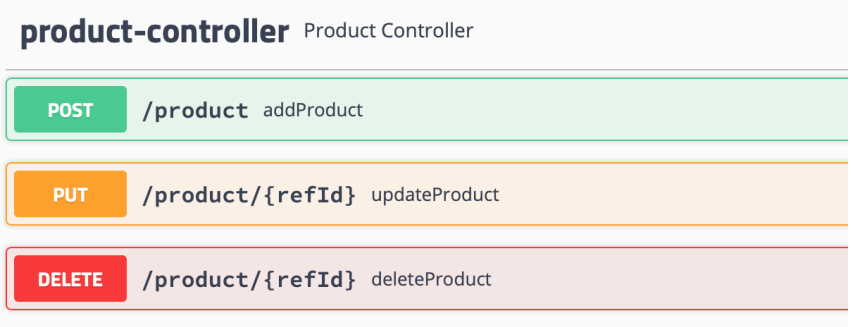


Figura 6.7: Endpoints para a adição de informação sobre um produto

```
public class ProductDto {

    private String ink;
    private String genre;
    private String price;
    private String category;
    private String reference;
    private List<String> features;
    private String shortDescription;
    private List<String> interiorComposition;
    private List<String> outsideComposition;
```

Figura 6.8: Dto para a inserção de informação sobre produtos no sistema

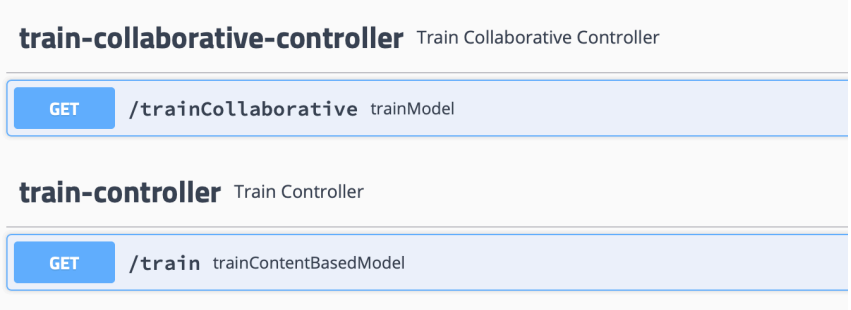


Figura 6.9: Endpoints que inicia o processo de treino no modelo colaborativo

O endpoint representado na Figura 6.9 permite dar inicio ao processo de treino do modelo colaborativo.

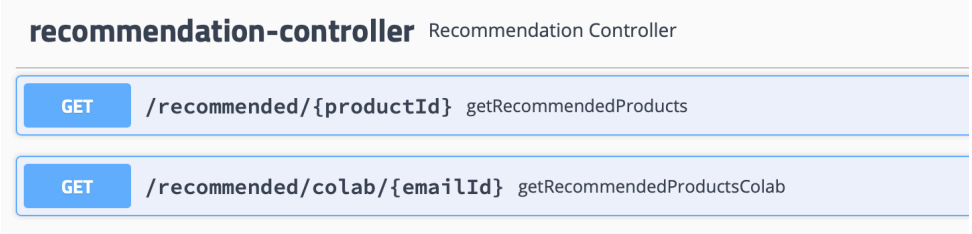


Figura 6.10: Endpoints relacionados com recomendações baseadas em conteúdo.

Os Endpoints relacionados com recomendações baseadas em conteúdo estão representados na Figura 6.10. Para a geração de uma recomendação baseada em conteúdo é fornecido o identificador do produto e serão retornado os 10 produtos mais "próximos".

Caso seja pretendida uma recomendação baseada no modelo colaborativo é fornecido o email do utilizador. Esse email é usado para procurar os utilizadores mais "próximos"(após o treino ter ocorrido). As recomendações que são retornada são baseadas nos produtos comprados pelo utilizadores mais próximos.

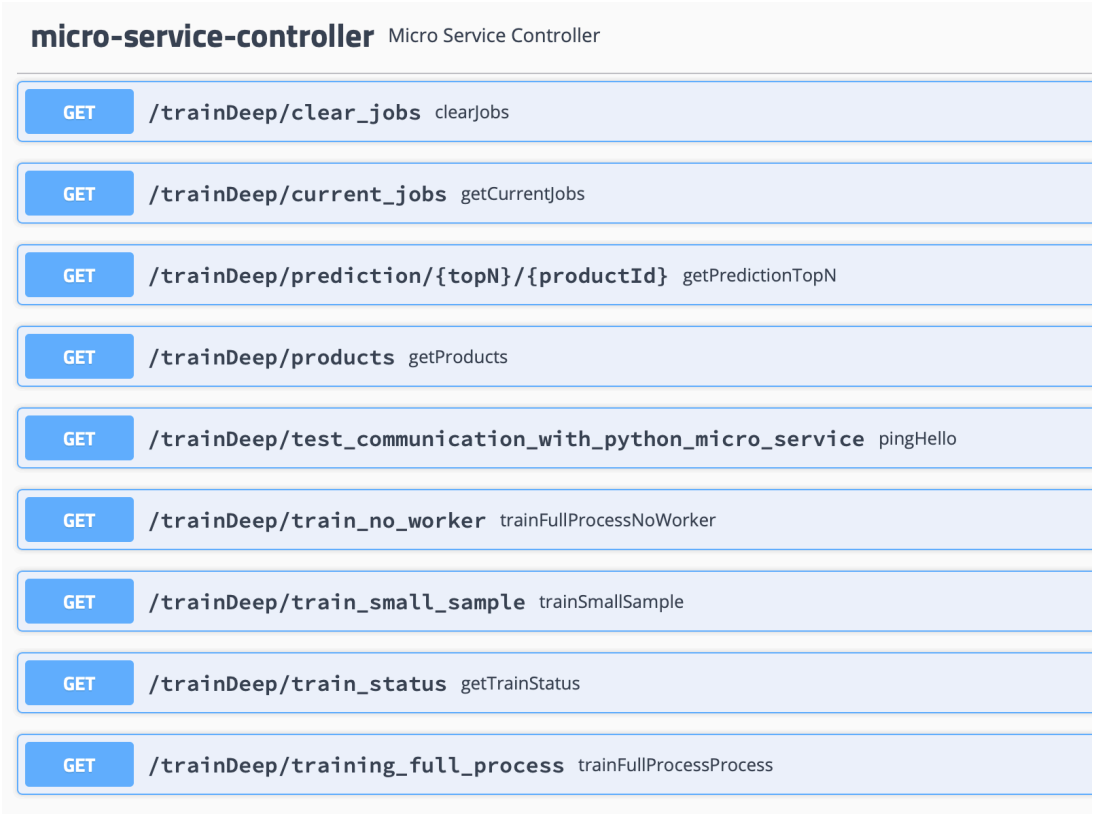


Figura 6.11: Endpoints relacionados com o micro-serviço escrito em python.

A Figura 6.11 representa os EndPoints que comunicam com o micro-serviço escrito em Python. Este código segue na anexo A.

Os endpoints presentes na figura 6.11 estão relacionados com o micro-serviço escrito em python e trata de todos o processo do modelo Machine Learning, desde o treino até a geração de recomendações que passam pelo mesmo. A adaptabilidade desta API deve ser salientada pois esta está preparado para que, com pequenos ajustes, possa tratar de todos os processos dos mais diferentes modelos ML.

Os endpoints disponibilizados são:

- clearJobs - Como o nome indica este endpoint elimina o processos existentes na queue de jobs a executar. Os principais jobs possíveis são: Enconding, training e re-training.
- currentJobs - Informação sobre que processos estão a decorrer.
- getPredictionsTopN - Este endpoint devolve as top N recomendações.

- trainNoWorked - Força o processo de treino no imediato em vez de colocar o processo na queue à espera da sua vez.
- trainSmallSample - Este endpoint é utilizado quando estamos perante um processo de treino muito demorado. Neste caso apenas é utilizador para o processo de treino um parta da informação disponível.
- trainStatus - Informação do processo de treino com a previsão de quando o processo terminará.
- trainingFullProcess - Cria um processo completo e insere-o na queue. Este processo inclui o processo de codificação e treino.

Capítulo 7

Caso de estudo

A procura por um dataset com os dados pretendidos foi um processo demorado, que levou a alguma indecisão sobre a ação a tomar. A empresa onde foi desenvolvido este projecto lida com um grande quantidade de dados. No entanto, por restrições relacionadas com a proteção de dados, esses dados não poderiam ser usados como caso de estudo. As alternativas à primeira opção seriam comprar dados ou utilizar algum dataset disponibilizado por alguma empresa. A opção de comprar os dados foi rapidamente excluída deixando-nos a opção final de utilizar dados disponibilizados por alguma empresa. Neste caso concreto foram usados dados disponibilizados por uma empresa de retalhista.

7.1 Materiais e análise dados

Para o caso de estudo foi utilizado um dataset que continha os dados das compras realizadas por uma empresa de retailer sediada no Reino Unido. Esses dados continham registos ao longo de um período de 8 meses. Cada registo contem informações tais como, InvoiceNo, StockCode, Description, Quantity ,InvoiceDate, UnitPrice, CustomerID ou Country.

- InvoiceNo - Referente ao número da fatura.
- StockCode - É referente ao código do produto.
- Quantity - Quantidade comprada de um determinado produto.
- UnitPrice - Preço unitário de cada produto.
- CustomerID - Identificação do utilizador no sistema.

- Country - Pais onde a compra foi efectuada.

Row ID	Invoic...	Stock...	Description	Quantity	InvoiceDate	UnitPri...	Custo...	Country
Row0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
Row1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
Row2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
Row3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
Row4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
Row5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
Row6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom
Row7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom
Row8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom
Row9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom
Row10	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.1	13047	United Kingdom
Row11	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.1	13047	United Kingdom
Row12	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12/1/2010 8:34	3.75	13047	United Kingdom
Row13	536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010 8:34	1.65	13047	United Kingdom
Row14	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12/1/2010 8:34	4.25	13047	United Kingdom
Row15	536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010 8:34	4.95	13047	United Kingdom
Row16	536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010 8:34	9.95	13047	United Kingdom

Figura 7.1: Amostra dos dados.

Após uma análise cuidada do dataset conclui-se que existem alguns aspectos que se podem observar.

- Número de clientes diferentes : 533
- Número de faturas diferentes : 718
- Número de países : 18
- Número de produtos diferentes : 1954

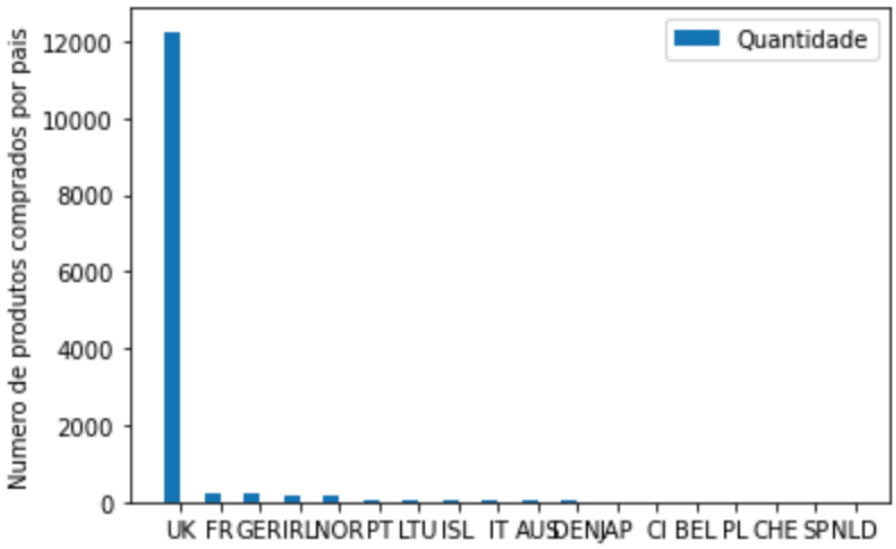


Figura 7.2: Relação entre produto e país de origem do comprador.

A figura 7.2, onde se representa a relação entre produto comprado e país de origem do comprador, deixa claro que a maioria dos compradores são originários do mesmo pais onde está fixada esta empresa de retail, neste caso, o Reino Unido.

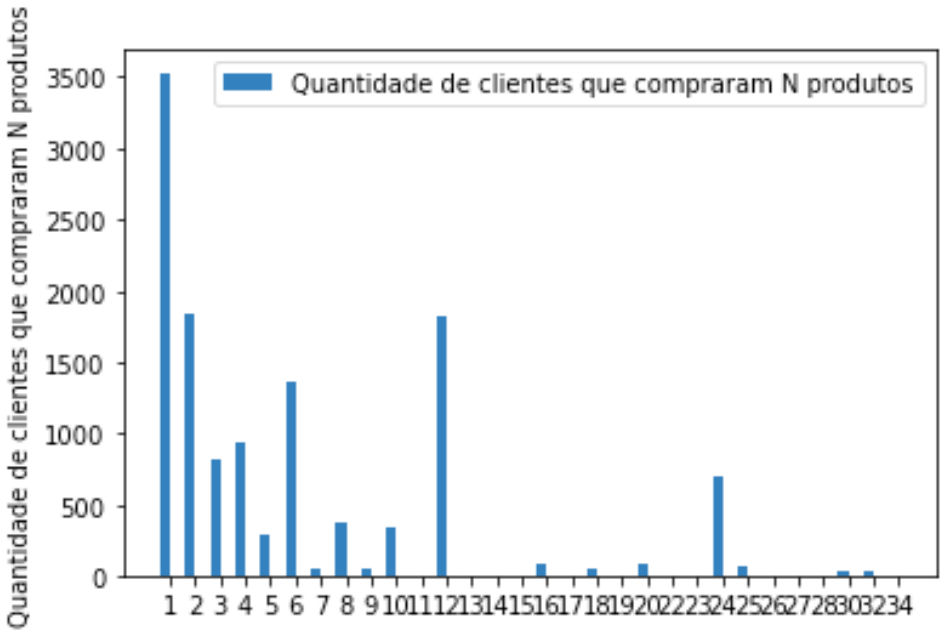


Figura 7.3: Quantidade de clientes que compraram N produtos

Como se pode verificar através da análise da Figura 7.3, onde se representa a quantidade de clientes que compraram N produtos, o mais comum é um cliente comprar apenas um produto. Contudo, consegue mostrar que comprar 6, 12 ou 24 são também quantidades bastante comuns. É algo particular no contexto de um loja online mas que se percebe por se tratar de um retail

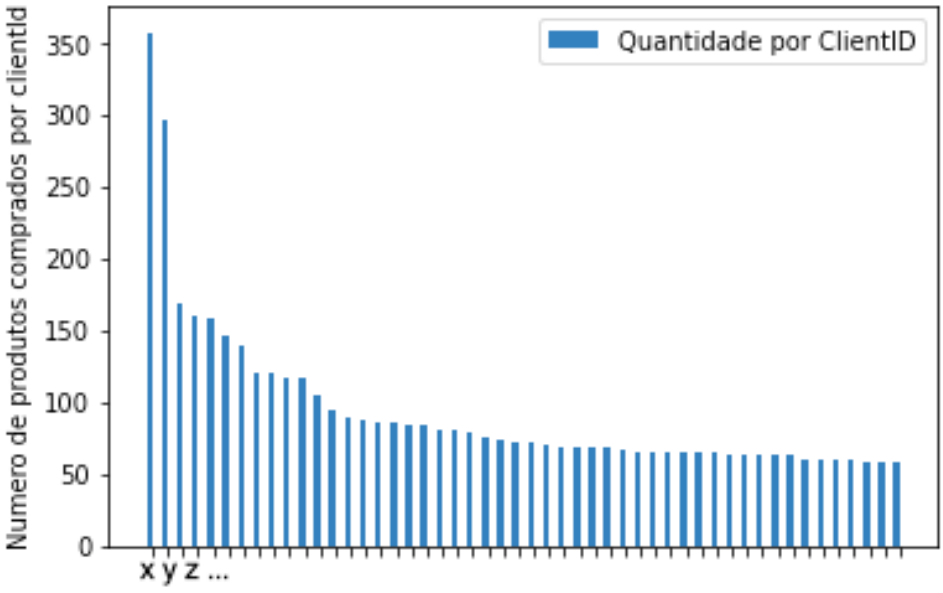


Figura 7.4: Quantidade por clienteId

O grande objetivo da análise da Figura 7.4, onde se representa a quantidade de produtos comprado por um cliente (identificado o mesmo ID), é percecionar se algum cliente era responsável pela maioria das compras. O gráfico revela-se equilibrado se excluirmos os clientId assinalados como x e y.

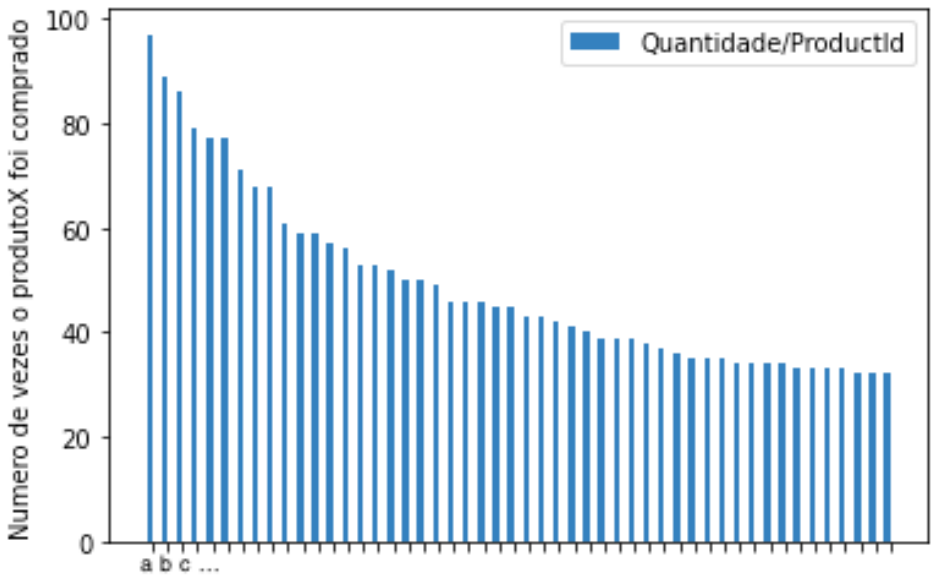


Figura 7.5: Quantidade por productId

A Figura 7.5, que representa a quantidade comprada por produto (identificado pelo ID), revela um equilíbrio entre os produtos mais e menos comprados. Esta conclusão ajuda-nos a tomar decisões quanto à construção do modelo de dados.

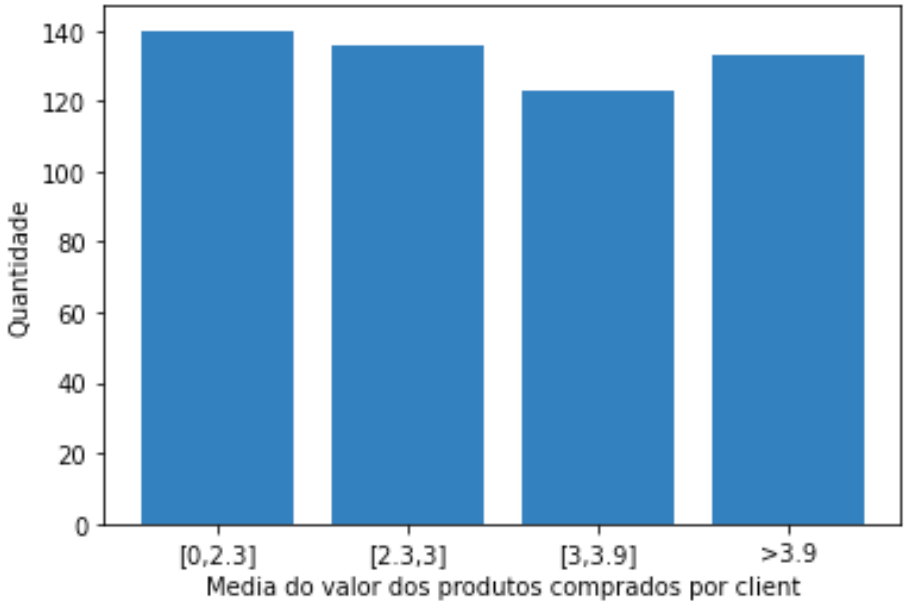


Figura 7.6: Media do valor dos produtos comprados por cliente

A Figura 7.6 apresenta-nos o valor médio dos produtos comprados por cliente. Os resultados estão dentro do que seria o esperado.

7.2 Métodos

7.2.1 Abordagem como um problema multilabel

Perante a análise dos dados foi decidido tratar o problema como um problema multilabel. Para isto, utlizamos a informação sobre os produtos comprados por cliente e tratámo-la como um problema multilabel. Como podemos observar no esquema da figura 7 foi inicialmente usada uma janela deslizante de valor 3 (equivalente a 3 produtos) e uma label também ela de valor 3 que neste caso corresponderia a 3 produtos.

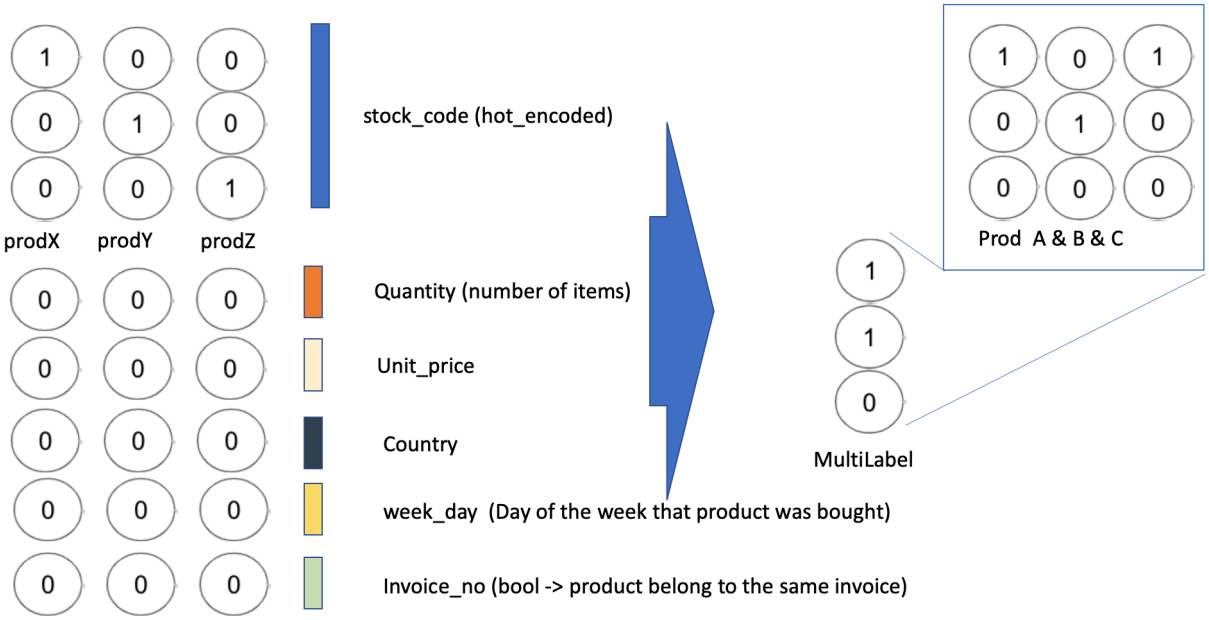


Figura 7.7: Abordagem multilabel

Como o esquema da Figura 7.7 demonstra, para a codificação dos dados foi utilizado o stock code depois de aplicada uma codificação hot encoded. Utilizamos ainda a quantidade do produtos comprados, o preço unitário, o país, o dia da semana e, ainda, se os produtos pertencem à mesma fatura.

7.2.2 Dataset de treino após transformação

Para a criação do dataset de treino houve um aspecto que se tornou fundamental, a ordem em que os produtos foram comprados. No entanto, quando uma série de produtos pertence à mesma factura são codificadas todas combinações de treino. Por exemplo, se considerarmos uma janela de tamanho 3 e tivermos um cliente que comprou o produto A,B,C num determinado instante e o produto D,E,F noutro instante todas a combinações entre A,B,C e D,E,F serão consideradas. No caso presente teríamos 8 combinações possíveis.

7.2.3 Treino

Para a primeira implementação da rede foi, como previsto utilizada um rede neuronal recorrente com várias camadas LSTM com o objectivo de tirar proveito da sequência a que os produtos foram comprados.

É importante ainda realçar que embora a sequência de produtos para o treino fosse fundamental, a forma como foi criado o dataset de treino, previa que era treinada não só a sequência ABC de produtos mas de igual forma a sequência ACB, CBA, etc.

Para o treino da primeira implementação da rede foram utilizadas duas camadas LSTM e uma camada Dense com 500, 100 e 500 neurónios respectivamente como podemos observar na figura 7.8. A Figura 7.8 representa o código Phyton usado.

```
def create_model_v1(n_steps,n_features, n_features_y, activation_func):
    input_shape= Input(shape=(n_steps, n_features))
    out=LSTM(500, return_sequences=True)(input_shape)
    out=Dropout(0.1)(out)
    out=LSTM(100, return_sequences=False)(out)
    out=Dropout(0.1)(out)
    out=Dense(500, activation='sigmoid', kernel_initializer="uniform")(out)
    out=Dropout(0.1)(out)
    out=Dense(n_features_y, activation=activation_func)(out)
    model = Model(inputs=input_shape, outputs=out)
    return model
```

Figura 7.8: Lstm model V1

Para o treino da última implementação da rede foi reduzido o número de camadas para apenas uma LSTM com 100 neurónios, como podemos observar na figura 7.9.

```
def create_model_v2(n_steps,n_features, n_features_y, activation_func):
    input_shape= Input(shape=(n_steps, n_features))
    out=LSTM(100, return_sequences=False)(input_shape)
    out=Dense(500, activation='sigmoid', kernel_initializer="uniform")(out)
    out=Dense(n_features_y, activation=activation_func)(out)
    model = Model(inputs=input_shape, outputs=out)
    return model
```

Figura 7.9: Lstm model V2

O excerto de código representado na Figura 7.10 indica a quantidade de épocas utilizadas para treino do modelo, assim com a activation funtion (função de activação) utilizada. Várias configurações foram testadas, sendo esta a que apresentou resultados mais próximos do esperado.

```
def compiletrain_(model, x_array, y_array, loss_fuction):
    metrics = ['accuracy', 'MeanSquaredError']
    model.compile(loss=loss_fuction, optimizer=function_optmizer, metrics=metrics)
    checkpointer = ModelCheckpoint(filepath="best_weights_v10.hdf5", monitor = 'val_loss', verbose=1,
                                   save_best_only=True)
    reduce_learning_rate = ReduceLROnPlateau(monitor='loss',
                                             factor=0.1,
                                             patience=2,
                                             cooldown=2,
                                             min_lr=0.00001,
                                             verbose=1)
    ####
    model.run_eagerly = True
    ####
    num=len(x_array)
    val=num-train
    x_train=x_array[0:train]
    val_data = x_array[train:train+val]

    history = model.fit( x_train, y_array, epochs=20, shuffle=False, batch_size=200, validation_split=0.2,
                         callbacks=[PlotLossesCallback(),checkpointer, reduce_learning_rate],
                         verbose=0)
    model.load_weights('best_weights_v10.hdf5')
    model.save('model_v10.h5')
    return history
```

Figura 7.10: Código em Python do Modelo compilado

No excerto código Phyton representado na Figura 7.11 podemos ver a criação do modelo a partir da função compiletrain_ acima referida.

```
history_categorical_crossentropy = compiletrain_(model,x_train_array, y_train_array, 'categorical_crossentropy')
```

Figura 7.11: Código em Python do compiletrain_

7.3 Resultados

A Figura 7.12 mostra a cross-entropy loss ao longo das épocas de treino do modelo, no conjunto de treino e no conjunto de validação. Como podemos observar, a loss de treino (linha azul) é monotónicamente decrescente ao longo das épocas de treino, indicando que o sinal de erro está a fluir eficientemente das últimas camadas para as anteriores (por back-propagation [34]). Simultaneamente, podemos observar que a loss de validação (linha laranja) decresce até à segunda época de treino, a partir da qual inverte esta tendência e cresce gradualmente. Este facto constitui um exemplo claro de um fenómeno de overfitting, pois o modelo aprende cada vez melhor a prever os dados onde é treinado, mas esta melhoria não se reflete em dados novos (conjunto de validação). Ademais, este fenómeno é expectável, pelo facto de os dados de treino ao nosso dispor serem extremamente reduzidos para o treino de redes neuronais, e pelo facto do número de parâmetros do modelo ser elevado (cerca de 1.8 milhões). Consequentemente, iremos usar o modelo no seu estado após a segunda época de treino pois mostra a melhor capacidade de generalização.

A Figura 7.13 mostra o erro quadrático médio ao longo das épocas de treino do modelo, no conjunto de treino e no conjunto de validação. Advertimos que a escala vertical (eixo dos yy) tem um alcance muito compacto, de 0.002021 a 0.002036. Assim sendo, observamos que esta métrica de erro toma valores muito próximos no conjunto de treino e no conjunto de validação.

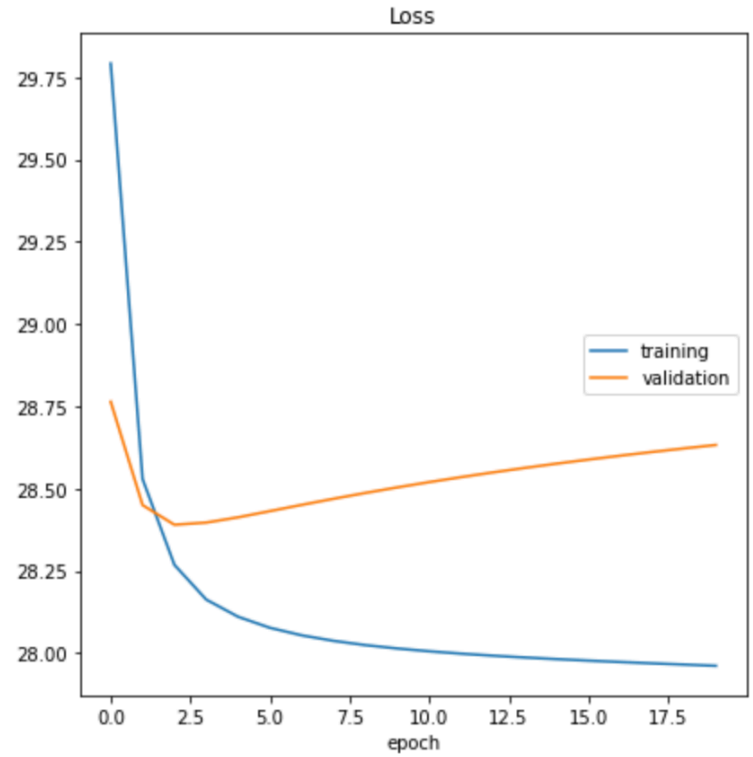


Figura 7.12: Métrica de cross-entropy loss do modelo durante treino.

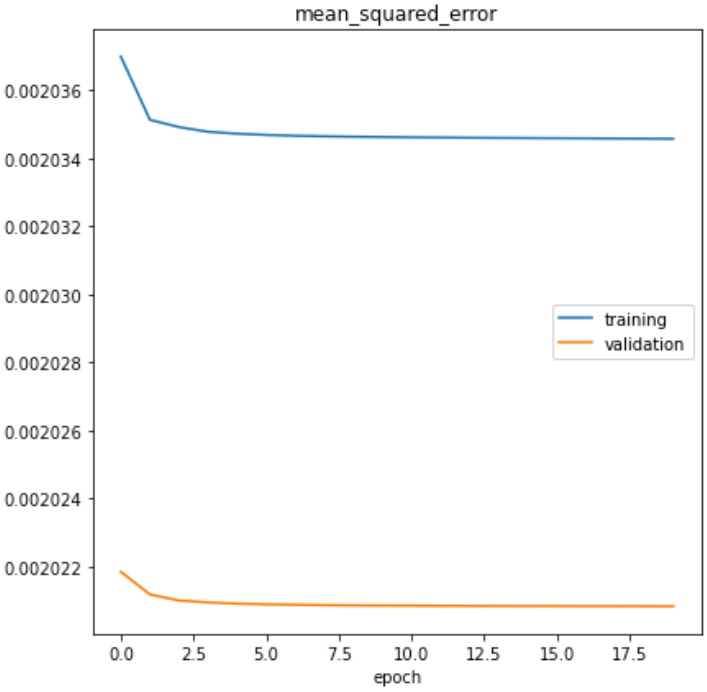


Figura 7.13: Erro quadrático médio do modelo durante treino.

Capítulo 8

Discussão e conclusões

Os sistemas de recomendação são cada vez mais usados para fornecer aos utilizadores recomendações acertadas sobre um determinado item. Eles tentam equilibrar fatores como precisão, novidade, dispersão e estabilidade nas recomendações. Os métodos de Filtragem Colaborativa (Collaborative Filtering - CF) desempenham um papel importante na recomendação, embora sejam frequentemente usados junto com outras técnicas de filtragem, como as baseadas no conteúdo (content-based), no conhecimento (knowledge-based) entre outros [35].

A CF baseia-se na maneira como os humanos tomam decisões ao longo da história: além de nossas próprias experiências, também baseamos as nossas decisões nas experiências e conhecimentos que podem chegar a cada um de nós por um grupo de conhecidos. O deep-learning, recorrendo ao uso de redes neuronais em várias camadas, tem sido muito usado e tem ajudado muito no alcance desses objetivos.

Assim, no trabalho aqui apresentado, foi concebido e desenvolvido um pipeline constituído por 4 passo principais: aquisição de dados, processamento, treino e geração de recomendações. Foi também desenvolvida uma API, em Java recorrendo ao uso da framework Spring e à base de dados MongoDB. Esta API facilita o processo de geração de recomendações e facilita a integração o processo concebido com outras aplicações e-commerce.

8.1 Problemas encontrados

O primeiro e mais difícil problema de lidar nesta dissertação foi, certamente, a escassez de dados que fossem permitidos usar para o estudo. A decisão acabou por recair na utilização de um dataset que tivesse presente todas as características dum sistema de compra e-commerce. No caso foi escolhido um dataset de uma empresa de retail de venda online sediada no Reino Unido. Ultrapassado o primeiro desafio, o foco passou por criar um sistema capaz de lidar com uma grande quantidade de dados e ainda assim

conseguir apresentar resultados relevantes em tempo útil. A escolha da linguagem apresentou também um desafio pois Java não é a linguagem mais popular na aplicação de métodos Machine Learning, o que obrigou a utilização de um micro-serviço em Python para, deste modo, tomar partido de toda a sua larga escolha de bibliotecas.

A escolha das métricas de avaliação dos Sistemas de Recomendação implementados foi, possivelmente, a maior dificuldade num projecto desta dimensão por não existirem boas comparações no que ao contexto do e-commerce diz respeito.

8.2 Discussão dos resultados

Os objetivos propostos pela empresa no contexto deste trabalho foram integralmente cumpridos, incluindo o desenvolvimento de uma API genérica para a geração de recomendações (colaborativas e baseadas em conteúdo). Para além disso, foi desenvolvido também um protótipo de geração de recomendações através de uma rede neuronal, integrado num caso de estudo de e-commerce. Relativamente ao caso de estudo, a integração foi bem sucedida, mas os resultados do treino da rede neuronal recorrente (RNN) não encorajam o seu uso num ambiente real. Apesar da experiência ter sido ambiciosa, demonstrou não ser eficaz treinar uma rede neuronal num dataset tão diminuto como o disponível para o caso de estudo. Ademais, a natureza esparsa do dataset, maioritariamente composto por dados categóricos, também dificulta o treino de redes neuronais [36].

8.3 Trabalho Futuro

Como trabalho futuro o objectivo passa aplicar o processo de recomendações aqui desenvolvido a outros DataSets para que consiga um treino melhor e tirar novas conclusões. Isto passa por ter acesso a uma maior quantidade de dados, maior qualidade e mais aproximado a uma aplicação e-commerce.

Referências

[1] F. Ricci, L. Rokach e B. Shapira. “Introduction to Recommender Systems Handbook”. Em: Recommender Systems Handbook. Ed. por F. Ricci, L. Rokach, B. Shapira e P. B. Kantor. Boston, MA: Springer US, 2011, pp. 1–35. isbn: 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_1. url: https://doi.org/10.1007/978-0-387-85820-3_1.

[2] P. Resnick e H. R. Varian. “Recommender systems”. Em: cacm 40.3 (mar. de 1997), pp. 56–58. doi: 10.1145/245108.245121. url: http://jmvidal.cse.sc.edu/library/resnick97a.pdf.

[3] Wavetec. “Wavetec (2019). Opinion Plus helps you discover the true voice of your customer - Wavetec. [online] Avaliable at: https://www.wavetec.com/news/opinion-plus-helps-you-discover-the-true-voice-of-your-customer/ [Accessed 12 May 2019]”. Em: dez. de 2019.

[4] A. Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. Em: Physica D: Nonlinear Phenomena 404 (2020), p. 132306. issn: 0167-2789. doi: https://doi.org/10.1016/j.physd.2019.132306. url: https://www.sciencedirect.com/science/article/pii/S0167278919305974.

[5] Deep Learning Book. https://www.deeplearningbook.com.br: Data Science Academy, 2021.

[6] S. Dargan, M. Kumar, M. R. Ayyagari e G. Kumar. “A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning”. Em: Archives of Computational Methods in Engineering. Vol. 27. https://doi.org/10.1007/s11831-019-09344-w: Springer, 20019, pp. 1071–1092.

[7] W. Xing e D. Du. “Dropout Prediction in MOOCs: Using Deep Learning for Personalized Intervention”. Em: Journal of Educational Computing Research 57.3 (2019), pp. 547–570. doi: 10.1177/0735633118757015. eprint: https://doi.org/10.1177/0735633118757015. url: https://doi.org/10.1177/0735633118757015.

[8] F. Isinkaye, Y. Folajimi e B. Ojokoh. “Recommendation systems: Principles, methods and evaluation”. Em: Egyptian Informatics Journal 16.3 (2015), pp. 261–273.

[9] Y.-C. Chen, L. Hui e T. Thaipisutikul. “A collaborative filtering recommendation system with dynamic time decay”. Em: The Journal of Supercomputing 77.1 (2021), pp. 244–262. doi: 10.1007/s11227-020-03266-2. url: https://doi.org/10.1007/s11227-020-03266-2.

[10] J. S. Breese, D. Heckerman e C. Kadie. “Empirical analysis of predictive algorithms for collaborative filtering”. Em: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc. 1998, pp. 43–52.

[11] P. B. Thorat, R. Goudar e S. Barve. “Survey on collaborative filtering, content-based filtering and hybrid recommendation system”. Em: International Journal of Computer Applications 110.4 (2015), pp. 31–36.

[12] B. Lika, K. Kolomvatsos e S. Hadjiefthymiades. “Facing the cold start problem in recommender systems”. Em: Expert Systems with Applications 41.4 (2014), pp. 2065–2073.

[13] Z.-K. Zhang, C. Liu, Y.-C. Zhang e T. Zhou. “Solving the cold-start problem in recommender systems with social tags”. Em: EPL (Europhysics Letters) 92.2 (2010), p. 28002.

[14] J. Lin. “The neural hype and comparisons against weak baselines”. Em: ACM SIGIR Forum. Vol. 52. ACM. 2019, pp. 40–51.

[15] M. F. Dacrema, P. Cremonesi e D. Jannach. “Are we really making much progress? A worrying analysis of recent neural recommendation approaches”. Em: Proceedings of the 13th ACM Conference on Recommender Systems. ACM. 2019, pp. 101–109.

[16] B. Hu, C. Shi, W. X. Zhao e P. S. Yu. “Leveraging meta-path based context for top-n recommendation with a neural co-attention model”. Em: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM. 2018, pp. 1531–1540.

[17] X. Li e J. She. “Collaborative variational autoencoder for recommender systems”. Em: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM. 2017, pp. 305–314.

[18] H. Wang, N. Wang e D.-Y. Yeung. “Collaborative deep learning for recommender systems”. Em: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM. 2015, pp. 1235–1244.

[19] C. Wang e D. M. Blei. “Collaborative topic modeling for recommending scientific articles”. Em: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM. 2011, pp. 448–456.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu e T.-S. Chua. “Neural collaborative filtering”. Em: Proceedings of the 26th international conference on world wide web. International World Wide Web Conferences Steering Committee. 2017, pp. 173–182.

[21] D. Liang, R. G. Krishnan, M. D. Hoffman e T. Jebara. “Variational autoencoders for collaborative filtering”. Em: Proceedings of the 2018 World Wide Web Conference. International World Wide Web Conferences Steering Committee. 2018, pp. 689–698.

[22] L. Zheng, V. Noroozi e P. S. Yu. “Joint deep modeling of users and items using reviews for recommendation”. Em: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM. 2017, pp. 425–434.

[23] S. Li, J. Kawale e Y. Fu. “Deep collaborative filtering via marginalized denoising auto-encoder”. Em: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM. 2015, pp. 811–820.

[24] H. Liang e T. Baldwin. “A probabilistic rating auto-encoder for personalized recommender systems”. Em: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM. 2015, pp. 1863–1866.

[25] R. Devooght e H. Bersini. “Collaborative filtering with recurrent neural networks”. Em: arXiv preprint arXiv:1608.07400 (2016).

[26] B. Hidasi, A. Karatzoglou, L. Baltrunas e D. Tikk. “Session-based recommendations with recurrent neural networks”. Em: arXiv preprint arXiv:1511.06939 (2015).

[27] B. Hidasi, M. Quadrana, A. Karatzoglou e D. Tikk. “Parallel recurrent neural network architectures for feature-rich session-based recommendations”. Em: Proceedings of the 10th ACM conference on recommender systems. ACM. 2016, pp. 241–248.

[28] D. Jannach e M. Ludewig. “When recurrent neural networks meet the neighborhood for session-based recommendation”. Em: Proceedings of the Eleventh ACM Conference on Recommender Systems. ACM. 2017, pp. 306–310.

[29] V. Bogina e T. Kuflik. “Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks.” Em: RecTemp@ RecSys. 2017, pp. 57–59.

[30] D. Kim, C. Park, J. Oh, S. Lee e H. Yu. “Convolutional matrix factorization for document context-aware recommendation”. Em: Proceedings of the 10th ACM Conference on Recommender Systems. ACM. 2016, pp. 233–240.

[31] Y. Wu, C. DuBois, A. X. Zheng e M. Ester. “Collaborative denoising auto-encoders for top-n recommender systems”. Em: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM. 2016, pp. 153–162.

[32] R. Devooght e H. Bersini. “Long and short-term recommendations with recurrent neural networks”. Em: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization. ACM. 2017, pp. 13–21.

[33] R. T. Fielding, R. N. Taylor, J. R. Erenkrantz, M. M. Gorlick, J. Whitehead, R. Khare e P. Oreizy. “Reflections on the REST Architectural Style and ”Principled Design of the Modern Web Architecture”(Impact Paper Award)”. Em: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. New York, NY, USA: Association for Computing Machinery, 2017, pp. 4–14.

isbn: 9781450351058. doi: 10.1145/3106237.3121282. url: https://doi.org/10.1145/3106237.3121282.

[34] S. Hochreiter e J. Schmidhuber. “Long short-term memory”. Em: Neural computation 9.8 (1997), pp. 1735–1780.

[35] J. Bobadilla, F. Ortega, A. Hernando e A. Gutiérrez. “Recommender systems survey”. Em: Knowledge-Based Systems 46 (2013), pp. 109–132. issn: 0950-7051. doi: https://doi.org/10.1016/j.knosys.2013.03.012. url: https://www.sciencedirect.com/science/article/pii/S0950705113001044.

[36] J. T. Hancock e T. M. Khoshgoftaar. “Survey on categorical data for neural networks”. Em: Journal of Big Data 7.1 (2020), pp. 1–41.

Apêndice A

Apêndice: Código Fonte

```
import numpy as np
from flasgger import Swagger
from flask import Flask, request
from redis import Redis
from rq import Queue

from service.Collaborative import CollaborativeDeepService, train_pre_processing, evaluating
from service.Decoding import decoding_with_heap_index
from service.PreProcessing import PreProcessing, STATS_FILE_NAME

app = Flask(__name__)
swagger = Swagger(app)

redis = Redis()
task_queue = Queue(connection=redis)

collaborativeDeepService = CollaborativeDeepService()
preProcessing = PreProcessing()
model = collaborativeDeepService.load_model()
encoded_product_id_to_numpy = preProcessing.encoding_product_id_to_numpy()



@app.route("/train")
def train():
    try:
        job_pre_id = task_queue.enqueue(train_pre_processing)
```

```
28      except ValueError:
29          return "Functions from the __main__ module cannot be processed by workers"
30      return f"Full process on queue with id \"{job_pre_id.id}\" and will start once the worker
        ↪ is started"
31
32
33  @app.route("/current_jobs")
34  def check_jobs():
35      actual_jobs = task_queue.jobs
36      return actual_jobs.__str__()
37
38
39  @app.route("/clear_jobs")
40  def clear_jobs():
41      actual_jobs = task_queue.empty()
42      return actual_jobs.__str__()
43
44
45  @app.route("/training_full_process")
46  def full_process():
47      try:
48          job_pre_id = task_queue.enqueue(preProcessing.get_input_output_encoded_sliding_window)
49      except ValueError:
50          return "Functions from the __main__ module cannot be processed by workers"
51      return f"Full process on queue with id \"{job_pre_id.id}\" and will start once the worker
        ↪ is started"
52
53
54  @app.route("/train_no_worker")
55  def full_process_no_worker():
56      preProcessing.get_input_output_encoded_sliding_window()
57      train_pre_processing()
58      return "Train complete"
59
60
61  @app.route('/train_small_sample')
62  def full_process__small_sample():
63      preProcessing.get_input_output_encoded_sliding_window(train_sample=True)
64      train_pre_processing()
65      return "Train complete"
66
67
68  @app.route("/train_status")
69  def status_pre_processing():
70      if open(STATS_FILE_NAME, "r").readable():
```

```
        text_file = open(STATS_FILE_NAME, "r")
        return text_file.read()
    return "Zero processes are running at the moment"


@app.route("/prediction_topN", methods=['GET'])
def prediction_top_x():
    product_id = request.args.get('productId')
    n = request.args.get('n')
    prod_encoded = encoded_product_id_to_numpy.get(product_id)
    if prod_encoded is None:
        return f"Product \"{product_id}\" doesn't belong to the list of products able to get
    ↪ recommendations." \
               f" Check /products to see list of products."
    try:
        prediction_encoded = model.predict(np.array([prod_encoded]))
        index_list = decoding_with_heap_index(prediction_encoded, int(n))
        map_index_id = list(map(lambda x: preProcessing.array_of_unique_products[x],
    ↪ index_list))
        return map_index_id.__str__()
    except ValueError:
        return "Mismatch between the provided input data and the model's expectations"


@app.route("/products", methods=['GET'])
def unique_products():
    return preProcessing.array_of_unique_products.__str__()


@app.route('/train_and_evaluate', methods=['GET'])
def train_and_evaluate():
    dic_stats = evaluating()
    return dic_stats.__str__()
```

Listagem A.1: Micro-service controller com Flask e Python

```

version: '3'
services:
  mongodb:
    container_name: mongodb
    image: mongo:3.6
    restart: always
    environment:
      - MONGO_DATA_DIR=/data/db
      - MONGO_LOG_DIR=/dev/null
```

```
    ports:
      - 27017:27017
    command: mongod --smallfiles --logpath=/dev/null
    networks:
      - network_mongo

  flask:
    container_name: flask
    image: flask10
    environment:
      DB_PORT_27017_TCP_ADDR: 27017
    ports:
      - 5000:5000
    networks:
      - network_mongo

  spring:
    container_name: spring_final
    image: spring_final
    ports:
      - 8081:8080
      - 8086:8085
    command: java -jar target/demo-0.0.1-SNAPSHOT.jar
    networks:
      - network_mongo

networks:
  network_mongo:
    external: true
```

Listagem A.2: Docker compose com 3 serviços.