Universidade do Porto
Faculdade de Engenharia
FEUP

Developing a tool to help retailers better position themselves in the food retail market

Deloitte - WeShare - Serviços de Gestão, S.A

Francisca Leão Cerquinho Ribeiro da Fonseca

Supervisor
Vera Miguéis
Company Supervisor
António Mota Lopes

Faculdade de Engenharia da Universidade do Porto,
Mestrado Integrado em Engenharia Informática e Computação

July 2020

Universidade do Porto

Faculdade de Engenharia

FEUP

Developing a tool to help retailers better position themselves in the food retail market

Francisca Leão Cerquinho Ribeiro da Fonseca

Mestrado Integrado em Engenharia Informática e Computação

July 2020

Abstract

Retailers have always tried to identify the products competitors offer, define the prices to charge for a product and properly define their market positioning. Although software that collect competitors' market prices and that present solutions to optimize prices already exist, none are capable of ensuring an entire online footprint of a particular product that has direct, indirect and location-specific competitors.

The aim of this thesis is the development of a rich data source, with permanently updated and reliable data, based on the online platforms that accompany the price changes of the products sold by several retailers. This project is supported by software as a service (SaaS) and has three main objectives: (1) to develop a search bot to collect products information from different online stores, (2) to match similar competitor products and develop a database and (3) to develop a graphical interface from data collected from online platforms, that will support decision making in sales, marketing and operations.

The solution was developed using Amazon Web Services (AWS) technologies, which provide a set of fully managed services used to build and run serverless applications. Through this service, it was possible to create a scalable, reliable, and flexible product, since AWS manages time and space. The architecture of the project has two main layers: data ingestion and data transformation. This last layer is divided into three processes: the creation of a database, cleaning, and standardization of data, and creation of the product match algorithm. For the data collection layer, two techniques were used: web-scraping and web crawler. To match the products, a comparison was made between the image and string attributes, such as name, brand, and packaging content. For image comparison, a pixel-by-pixel calculation was used to determine the differences between two input images. To assess the similarity between string attributes, two approaches were studied: one that uses word embeddings and another that uses the Levenshtein Distance. Furthermore, two approaches to compare the string attributes were used: the first calculates the similarity of the values of the string attributes individually and sums the similarities returned and the second combines the value of the three string attributes into a single string separated by a space. After being analyzed and tested, comparing the string attributes together and using Levenshtein distance to assess similarity proved to be the best approach. Through the product matching algorithm, it was possible to build a database with all the existing products from different retailer's online stores. For each product, the information about who has it is stored in that same database. With this information, it was possible to create a dashboard that enables retailers to analyze their competitors.

Summing up, this project enabled to create a solution that can provide retail companies with the information they need to take real-time action for competitive excellence. The result of the project is a dashboard to support decision making, namely pricing.

Keywords: Product Matching, BI, Product Optimization, SaaS, Real-Time, NLP

Resumo

Os retalhistas precisam, diariamente, de identificar os produtos que os concorrentes oferecem, definir preços a serem cobrados por um produto e definir adequadamente o seu posicionamento no mercado. Apesar de já existirem alguns software que recolhem os preços praticados e que oferecem soluções para optimizar os preços, não existe nenhum capaz de assegurar toda a oferta online de um determinado produto.

O objetivo desta tese é o desenvolvimento de uma fonte de dados rica, atualizada e confiável, com base nas plataformas online que acompanham as variações de preço dos produtos vendidos por vários retalhistas. Este projeto é suportado por um software as a service (SaaS), tendo três objetivos: (1) desenvolver um bot que faça a recolha automática de detalhes de produtos de diferentes lojas online, (2) fazer o match entre produtos semelhantes da concorrência e construir uma base de dados e (3) desenvolver um dashboard de suporte a tomada de decisões em vendas, marketing, atendimento ao cliente e operações.

A solução foi desenvolvida usando tecnologias da Amazon Web Services (AWS), que fornece um conjunto de serviços para criar e executar aplicativos sem servidor, permitindo criar um produto escalável, confiável e flexível. A arquitetura do projeto possui duas camadas: recolha e transformação de dados. Esta última camada é dividida em três processos: criação de uma base de dados, limpeza e padronização de dados e criação do algoritmo *product match*. Para recolher os dados foram utilizadas duas técnicas: web-scrapping e web crawler. Para fazer a correspondência entre os produtos, foi feita uma comparação entre as imagens e atributos de *string*, como nome, marca e conteúdo da embalagem. Para a comparação de imagens, foi usado um cálculo pixel por pixel para determinar as diferenças entre duas imagens. Para avaliar a semelhança entre os atributos das strings foram estudadas duas abordagens: uma que usa word embeddings e outra que usa a distância de Levenshtein. Além disso, foram utilizadas duas formas de comparar os atributos de *string*: a primeira calcula a similaridade dos valores dos atributos individualmente e soma as semelhanças retornadas e a segunda combina o valor dos três atributos numa única *string* separada por um espaço. Após ser analisado, foi provado que a melhor abordagem é aquela que compara os atributos de *string* numa única *string* e usa a distância de Levenshtein para avaliar a similaridade. Através do algoritmo de product match, foi possível criar uma base de dados com todos os produtos existentes nas diferentes lojas online. Para cada produto, as informações sobre quem o possui são armazenadas na mesma base de dados. Deste modo, foi possível criar um dashboard que permite analisar a concorrência.

Resumindo, este projeto permitiu criar uma solução que fornece às empresas de retalho alimentar as informações necessárias para que realizem ações em tempo real, obtendo excelência competitiva. O resultado do produto é um dashboard atrativo que apoia na tomada de decisão, nomeadamente o preço.

Palavras-Chave: Product Matching, BI, Product Optimization, SaaS, Real-Time, NLP

Acknowledgements

I would like to thank my advisors, Professor Vera Miguéis for the constant dedication, persistence, and demand and António Mota Lopes for the guidance and sharing of knowledge in the development of the project.

To my parents for the education they gave to me and for having always believed in me and my abilities.

To my siblings and boyfriend for their support. My friends who accompanied, supported, taught and strengthened me during my academic career, namely Mariana Silva, Xavier Fontes, and Luís Saraiva. A special thanks to Mariana Silva for all the support and friendship she provided me during my academic career.

In addition, I would like to thank Cláudia Pimenta, whom I met at Deloitte but who has always supported me and contributed in large part to my personal and professional growth within the company.

Finally, I could not fail to mention my grandmother who, despite not having been present in recent academic years, has always accompanied me during my student life and has always contributed to my growth.

"I can accept failure, everyone fails at something.
But I can't accept not trying."

Michael Jordan

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
AWS	Amazon Web Services
CA	Competitor Analysis
CSS	Cascading Style Sheets
DTTL	Deloitte Touche Tohmatsu Limited
EAN	European Article Number
GTIN	Global Trade Item Number
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
NLP	Natural Language Processing
SaaS	Software as a Service
TF-IDF	Term Frequency – Inverse Document Frequency
URL	Uniform Resource Locator
VM	Virtual Machine
WEB	Word Wide Web
XHR	XMLHttpRequest

Chapter 1

Introduction

This chapter provides an introduction to this dissertation by defining its scope, motivations and goals. Section 1.1 outlines the context of the problem and the environment in which this is developed. Section 1.2 defines the main forces driving this thesis. Section 1.3 explains the problem in detail and describes the objectives. Lastly, Section 1.4 explains the composition of the rest of the document.

1.1 Context

Retail is the process of selling consumer goods or services to customers through multiple channels of distribution aiming for customer consumption rather than re-selling. The retail industry is a broad industry, with a myriad of different types of companies falling under this industry category. Food retailing is an activity of enormous economic significance which, in recent years, has been exposed to significant structural changes in response to a combination of economic forces, consumer trends, competitive initiatives, technological developments and environmental regulations [1]. It is a constantly growing sector, namely through new competitive channels, formats and outlets.

In the past, the small food trade guaranteed the supply of the entire population. The retail market was characterized by having a large number of small establishments, and the food distribution network was characterized by the existence of a large number of small distributors, being covered in a restricted distribution area [2].

However, in the 1980s, food distribution in Portugal began to suffer some changes due to the increase in large distribution. Consumers started looking for large stores in which they began to enjoy a wide variety of products (food and non-food), in one space. Allied to the size of these stores, this period was also characterized by the growing extension of economic groups reinforcing negotiating capacity of these companies with their suppliers. As a result of this situation, large quantities of products, including food products, were sold at lower prices to consumers, due to the favorable conditions obtained with the food chains and their suppliers [2].

In the second half of the 1980s, Portugal's entry in the European Union brought a significantly positive impact for the food sector due to the liberalization of markets and the sharp growth in imports of more sophisticated food products [2].

Currently, and according to 2018 data, Sonae MC leads the food retail industry in Portugal, with the most diversified format portfolio. Sonae MC is the leader in market share, diversity of formats, location in stores, electronic commerce for food retail and finally price positioning, characterized by a strong promotional action. The figure below demonstrates the growth of the retail industry in Portugal, between 2017 and 2018.

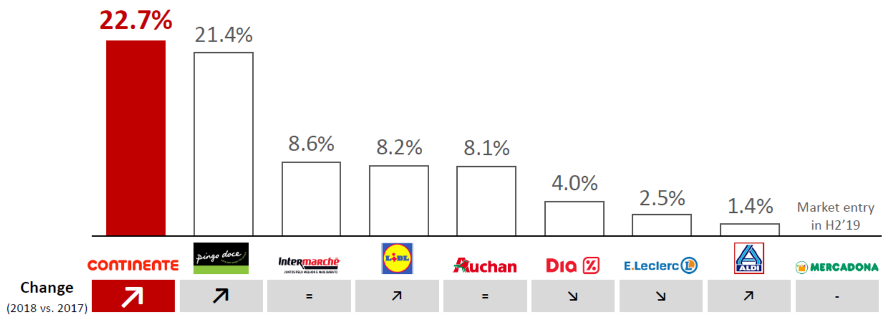


Figure 1.1: Portugal Market Share
Source: Planet Retail RNG as of March 2019.

The growth and distribution of supermarkets depends on a number of factors such as the consumption habits of the local population, state of the economy, and central and local government regulations. In the past, the consumer would only have access to small grocery stores and markets, however throughout the years, many other channels have emerged, as hypermarkets, or the rapidly growing e-commerce channel. Hypermarkets, for example Continente, are characterized by having large store sizes, which enable the existence of a great diversity of products and brands, in an organized and appealing space. Hypermarkets are also frequently characterized by great promotional offers and other associated services, such as parking and loyalty cards, and are normally visited by customers in a biweekly or monthly basis. Supermarkets, such as Continente Bom dia and Pingo Doce, have less product diversity than hypermarkets, as they are usually smaller establishments. The existence of affordable prices, combined with the location and convenience factor, determine that supermarkets are normally associated with daily or weekly purchases. The discount format, like MiniPreço, corresponds to a type of establishment dedicated to customers who consider price as driving factor when shopping. They are characterized by spaces that are less appealing and less organized, due to the existence of a fewer diversity of products and a great investment in white label products with more accessible prices.

However, in recent years, there is another format that has been growing, the online format also known as e-commerce. In this channel the goods are therefore presented in Internet stores. Customers generally place their orders through electronic checkout or can use traditional email or communication methods (for example, mail, phone and fax) to do so. There are three key dimensions that affect consumer satisfaction with online shopping activities and subsequent purchase: user experience (for example, the quality of broadband connection and website design), transaction (for instance, purchase price, convenience, warranty, entertainment and evaluation) and service (such as, order processing, delivery and after-sales service). User experience is closely linked to the general satisfaction, as it acts as a motivating force for the consumer to continue browsing, searching and buying on a specific website. Meanwhile, the quality of the transaction process also plays a crucial role, as convenience, value and security are essential requirements to the online consumer. Finally, the level of compliance determines consumer confidence in online transactions and can also help provide greater convenience and value for electronic buyers.

More customers now expect a seamless, personalized shopping experience. Whether they are buying in person or online, consumers search for a quick and easy experience, something that is being facilitated with retail technologies. In an ever-changing competitive environment, grocers and online food retailers are constantly innovating and utilizing technology to ease and improve the consumer shopping experience. The increasing volume of rapidly expanding data is forcing online retailers to introduce data analytic platforms, as a means to gain a transparent overview of the market. Companies need to spend less time analyzing data and more time gaining meaningful insights from it.

Advanced Analytics, relates to the science of forecasting future trends based on the study of present-day and past data. Today, inexorably, it is making inroads into the retail sector. Grocery retailers and supermarkets are increasingly turning to this data-driven science to help them get a leg up over the competition. Additionally, Advanced Analytics is becoming one of the key drivers of profit, as it represents a portfolio of tools, techniques, and organizational capabilities that can be applied to specific decisions across a wide range of business concerns. Advanced Analytics can help retailers improve the way they analyze, set, and deliver pricing in a sustainable and predictable manner. With a broad data set, analytics can allow retailers to derive valuable insights by measuring differences in demand across customer segments, identifying key value items, clustering stores into zones, and assessing shopping behavior across channels. These findings allow retailers to develop a tactical framework to set prices while accounting for distinctions across customers, channels, competitors, and categories [3].

Companies need a comprehensive and structured approach to optimize their category planning — one that combines data-based insights across customers, competitors, pricing, promotions and channels with Advanced Analytics for analysing and scenario planning. Such an approach can identify growth opportunities in the category, inform negotiation strategies, and, ultimately, enhance relationships with customers, improving both margins and sales. Analytics software is a class of tools that leverages data to create context-rich, actionable insight. Business analytics is focuses on the questions – why is this happening, what if the trends continue, what will happen next (predict), and what is the best that can happen (optimize).

In the new normal of retailing, where the consumer is value-conscious, always-on, mobile-enabled, socially active and channel-agnostic, retailers are finding it difficult to differentiate themselves based on traditional factors such as price, promotions, location and assortment alone. On the other hand, with consumers themselves reliant on technology – personal computers, tablets, mobile and smart phones, the Internet, and social media – they are leaving behind digital breadcrumbs and displaying their "digital body language". Combined with how consumers interact with retailers across their own channels, loyalty programs, promotions and customer service, this mass of consumer data offers retailers their best chance yet to really know their customer [4].

Knowing the customer better than the competition and having the ability to orchestrate business decisions at close to real-time speed is the new retail competitive battlefield and business analytics can be one of the strongest weapons in a retailer's arsenal. Business analytics will only evolve further into a strategic capability that sits at the intersection of customer preferences, business strategy, and business processes. Insights will be deeply embedded across a retailer's functional value chain, giving it both the ability to be investigative and predictive (strategic), as well as the adeptness to be efficient and agile (operational) [4].

1.2 Motivation

The retail sector is constantly changing and retailers feel compelled to increase the efficiency in operations, revenues, market shares, as well as offering a consistent and integrated omnichannel experience. The battle for consumers' grocery spend is more intense than ever, with new store concepts, pricing strategies, delivery formats and market volatility, putting retailers under pressure to deliver increased value with limited costs.

Nowadays, retailers have the necessary services, such as price comparison platforms, required to analyse the price at which a specific product X is being sold at competitors' stores. Despite this, there are two significant limitations regarding these services' functionalities. To begin with, the first limitation relates to their inability to adapt, added to the complexity and re-occurrence of maintenance required to ensue their functionality. In addition, it is almost impossible to build an effective price strategy by means of price comparison engines, since the analysis they provide is neither of a good quality nor complete.

Despite the existence of the services mentioned above, there is currently no software in the Portuguese food industry capable of providing near real-time information on what is happening in the market. Retailers need to be one step ahead of their competitors and with the constant growth of technology, it is necessary to develop a software that augments management efficiency and effectiveness.

This thesis focuses on providing retail companies with the crucial information needed to take action in real-time, for competitive changes in any frequency required, with a focus on the online retail channel. More specifically, this project aims to develop a technology capable of identifying similar or equivalent products available in retailers websites and construct a database with relevant data characterizing those products, namely their price. For this, the biggest difficulty lies in the handling of highly heterogeneous product descriptions and a shortfall in standards for clear product identification (GTIN).

A well-designed software as a service (SaaS) will enable a large improvement over the current methods of managing food retail and will allow quick responses to high-level business decisions. Furthermore, it will improve staff efficiency, freeing staff to concentrate on their primary job functions, rather than repetitive data entry tasks.

The project's development will be undertaken at Deloitte Consultores S.A., a Portuguese member firm of DTTL which provides audit & assurance, consulting, financial advisory, risk advisory, and tax.

1.3 Problem and Specific Goals

Price is one of the dimensions that affects companies in meeting objectives, always trying to answer questions such as: what is the pricing strategy that should be adopted?, how does the location influence prices?, what is the composition of the final price (with-/without discounts, with/without associated services)?, what is the trend of the price of a product in a certain period?

These issues pose enormous challenges to the companies in this market, namely with the increasing use of digital channels by consumers to compare prices in the market, identifying promotions/campaigns, or even identifying trends.

This project will seek to support the definition of the business strategy and aid decision makers in defining prices and designing promotional strategies, using an application based on different technologies that enables the capture of, near real-time, information on price and product ranges of competitors in the food-retail market. In addition to allowing the monitoring of direct and indirect competition in the food retail market through online channels, retailers will be able to have a better view and understanding of the global market.

Thus, this project has three main objectives:

1. To develop a search bot to collect details of retail products from different online stores;
2. To match similar competitor products and develop a database;
3. To develop a data interface from data collected from online platforms, that will support decision making in sales, marketing, customer service, and operations.

Essentially, the objective is to search and collect, automatically, food-retail product data from Portuguese online stores and implement a use case of Product Match that is transversal to any retail company in the sector.

From a user's point of view, the output of the project should be an iterative dashboard that allows the retailer to understand their market position: analyzing prices and category composition, coverage of assortments and categories, comparison of promotions and framing in the product portfolio. This aims to decrease the time needed to understand competitors' pricing strategies, improve efficiency in pricing by product and store type and increase the effectiveness of promotion planning. This dashboard allows the retailer to ensure parity with competitors by carefully analyzing the data collected to discover and address trends that may soon help or hurt the business.

1.4 Report Structure

This thesis is, then, developed in the following manner: Chapter 2 introduces the concepts that support this report. Chapter 3 presents an inspection of the current state of the art associated with the development of similar projects. Chapter 4 describes the entire solution developed for the problem mentioned above. Chapter 5 describes the strategy that was used to validate the work developed and the results obtained. At last, Chapter 6 reflects the results of the project, its goals and future development.

Chapter 2

Background

This chapter introduces significant concepts that will support the comprehension of this report. Section 2.1 describes what is SaaS. Section 2.2 explains the most common methods to extract data from the internet. Section 2.3 describes the potential of a serverless when associated with Saas. Section 2.4 describes the applicability of price optimization in retail. Section 2.5 introduces the product match concept. Finally, Section 2.6 summarizes this chapter.

2.1 Software as a Service (SaaS)

SaaS is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. Unlike traditional packaged applications that users install on their computers or servers, the SaaS vendor owns the software and runs it on computers in its data center [5].

Traditional software is conventionally sold as a perpetual license with an up-front cost (and an optional ongoing support fee). In contrast, SaaS providers generally price applications using a subscription fee, most commonly a monthly fee or an annual fee. Consequently, the initial setup cost for SaaS is typically lower than the equivalent enterprise software.

SaaS guarantees easier, speedier and cheaper implementations. The fact that the software's upgrades are automatically pursued in a seamless manner (as there is no need to customize the software) ensures that users always access the most recent version of the program, without having to re-install it. In traditional packaged software solutions, this automatic upgrade is not possible, whereas in SaaS, the vendors just push the upgrades and updates out to the customer base.

2.2 Web scraping vs Web crawling

The process of creating a SaaS goes through different phases, of which the starting phase consists of data collection. In this phase, data is collected from several different sources, in the best quality manner. Without data collection, no transformation can occur, leading data to be one of the crucial resources that today's businesses have. Despite this, many still lack to give data the importance it should have.

There are many ways to gather information or data from the internet. Of those many ways, two of the most popular ones are web crawling (or data crawling) and data scraping (or web scraping). While both web crawling and data scraping are essential methods of retrieving data, the information needed and the processes involved in the respective methods differ in several ways.

Web scraping and crawling services can come in handy for retail industry at this juncture of technological growth in the retail space. These services provide the ability to extract data from any website and transform it into useful informational patterns and statistics. These techniques offer businesses competitive advantage of key industry trends that company needs to follow. Web scraping and crawling services hold significance in monitoring the physical market stores as well as the e-commerce variants. Web scraping services help in collecting retail data from different websites and use the collected data to perform market research and data analysis [6].

Web crawling combined with data mining helps retailers in extracting key trends and following websites used by competitors. Using these techniques, businesses are gaining wisdom from the strategies of their competitors, which help them to predict the future trends. Scrape retail data aid companies to gain visibility across purchasing markets, competitors and consumers, which eases in keeping pace with demand, even if it shifts quickly. Collecting and analyzing retail data aids in following latest retail industry trends. It helps in monitoring the competition and discovering major insights to enhance market share.

In order to understand the difference between the two methods, the subsection Section 2.2.1 and Section 2.2.2 explain each method, and finally the Section 2.2.3 explains the differences between them.

2.2.1 Web Scraping

Web scraping is a relatively new method for collecting online data. The term describes the automated process of accessing websites and downloading specific information, such as prices. Allowing the creation of large, customised data sets at low costs, web scraping is already applied for scientific and commercial purposes in many areas, such as marketing, industrial organisations, or inflation measurement. With the increasing number of prices published online and as online grocery retail is slowly gaining market share in many parts of the world, web scraping may be a promising alternative to get data for food price research [7]. The figure below gives a glance at the web scraping architecture.

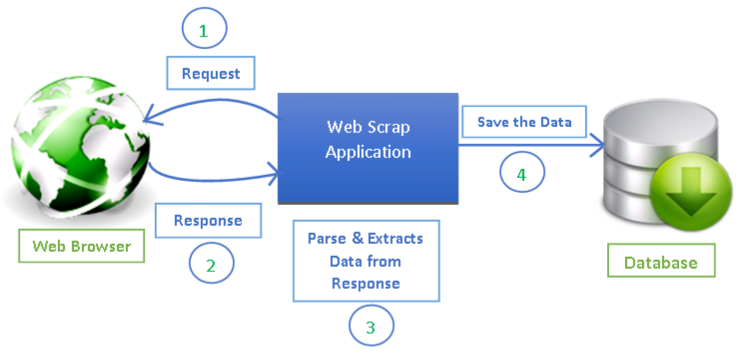


Figure 2.1: Architecture of a standard Web Scraping
Source: Pinterest - softprodigy.com [8]

2.2.2 Web Crawlers

On the other hand, Web crawling means accessing web content and indexing it via hyperlinks, thus, only the URL but no specific information is extracted. Instead, the full content is made available through the hyperlink but is generally not archived.

A Web crawler, sometimes called a spider or spiderbot and often shortened to crawler, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing (web spidering) [9]. They are called “web crawlers” because crawling is the technical term for automatically accessing a website and obtaining data via a software program.

These bots are almost always operated by search engines. The application of a search algorithm to the data collected by web crawlers enables search engines to provide relevant links in response to user search queries, by generating the list of webpages that show up after a user types a search into Google or another search engine.

Web crawlers are a central part of search engines, and details on their algorithms and architecture are kept as business secrets. When crawler designs are published, there is often an important lack of detail that prevents others from reproducing the work. There are also emerging concerns about “search engine spamming”, which prevents major search engines from publishing their ranking algorithms. The graph below illustrates the architecture of a web crawler.

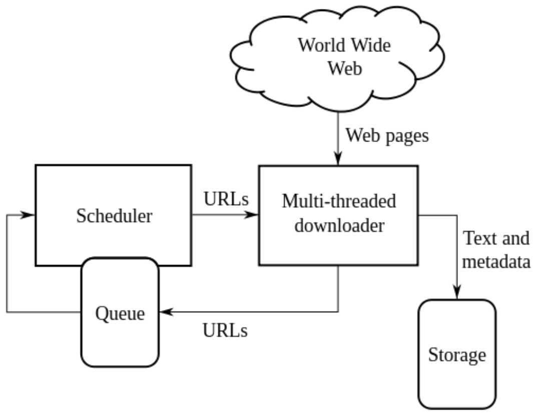


Figure 2.2: High-level architecture of a standard Web crawler
Source: bcc.ime.usp.br

2.2.3 The difference between web scraping and web crawling

All things considered, web scraping involves a script that accesses the websites hosting the data, finds the relevant, previously defined elements and then downloads, and stores them in structured data sets.

Table 5.3 shows the main differences between Web Scraping and Web Crawling.

	Web scraping	**Web crawling**
Process	Automatically requesting web documents and collecting information from them	Repetitively finding and fetching hyperlinks starting from a list of initial URLs
Target information	Pre-defined data on specific websites	URLs to access all kinds of information, depending on search request
Output	Downloaded data in structured format	Indexed hyperlinks, stored in database
Use	Data collection (e.g. price series)	Ad hoc requests (e.g. search engines, price comparison tools)

Table 2.1: Distinction between web scraping and web crawling

Figure 2.3 gives a schematic overview of the several steps required to build a web scraper. Technically, the script can be coded in most of the common programming languages, like python. Although such code elements are certainly helpful as building blocks, there is always a need to adapt the script to the targeted website, depending on how the website is set up, the page structure, authentication requirements, etc., because many websites use reactive elements, so simple HTTP requests cannot be used. This is especially relevant in large online food retail websites mostly have online catalogues but no application programming interface to access their food prices directly. Therefore, the script needs to navigate a “real” web browser and “click” through the website [7].

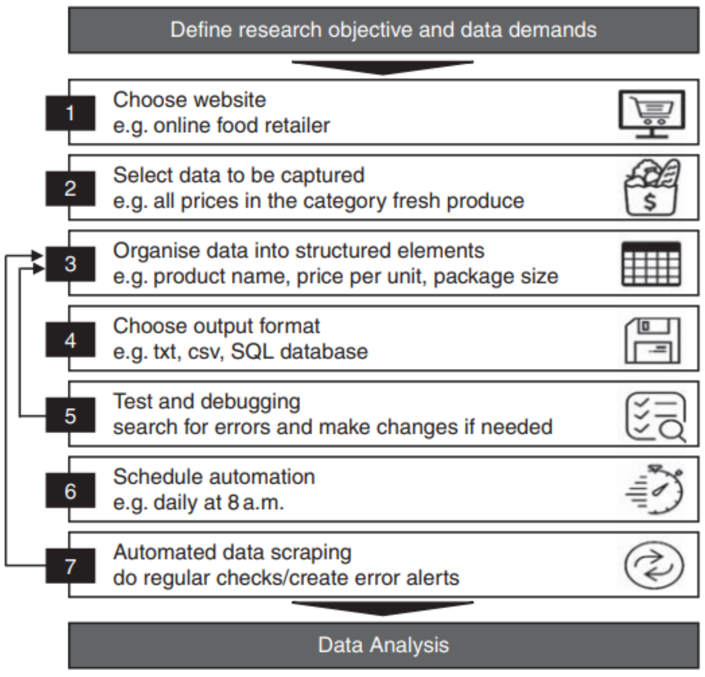

Define research objective and data demands
1
Choose website
e.g. online food retailer
2
Select data to be captured
e.g. all prices in the category fresh produce
3
Organise data into structured elements
e.g. product name, price per unit, package size
4
Choose output format
e.g. txt, csv, SQL database
5
Test and debugging
search for errors and make changes if needed
6
Schedule automation
e.g. daily at 8 a.m.
7
Automated data scraping
do regular checks/create error alerts
Data Analysis

Figure 2.3: Schematic web scraping procedure
Source: Representation of Judith Hillen [7]

2.3 Serverless Applications

Software as a Service has been transformed significantly with serverless computing. This is a boom in a continually shifting landscape of customers and load profiles, that essentially virtualizes runtime and operational specifics through a third-party provider. With serverless SaaS, developers can focus on their business logic and attain faster time-to-market [10].

Serverless is an application design and deployment paradigm that is event-driven and incorporates scalable cloud services as computing resources from a third-party provider. With serverless computing, developers can drop in code, create backend applications and event handling routines, and process data without concerns about servers, virtual machines (VMs), or the underlying computing resources which are maintained by the provider.

Traditionally, physical servers were prevailing, where with just a single core server, multiple physical devices were connected with the main database system. The server actually worked as a connecting link between those devices and the database. Serverless application security can be greater than that of a traditional web application due to the ability and leading practice of limiting access to resources on a function-by-function level. Given the complex nature of this task, permissions are most successfully implemented in serverless applications when considered as part of the design of the application and built in from the start. Serverless architectures offer significantly reduced complexity and engineering lead time. Service providers follow a pay-as-you-go model that accounts for the actual amount of resources consumed by the customers' applications without associating costs to idle, down-time.

Figure 2.4 is a diagram of a traditional model that relies on configured cloud servers compared to a serverless model.

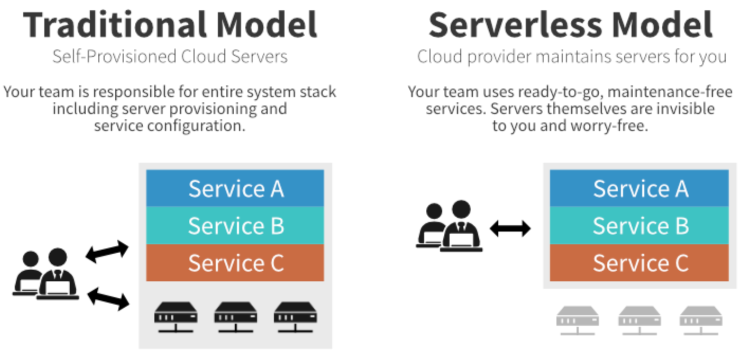

Traditional Model
Self-Provisioned Cloud Servers
Your team is responsible for entire system stack including server provisioning and service configuration.
Service A
Service B
Service C
Serverless Model
Cloud provider maintains servers for you
Your team uses ready-to-go, maintenance-free services. Servers themselves are invisible to you and worry-free.
Service A
Service B
Service C

Figure 2.4: Traditional vs Serverless Architecture
Source: zetavisual.com - cloud-architecture [11]

Serverless has wide applicability in retail, yet, its adoption must be incremental and well thought through. The first few use cases should be smaller and simpler processing, before aiming to target complex functions (such as with workflows and state management). Event-driven and pipeline-based architecture also lends itself well to the decoupling of processing and makes it easier to implement serverless functions [12].

2.4 Price Optimization in Retail

Pricing is an important decision-making aspect after the product is manufactured. Price determines the future of the product, acceptability of the product to the customers and ultimately the return and profitability from the product. It is a tool of competition [13]. Generally, pricing strategies include the following five strategies:

1. **Cost-plus pricing** - imply calculating company costs and adding a mark-up.
2. **Competitive pricing** - setting a price based on what the competition charges.
3. **Value-based pricing** - setting a price based on how much the customer believes the company is worth.
4. **Price skimming** - setting a high price and lowering it as the market evolves.
5. **Penetration pricing** - setting a low price to enter a competitive market and raising it later.

Particularly regarding competitive pricing, the business may sell its products at a price above or below such benchmark. Setting a price above the benchmark will result in higher profit per unit but might result in less units sold as customers would prefer products with lower prices. On the other hand, setting a price below the benchmark might result in more units sold but will cause less profit per unit [14].

The main dilemma for retailers when trying to accurately price their products is when they attempt to tackle this question: What is a fair price for this item considering the market, the current time of year, demand, and the product's attributes? The fact that these factors are constantly changing makes this question to become increasingly difficult to answer.

Pricing optimization software is suitable for businesses of all sizes, from an online clothing store to a supermarket chain. For this, a whole set of complex data is used. This data is obtained as a result of monitoring the price range in the market, the range of products available and pricing strategies conducted by competitors. For the success of this process, it is necessary to collect data from competitors, such as prices, discounts, and promotions. Successful price analysis and its strategy depend on the quality of the information collected. The more data sources a retailer uses to set prices, the more accurate they will become [15].

When implemented correctly, competitive based optimization solutions are powerful tools that offer retailers a competitive advantage, not only by having a tool capable of setting optimal prices and promotional products that achieve merchandising and marketing goals, but also by ensuring that the entire catalog is sold [15].

2.5 Product Match

In order to develop competitive based optimization approaches, companies need to match the products offered by the competitors with the products company offers. This match is a very difficult task. Retailers use product matching, which is basically a mix of algorithmic and manual techniques to recognize and match identical products from different sources. Many retailers try to do it themselves, but since it takes up so much time and resources, it can distract them from their core businesses. This is how the Product Match concept was born, which makes it possible to match the same product as sold by multiple retailers.

Product matching is a process in which one is given two products with all available data-points from each source and is able to determine if these products are identical. Thus, it aims to develop an algorithm that decides whether two product descriptions from different stores describe the same product or not.

Product matching can be done manually or automatically. In manual product matching, a team goes through many websites and tries to find a perfect match for the wanted product. When performing this task, the team does not only check product names, but also technical specs and product images in order to ensure that a match is indeed a proper one. Contrarily, automated product matching (automatch) is a continual automated product matching process based on an algorithm [16].

In the development of this project, the Product Match concept is used to assist the retailer in market analysis, by allowing for the easy comparison of offers held by different vendors. The development of the Product Match algorithm is done before using Advance Analytics. It is necessary to make a match between competitor products and original products, to make it possible to study and analyze the data resulting from the Product Match algorithm.

2.6 Summary

Retailers that make the transition to SaaS can look forward to transforming every aspect of their organization for the better, from the customer experiences they provide, to how their employees work, to the underlying business processes which support their operations. SaaS is a perfect solution, because the applications would connect to other components easily, thus augmenting the systems at little cost.

Furthermore, with the help of web scraping experts, companies can increase the chances to collect the necessary data to do better business, sales, inventory and marketing decisions. Posteriorly, analyzing web data helps companies derive insights as to how to increase traffic and sales, manifest ideal conversation rates, and ultimately, lead to closure in sales. Including data matching and record linkage tools in companies' data quality management framework is one proven way to minimize the negative impact of bad data. Data matching tools automate the process of segmenting the raw data through multiple layers, profiling, cleansing, deduplicating, and merging it for accurate insights through analytics.

Chapter 3

State of the Art

This chapter describes the state of the art of the project. Firstly, section 3.1 introduces price comparison platforms and it current state of art. Secondly, Section 3.2 describe the current state of the art of the product match concept. Thirdly, Section 3.3 describes the power of advanced analytics and price monitoring tools and its current state of art. Finally, Section 3.4 summarizes this chapter.

3.1 Price Comparison Platforms

In today's hyper-competitive e-commerce scenario, companies grab every opportunity to attract potential customers. Price comparison websites are one of the opportunities e-retailers leverage to attract customers. A few years back, a typical online buyer looking to buy apparel or an electronic gadget would scour the web to compare prices from different e-commerce sites and look for the best deals available. Prices of the same commodity would differ over different sites which in turn prompted sites to monitor price changes in real-time using analysts.

Price comparators are tools that enable users on the internet to identify different product prices practiced by different e-commerce stores. By doing so, these comparators help consumers through their purchase journey, allowing them to find the best deals and offers [17].

On the consumer perspective, price comparison websites work in a similar manner as a typical search engine. The main difference held in price comparison websites is the fact that they will show products side-by-side, and additionally, display how these products stack up against one another in terms of pricing. In contrast, typical search engines generally serve up straightforward results without sorting these by price or comparing them to other products. On the business side, each price comparison site is a little different. Essentially, companies pay a certain fee, submit the required content (product photos, URL, price, description, etc.), and list their products. Companies products will then show up for relevant searches along with comparable products, ranked by price. These platforms are useful to inform the different pricing strategies applied in the market, which is key for businesses. Companies do not want to be in an uncompetitive price position, charging much higher prices than their competition, for similar products [18]. Price comparison mechanisms help retailers to increase brand visibility, allowing them the ability to compare different prices and promotional activities to those of competitors, while also enabling them to identify how popular their products are versus their competition.

Based on various internet searches, including articles that evaluate the popularity of different Price Comparators [18] [17] [19], PriceGrabber [20], and KuantoKusta [21] are described bellow.

3.1.1 Price Grabber

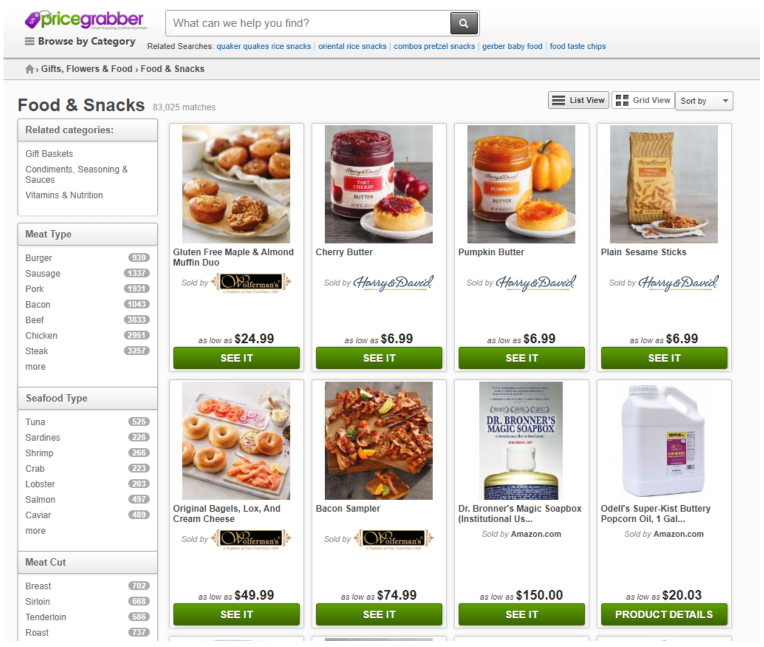


Figure 3.1: pricegrabber.com

PriceGrabber is a leading distributed e-commerce platform and shopping site that connects millions of shoppers each month with thousands of merchants. Consumers use PriceGrabber.com to shop and compare products prices from different merchants. In addition to being a comparison shopping site, PriceGrabber is a leading distributed e-commerce platform with a network of over 400 digital publisher partners. PriceGrabber is a unit of CPL Holdings LLC and headquartered in Los Angeles, CA with operations across North America and in the United Kingdom [22].

Price Grabber presents on the main page, with a top-down approach (Appendix A): a bar at the top with small menus of auxiliary information to the entire site, a search bar, the categories of products and services, two advertising banners, most popular categories and, in the bottom, contact information, support and more.

The user can start looking for a product through the search bar, categories, or also use the advertising banners that take him directly to the product's shopkeeper, without the opportunity to make any kind of comparison. However, if the search is done by normal means, that is, through the search bar or the categories, the user will be presented with an interface with a list of possible products to compare. The difference between PriceGrabber and KuantoKusta, is that it adds a way of comparison, where the user can choose up to four products from the presented list to compare them side by side. In addition to the list of products, each product on the list features has a button to view the product directly in the store, or compare it directly with other similar products from different e-commerce stores, and in turn, this entire process is followed by a sidebar with options for filtering product characteristics, such as: price, brand and others.

3.1.2 KuantoKusta

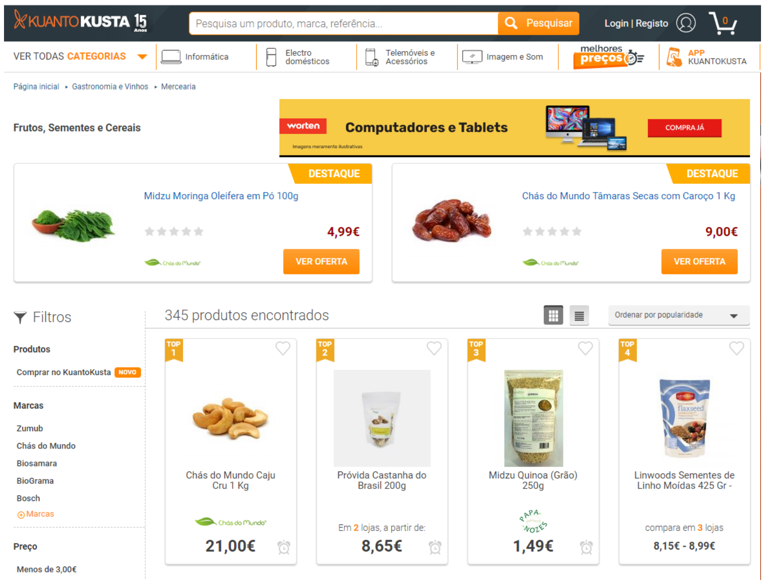


Figure 3.2: kuantokusta.pt

Kuantokusta helps consumers to compare the products displayed by traders, in order to find the best offers of products and services available. It is the Portuguese leader of price comparators.

The user can start searching for a product using the search bar, advertising banners and/or categories. If the user uses the search bar, he will be presented with the results interface with a product listing. If the user uses advertising banners or categories, he will be redirected directly to the page where the product is sold. The figure 3.3 shows the result of a search, that presents a list with products of the same kind, ordered by the best price, with the logo and hyperlink of the store that sells the product.

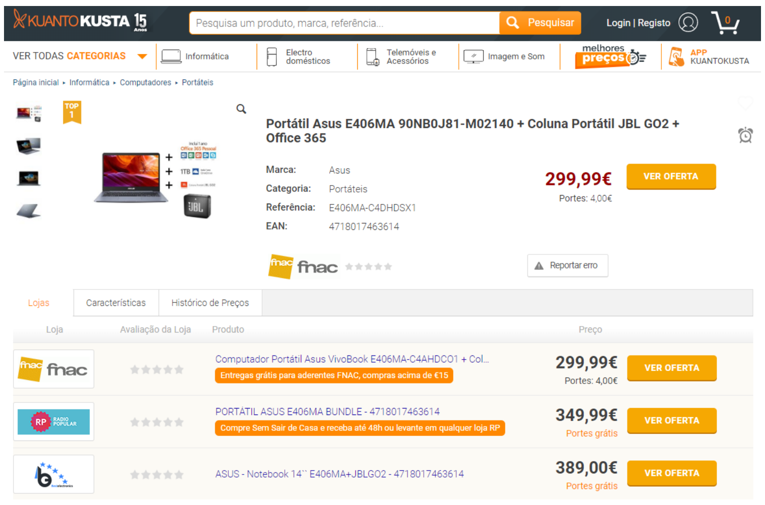

KUANTOKUSTA 15 Anos
Pesquisa um produto, marca, referência...
Pesquisar
Login | Registo
VER TODAS CATEGORIAS
Informática
Electro domésticos
Telemóveis e Acessórios
Imagem e Som
melhores preços
APP KUANTOKUSTA
Página inicial • Informática • Computadores • Portáteis
TOP 1
Portátil Asus E406MA 90NB0J81-M02140 + Coluna Portátil JBL GO2 + Office 365
Marca: Asus
Categoria: Portáteis
Referência: E406MA-C4DHDSX1
EAN: 4718017463614
299,99€
Portes: 4,00€
VER OFERTA
fnac
Reportar erro
Lojas
Características
Histórico de Preços
Loja
Avaliação da Loja
Produto
Preço
fnac
Computador Portátil Asus VivoBook E406MA-C4AHDCO1 + Col...
Entregas grátis para aderentes FNAC, compras acima de €15
299,99€
Portes: 4,00€
VER OFERTA
RP RADIO POPULAR
PORTÁTIL ASUS E406MA BUNDLE - 4718017463614
Compre Sem Sair de Casa e receba até 48h ou levante em qualquer loja RP
349,99€
Portes grátis
VER OFERTA
ASUS - Notebook 14'' E406MA+JBLGO2 - 4718017463614
389,00€
Portes grátis
VER OFERTA

Figure 3.3: Search results from KuantoKusta

3.2 Product match for building a price comparison platform

To build a price comparison engine, it is necessary to compare and display the data. There are many ways to collect data for comparison. Price comparison sites can collect data directly from merchants. Retailers who wish to list their products on the website, provide their product and price lists through commercial feeds, which are compared to the original database. As some sites provide data through Application Programming Interface (APIs), it is only necessary to invoke suitable endpoints to get data. Scraping the Web, which was discussed earlier, is another option for collecting data that focuses more on transforming unstructured data on the Web to a structured result. Many companies today opt to invest certain resources in collecting information about their competitors from the web and other channels. This is a routine procedure that companies tend to do, in order to keep track of what their competitors are doing, what products and services they offer, and any news that concern about them. This is usually done by the marketing team through manual navigation.

After collecting data from the different stores, it is necessary to make a correspondence between the different products. Most of the solutions discussed above use the product match concept to match products. Product matching aims to build algorithms that perform the intelligent task of deciding whether two products information from two different stores describe the same product or not.

Matching a seller listed item to an appropriate product has become a fundamental and one of the most significant steps for e-commerce platforms [23]. Lots of studies have been focused on structuring the products inventory by extracting and enriching attribute-value pairs from various sources, and feed them into a matching function [24] [25]. In addition, there are also several studies that use machine learning to solve the problem, as is the case with the solutions presented in the sections below. These sections have been divided into two different approaches: Product matching based on feature matching and Product matching based on binary classification.

3.2.1 Product Matching based on feature matching

Products can be described in terms of their features such as brand, color, size, etc. Using different attributes that are available in different sources, it is possible to make a match between attributes and discover the similarity between products. For example, titles of matching products may not be identical but contain semantically alike tokens. On the other hand, mismatching products may differ on a single attribute and consequently their corresponding titles may differ by as little as one character. High price differential may be a strong indicator of a mismatch but by itself is rarely conclusive since identical products may have highly varying prices across different sellers while certain kinds of mismatched products may still have very close or equal prices [26].

Ajinkya More [26] proposes an approach that leverages product information that the author deem the most reliable — title, description, images, and price to arrive at a matching decision. The system consists of several components and depending on the specific use cases, some or all of these components may be leveraged.

Neural networks and deep learning techniques are used to identify and learn from both similarities and differences, and to create word-level embedding to the creation of a system of representation for common words. A distinctive feature of deep neural networks is that they learn to extract what is invariant about data with respect to a given task. For instance: a neural network that should recognize cats on photographs can learn what pictures with cats all have in common — the “cattiness” of an image [27]. Petar Ristoski et al.[28], use neural language models and deep learning techniques in combination with standard classification approaches for product matching and categorization. The authors inspect the use of different models for three tasks: (1) Matching products described with structured annotations from the Web (2) Enriching an existing product database with product data from the Web, and (3) Categorizing products. For those tasks, the authors employ Conditional Random Fields for extracting product attributes from textual descriptions, as well as Convolucional Neural Networks to produce embeddings of product images.

On the other hand, Rosie Hodd [29] uses machine learning approach to provide high accuracy and scalability with the world’s single biggest source of real-time retail data. Natural language processing (NLP) is a sub-field of machine learning, which aims to understand how computers analyze human language. One common NLP task is word embedding, mapping words to vectors that represent them, and many models exist to perform this task. At their most complex, this can be a model trained on millions of words or phrases to produce fixed length vectors based on features learned during training, where the representation of a word is distributed across several elements in a vector.

Juan Li et. al [30] proposed a neural product matching model by considering the respective characteristics of product titles and attributes, and the authors cast the problem as ranking and classification scenarios. The authors also tried different classical classification algorithms and ranking algorithms to combine their neural matching features and some other features.

Finally, Cenk Çorapcı [2] implemented a model capable of recognizing whether two product titles are the same or not. The model consisted of an embedding layer that learns character embeddings for every character input sequence on the fly, two siamese layers of one dimensional convolutions, a concatenation layer for the outputs of the convolutional layers and feed forward layers that later connect to a single sigmoid output. The output neuron gives a similarity score between 0 and 1, 1 meaning the two titles are same. The results were pretty promising and had a great likelihood of being successful.

3.2.2 Product Match based on binary classification

Another widely used approach to compare two products and assess whether they are identical or not is the classification method. In this scenario, two product descriptions are given, including their titles and attributes and the result is in two classes: 1 - represents the two products are identical and 0 - represents that they are different.

A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a dataset [31]. In machine learning, classification problems are one of the most fundamentally exciting and yet challenging existing problems. The implications of a competent classification model are enormous. These models are leveraged for natural language processing text classification, image recognition, data prediction, reinforcement training, and a countless number of further applications.

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

Two denominations are possible, depending on the number of possible classes in the output - one used when the output is superior to two, and another one used when there are only two possible classes. In the first output case, each matrix row represents the number of instances in a predicted class, while each column demonstrates the number of instances in an actual class (or vice versa). In the second output case, the matrix displays two rows and two columns. These report the number of false positives, false negatives, while representing also the number of true positives and true negatives [32].

Semantics3 [33] defends that the basic immutable rule of product matching is to minimize false positives. All algorithmic end-goals point to this result over time in an iterative fashion. A false positive is generated when the algorithm indicates that two products are identical, but are not.

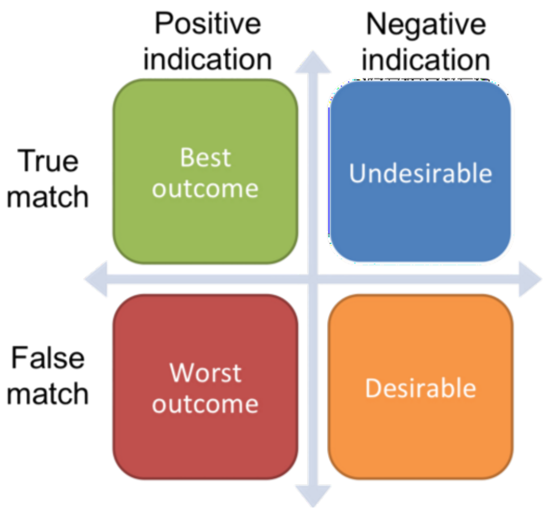


Figure 3.4: Classification: True vs. False and Positive vs. Negative

In order to minimize the false positives, the author employs nine different strategies in combination, as a series of "safety valves". A product is pushed through this series of safety valves if it passes the test (Yes or No), or if the algorithm lacks sufficient data to make a judgement (Don't know). The product-matching "safety valves" include the following: image matching, brand normalization, unique identifier (EAN checks) and keywords similarity.

The governing principle behind safety valves is to invalidate dissimilar features in products. A product that fails the test is immediately discarded. Additionally, some checks can be difficult to do and analyse, like image matching, keyword similarity checks and brand. Feature normalization are not straightforward, different retailers would use different formats and nomenclature to describe products.

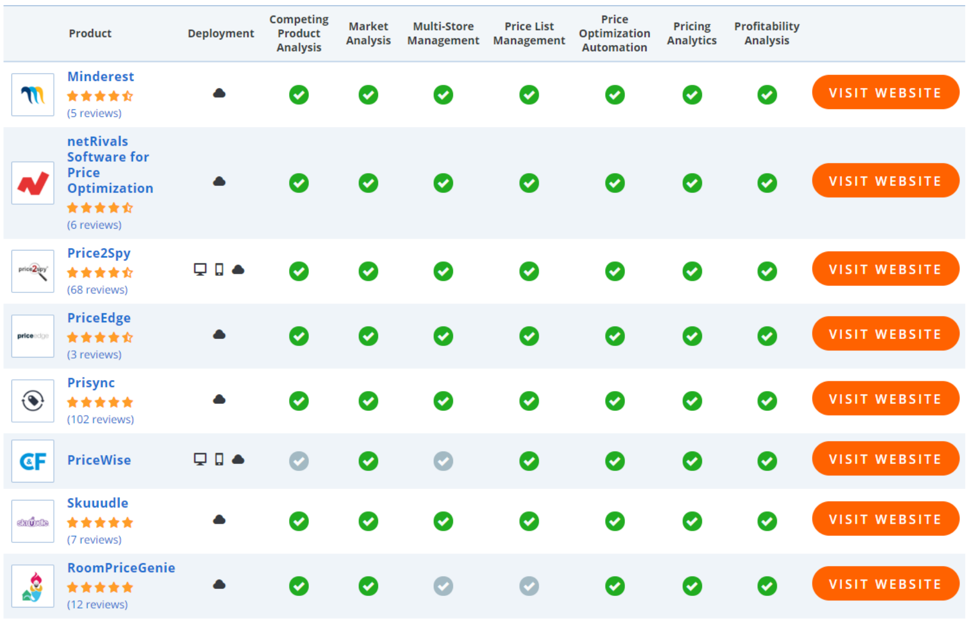
3.3 Competitor Analysis tools

The end result of this project is a dashboard with real-time analysis of what is happening in the market. Although this may seem similar to the price comparison tools, it differs from these in the sense that it is more focused for retailer use than for customer use, having a more analytical component, aiming to understand the behaviour of products, promotions and prices. The tools as the one created throughout this project are named Competitor Analysis (CA) tools, and play a crucial part in the strategic planning process. Spying every move of the competitors, offers companies an advantageous position in decision-making. After collecting data from different platforms and matching products it is necessary to analyse what the competitors are doing and what products and services they offer. In the past CA was conducted manually and it was a tedious process. For e-commerce businesses in particular, a competitive analysis can help retailers identify industry trends and determine price strategies [34].

According to Forrester Consulting [35], 81% of buyers compare the offers of several stores in search of a better bargain. Retailers that can collect and analyze market data, map their position against competitors, and offer optimal prices are the ones who catch these buyers' eyes first and foremost.

Competitor price monitoring tools uses data science and machine learning to produce competitive pricing by considering a number of internal and external parameters such as competitor pricing, market, demand and supply. Price monitoring is the action of consistently keeping an eye on competing for product prices that match with companies assortment in a systematic and organized way.

As we can see in the figure 3.5, there are plenty of pricing analysis software solutions available on the market today [36].



Product	Deployment	Competing Product Analysis	Market Analysis	Multi-Store Management	Price List Management	Price Optimization Automation	Pricing Analytics	Profitability Analysis	
Minderest ★★★★½ (5 reviews)	☁	✅	✅	✅	✅	✅	✅	✅	VISIT WEBSITE
netRivals Software for Price Optimization ★★★★½ (6 reviews)	☁	✅	✅	✅	✅	✅	✅	✅	VISIT WEBSITE
Price2Spy ★★★★½ (68 reviews)	🖥 📱 ☁	✅	✅	✅	✅	✅	✅	✅	VISIT WEBSITE
PriceEdge ★★★★½ (3 reviews)	☁	✅	✅	✅	✅	✅	✅	✅	VISIT WEBSITE
Prisync ★★★★★ (102 reviews)	☁	✅	✅	✅	✅	✅	✅	✅	VISIT WEBSITE
PriceWise	🖥 📱 ☁	✓	✅	✓	✅	✅	✅	✅	VISIT WEBSITE
Skuuudle ★★★★★ (7 reviews)	☁	✅	✅	✅	✅	✅	✅	✅	VISIT WEBSITE
RoomPriceGenie ★★★★★ (12 reviews)	☁	✅	✅	✓	✓	✅	✅	✅	VISIT WEBSITE

Figure 3.5: Pricing Tracking Tools
Source: capterra.com - Pricing Tracking Tools [36]

Even though their complexity, accuracy, and reliability may vary, the three fundamental processes underlying the work of pricing analysis software are the same. These include: (1) Data collection, (2) Procession and analysis and (3) Recommendations delivery. There is no single set of features and benefits offered by software vendors. Choosing a specific solution depends primarily on a retailer's business goals. Many of these price analysis software offer many benefits for retailers, such as: customized data sets collected in real-time, highly accurate exact and similar product matches, and total visibility of market changes.

Revionics is an American-based pricing software provider for retailers and specializes in AI-based pricing software. Revionics Price Optimization utilizes AI, data, and business strategies to offer optimal prices for products. They offer clients the ability to customize categories, channels, and items according to their business needs. Revionics price optimization software provides insights such as consumer demand forecasts, price sensitivity reports, changes in costs, and competitive price positions which help retailers set optimal prices for their products. Their price optimization software can help companies avoid price wars, which is a common issue in highly competitive retail markets. By giving a detailed insight into competitor price changes and the impact this can have on-demand and sales, Revionics can make companies more competitive across all channels. Revionics promises transparency into what rules are used for price recommendations and gives a comprehensive "confidence score" of the recommendations given. Ultimately the user decides whether or not to accept the price recommendations given by their price optimization software, but Revionics assures that the price recommendations their software provides will result in profit and customer satisfaction [37].

In price monitoring, the online supermarket sector faces a different paradigm than the rest. This sector contains an almost infinite catalog size and a product stock that is as variable as it is sensitive, meaning that online supermarket price comparison must be treated in a completely different way. In this type of market not only other supermarkets participate, but other giants of Internet sales are also sticking their noses in and competing in the shipment of household products and groceries to consumers' homes. In any case, whatever competition the supermarket faces, the main stumbling block that must be overcome in this area is the numerosity of the products on sale. This makes it practically impossible to track prices by hand and preventing, in many cases, the possibility of maintaining competitiveness in the face of certain offers and price reductions. Minderest's supermarket price comparison software is capable of matching products by equivalence even if they are not identical to each other. In this way it is possible to make an effective price comparison between the different products of each supermarket [38]. Minderest is already trusted by major national and international accounts, such as Carrefour and the Auchan Group. One of the strengths of Minderest's software is its supermarket-specific matching technology. With this, the software can match fresh products by similarity or equivalence, as well as private label products. Matching own brands is one of the most complex, as it requires the identification of the alternatives to be covered by each product, so it is undoubtedly a high added value for the tool. Online supermarkets now also need to adapt to the new online competitiveness, and to do so it is necessary to have the right technology to keep them in the battle to attract, capture, and retain customers.

3.4 Conclusion

Although there are several data collection and analysis platforms on competitors' products, operating in diverse industries throughout many cities in Europe, there is yet to be one to focus in the Portuguese food retail sector. These platforms require extremely high maintenance - starting right from the simple data collection and analysis from all the different websites. Each retailer presents their product information in a different way, and the information may change from day to day, thus, requiring a high level of maintenance. The development of this project is still an innovative one in Portugal, regardless of the fact that the product may be transversal to any country or industry. Despite this, its proximity to the Portuguese market and business models makes it more suitable to fit the Portuguese retail sector, for which it was developed. An example of this strong suitability related to the fact that the display of data and strategies shifts from market to market, and from retailer to retailer - having been developed specifically for Portugal, this product brings a significant advantage in Portugal when compared to similar products existent around Europe, due to the fact that no one actually knows the Portuguese market as well as Portuguese people themselves.

Chapter 4

Implementation

The main focus of this project is aimed at providing Portuguese retailers in the online food sector with real-time information regarding what is happening in the market. In order to do so, the development of a SaaS that is transversal to any retail company is the main goal. By utilising this SaaS, retailers will be able to analyse price evolution over time, and consequently, discover which price strategy each of their competitors applies and which companies are monitoring each other. Through the development of the project, the most significant challenge faced is to match the products from different retailers, as it is not trivial and requires a thorough analysis of the different products. When completed, the project's output should be an iterative dashboard that allows the retailer to understand their market position: analysing prices and category composition, coverage of assortments and categories, comparison of promotions, and framing in the product portfolio.

The following chapter will enable a better understanding of the implementation of this SaaS, by describing the entire solution developed for the problem mentioned above. Initially, the high-level overview of the developed solution is described, briefly covering the technologies and methodologies chosen to make the whole system function. Posteriorly, each of the software components is thoroughly analyzed so that the project's architecture and algorithms are clear. Doing so will enable the understanding of how the whole system will be able to deliver the proposed features.

4.1 Software Overview

This software has been designed to automatically collect product data from different online retail stores and implement a product match capable of identifying a match between competitor products.

The development of this SaaS is divided into four phases: (1) data collection from different online stores, (2) data cleaning and standardization, (3) product match and (4) the implementation of an analytical layer. This last layer is developed after the completion of the product match algorithm, and is implemented through the use of the Microsoft Power BI service [39]. This service is used to find insights about the data. Power BI can help connect disparate data sets, transform and clean the data into a data model and create charts or graphs to provide visuals of the data.

The software is developed using Python and AWS technologies. Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 175 fully featured services from data centers globally. Millions of customers — including the fastest-growing startups, largest enterprises, and leading government agencies

— are using AWS to lower costs, become more agile, and innovate faster. Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing [40].

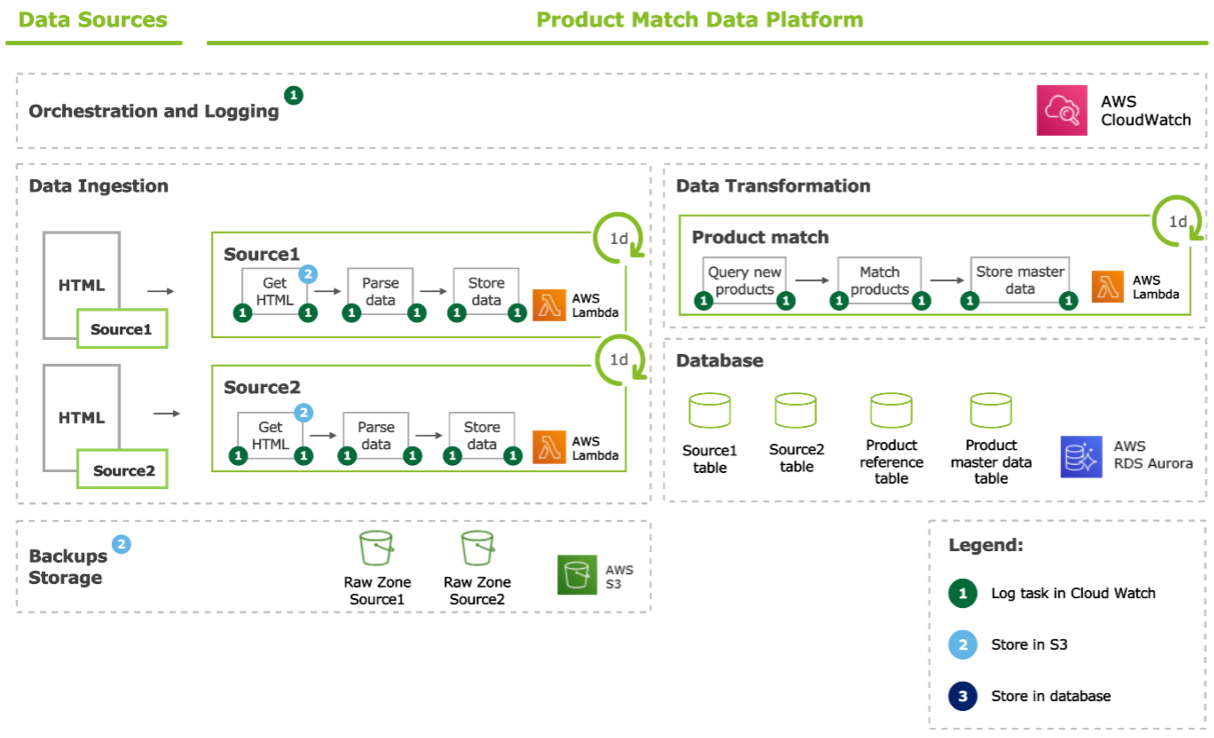


Figure 4.1: Solution Architecture

Figure 4.1 is a diagram of the project architecture, highlighting the two main processes: data ingestion and product match. To better comprehend the layout, the technologies used are explained below. Each process is executed through AWS *lambda*. **AWS *lambda*** is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. It is a computing service that runs code in response to events and automatically manages the computing resources required by that code. For each of these processes represented by *lambdas* to be executed daily, the **AWS CloudWatch** service was used as a scheduler (running the task once a day) and a task recorder (event manager). CloudWatch collects operational and monitoring data in the form of logs, metrics, and events, providing a unified view of AWS resources, applications, and services that run on AWS and on-premises servers. This resource is represented at the top of figure 4.1, and all processes using this service are identified with the corresponding number in the legend (1). Data from different retailers' websites collected daily is saved as it comes from the source in the **Amazon S3 bucket**. Amazon Simple Storage Service (Amazon S3) is an object storage service. This process is represented in the "Backup Storage" section identified in Figure 4.1, and all methods using this service are identified with the corresponding number in the legend (2). After that, the data is processed as desired and saved in a PostgreSQL database. This process is also identified in Figure 4.1, labeled "Database", and all methods using this service are identified with the corresponding number in the legend (3). To make the configuration of this database with the AWS service easier, **Amazon RDS** was used. Amazon RDS supports a variety of database mechanisms for storing and organizing data and helps with database management tasks such as migration, backup, recovery, and patching. The integration of these processes is made easy by the fact that it is a serverless application. The Serverless Framework was designed to provision AWS *lambda* Functions, events, and infrastructure resources safely and quickly. It does this via a couple of methods intended for different types of deployments.

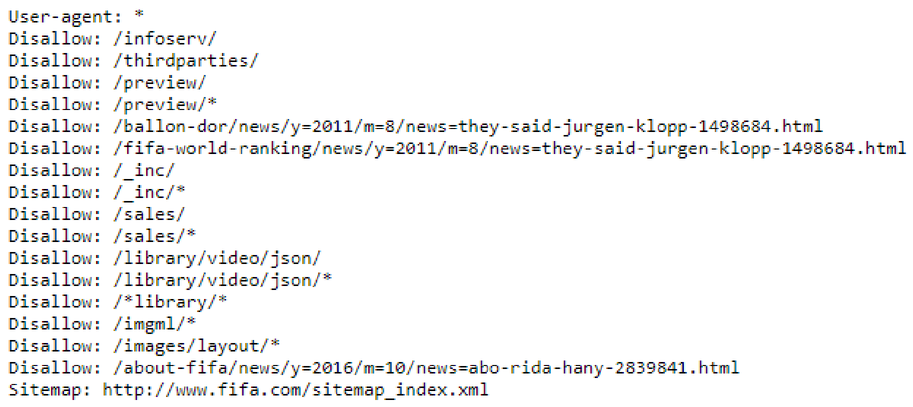
4.2 Main Processes

For a better understanding of the architecture, the following sections describe the implementation of each of the processes mentioned above.

4.2.1 Data Ingestion Process

The development of the Product Match algorithm begins with the collection of information through web crawlers and feeds. This is a challenge, because different retailers use different attributes and categorization. To collect information from different websites, two methods were used: web crawler and web scraping. The web scraping technique was used to extract structured information from a web page, usually by means crafted specifically for the target website. If it is not possible to find a fixed list of URLs to copy, because some sites protect this information, the web crawler technique was used, which finds and searches for web links from a list of starting URLs. From that initial starting point, the crawler will go through the pages of a website, following the links, finding new pages and extracting content relatively indiscriminately.

When collecting data from a web, it is necessary to check the authorization to do so. The privacy policy and the Terms and Conditions may contain some information about scraping the web, however, the best source is *robots.txt*. It is a file that indicates which parts of the site cannot be scrapped. The information contained in the file is not legally binding. However, one must respect the website owner's wishes and adapt to the information contained therein. The file is stored at the root of the website and it is necessary to add */robots.txt* at the end of the link to access it (example: www.example.pt/robots.txt).

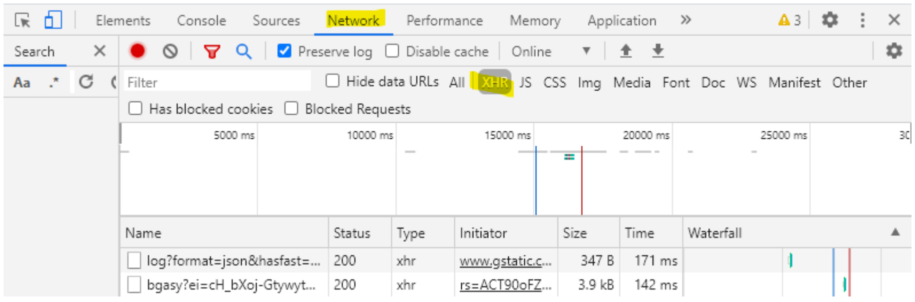


```
User-agent: *
Disallow: /infoserv/
Disallow: /thirdparties/
Disallow: /preview/
Disallow: /preview/*
Disallow: /ballon-dor/news/y=2011/m=8/news=they-said-jurgen-klopp-1498684.html
Disallow: /fifa-world-ranking/news/y=2011/m=8/news=they-said-jurgen-klopp-1498684.html
Disallow: /_inc/
Disallow: /_inc/*
Disallow: /sales/
Disallow: /sales/*
Disallow: /library/video/json/
Disallow: /library/video/json/*
Disallow: /*library/*
Disallow: /imgml/*
Disallow: /images/layout/*
Disallow: /about-fifa/news/y=2016/m=10/news=abo-rida-hany-2839841.html
Sitemap: http://www.fifa.com/sitemap_index.xml
```

Figure 4.2: Example of robots.txt file

Figure 4.2 gives an example of a *robots.txt* file. The function of the *User-agent* command is to list which robots must follow the rules indicated in the *robots.txt* file. The *Disallow* and *Allow* is a directory or page, referring to the root domain, which may or may not be tracked by the aforementioned user agent.

The next step is to request page content. One of the approaches used to scrap information was to send an HTTP request, usually via Requests, to a web page, and then analyze what is returned to access the desired information. However, this type of approach is not always possible, as it depends on how the web framework was developed. There are two sides of web development – the server side and the client side. A server side framework typically uses a programming language with a compiler and runs on a web server, such as Node and PHP. On the other hand, the client-side framework is usually a JavaScript library and runs in a web browser. In this case, the server sends the static content (the HTML, CSS, and Javascript), but the HTML is only a template - it does not hold any data. Separately, the Javascript in the server response fetches the data from an API and uses it to create the page client-side. With a client-side application, the browser is doing much of the work. In this case, the browser renders HTML, so it is possible to use it to see where the data comes from using its internal developer tools, which gives developers access to the inner workings of the browser and web applications. If the website loads data dynamically, it typically uses XHR (XMLHttpRequest). This is the type of request used to fetch XML or JSON data. It is used to send HTTP or HTTPS requests directly to a web server and load the server's response data directly back into the script. Figure below shows how these methods are found in the programmer's tools.



☑ Preserve log ☐ Disable cache ☐ Hide data URLs ☐ Has blocked cookies ☐ Blocked Requests

Name	Status	Type	Initiator	Size	Time
log?format=json&hasfast=...	200	xhr	www.gstatic.c...	347 B	171 ms
bgasy?ei=cH_bXoj-Gtywyt...	200	xhr	rs=ACT90oFZ...	3.9 kB	142 ms

Figure 4.3: Find XHRs requests using Chrome developer tools

Inspecting the developer tools is important to find out if it is possible to collect data from a retailer with a simple HTTP request. When it is possible to find XMLHttpRequests that returns all the necessary information, it means that the web scraping technique was used. To extract this data it is necessary to grab the request URL an use it to make an HTTP request. Different APIs return different types of data. For instance, some might send XML and others in JSON. The results can be stored in an array of arrays and others in an array of maps or dictionaries and some may not even return column headings. Things change between websites and it is necessary to analyze and treat the information individually. At the end of this process, as was said earlier, the product data in JSON format is stored in an AWS S3 Bucket and then the resulting objects are parsed and inserted into the database as desired.

Alternatively, when it was not possible to capture the request to the API, the web crawler method was used. In this case, Scrapy, a free web tracking structure and open source developed in Python, was used.

One of the biggest problems that the web crawler can face is being blocked by the website it is scraping. Many large sites have software to detect when there is a suspicious number of requests coming from an IP address, as this usually indicates some kind of automated access - it could be a scratch or something related to security. In this way, proxies were used to ensure that the crawler is not blocked or banned. They act as a bridge between the origin and destination of a request, as shown in the image below.

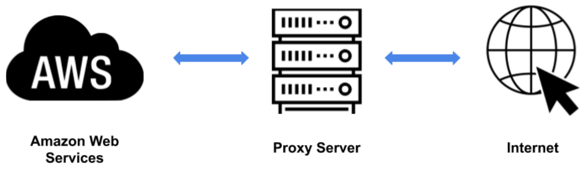


Figure 4.4: Proxy connection

In addition to masking the original IP address, another major benefit of using proxies with a web crawler is getting past rate limits on the destination site. That is, if many requests arrive from an IP address in a short period of time, the site will return some type of error message to "block" future requests from that client for an IP address for a while. Since more than a few thousand pages of content are being ingested from a major destination site, rate limits will likely be encountered at some point. To circumvent this type of restriction, a large number of requests were distributed evenly over a large number of proxy servers. Then, the destination website will see only a few requests coming from the IP address of each individual proxy server, which means that all of them will remain below the rate limit, while the crawler is still able to ingest the data for many requests at once.

The data ingestion and parse process were monitored using AWS Step Functions, which allows the coordination of multiple AWS services in serverless workflows. The process of each source is carried out in parallel. For each of them, the process is the same: first the data is collected and then the data is parsed. If data collection fails, the information is not parsed. The best time to access the site was studied, and through the AWS CloudWatch service mentioned above, it was possible to monitor this process and every day at the same time this step function runs.

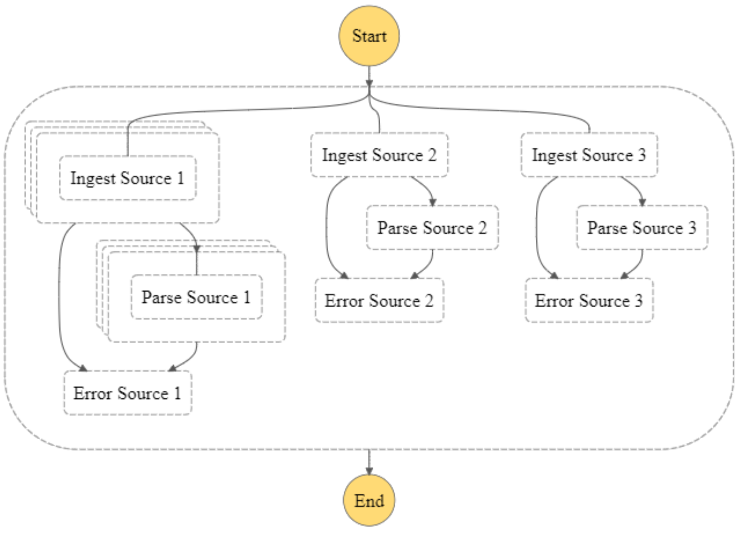


Figure 4.5: Ingest and Parse Process - Step Function

Figure 4.5 shows the execution of this process. The step functions record the status of each step, so when something goes wrong it is possible to quickly diagnose and debug problems. In this case, when the ingest process fails, the parse process is not performed and ends in an error state. And the same goes for the parse process, if it fails, the process ends in an error state and the developer receives an e-mail and a message on the cell phone, notifying them that the process has failed. In this way, errors can be corrected daily and there is a guarantee that no errors remain to be corrected. This is possible through the use of the **Amazon Simple Notification Service (Amazon SNS)** that enables message filtering and fanout to a large number of subscribers, including serverless functions, queues, and distributed systems.

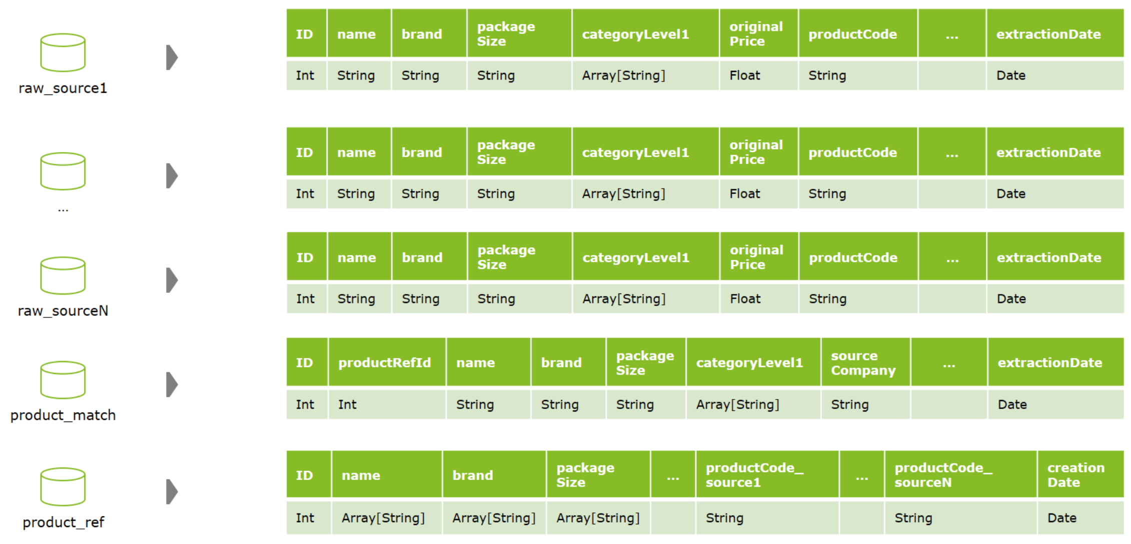
One of the biggest drawbacks of using *lambda*s is that they have a time limit of 15 minutes. Often this time is not enough to extract data from a source and in these cases the data was extracted and/or parsed by category. That is why in Figure 4.5 the data ingestion process from source one, has more than a rectangle (*lambda*). It was done as an iterative process, called map state. Map state can be used to run a set of steps for each element of an input array. While the parallel state executes multiple branches of steps using the same input, a map state will execute the same steps for multiple entries of an array in the state input.

4.2.2 Data Transformation Process

The data transformation process goes through three phases: (1) creation of a database that supports the Product Match, (2) cleaning and standardization of data (3) development of a product match algorithm. For a better understanding of the problem, each of the phases will be described below.

4.2.2.1 Database

A database was created in order to aid in the construction of the algorithm. To begin with, as previously mentioned, each of the retailer's collected website has been stored on a table created for that purpose. Each of these tables from different retailers are identified in *raw_ source 1 to N*, as is illustrated in Figure below.



ID	name	brand	package Size	categoryLevel1	original Price	productCode	...	extractionDate
Int	String	String	String	Array[String]	Float	String		Date

ID	name	brand	package Size	categoryLevel1	original Price	productCode	...	extractionDate
Int	String	String	String	Array[String]	Float	String		Date

ID	name	brand	package Size	categoryLevel1	original Price	productCode	...	extractionDate
Int	String	String	String	Array[String]	Float	String		Date

ID	productRefId	name	brand	package Size	categoryLevel1	source Company	...	extractionDate
Int	Int	String	String	String	Array[String]	String		Date

ID	name	brand	package Size	...	productCode_ source1	...	productCode_ sourceN	creation Date
Int	Array[String]	Array[String]	Array[String]		String		String	Date

Figure 4.6: Database

Added to this, two tables were created, namely *product_ match* and *product_ ref.*

The **product_ref** table stores all the products that exist in the market - each row represents a different product and the retailers that have this product are identified in their respective column. These columns are identified in the table by *productCode_ source1* to *productCode_ sourceN*. The value of this attribute in the row corresponds to the product code used for the retailer. This table is used to assign a unique identifier to each product in the market and to find the match between the products. On a daily basis, the products of each retailer try to find the match with the products that exist in *product_ ref.* If it exists, that product is changed by adding that retailer's product code in the corresponding column. Otherwise, it means that this product is not yet identified as a product in the market and therefore a new product/row is added to this table. It is crucial that this table is able to store enough information to find the match between products, and thus, only the relevant attributes to match are stored in. The values of these attributes need to be stored for all retailers that contain each product, and so, this information is stored in an array, in order to allow for more than one attribute value.

On a daily basis, the **product_match** table is used to store all products from different retailers. Each product stored in this table is identified by the *sourceCompany* and *productRefId* columns, identified in Figure 4.6. The *sourceCompany* value is used to identify the retailer and the *productRefId* is used to identify which product from *product_ ref* it corresponds to.

Over the time, *product_ ref* begins to stabilize because new products are rarely added. In contrast, *product_ match* is always growing daily. The *product_ match* table is used to make analyses and understand the position of different retailers in the market. The relationship between these two tables is illustrated in the UML diagram displayed below.

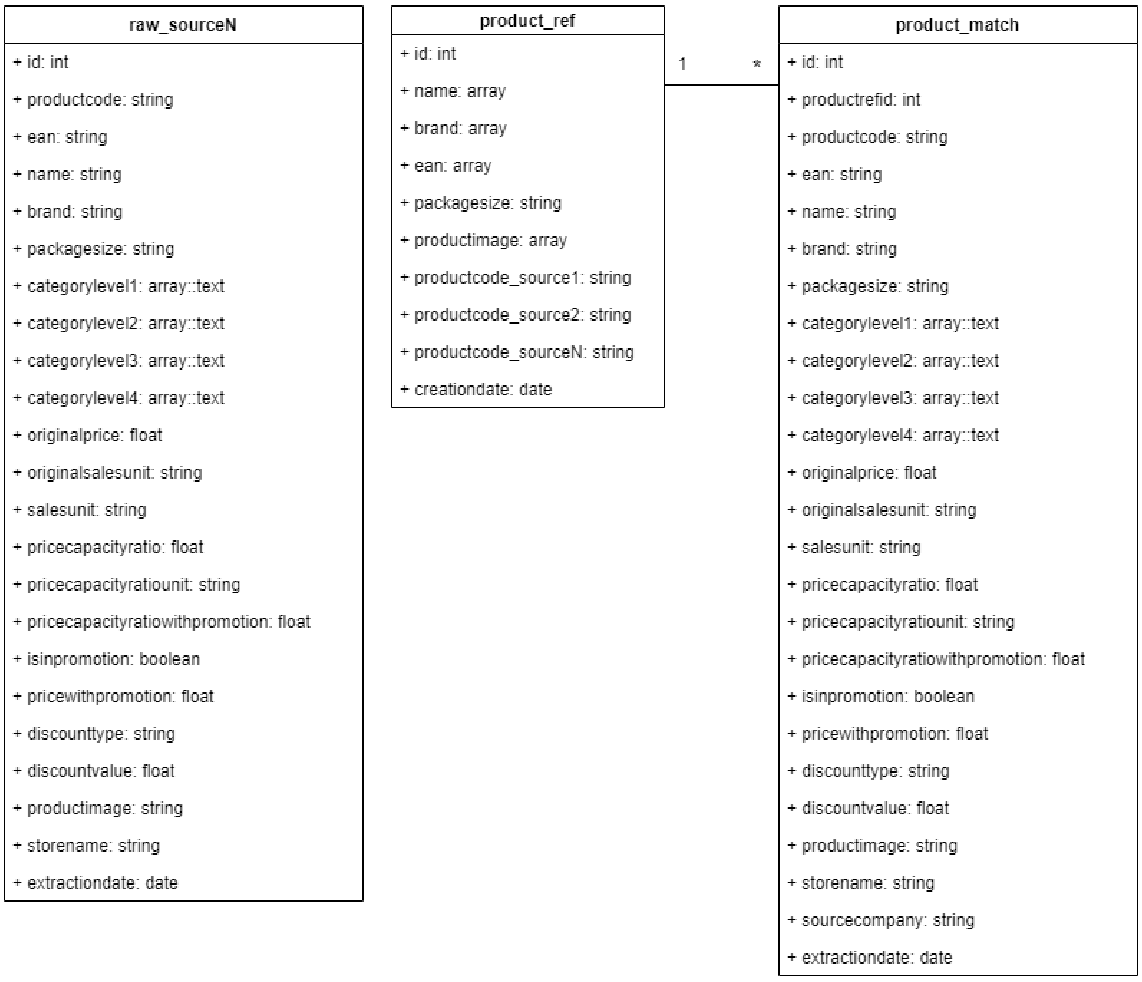


Figure 4.7: UML Diagram

4.2.2.2 Data cleaning and standardization

Data cleaning is the process of ensuring that data is correct, consistent and usable by identifying any errors or corruptions in the data, correcting or deleting them, or manually processing them as needed to prevent the error from happening again. Data cleaning might seem dull and uninteresting, but it is one of the most important tasks before implementing an algorithm. If the data is corrupted, then it may hinder the process or provide inaccurate results. Having wrong or bad quality data can be detrimental to processes and analysis.

Product offers are particularly challenging to match, as they are often highly heterogeneous and of different data quality. In particular, there is a huge degree of heterogeneity since product offers come from thousands of merchants using different names and descriptions of the products. Besides, offers often have missing or incorrect values and, in most cases, are not well structured, but mix different product characteristics in text fields, such as the product name. When using multiple data sources, a lack of standardization is common. As an important step in data integration and to obtain good results, it is necessary to standardize and clean the data as much as possible so that the products are represented in the same way.

On online e-commerce platforms, products are usually described by their product titles and product attributes. A product title usually contains the key information about the product and it is generally a brief and free text. Attributes are provided to specify more details about products, and they are usually in the format of name-value pairs from data catalog or structured tables on web pages. The data cleaning process involves analyzing the data collected from different sources and standardizing it to a uniform format for each value. After an exhaustive analysis of the data, a function transversal to any source was created, which is the standardization of each of the attributes values. Among the various attributes collected on retailers' platforms, the name, brand and packaging were chosen to standardize, as they are the ones that show the greatest discrepancy to the human eye, as we can see in Figure 4.8.

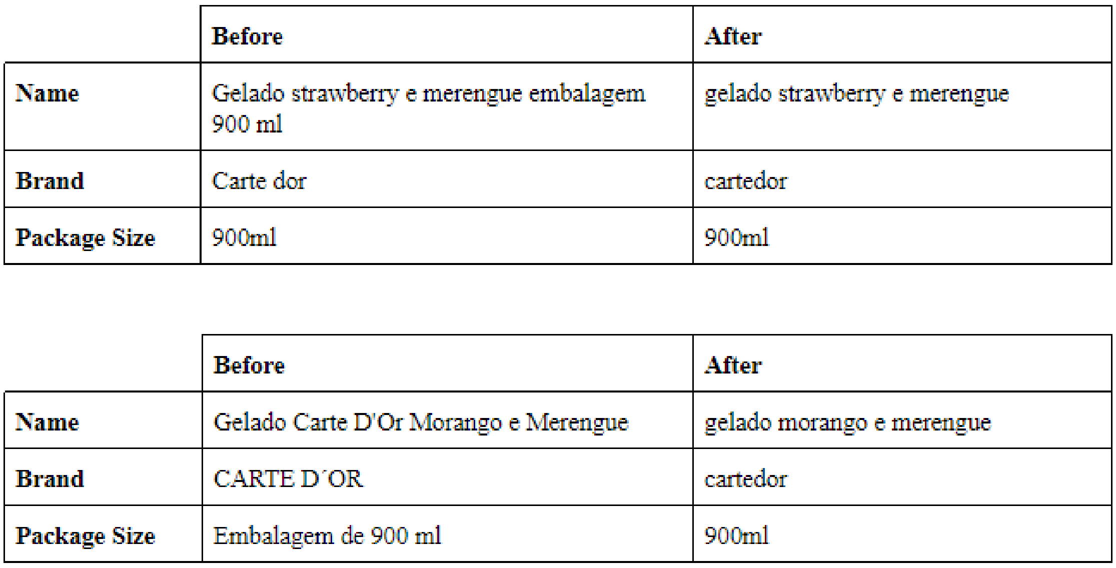
Image	Name	Brand	Package Size
a)	Gelado de manteiga de amendoim embalagem 460 ml	Haagen-dazs	460ml
	Peanut butter crunch	Häagen-Dazs	Embalagem de 460 ml
b)	Farinha de pão mix b sem glúten embalagem 1 kg	Schar	1kg
	Preparado para Pão Sem Glúten Schär	SCHAR	Embalagem de 1 kg
c)	Spray mousse lixívia + desengordurante embalagem 700ml	Neo blanc	700ml
	Lixívia Spray Mousse	Neoblanc	Embalagem de 700ml
d)	Snack para Cão Pequeno Dentalife	DENTALIFE	69g
	Snack higiene oral diária para cão mini	Purina One	Embalagem de 69g
e)	Bebida de cereais com Fibra Nestlé Bolero	BOLERO	200g
	Mistura simples solúvel bolero	Nestlé	Embalagem de 200g
f)	Iogurte Morango	Pastoret	emb.125 gr
	Pastoret Iogurte Artesanal de Morango embalagem 125 g	Pastoret	125g

Figure 4.8: An example of six products sold on two platforms. The columns refer to their attributes and their values are listed under them.

In Figure 4.8, each row identified with a letter corresponds to a different product and each column to different attributes, the separation of a cell into two rows is used to differentiate the attribute value given by different retailers. In this Figure, the title attributed by one of the sources to the product (a) is *Gelado de manteiga de amendoim embalagem 460 ml*, which refers to a kind of ice cream made by *Häagen-Dazs*. In addition to the product name, this title also contains information about the net weight and other main product specifications. One challenge of the product match is the fact that as products are described in different manners according to each sellers' demands, their products' title formats and attributes are different. Another difficulty of product matching is the synonyms, these not only appear in product titles, but also appear in product attributes.

In order to standardize the products attributes, for each of the values the punctuation was removed and everything was changed to lowercase. In the case of the product name attribute, if the package content and brand were contained in the name, they were eliminated. In addition to the name, the brand attribute can also be difficult to standardize because many sources write it in different ways or identify the same brand differently. An illustrative example of this scenario is shown in Figure 4.8. For example, in product (d) *Snack for small dog Dentalife*, one of the sources uses the brand *Purina* and the other use *Dentalife*, because *Dentalife* is the range of oral hygiene products from *Purina*. In these situations, it is not possible to standardize the product brand. However, in an attempt to increase the similarity between them, spaces and prepositions have been removed.

Figure below illustrates the transformation of two product labels that represent the same product but on different retailers' websites.



	Before	**After**
Name	Gelado strawberry e merengue embalagem 900 ml	gelado strawberry e merengue
Brand	Carte dor	cartedor
Package Size	900ml	900ml

	Before	**After**
Name	Gelado Carte D'Or Morango e Merengue	gelado morango e merengue
Brand	CARTE D´OR	cartedor
Package Size	Embalagem de 900 ml	900ml

Figure 4.9: An example of the transformation of the same product sold by different retailers

4.2.2.3 Product Match

Product matching is a challenging variation of entity resolution to identify representations and offers referring to the same product. The products collected from different retailers have different attributes and not all the retailers provide information about all the attributes values on their APIs or websites. If the EAN (European Article Number) attribute exists, the match is direct. EAN number is a numeric barcode number which is used in global trade or business to identify a specific retail product type and packaging configuration from a specific manufacturer. On the other hand, if the EAN does not exist, it is necessary to match the remaining attributes.

A product can be identified using different attributes, but not all are relevant to the Product Match algorithm. According to retail experts, attributes such as price and category do not become so relevant. The price does not discriminate well against products because there are many products that are not the same but have the same or very similar prices, and even the same products also have very different prices such as detergents and diapers. The category is also not a good attribute to consider, because retailers use different product categorization.

After an exploratory data analysis and consultations with experts of retail, the following attributes were chosen for the algorithm: name, brand, package content and image. In these attributes the probability of finding the largest number of tokens in common is greater. A token in NLP is a string of contiguous characters between two spaces, or between a space and punctuation marks. A token can also be an integer, rational number, or a number with a colon. These attributes are not identified by the different retailers always in the same way and therefore, before being used for the algorithm, they go through a process of standardization and analysis previously mentioned in section 4.2.2.2. The image turns out to be the one that immediately pops into view and the human immediately realizes whether it is the same product or not. As illustrated in Figure bellow, even this attribute becomes ambiguous to compare, because it can vary from retailer to retailer in perspective, clarity and tone for example. If the image has a transparent background, a white background has been added. To obtain a higher similarity value between the images, it was studied whether the match between the images was more accurate with or without cutting the image boundaries. This transformation is shown in Figure 4.10 displayed below. In addition, the images were scaled to the same size.

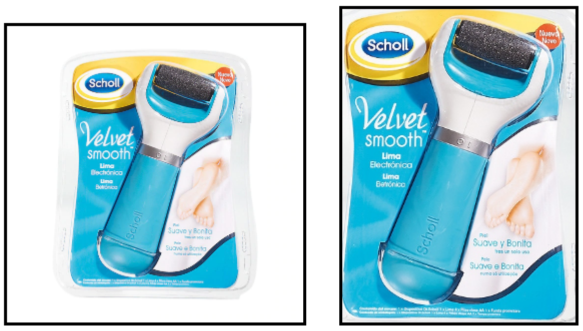


Figure 4.10: An example of a product image transformation process. On the left the original image is represented and on the right the result of the image after the transformation.

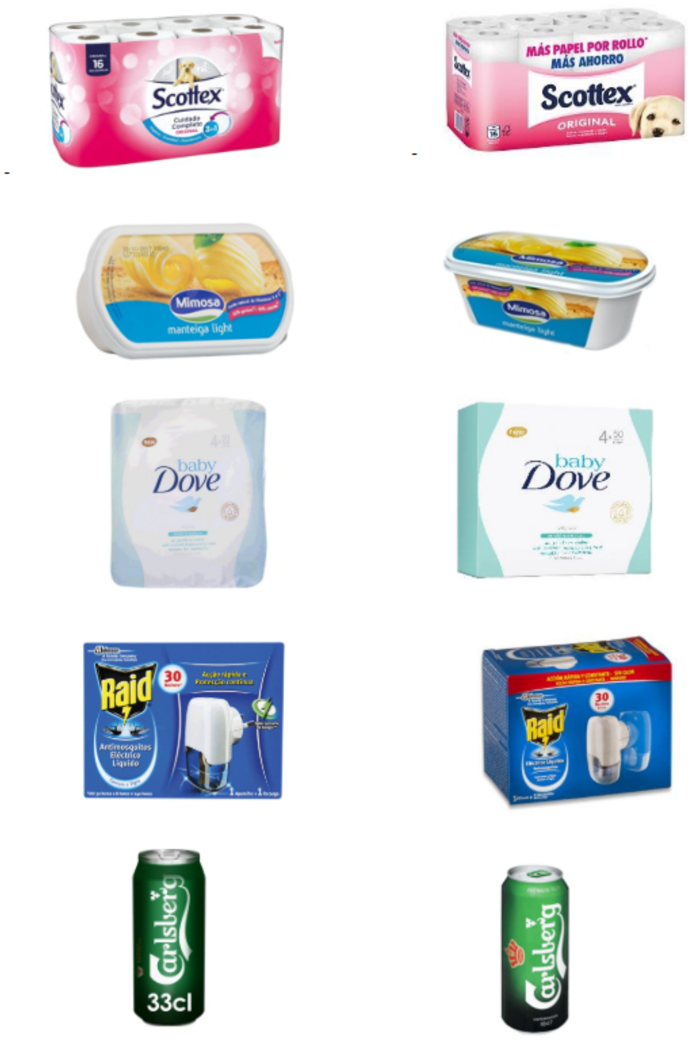


Figure 4.11: An example of images referring to the same product sold by different retailers

The attributes name, brand, package content and image differ so much from retailer to retailer that when compared alone they are not enough to decide whether the products match. Thus, the proposed algorithm aggregates these four attributes and, depending on the similarity value returned, decides whether the products match or not.

These attributes are stored in the *product_ ref* table, more specifically, in an array. The data first passes through a data cleaning process, because the more uniform the attribute is, the higher the similarity value after the match. The value given for an attribute by each retailer is stored in that array, so the size of that array corresponds to the number of retailers that the product has. As previously mentioned, the product offerings are extremely heterogeneous, so the larger the number of attribute values stored in the table *product_ ref*, the greater will be the possibility of obtaining a higher similarity value in future matches. For example, when evaluating a product, it may be more similar to the value given by retailer A than by retailer B. Concluding, when a new product comes in to combine it, it is compared with the different attribute values that are stored in the *product_ ref* table and chooses the value with which it is most similar. The higher the number of values to compare the new product with, the higher the possibility of obtaining a higher similarity value.

The following pseudocode, describe the complete algorithm.

Algorithm 1: Product matching algorithm pseudo code

```
Input: Product data from a retailer
initialization;
if the product code is in product_ref ;
 then
    the retailer's product has previously matched;
    add product to product_match;
else
    if The product has an EAN ;
     then
        if The product EAN matches any existing EAN in the table ;
         then
            change the row of the product_ref table it matched by adding the
              product code to the corresponding retailer's column;
            add product to product_match;
        else
            go to MatchProducts function;
        end
    else
        go to MatchProducts function;
    end
end
Function MatchProducts(product, product_reference):
    if The similarity of the combination of attributes brand name, size and package
      image is greater than x ;
     then
        change the row of the product_ref table it matched by adding the product
          code to the corresponding retailer's column;
        add product to product_match;
    else
        add this new product to product_ref;
        add product to product_match;
    end
```

Many approaches have been used to assess the similarity between attributes, for which the results are presented in Chapter 5.

The computation of the similarity between two images is done through an algorithm that first calculates the absolute value of the pixel-by-pixel difference between the two images, which results in a different image. This algorithm then adds all the different pixels, putting the histogram value, which plots the number of pixels for each tonal value. Finally, it calculates a similarity percentage based on a black and white image of the same size.

Two approaches were used for string attributes. Firstly, an approach that uses Word Embeddings, which is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. Secondly, the Levenshtein Distance, which is a string metric for measuring the difference between two sequences.

Word Embedding Approach

A word embedding is a representation of a word, and by extension a whole language corpus, in a vector or other form of numerical mapping. This allows words to be treated numerically with word similarity represented as spatial difference in the dimensions of the word embedding mapping. Similarities between words are calculated using cosine similarity measures, estimated based on their corresponding word vectors. As illustrated in Figure 4.14, words often used in the same contexts end up in similar locations in the vector space, on the assumption that words that get used similarly mean similar things.

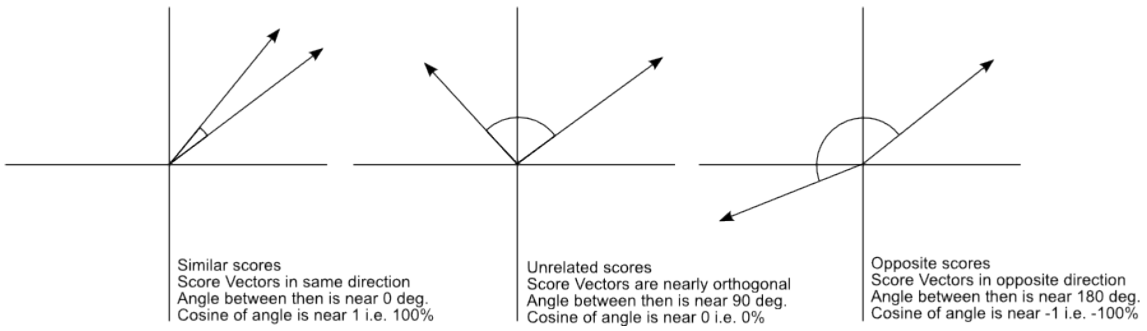

Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%
Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%
Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

Figure 4.12: Cosine similarity for similarity between words

The cosine of two non-zero vectors can be derived by using the *Euclidean dot product* formula:

$$\boldsymbol{A} \cdot \boldsymbol{B} = \|\boldsymbol{A}\|\|\boldsymbol{B}\| \cos\theta$$

Given two vectors of attributes, A and B, the cosine similarity, $\cos\theta$, is represented using a dot product and magnitude as:

$$simalirity = \cos\theta = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}}$$

where $\mathbf{A}_i$ and $\mathbf{B}_i$ are components of vector $\mathbf{A}$ and $\mathbf{B}$, respectively.

The resulting similarity measure ranges from -1, meaning exactly opposite, to 1, meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

Levenshtein Distance Approach

The Levenshtein distance between two words is the minimum number of single-character edits required to change one word into the other. These edits can be insertions, deletions or substitutions.

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if \min(i,j) = 0, \\ min & \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a \neq b)} \end{cases} \quad otherwise. \end{cases}$$

It is also possible to calculate the Levenshtein similarity ratio based on the Levenshtein distance. This can be done using the following formula:

$$\frac{(|a| + |b|) - lev_{a,b}(i, j)}{|a| + |b|}$$

where |a| and |b| are the lengths of sequence a and sequence b respectively.

The product match algorithm is not trivial, as previously mentioned. The products are heterogeneous and follow different categorization procedures, so the match process becomes complex to solve. Some tests were done to try to come up with the best algorithm. The process of comparing images takes longer than the process of similarity between strings. Consequently, the first approach taken is to make a comparison between each of the attributes individually (name, brand and content of the package). For each of these comparisons the similarity value returned varies between 0 and 1 and then the values returned for each of these comparisons are added together. Thus, the maximum value that this sum can take is three. If this comparison between strings results in a similarity value greater than 1.5, a comparison is made between images, in order to decide if the products match or not. The scheme shown below presents the comparison between the string attributes used in both approaches used to calculate similarity: word embeddings and Levenshtein distance.

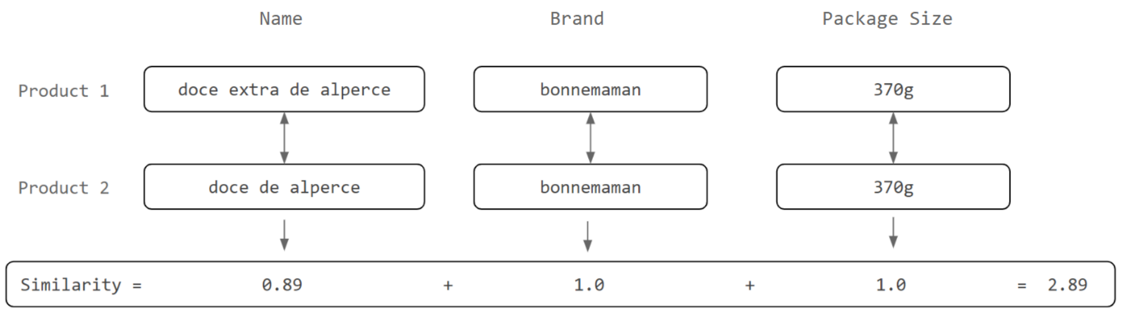


Figure 4.13: Comparison between strings attributes - First Approach

In order to check if there were better results when applying a different approach, a second approach was applied. In this, as illustrated in Figure bellow, the attributes name, brand and content of the package are aggregated in a single string and separated by a space.

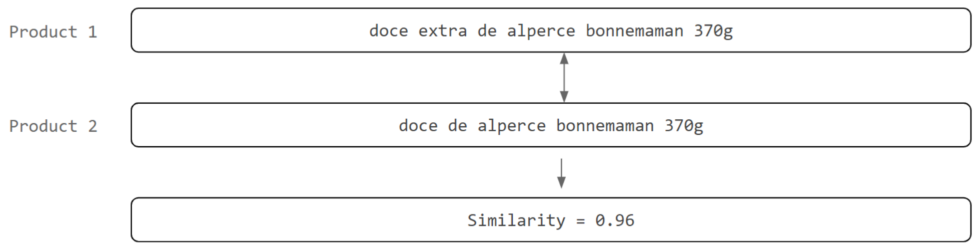

Product 1
doce extra de alperce bonnemaman 370g
Product 2
doce de alperce bonnemaman 370g
Similarity = 0.96

Figure 4.14: Comparison between strings attributes - Second Approach

4.3 Conclusions

This chapter described the details regarding the implementation of the solution. It describes in detail the design of the solution architecture, as well as, the different implementation processes. The data ingestion process is a process that requires enormous maintenance from the developer, since at any time the websites can be changed and the information may need to be collected differently. In addition, it is necessary to implement anti-scrapping techniques. On the other hand, the process of processing the data and implementing the Product Match algorithm is a complex and difficult issue to solve. Some approaches for solving the problem were proposed, however, there are still possible improvements. The solution was integrated and monitored through AWS technologies.

Chapter 5

Results

This chapter aims to show the results obtained in the project, referring to the product match algorithm and the final dashboards. Firstly, Section 5.1 describes the collected dataset. Secondly, Section 5.2 describes the results obtained in the Product Match algorithm validation process. Thirdly, Section 5.3 provides an overview of the dashboard creation process. Finally, Section 5.4 contains a number of final comments on the methodology used and the results obtained.

5.1 Exploratory Data Analysis

The data set under review contains information collected from seven retailers, three of which were made through the Web Scraping process and four through the Web Crawler process. The table below shows the number of different products collected from each retailer.

	Number of Different Products
Retailer A	41 816
Retailer B	10 490
Retailer C	23 503
Retailer D	24 424
Retailer E	6 783
Retailer F	2 266
Retailer G	4 322

Table 5.1: Number of Different Products by Retailer

Each product has 22 attributes, however only three out of seven retailers have the EAN code associated with the products. The table below illustrates the different attributes that a product can have.

Attribute	**Type**	**Description**
productcode	string	Product code
name	string	Product name
brand	string	Product brand
packagesize	string	Product package size
originalprice	float	Product original price
originalsalesunit	string	Product original sales unit
salesunit	string	Product sales unit in promotion
pricecapacityratio	float	Price capacity ratio
pricecapacityratiounit	string	Price capacity ratio unit
pricecapacityratiowithpromotion	string	Price capacity ratio with promotion
isinpromotion	boolean	The product is on sale or not?
discountType	string	Product discount type
discountValue	float	Product discount value
pricewithpromotion	float	Product price with promotion
categorylevel1	array	Product level 1 category array
categorylevel2	array	Product level 2 category array
categorylevel3	array	Product level 3 category array
categorylevel4	array	Product level 4 category array
productimage	string	Product image URL
storename	string	Product store name
ean	string	Product EAN code
extractiondate	date	Product extraction date

Table 5.2: Product attributes of each retailer

5.2 Product Match Algorithm

To evaluate the performance and quality of the algorithm, data from the products that matched through the EAN were used as *ground truth* data. *Ground Truth* is a common terminology that is widely used in various fields to refer to any kind of information provided by direct observation. Matching products through their EAN allows one to understand whether the algorithm is, indeed, fully functioning. By using the data for which there is a match between product and EAN, one can then proceed to analyse if the algorithm also classifies them as a match, and thus, is correctly functioning. In contrast, if by any chance, the algorithm is erroneously classifying as a match or not match, this situation is easily detected through this comparative analysis.

The first subsection 5.2.1 presents in detail the exploratory analysis of the data. The following two subsections show the results of the algorithm for the different approaches of combining the attributes name, brand, packaging content, and image.

5.2.1 Exploratory Analysis of Ground Truth Data

The first step before evaluating the results obtained is to make an exploratory analysis of the data, to better understand the characteristics of the data with which the algorithm works. As shown in the table below, the data set contains information collected from retailers that have the EAN code as a product attribute and represents 33 067 different products.

	Value	**Percentage (%)**
Total number of products	33 067	100 %
Number of Products Without EAN	739	2.23 %
Number of Products That Match	6 206	18.77 %
Number of Products That Not Match	26 861	81.23 %

Table 5.3: Exploratory Data Analysis

In addition to understanding how many products the data set has, as well as those that were classified as a match or not, it is necessary to verify the information the data set has relative to the products attributes. There are retailers who do not have complete information about their attributes for all products. To match products through the EAN, three retailers were used. As previously mentioned, after the data cleaning process, the values of the attributes that are used to match are stored in an array in the *product_ref* table. Thus, the size of that array corresponds to the number of retailers that offer that product. Therefore, since the data set used to evaluate the results comes from the product collection of three different retailers, the maximum number of attribute values that a corresponding attribute can have is three.

The pie chart below shows the percentage of attribute values that the set of matched products has for the corresponding products. After a detailed analysis of the graph presented, it is proven that only the brand has no assigned value to some products. However, it represents 9 products, which corresponds to 0.15% of the products that are classified as match (6 206).

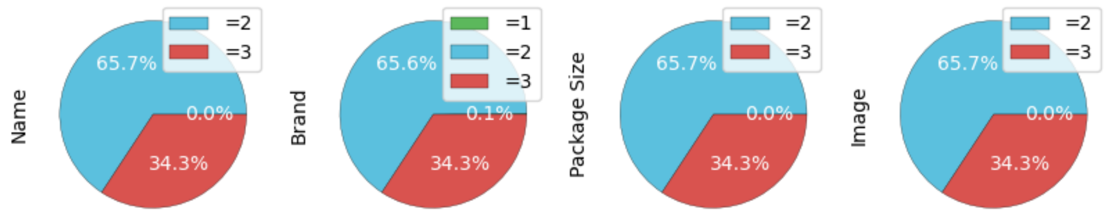


Figure 5.1: Attribute evaluation for those matching products

5.2.2 Match between images

As previously mentioned, to improve the efficiency of the match between the images, two approaches were studied: one in which a cut of the image limits was applied before making the match and another where the cut was not applied. From the results obtained, it was possible to conclude that not applying the cut improves the comparison between the images. Through the use of the data that matched through the EAN (6206), it was possible to conclude over the Figure 5.2 that the similarity value was higher with cutting in 4254 cases, which corresponds to 68.55% of the data (see Figure 5.2).

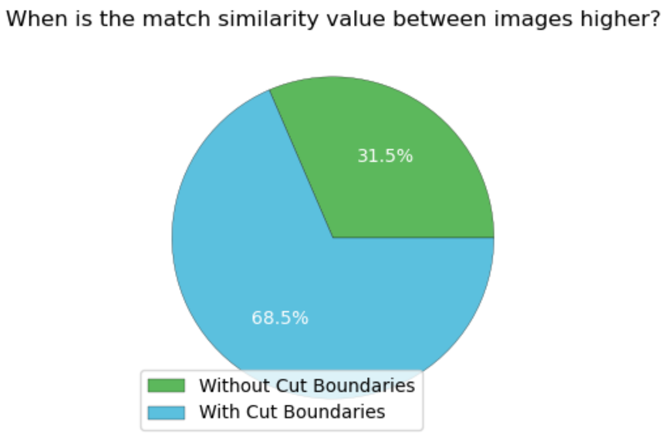


Figure 5.2: Comparing cut boundaries or not for image matching

5.2.3 Match between string attributes

This section compares the results obtained through the fourth approaches used to match string attributes. As the process of comparing images is slower and it is necessary to guarantee a better performance of the algorithm, a validation of the approaches was done first without using the comparison between images. Through this validation, it is possible to understand from which similarity value it makes sense to use image comparison. The results obtained with and without the image boundaries cut are also presented, where it is possible to conclude that better results are obtained with cutting.

The histograms shown below present the results obtained through the different approaches of the algorithm. The histogram on the top represents the approach without the image matching. On the bottom left is the approach for image matching without boundaries cut and on the right with boundaries cut. These histograms are normalized because the height of the histogram bar represents the proportion of the data in each similarity value. The normalized count is the count of a similarity value divided by the total number of observations. For this normalization, the area (or integral) under the histogram is equal to one. The x-axis represents the similarity value given by the algorithm.

5.2.3.1 First Approach - Comparing string attributes individually and using Word Embedding to evaluate the similarity

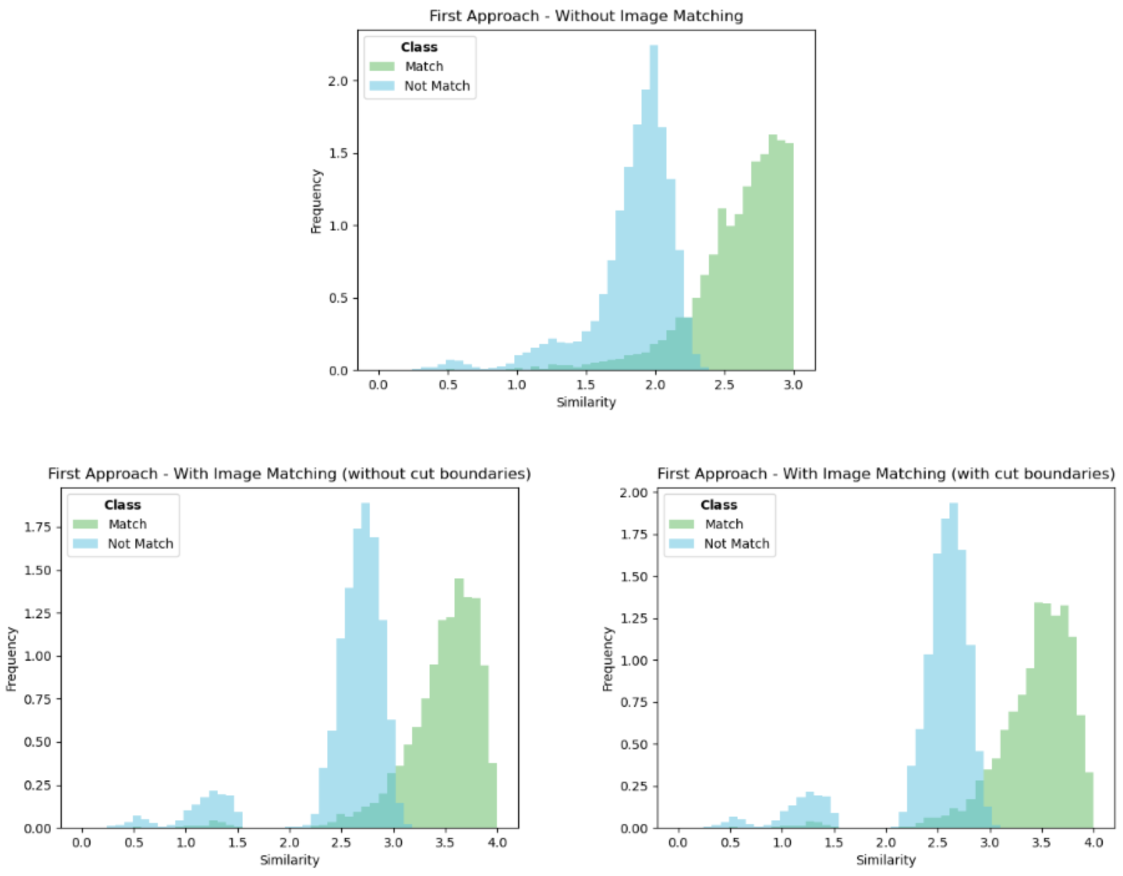
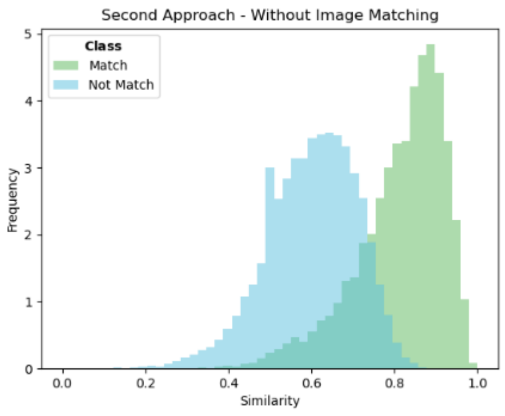
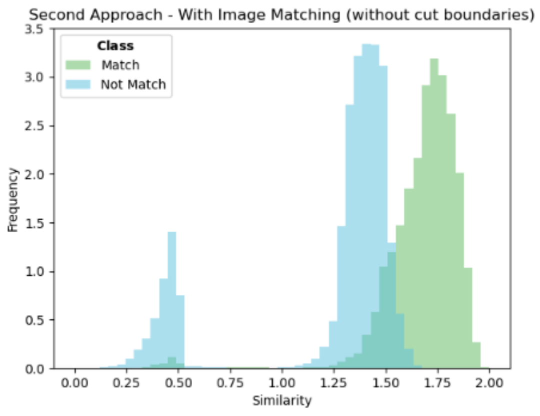


Figure 5.3: Comparison between string attributes individually and use of Word Embedding to evaluate the similarity. On the top without image matching, and on the bottom with the match between images.

Through this graph it is possible to conclude from the value 1.5 there are many data misclassified. Thus, the approach used to solve the problem was when the value of the sum of string attributes similarities is greater than or equal to 1.5, a comparison between images is made and the value is added to the sum made previously. This last approach is illustrated in the images on the bottom, where it is possible to conclude by comparing the image on the top with the images on the bottom that using the match between image improves the performance of the matching algorithm. However, comparing the images below, it is possible to conclude that in this approach, cutting the image boundaries improves the algorithm. With the cut of the image boundaries, 3.14% of the data is misclassified, while without the cut it is 3.68% of the data.

5.2.3.2 Second Approach - Comparing string attributes together and using Word Embedding to evaluate the similarity





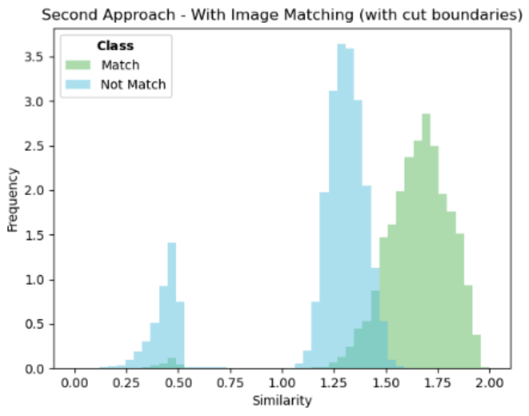


Figure 5.4: Comparison between string attributes together and use of Word Embedding to evaluate the similarity. On the top without image matching, and on the bottom with the match between images.

Similar to the way the results obtained in the first approach were studied, when analyzing the histogram represented at the top of the Figure 5.4, we conclude that from the value 0.5 there are misclassified data. Therefore, and in order to minimize this number, the comparison between images was added, which significantly reduces the number of poorly classified data. Furthermore, after comparing the match between the images with or without the cut, we conclude that applying the cut improves the product match algorithm.

5.2.3.3 Third Approach - Comparing string attributes individually and using Levenshtein Distance to evaluate the similarity

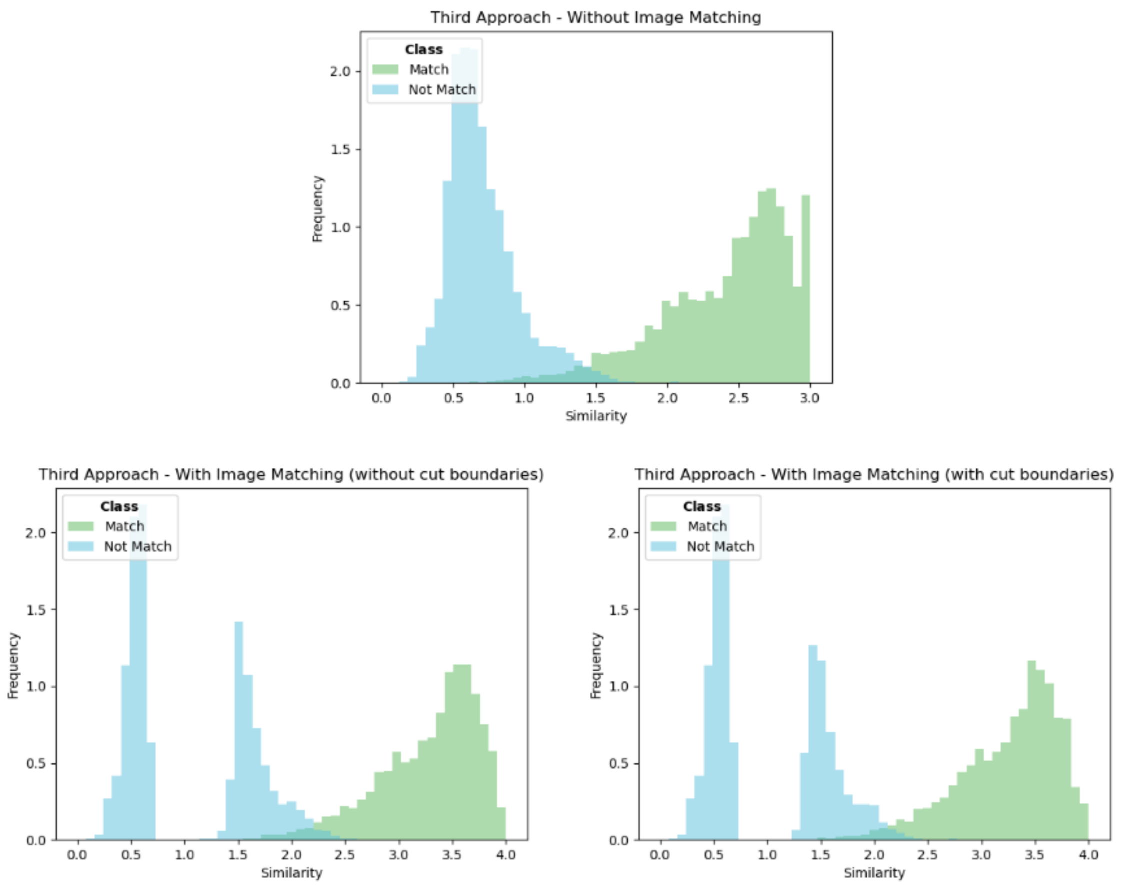


Figure 5.5: Comparison between string attributes individually and use of Levenshtein Distance to evaluate the similarity. On the top without image matching, and on the bottom with the match between images.

Making a detailed analysis of the three histograms represented in Figure 5.5, it is possible to conclude from the histogram represented at the top that from the 0.6 similarity value there are misclassified data and the match between the images can be used to improve the product match algorithm . On the other hand, when analyzing the histograms represented at the bottom of the image, we conclude that applying the cut boundaries of the image improves the algorithm. With this cut, 2.60% of the data is misclassified, while without the cut it is 3.19% of the data.

5.2.3.4 Fourth Approach - Comparing string attributes together and using Levenshtein Distance to evaluate the similarity

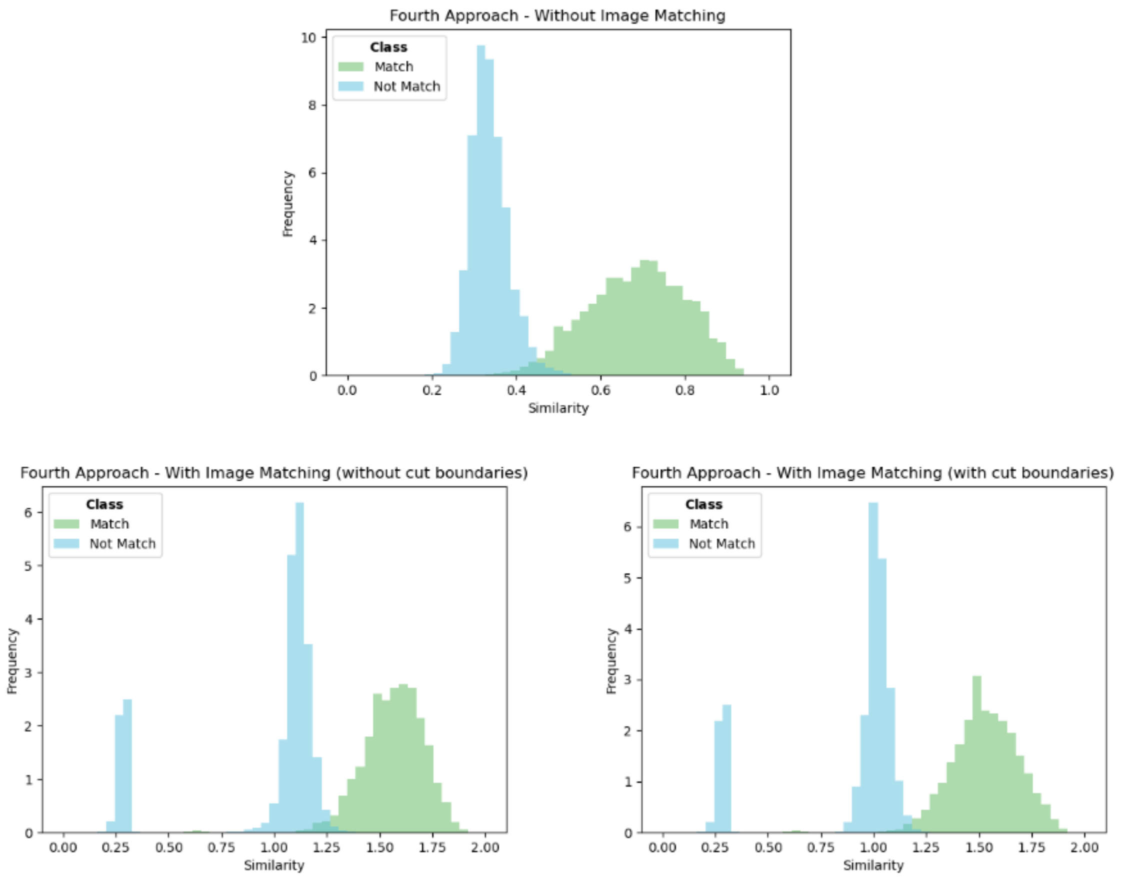


Figure 5.6: Comparison between string attributes together and use of Levenshtein Distance to evaluate the similarity. On the top without image matching, and on the bottom with the match between images.

Through this graph, it is observable that from the value 0.3 there are many data misclassified. Thus, when the value of the sum of string attributes similarities is greater or equal to 0.3, a comparison between images is made and the value is added to the sum made previously. This last approach is illustrated in the images on the bottom, it is possible to conclude by comparing the image on the top with the images on the bottom that this approach enables to reduce the number of misclassified data. However, comparing the images below, it can be observed that in this approach, i.e. cutting the image boundaries, improves the algorithm. With the cut of the image boundaries, 1.55% of the data is misclassified, while without the cut it is 2.44% of the data.

5.2.3.5 Comparison between the different approaches used

The table below shows the number of poorly classified data for each of the approaches. These data are represented with a darker color in Figures 5.3, 5.4, 5.5 and 5.6.

	First Approach	**Second Approach**	**Third Approach**	**Fourth Approach**
Without Image Matching	1 605 (4.85%)	4 785 (14.47 %)	1260 (3.81 %)	1840 (5.56 %)
With Image Matching (without cut)	1 217 (3.68 %)	2 738 (8.28 %)	1 054 (3.19 %)	809 (2.44 %)
With Image Matching (with cut)	1 038 (3.14 %)	1 942 (7.44 %)	861 (2.60 %)	513 (1.55 %)

Table 5.4: Amount of misclassified data for the fourth approaches

Through a detailed analysis of Table 5.4, it is possible to conclude that, for all approaches, to consider the comparison between images and the cut of boundaries image improves the performance of the algorithm. However, the number of misclassified matches is lower in the fourth approach than in the others. In this fourth approach, only 1.55% of the data is misclassified, which represents a very small proportion of the total. Many of these misclassified data are quite heterogeneous and even a human could have difficulties understanding whether they represent the same product or not. The figure below illustrates some examples of misclassified products.

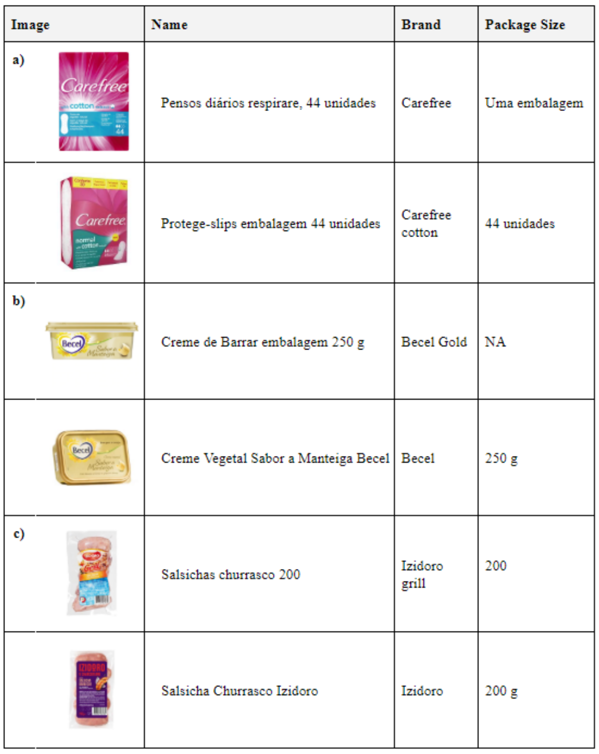


Image	**Name**	**Brand**	**Package Size**
a)	Pensos diários respirare, 44 unidades	Carefree	Uma embalagem
	Protege-slips embalagem 44 unidades	Carefree cotton	44 unidades
b)	Creme de Barrar embalagem 250 g	Becel Gold	NA
	Creme Vegetal Sabor a Manteiga Becel	Becel	250 g
c)	Salsichas churrasco 200	Izidoro grill	200
	Salsicha Churrasco Izidoro	Izidoro	200 g

Figure 5.7: An example of misclassified products.

5.3 Dashboards

As previously mentioned, the final result of the project are dashboards to support a market analysis. In order to create these analysis, it is necessary to establish relationships between the tables constructed to calculate the results accurately and present the correct information in the reports. Data relationship model is an important model design topic that is essential to deliver intuitive, accurate, and optimal models. The Appendix B presents an overview of the relational data model created. In addition, measures were also created, which are columns in the fact table that store values to be summarized. Measures quantification may involve simple column aggregations or sophisticated formulas that override filter context and/or relationship propagation.

When working with databases, one of the most important tasks to be performed is the performance optimization. The *product_match* table was used as a support, and from it materialized views were created in order to construct tables with information related to the retailer for which the analysis is being carried out. A *Materialized View* is a real table in the database that is updated whenever an update occurs in any table used by the query. In summary, a materialized view is used when the performance of searches in the view is more important than the performance of tables recording. In this sense and with the objective of increasing the performance of PowerBI, materialized views are responsible for processing and transforming data and PowerBI only reads it.

Consider the following examples of the created materialized views:

- Materialized view with the products that the retailer has in common with the market, those that only the retailer has, and those that the retailer does not have, grouped by date, respective retailer and product.
- Materialized view with promotions in common with other retailers, grouped by date, respective retailer and product.
- Materialized view with the categories that the respective retailer has in common with the market, those that the retailer does not have and those that only the respective retailer has.

The figures below illustrate some examples of the dashboards created. There are a total of 25 different pages, although only three are represented below. The layout of these is as follows - with a right to left approach: (1) the analysis period selected by the user, where they can choose to select either a data range, or the previous 7, 15 or 30 days, and (2) the three analysis dimensions - product, price and promotions. In addition to this navbar common to all dashboards, the dashboard has a side menu presented on the left where the user can choose to analyze the market, a category, a specific product or a basket.

Figure 5.8 illustrates a dashboard with an overview of the analysis of products in the market, along with each retailer's range coverage. This enables the retailer for whom the analysis is being carried out to understand, in a detailed manner, which products they have in common with the rest of the market. Added to this, the dashboard also enables the retailer to understand not only which products they do not have, when compared to what exists in the market, but also those which they are the only retailer to have.

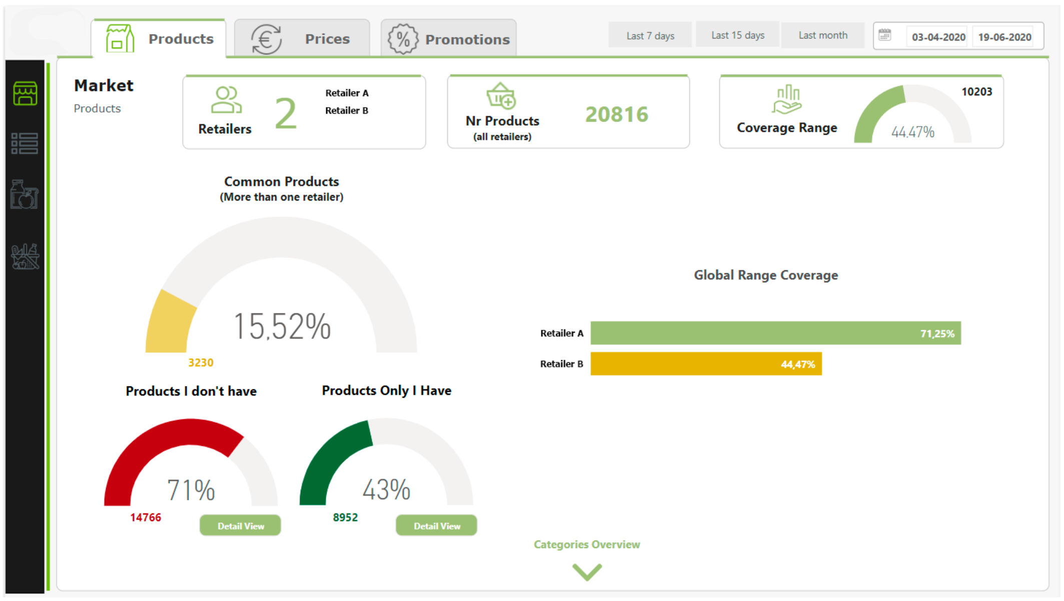


Figure 5.8: Dashboard overview of Market Products

The dashboard present in Figure 5.9 illustrates an overview of the existent market categories, allowing the user to analyze the range coverage the retailer has for each category.

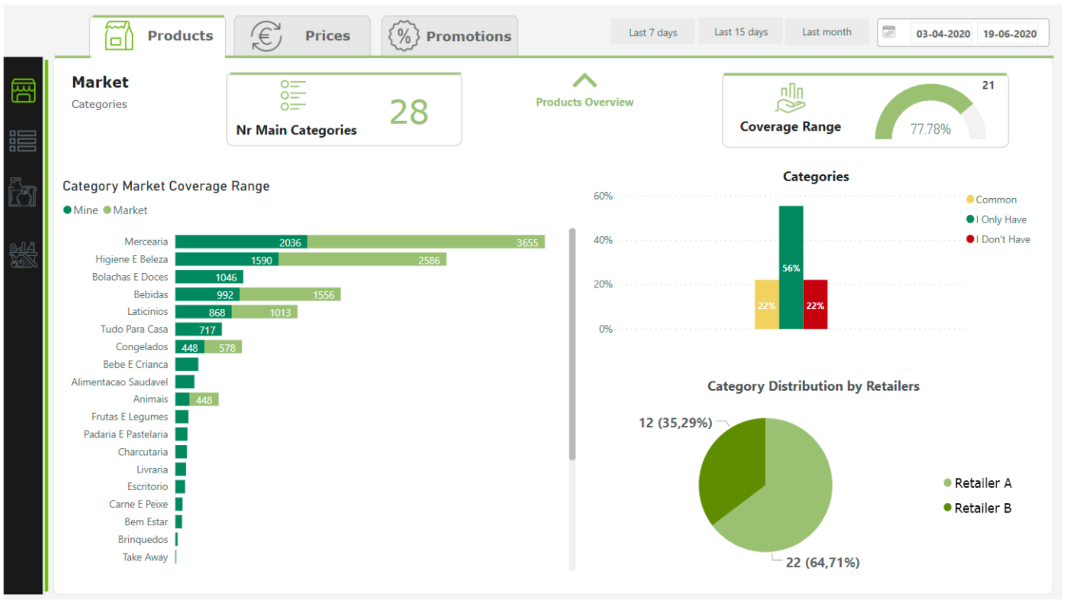
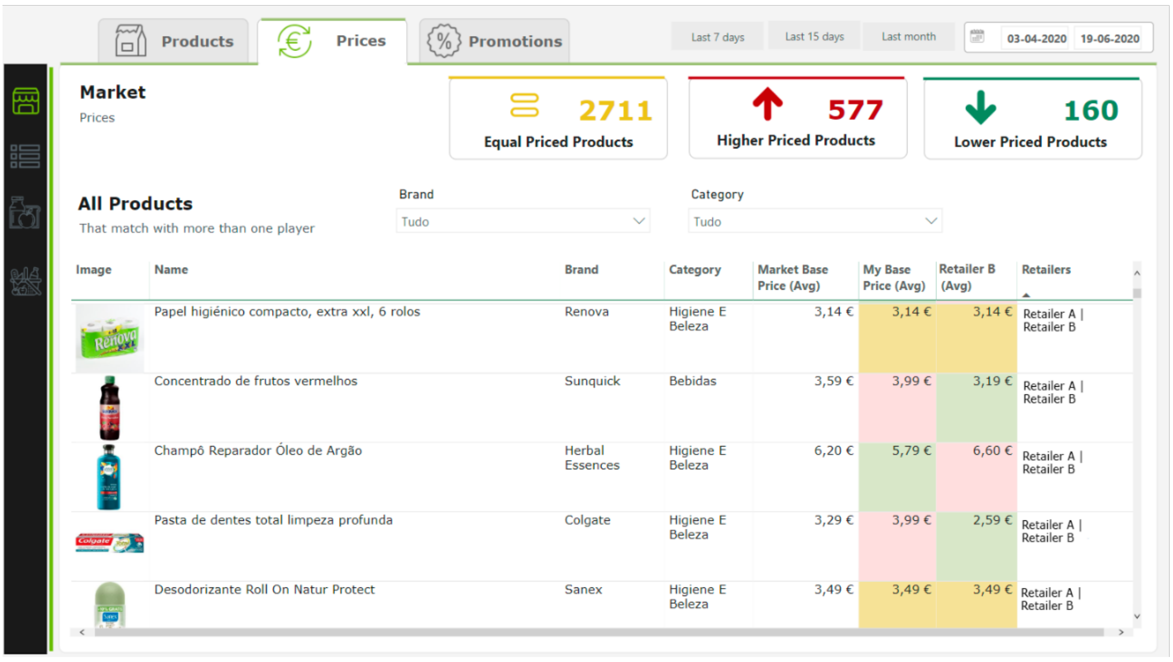


Figure 5.9: Dashboard overview of Market Categories

The dashboard in Figure 5.10 allows the user to analyze the prices of products that the user has in common with at least one other retailer. In the table shown in the figure, yellow represents products for which the respective retailer and another retailer in the market charge the same price, red demonstrates that the respective retailer charges a higher price than another retailer, and green shows that it charges a lower price.



Market
Prices

2711 Equal Priced Products

577 Higher Priced Products

160 Lower Priced Products

All Products
That match with more than one player

Brand: Tudo

Category: Tudo

Image	Name	Brand	Category	Market Base Price (Avg)	My Base Price (Avg)	Retailer B (Avg)	Retailers
	Papel higiénico compacto, extra xxl, 6 rolos	Renova	Higiene E Beleza	3,14 €	3,14 €	3,14 €	Retailer A | Retailer B
	Concentrado de frutos vermelhos	Sunquick	Bebidas	3,59 €	3,99 €	3,19 €	Retailer A | Retailer B
	Champô Reparador Óleo de Argão	Herbal Essences	Higiene E Beleza	6,20 €	5,79 €	6,60 €	Retailer A | Retailer B
	Pasta de dentes total limpeza profunda	Colgate	Higiene E Beleza	3,29 €	3,99 €	2,59 €	Retailer A | Retailer B
	Desodorizante Roll On Natur Protect	Sanex	Higiene E Beleza	3,49 €	3,49 €	3,49 €	Retailer A | Retailer B

Figure 5.10: Dashboard overview of Market Prices

5.4 Conclusions

This section describes the set of data collected, as well as the sample chosen to validate the algorithm. In addition, the results obtained in the data transformation layers, product correspondence and analytical layer (dashboards) are presented. It was concluded that the best approach for the product match algorithm is the one that compares string attributes together and uses Levenshtein Distance to evaluate the similarity, since it presents a lower number of misclassified data.

Chapter 6

Conclusion

Food retail is a constantly growing sector with a huge diversity of store formats. In an ever-changing competitive environment, grocers and online food retailers are constantly innovating and utilizing technology to ease the consumer shopping experience. Retailers compete hard for every customer, and it is customer demand and behaviour which has driven these fundamental changes. The internet and the digitization of commerce have fundamentally changed the retail sector from a local to a global market. The sector has evolved into an omni-channel sector, combining all forms of sales (physical stores and e-commerce), as well as online retail platforms.

During this work, this sector was characterized and it was exposed the difficulty that retailers have in better positioning themselves in the food retail market. In response to this problem, this project culminates in the development of a software that provides retailers with real-time information about products, prices and promotions in the market. In this sense, the aim of this project reflects the development of a rich data source, with permanently updated and reliable data, based on the updated online platforms that accompany the price changes of the products sold by several retailers.

The development of this project is based on the "shopping" process, which has always been done by retailers in physical stores. This process enables to understand the range of prices that the retailer's competitors offer in the market, and thus, validate whether their offer is consistent and/or differentiating and whether the pricing policy is adequate and/or competitive. This project focuses on the online format of different retailers in the food retail market and is a project facing a huge variety of features, requiring thorough maintenance and analysis. For the development of this solution, Amazon Web Services (AWS) technologies were used, which provides many different services that can be components of a serverless application. The great benefit of using AWS is that it is a safety and economic infrastructure. Unlike traditional datacenters, AWS offers unlimited flexibility and scalability. Thus, it was possible to use the most varied technologies from AWS to monitor and automate all processes of the project. One of the common use cases for AWS technologies is Amazon cloud storage, which offers several types of storage, allowing companies to make their own decisions based on their needs. In the development of the project, this service was used to store daily data collected from the websites of different retailers. To collect data from different retailers' websites, two techniques were used: web-scraping and web crawler. In addition to collecting prices and automating all processes, which is one of the most important phases of the project, there is another focus - the product match process. The concept of Product Match is not only relevant for retailers, but also something that is already used as a business tool on a recurring basis, although not always standardized, automated and scalable. The development of this project allowed the construction of a solid basis for this tool, monitoring all processes and studying different approaches to the match of products.

Before matching the products, it was necessary to make a detailed analysis of the products sold by the different retailers. After this analysis, the attributes name, brand and content of the packaging were standardized in order to facilitate the process of comparing attribute values.

To match the products, a comparison was made between images and string attributes, namely the name, brand, and packaging content. For image comparison, a pixel-by-pixel calculation was used to determine the differences between two input images. For the comparison between string attributes, two strategies were studied. Firstly, an NLP technique based on semantic similarity was used, which is calculated based on two semantic vectors. The similarity is determined using the cosine distance between two vectors. Secondly, an approach that use the Levenshtein Distance is used. This is a string metric for measuring the difference between two sequence. The second strategy proved to be the best as enables to obtained the lowest number of misclassified data.

In addition to the two approaches used to assess the similarity between string attributes, two different strategies were also studied to compare these attributes. These approaches differ in comparing string attributes. The first approach of comparing string attributes compares the three attributes individually and adds the similarities returned. On the other hand, the second approach joins the three attributes in a single string separated by a space and makes the comparison. After being analyzed and tested, comparing the string attributes in a single string and using Levenshtein Distance to assess similarity proved to be the best approach. Through the product matching algorithm, it was possible to build a database with all the existing products from different retailer's online stores. For each product, the information about who has it is stored in that same database. With this information, it was possible to create a dashboard that aims to help retailers analyze their competitors.

6.1 Main Difficulties

There were several difficulties while developing this SaaS. First, the software developed requires enormous maintenance and data collection must be carried out with the greatest possible rigor. In this process of data collection, data can often go wrong. The websites do not always have the main attributes exposed to the customer and often the values of the attributes are wrong, making the Product Match algorithm difficult as well as the analyzes made later. Furthermore, many sites protect themselves using anti-scraping mechanisms to avoid being attacked by scraping programs on the web, so it was necessary to find solutions to bypass anti-scraping techniques and avoid being blocked by anti-scraping systems. Secondly, one of the biggest challenges and difficulties of this project is the Product Match algorithm. It is not a problem with a trivial resolution and requires huge data analysis and transformation. Not all products are cataloged in the same way and have different attributes values. Online supermarket catalogues, as mentioned before, are characterised by having innumerable quantities of different products. One of the most difficult product categories to match is fresh products, which have different suppliers, different base prices and, therefore, different profit margins for the company.

6.2 Main Contributions

The main contribution of this work were the construction of a solid basis for solving the Product Match problem. The use and integration of AWS technologies allowed to monitor the main processes of this project, namely the process of data ingestion and transformation. Retailers can only make the right decisions based on data if the data used is correct. Without sufficient data quality, the data is practically useless and sometimes even dangerous. This project contributed largely to the completeness, consistency, validation and accuracy of the Product Match problem. Furthermore, this project contributed to a simple approach to overcome the problem of similarity between products.

6.3 Future Work

In the future, improvements and additional functionalities may be applied to all layers of this service. One feature that could potentially be added, in the data ingestion process, could be to provide the retailer with the ability to add product prices practices in the different physical stores. This process would be executed by a member of the retailer company who travels to physical stores to study the prices charged by different competitors. Added to the different products, prices and promotions practiced by different retailers and store location is another decisive factor in product pricing. By enabling the addition of physical store prices' inclusion, by location, one would ensure a more complete and thorough market analysis.

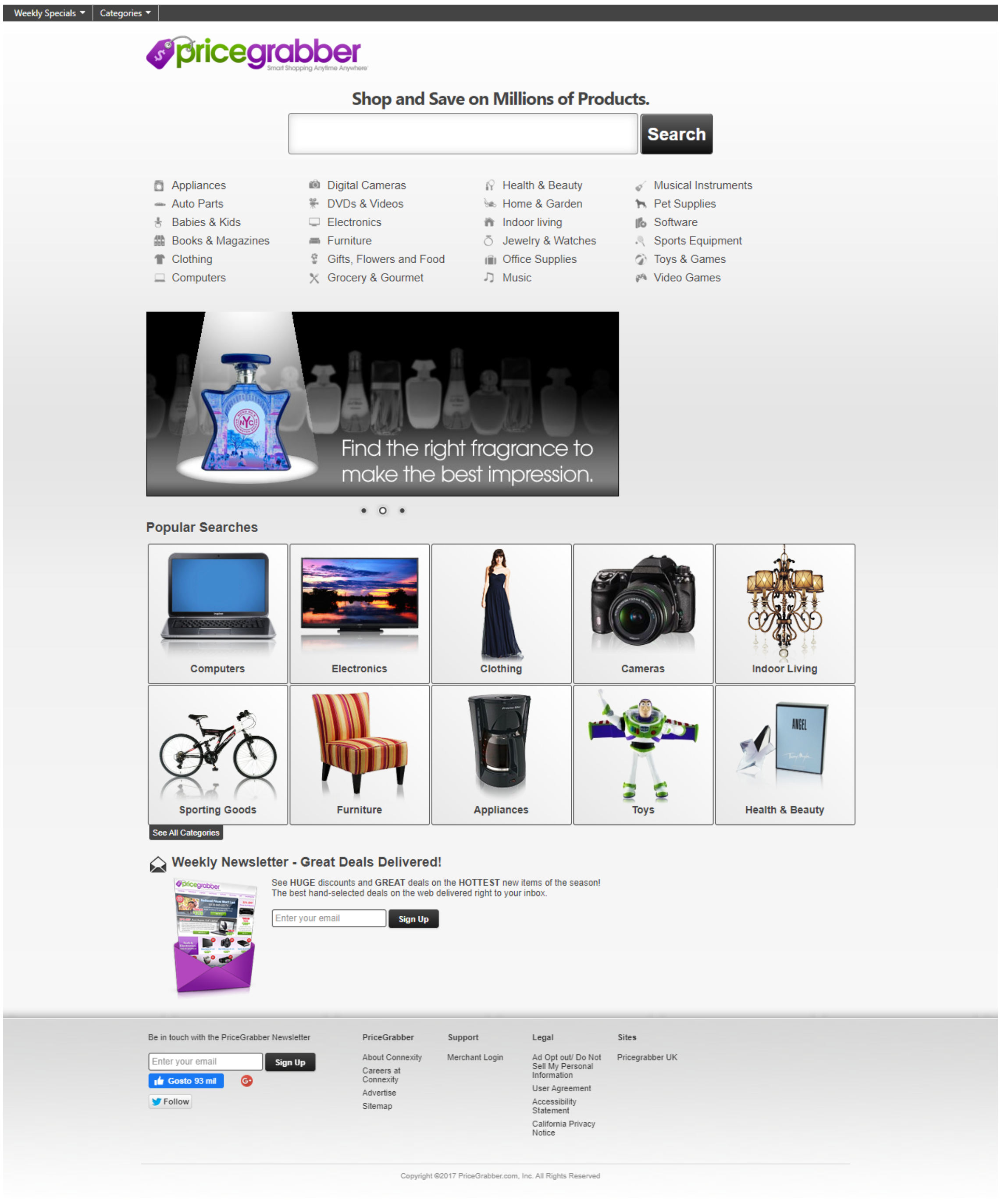
Added to this, improvements may also be made in the data transformation layer and the product match algorithm, in order to enhance their performance and quality. Regarding the development of the algorithm, other approaches that make use of Artificial Intelligence and Machine Learning can be used. A neuronal network, for instance, could be built, that receives a sample database of classified products, learns from their different attribute values, and is then able to classify two products as a match or not match, in the future.

Finally, in the analytics layer the product can be scaled both vertically and horizontally. There are several strategies and analytical calculations that can be done to provide the retailer with competitive insights. These strategies can simulate and optimize pricing using Machine Learning science. This layer may, for example, be able to provide suggestions on best price practices, considering the company's pricing strategy. This would allow the retailer to gain a profitable pricing advantage with competitive positioning, improve channel strategy management, category and item level detail. Another feature could be, for example, identifying direct competitors based on competitive prices and elasticity, determining the relative importance of each competitor. Additionally, at a promotional level, a detailed analysis of retailer performance could be conducted, aimed at understanding the impact of each offer on profits, with important metrics for the business.

Appendix A

Price Grabber Main Page

This appendix contains an overview of the PriceGrabber platform homepage.



Shop and Save on Millions of Products.

Weekly Newsletter - Great Deals Delivered!

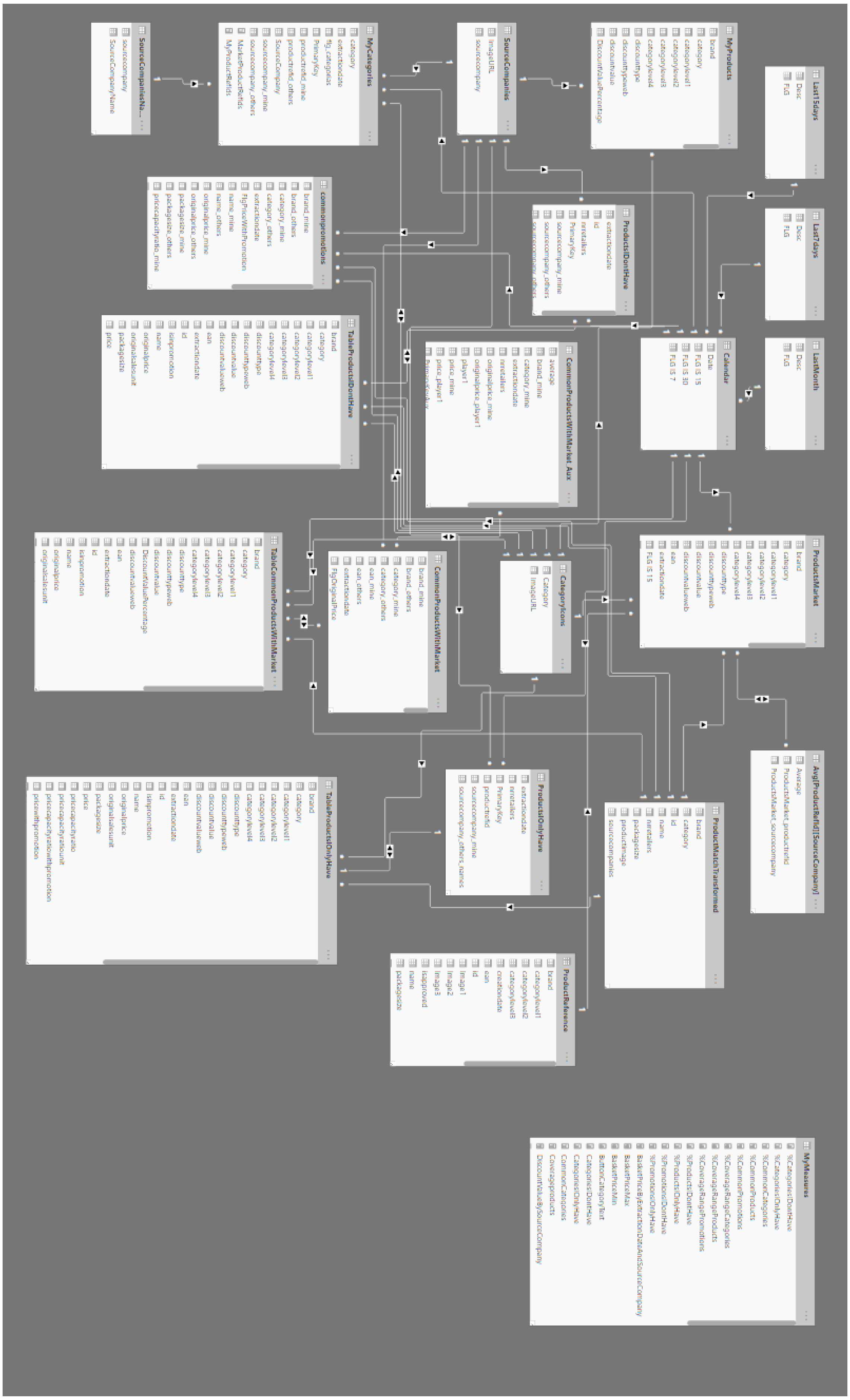
See **HUGE** discounts and **GREAT** deals on the **HOTTEST** new items of the season!
The best hand-selected deals on the web delivered right to your inbox.

Copyright ©2017 PriceGrabber.com, Inc. All Rights Reserved

Appendix B

Data Relationship Model in PowerBI

This appendix contains an overview of the Data Relationship Model created to build the final dashboards.


SourceCompaniesNa...
MyCategories
SourceCompanies
MyProducts
Last15days
commonpromotions
ProductsIDontHave
Last7days
TableProductsIDontHave
CommonProductsWithMarket_Aux
Calendar
LastMonth
TableCommonProductsWithMarket
CommonProductsWithMarket
CategoryIcons
ProductsMarket
Avg[ProductsRefId][SourceCompany]
TableProductsIOnlyHave
ProductsIOnlyHave
ProductMatchTransformed
ProductReference
MyMeasures

References

[1] Handbook of Water and Energy Management in Food Processing. Food retailing. 2008.

[2] Cenk Çorapcı. Product matching with deep learning. 9 May 2019.

[3] Deloitte. Analytics in retail going to market with a smarter approach. 2013.

[4] EKN Benchmark Study. The future of retail analytics. 2013.

[5] Meridith Levinson. Software as a service (saas) definition and solutions. May 2015.

[6] Prompt Cloud. Web crawling for the retail industry. April 21, 2015.

[7] Judith Hillen. Web scraping for food price research. November 2019.

[8] Pinterest softprodigy. Webscraping architecture - web browser, crawlers, webs.

[9] Web Crawler. Web crawler — Wikipedia, the free encyclopedia. [Online; accessed 1 March 2020].

[10] Cambridge Technology. Serverless applications – the next step in the evolution of saas. Oct 23, 2019.

[11] Zeta Visual. *Cloud Architecture*, (accessed April 22, 2020). `https://www.zetavisual.com/cloud-architecture/`.

[12] Kumod Jha. Serverless computing and a case study in retail. Oct. 08, 18.

[13] Mayuri K. Importance of pricing.

[14] Bdc. Competitor based pricing strategy: competition based pricing for saas.

[15] Joe Skorupa. Pricing with confidence custom research. *RIS News*, March 2016.

[16] Price2Spy. Manual vs. automated product matching: Which one is the best for your ecommerce business. March, 2019.

[17] Jhonny Alexander Aldeia de Jesus. Comparador de preços inteligente. June 2015.

[18] Mark Hayes. 17 best price comparison engines to increase ecommerce sales.

[19] eBizMBA Inc. Top 15 best comparison shopping websites. February 2020.

[20] PriceGrabber. *PriceGrabber.com*, (accessed April 15, 2020). `http://www.pricegrabber.com/`.

[21] KuantoKusta. *KuantoKusta.pt*, (accessed April 15, 2020). `https://www.kuantokusta.pt/`.

[22] crunshbase.com. *Crunshbase*, (accessed May 15, 2020). `https://www.crunchbase.com/organization/pricegrabber#section-overview`.

[23] Yutao Zhu Xiaochen Zuo Juan Li, Zhicheng Dou and Ji-Rong Wen. Deep cross-platform product matching in e-commerce.

[24] Ajinkya More. Attribute extraction from product titles in ecommerce.

[25] Yan Liu1 Marko Krema Andrew Fano Rayid Ghani, Katharina Probst. Text mining for product attribute extraction.

[26] Ajinkya More. Product matching in ecommerce using deep learning. Sept 2017.

[27] Intelligente Node. How to classify, match products with machine learning. November 2018.

[28] Peter Mika Petar Ristoski, Petar Petrovski and Heiko Paulheim. A machine learing approach for product matching and categorization. 2016.

[29] Rosie Hood. Unravelling product matching in retail with ai. Nov 26, 2019.

[30] Yu-Tao Zhu Xiaochen Zuo Juan Li, Zhicheng Dou and Ji-Rong Wen. Deep cross-platform product matching in e-commerce. 13 August 2019.

[31] Kirill Fuchs. Machine learning: Classification models. Mar 28, 2017.

[32] Confusion matrix. Confusion matrix — Wikipedia, the free encyclopedia. [Online; accessed 10 March 2020].

[33] Semantics3. Product matching: You've not heard of it, but its powering your price comparison engine. 14 April 2015.

[34] Simon Fong. Framework of competitor analysis by monitoring information on the web. February 2012.

[35] Forrester Consulting. How dynamic pricing is revolutionizing retail.

[36] Capterra. Pricing tracking tools.

[37] Priceoptimization.org. *Revionics*, (accessed April 10, 2020). `https://priceoptimization.org/providers/revionics/`.

[38] Minderest. *Price Comparison Engine for Supermarkets*, (accessed June 15, 2020). `https://www.minderest.com/price-comparison-engine-supermarkets`.

[39] Power BI. *Microsoft Power BI*, (accessed June 20, 2020). `https://powerbi.microsoft.com/pt-pt/`.

[40] AWS. *Amazon Web Services (AWS)*, (accessed May 20, 2020). `https://aws.amazon.com/`.