TU WIEN Informatics

Extracting structured data from semi-structured computer screen specifications in German

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Matthias Hagmann, BSc
Matrikelnummer 01526023

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Univ.Ass. Gábor Recski, PhD

Wien, 2. April 2024

Matthias Hagmann	Gábor Recski

TU WIEN Informatics

Extracting structured data from semi-structured computer screen specifications in German

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Matthias Hagmann, BSc
Registration Number 01526023

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Ass. Gábor Recski, PhD

Vienna, April 2, 2024

Matthias Hagmann	Gábor Recski

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Matthias Hagmann, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. April 2024

Matthias Hagmann

Kurzfassung

Für elektronischer Produkte, wie Smartphones, Notebooks oder Computerbildschirme, gibt es eine Menge konkurrierender Geräte unterschiedlicher Hersteller. Benötigt man ein neues Smartphone, so ist es praktisch unmöglich alle relevanten Marken zu finden, deren Produktseiten zu durchforsten und sich über jedes einzelne Gerät zu informieren. Die relevanten Produkte können anhand von Eigenschaften, wie der maximalen Gerätelänge, gefiltert und eingegrenzt werden. Preis- und Produktvergleichsportale wie Geizhals und Idealo bieten dafür Filter-, Sortier- und Vergleichsfunktion an und eignen sich, um einen Überblick über aktuelle Modelle zu verschaffen. Auf welche Quellen dafür zurückgegriffen wird und wie die Aufbereitung funktioniert, ist nicht öffentlich bekannt. Die Daten werden jedoch von Menschen gepflegt und somit kontinuierlich manuell verbessert.

Diese Diplomarbeit untersucht die Frage, ob Produktdaten von Online-Shops ausreichen, um detaillierte Produktspezifikationen für ein Produkt zu erstellen. Die Untersuchung fokussiert sich dabei auf die Kategorie der Computerbildschirme und Onlineshops aus dem deutschsprachigen Raum, die auf Geizhals vertreten sind.

Im Rahmen der Arbeit wurde eine automatisierte Pipeline zur Datenextraktion von Produktwebseiten und Weiterverarbeitung in eine strukturierte und vereinheitlichte Form implementiert. Zudem wurde der ComputerScreen2023 Datensatz erstellt. Dieser enthält Produktdaten von 32.227 Produktseiten mit mehr als 2.000 unterschiedlichen Computerbildschirmen und deren Referenzspezifikation von Geizhals. Die Pipeline kombiniert Daten von mehreren Onlineshops, um erstellt daraus möglichst präzise Produktspezifikationen. Die erzeugten Daten lassen sich für Produktvergleichswebseiten mit Filter-, Sortier- und Vergleichsfunktion nutzen.

Die Leistung der Pipeline wurde durch ein Experiment mit dem ComputerScreen2023-Datensatz ermittelt. Die Basis-Pipeline-Implementierung, die ausschließlich reguläre Ausdrücke zur Datenextraktion nutzt, erreichte dabei eine Genauigkeitsquote von 59,67 % und eine Vollständigkeitsquote von 47,54 % pro Eigenschaft. Damit extrahiert die Pipeline durchschnittlich 14,5 korrekte Eigenschaften pro Produkt. Durch Nutzung eines Machine-Learning-Modells für die Extraktion, oder einer manuellen Anpassung der Konfiguration konnten die Werte leicht verbessert werden. Die Auswertung basierend auf 15 Eigenschaften zeigt das wahre Potenzial der Pipeline und erreicht eine Genauigkeitsquote von 89,13 % pro Eigenschaft und eine Vollständigkeitsquote von 72,01 % pro Attribut. Das ergibt einen F1-Wert von 79,66 %.

Abstract

For electronic products such as smartphones, notebooks or computer screens, there are a lot of competing devices from a wide range of manufacturers. If you need a new smartphone, it is practically impossible to find all the relevant brands, browse their product pages and inform oneself about each individual device. The number of products in question can be reduced by desired product properties, such as the maximum device length. Price and product comparison portals like Geizhals and Idealo offer filtering, sorting and comparison functions for this purpose and are suitable for providing an overview of current models. Their databases contain a huge amount of products with detailed product specifications. It is not publicly known which sources they have and how the processing works. However, the data is maintained by people and is therefore continuously improved manually.

This thesis examines the question of whether the data from some online stores is sufficient to create equally detailed product specifications as Geizhals and implements an automated pipeline for this purpose. Our investigation focuses on the category of computer monitors and online stores from German-speaking countries that are represented on Geizhals.

As part of the work, an automated pipeline for extracting data from product websites and processing it into a unified, structured data format was implemented. In addition, the ComputerScreen2023 data set was created. It contains product data from 32,227 product landing pages with more than 2,000 different computer screens and their reference specification from Geizhals. The pipeline combines data from several online shops to create product specifications that are as precise as possible. The resulting structured data can be used to build product comparison websites with filter, sort and comparison functionality.

The performance of the pipeline was determined by experimental analysis using the ComputerScreen2023 dataset with reference data for more than 2,000 computer screen. The base pipeline implementation solely relies on regular expressions for data extraction and achieved an attribute precision of 59.67 % and a recall of 47.54 % and mined an average of 14.5 correct properties per product. With the addition of a machine learning model for extraction, or a manual adjustment of the configuration, the scores slightly raise. Based on a selection of 15 attributes, the pipeline shows its true potential with an attribute precision of 89.13 % and a recall of 72.01 %, which results in an F1 score of 79.66 %.

Contents

CHAPTER 1

Introduction

In this thesis, I aim to solve a long-standing problem with relevance for product and price comparison websites. I present an information extraction pipeline, which mines data from online shops and creates well-structured and unified product data. The focus is on the category of computer screens, with the goal of retrieving a high number of valid attributes. The processed data is solely based on German-speaking shops, so the resulting specifications and property names are also in German. However, the implementation is adaptable for other languages and product categories as well. Therefore, attribute-value pairs of product specifications are extracted from the merchants' product landing pages. Based on predefined attributes and a regular expression for each attribute, this data is processed and converted into structured data. Combining data of the same product from multiple shops leads to one specification per computer screen.

In order to test and evaluate the pipeline, a dataset containing over 2000 different computer screens has been created by me. Therefore, the monitor category of the product and price comparison website Geizhals has been used as starting point. For each monitor, offers from multiple online shops are linked and a comprehensive table with product specifications are given on the Geizhals website. For the *ComputerScreen2023* dataset, these specifications are stored as reference data and 32,227 product specifications extracted from the linked online shops are stored in a JSON file as well. Thus, all the gathered data comes from freely accessible sources on the internet. The online shop's source code has been used as raw data for the extraction pipeline, and the reference data was used for the evaluation of it. A slight improvement is noticeable with the addition of machine learning model, or manual tuning of the configuration.

1.1 Problem Statement and Motivation

With online shopping, a large selection of products from various categories are available and advertised to everyone. But how to find the smartphone that best suits from an

overwhelming selection?

Online product catalog websites like Geizhals and Idealo can help with that. They provide a huge product database with goods from various categories, reaching from every sort of electronics up to clothing. Their product landing pages list several merchant offers for the identical product. Comparison and filter functionality make it easier to find the desired product. To provide this functionality, they need structured and unified product information. Based on that, numerous properties can be selected to thin out the gigantic selection of smartphones. Should it be a handy device? Then pick a maximum length of 150 mm and two thirds of around 3,000 listed smartphones are out [AG]. Furthermore, pick an operating system and only several hundreds are left. Additionally, cut off devices older than two years and limit the price tag. Only a handful of devices are left, which can be examined in detail. The same approach also works for computer screens, but with different properties. Pick a range for the screen diagonal and select the ports it should have. Further options to pick from can be ergonomic features or the panel type.

However, product catalogs require workers to manually add properties and constantly enhance the product information in their database. A completely manual approach is not feasible for millions of products. Therefore, a fully automated method of data collection and processing must be pursued. In contrast to extracting an average of 13.3 attributes per product [NFP+11], the pipeline will collect more properties for the category of computer screens and get close to the attribute number and details as Geizhals provides. Although there are many related research topics based on product data extraction, I found insufficient literature on gathering specifications from German-speaking product websites. Furthermore, I could not find a solution that aims for the variety of properties and data quality, which Geizhals seems to reach with additional manual labor [Gei]. For its competitor Idealo it is rumored that even 500 data maintainers keep their product catalog up to date [KK22]. So there seems to be a lot of room to automate these processes.

In this thesis, I propose a solution to create such structured data from shop landing pages automatically. Thus, the research questions I tackle in this thesis are:

- Is it possible to create high quality product specifications solely from merchant landing pages?
- Does Geizhals use more data sources, than I had, to create their detailed product specifications?

1.2 What is semi-structured data?

This thesis distinguishes between the following three data formats.

1.2.1 Plain text

Some product properties might be given in text paragraphs, where they are hard to retrieve. With a regular expression, collecting lengths like ’20 mm’ is simple, but you

also need to find the property name to which it corresponds to. This is challenging, as it can be the height of a small screen, the thickness of the delivered package or the range of the height adjustable stand.

1.2.2 Semi-structured data

According to the merchant data, far more product properties are based on lists, such as attribute-value pairs. They contain many properties and are easier to collect, than the sparse data from plain texts. Thus, the focus of this work is on partly structured, but inhomogeneous data, as this will bring the best results with limited resources. Lists are often visualized as bullet points with text, where each point holds related product data.

Lists on websites are meant to be characterized by an unordered list tag *<ul>* and multiple list item tags *<li>*, which are semantic tags in HyperText Markup Language (HTML). The same applies to tables, where a *<table>* tag should wrap the row elements *<tr>*, which are further split into columns using the table data *<td>* tags. However, many websites do not use semantic HTML and rely on their own way to structure the HTML code, often with plain * * tags. This makes it harder to extract property names and values from lists of attribute-value pairs automatically.

Although the data structures are similar, the values are not unified. As visualized in Figure 1.1 and Figure 1.2 the property titles differ and also the values are given in different formats. In this thesis, such data is called semi-structured.

1.2.3 Unified and structured data

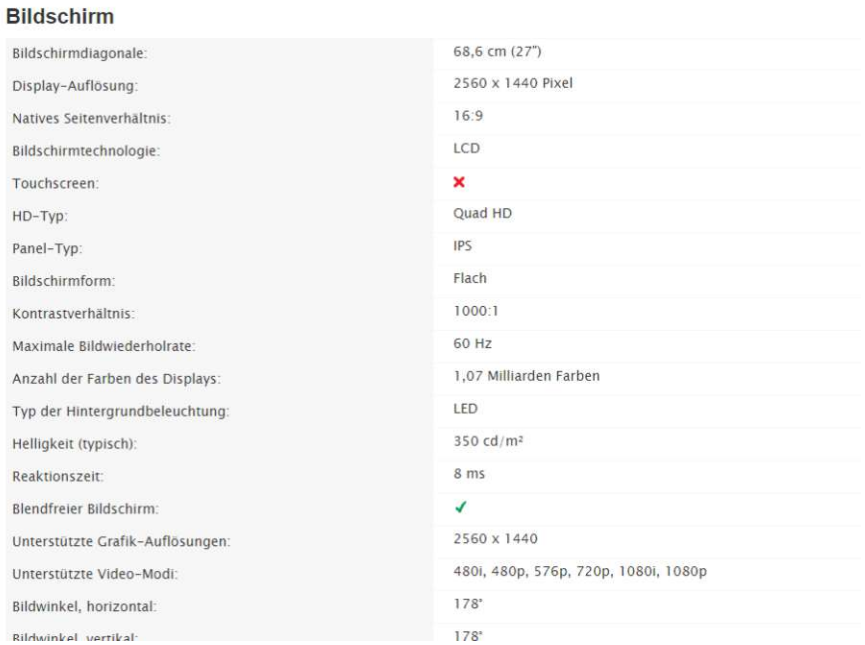
A data format that follows strict rules, is called structured data in this thesis. Unified and structured data is comparable to a dictionary in Python, with a fixed set of allowed keys and also distinct format of the values. An example is shown in Figure 4.10.

1.3 Contribution

I tackle the problem of information extraction and build an automated pipeline for the category of computer screens. In contrast to other research papers, I focus on semi-structured data extracted from product landing pages in German language. Therefore, the research data will be based on shops with product offers for Austria and Germany.

The thesis shows the positive sides and the shortcomings of a traditional extraction approach based on manually crafted regular expressions. According to preliminary tests, specifications like the screen resolution and screen diagonal are well retrievable with regular expression, as the values mostly conform to a standard format. In contrast to that, the ports of a computer screen are hard to gather with these conventional methods, as they are specified in highly different formats (compare Figure 1.3 and Figure 1.4).

Building on this knowledge, I aim to mitigate this problem with a machine learning approach, based on *Natural Language Processing* (NLP). I fine-tune a *Bidirectional*



Bildschirm

Bildschirmdiagonale:	68,6 cm (27")
Display-Auflösung:	2560 x 1440 Pixel
Natives Seitenverhältnis:	16:9
Bildschirmtechnologie:	LCD
Touchscreen:	✗
HD-Typ:	Quad HD
Panel-Typ:	IPS
Bildschirmform:	Flach
Kontrastverhältnis:	1000:1
Maximale Bildwiederholrate:	60 Hz
Anzahl der Farben des Displays:	1,07 Milliarden Farben
Typ der Hintergrundbeleuchtung:	LED
Helligkeit (typisch):	350 cd/m²
Reaktionszeit:	8 ms
Blendfreier Bildschirm:	✓
Unterstützte Grafik-Auflösungen:	2560 x 1440
Unterstützte Video-Modi:	480i, 480p, 576p, 720p, 1080i, 1080p
Bildwinkel, horizontal:	178°
Bildwinkel, vertikal:	178°

Figure 1.1: Semi-structured product specifications from a shop

Produktbeschreibung	Dell UltraSharp U2722DE - LED-Monitor - QHD - 68.47 cm (27")
Gerätetyp	LED-hintergrundbeleuchteter LCD-Monitor - 68.47 cm (27")
Energie Effizienzklasse	Klasse E
Leistungsaufnahme im Ein-Zustand	20.4 W
Leistungsmerkmale	USB 3.2 Gen 2/USB-C Hub, USB PD 90 Watt
Bildschirmtyp	IPS
Seitenverhältnis	16:9
Native Auflösung	QHD 2560 x 1440 bei 60 Hz
Pixelpitch	0.2331 mm
Helligkeit	350 cd/m²
Kontrast	1000:1
Reaktionszeit	8 ms (normal); 5 ms (schnell)
Farbunterstützung	1,07 Mrd. Farben
Eingangsanschlüsse	HDMI, DisplayPort, USB-C
Einstellungen der Anzeigeposition	Höhe, Pivot (Rotation), Drehung, Neigung
Bildschirmbeschichtung	Blendfrei, 3H Hard Coating
Spannung	Wechselstrom 120/230 V (50/60 Hz)
Abmessungen (Breite x Tiefe x Höhe) - mit Fuß	61.134 cm x 18.5 cm x 38.513 cm
Umweltschutzstandards	TCO Certified Displays 8, TCO Certified Edge Displays, ENERGY STAR-qualifiziert
Kennzeichnung	Plug and Play, RoHS, NFPA 99, TUV Rheinland, BFR-frei, DisplayPort 1.4, HDCP 1.4, PVC-free
Services im Bundle	3 Jahre Advanced Exchange-Service
Entwickelt für	Latitude 5320, 5520; OptiPlex 3090; Precision Mobile Workstation 7560; XPS 13 9310, 15 9510

Figure 1.2: Semi-structured product specifications from another shop

Anschlüsse und Schnittstellen

Integrierter USB-Hub:	✓
HDMI:	✓
USB-Hub-Version:	3.2 Gen 2 (3.1 Gen 2)
USB-Upstream-Porttyp:	USB Typ-C
Anzahl der vorgeschalteten Steckplätze:	2
Anzahl der Upstream-Ports USB Typ-C:	2
Anzahl der nachgeschalteten Steckplätze vom Typ USB-A:	4
Anzahl der nachgeschalteten Ports vom Typ USB-C:	1
USB-Typ-C DisplayPort-Wechselmodus:	✓
USB-Stromversorgung bis zu:	90 W
DVI Anschluss:	✗
Anzahl HDMI-Anschlüsse:	1
HDMI-Version:	1.4
Anzahl DisplayPort Anschlüsse:	2
DisplayPorts-Version:	1.4
Audioausgang:	✓
Kopfhörer-Anschluss:	3,5 mm

Figure 1.3: Port specific properties from a shop

Anschlüsse

Anschluss-Typ	DisplayPort 1.4, HDMI 1.4, USB-C
Weitere Anschlüsse	RJ-45 (LAN), Kopfhörer-Ausgang
HDMI	1 ×
DisplayPort	1 ×
USB-C	3 ×
USB	4 ×

Figure 1.4: Port specific properties from another shop

Encoder Representations from Transformers (BERT) transformer model and use it for token classification in order to extract the relevant port details for HDMI and DisplayPort. The aim of this work is a generalizable solution, which is applicable for all shops. This is not the most straightforward solution, but with the greatest benefit in practice. The evaluation results from chapter 6 will reveal shortcomings of the proposed pipeline and give hints for further improvements. Secondly, the results will show, if merchant websites deliver enough data to create detailed specifications.

Additionally, I evaluate the pipeline with a manually enhanced pipeline configuration.

1.3.1 ComputerScreen2023 dataset

I offer a dataset with reference specifications for 2,555 different computer screens and 32,227 JSON files with product data extracted from product landing pages of several

online shops. Initially, the raw HTML source code of more than 45,000 pages has been gathered from more than 200 online shops, which sell their goods in Austria and Germany. I have collected links to the merchants by crawling the offers from Geizhals product landing pages, as shown in Figure 3.1. Due to the size of all these HTML files together, I provide the extracted, but unedited product data as attribute-value pairs from the 30 most common shops as JSON. Due to the fact, that the extraction of the raw specifications from HTML is based on manually crafted CSS selectors for each shop (compare Figure 4.2), I am rather certain to catch the main attribute-value pairs of a shop. However, some merchants add additional information like EAN in secondary listings, which I do not collect. I provide the parser configuration for these shops, but do not make efforts to fully automate and generalize this extraction for of all shops.

Ground truth for evaluation

The product specifications from all the Geizhals product detail pages, shown in Figure 3.1, are also saved as reference data. They contain dense specification which are necessary for the evaluation of the pipeline. This reference data is also processed in the base pipeline, in order to convert it to a compatible, structured format for comparison with the extracted data from the landing pages.

CHAPTER 2

Related work

I propose an automated pipeline to gather structured computer screen specifications from semi-structured data. My base pipeline uses manually crafted regular expression to collect property values. Further processing puts the data into a predefined structure, in order to filter, search or compare products based on their specification. Afterward, monitors can be filtered to show, e.g. all computer screens with a diagonal between 27 and 32 inches providing DisplayPort.

Using natural language processing, I want to improve the extraction quality with a token classifier. According to my tests, the ports are especially hard to extract and even require knowledge to propose a useful catalog structure. For example, a USB-C port on its own is able to deliver data, various amounts of power and a DisplayPort signal. The hard part is, that the schema provided by merchants differ completely, which makes the extraction of ports with all their details especially hard.

The evaluation of a fully automated pipeline as proposed in [NFP+11] gathers an average of 13.3 attributes for products with more than ten merchant offers. According to preliminary tests, the pipeline is able to extract a lot more attributes, even with data from single merchants. However, I can not evaluate the pipeline with other datasets for comparison, as I could not find one for the domain of computer screens and also none using German language.

2.1 Extracting attribute-value pairs from product specifications

Petrovski et al. aim to solve the problem of product specification extraction from HTML tables and lists [PB17]. Product comparison portals show product specifications together with offers from many shops. An offer consists of a title and a free-text product description containing some product attributes. Additionally, such an offer may contain product

specifications as tables or lists with many more product properties. Extracting these attribute-value pairs is necessary for product matching, categorization, faceted product search and product recommendation.

Dataset Their evaluation and training dataset consists of the Web Data Commons Gold Standard for Product Matching and Product Feature Extraction [PPMB15] with 564 products. It contains annotated specification lists and gold labels for product data extraction.

The implementation of specification extraction consists of two steps. First, the detection of specification tables or lists. Therefore, they use a binary classification model based on SVM with a linear kernel. The classifier relies on several features, such as the number of links or the average ratio between numerical and alphabetical characters in a cell. Secondly, for the extraction process, they use supervised learning to identify attribute and value columns. The data then gets extracted from these HTML fragments.

Their table extraction approach outperforms DEXTER [QBD+15] in F-score, which is based on precision and recall, by 10%. The detection of tables with more than a single attribute-column is claimed to lead to the significant increase. For lists, the improvement to DEXTER is 3%.

The challenges for future research directions are specification extraction from lists. This is more difficult compared to tables, as they sometimes do not even contain any type of delimiter.

2.2 Extraction of product specifications beyond tables and lists

Product specifications in structured tabular blocks are often present on E-commerce product listings. Thus, extraction of these specifications from product landing pages is a big benefit in product catalog creation and for product search. However, websites rely on different HTML tags like `<table>`, `<div>` or `<ul>` to render them, which makes the specifications harder to find and extract.

Gangadhar et al. [GK22] suggest an approach that identifies and extracts product specifications as attribute-value pairs in two stages. In addition to that, their approach is not limited to tables and lists.

Existing product datasets with specifications seemed limited to these two HTML building blocks. Thus, they created a labeled dataset of product specifications gathered from 14,111 product websites, that are composed of any kind of HTML blocks.

An efficient SVM model with a linear kernel is used as pre-filter, leaving only a small amount of HTML blocks for further classification. For the remaining blocks, a CNN model with word embeddings is used to classify them as specification or not. The attribute-value

extraction is the second stage and relies on wrapper induction techniques based on known attribute names and values.

The evaluation of the specification and non-specification detection shows a 1.6 % improvement of F1-score compared to the approach by Petrovski et al. [PPMB15]. However, the extraction task, evaluated for tables, performed 6.7 % worse. Nevertheless, the approach shows its ability to extract attribute-value pairs from specification blocks built from unknown HTML tags.

Specification blocks on product pages consist of various HTML tags. Without a limited standard set of HTML elements, it remains a challenge to find the specifications and extract data automatically.

2.3 Synthesizing products for online catalogs

Product search engines like Google Shopping require an extensive product catalog with structured properties in order to be successful. However, adding new products is a challenging task that requires automation to keep up with all the updates. Therefore, Nguyen et al. present an automated end-to-end solution on product synthesis for online catalogs [NFP+11]. Based on a set of merchant offers, new products should be identified and added to the product catalog, along with its structured attributes.

Their product synthesis pipeline consists of four big components. Merchants provide offer feeds with the title and URL to the product landing page. The latter website contains the product specifications in a structured way, e.g. as HTML table, but the schema varies between merchants and products. So, the first step consists of the product categorization based on its title, as each product category has its own attributes of interest. From the extracted attribute-value pairs, the schema reconciliation component maps merchant attribute names with the attributes, that already exist in the catalog. For example, "Hard Disk Size" is mapped to "Capacity" from the catalog. Third comes the clustering component, which groups offers of the identical product, based on key attributes like the *Manufacturer Part Number*(MPN). Lastly, the data from one cluster is merged to a single product specification using majority voting.

Dataset The dataset they used for evaluation is based on more than 800,000 product offers. It consists of web pages, gathered from product pages of various shops listed on Bing Shopping. It contains data from more than 1000 merchants and 400 categories, including electronic goods, such as computers and hard drives.

The evaluation results show, that 287,135 products (with 1,126,926 product attributes) were synthesized out of 800,000 products given as input. From a sample of 400 products, consisting of 1,447 attribute-value pairs the evaluation shows an attribute precision of 92 % and a product precision of 85 %.

The challenge of product synthesis lays within the heterogeneity of data from merchant offers. Shops rely on different attribute names and schemas, which often lack a variety

of attributes. Therefore, locating the manufacturer's product page to extract relevant information should enrich the specifications.

2.4 Overview of related datasets

My paper also describes the creation of a dataset with computer screens, which is called ComputerScreen2023 dataset. The two datasets with electronic goods used in the related papers are summarized in the comparison Table 2.1. In contrast to my dataset, they contain data in English.

Dataset name	**Categories**	**Data format**	**Products**
Bing search	498 (laptops, hard drives)	Web page	856,781
WDC Gold Std.	TVs, headphones, phones	Web page + annotation	564

Table 2.1: Overview of related product datasets

CHAPTER 3

The ComputerScreen2023 dataset

In order to build and run an automatic specification extraction pipeline, data is necessary to test and evaluate it. Due to my focus on the niche of German and Austrian shops, datasets with relevant product data were nowhere to be found. However, with publicly available data for various electronic goods on the Internet, I decided to collect data on my own and build a dataset with product specifications in German language. As the goal of the thesis are dense product specifications, I focused on one category. I selected computer screens due to familiarity with their specifications and particularities, as well as their wide variety of relevant properties.

For this dataset, I automatically gathered data from all computer screens listed on Geizhals [AG]. There, each product has its own landing page with detailed structured specifications, as well as offers of this product from several merchants. Figure 3.1 shows a screenshot from such a landing page. I grabbed the product specification table for every computer screen and saved it as reference data. In addition, a list of product offers from several online shop is shown on the bottom. Each of them lists their shop name, the price and a link to their online shop. Based on these links to a shop's product landing page, I gather the page source (HTML) and extract product specification tables and lists. The resulting raw data is stored in JSON files and is later used as input for my proposed pipeline. These two types of data, the reference from Geizhals and the product specifications from multiple shops, build up the foundation of my *ComputerScreen2023* dataset.

It contains data for more than 2,000 different monitors and over 30,000 raw product specifications extracted from German-speaking online shops. For both of them, data is stored as attribute-value pairs in a JSON format. The ComputerScreen2023 dataset is available as part of the source code on GitHub [Hag].

The dataset solely contains public data, collected on October 1, 2023. Geizhals listed 2,555 different computer screens on that day, which are all part of this dataset. In

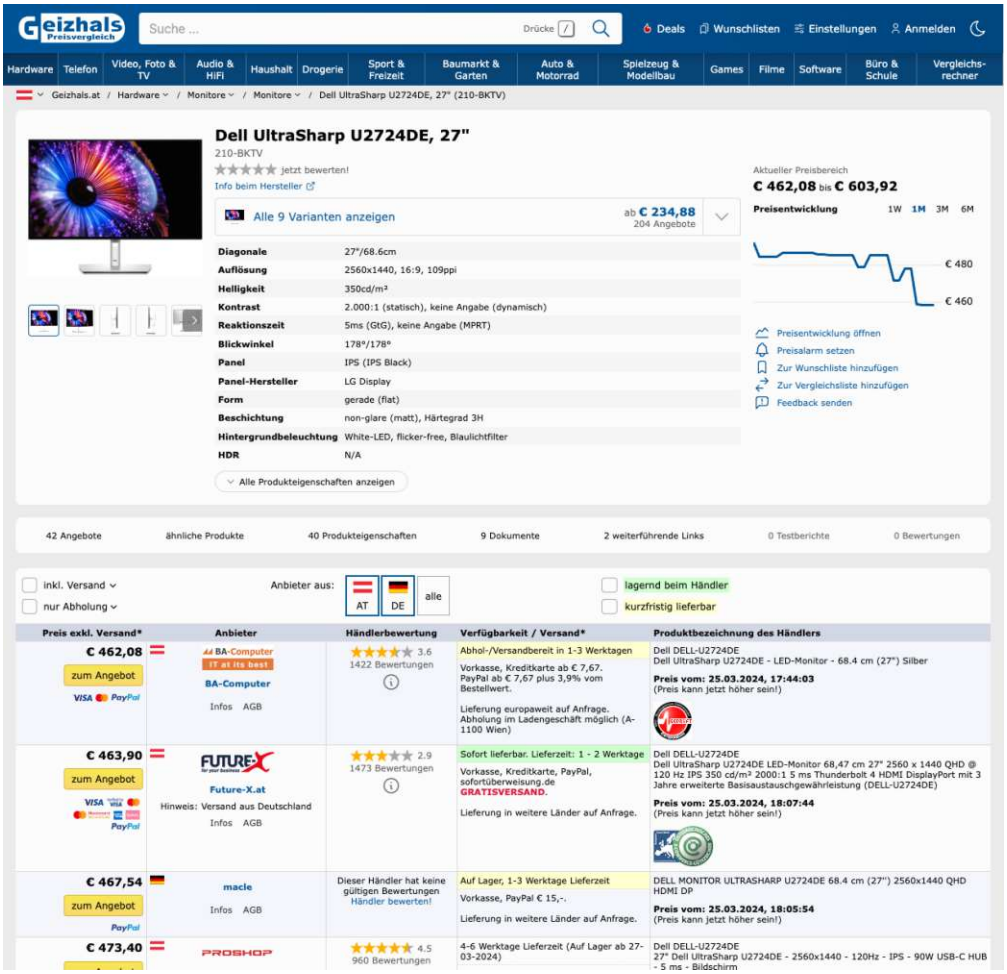


Figure 3.1: Geizhals product page with detailed specifications and merchant offers

addition to the reference data from Geizhals, 45,460 product landing were gathered from more than 250 online shops. For the most common 30 shops, manually crafted parser configurations allowed me to extract product specifications from tables or lists. The unedited attribute-value pairs are part of the dataset and provided as JSON files. The huge number of HTML files makes them quite big and are thus not part of the dataset.

3.1 Public data

The starting point for dataset creation was the price comparison website Geizhals. It is a freely accessible product catalog website with a focus on price comparison of electronic goods and shows an extensive list of product specifications. The website also offers search and filter capabilities, with the granularity of filter options depending on the product category. To provide this, the website must have well-structured and unified data in their

backend. On product pages, Geizhals shows these specifications in a list with property title and according values. It is worth to note, that Geizhals groups several related property values in a single field. For example, the color gamut for sRGB, DCI-P3 and more are all shown in the same row, separated by a comma.

In addition, it lists online shops that offer this product, together with the price and link to the offer. Keep in mind, that the website is profit oriented and having a shop listed comes with a cost. Hence, the dataset certainly does not contain all websites, that sell the product in Austria and Germany and solely pays attention to the shops listed on Geizhals. However, most well known shops are listed there, and it can be believed, that the vast majority of computer screens sold in Austria in 2023 are contained in the ComputerScreen2023 dataset.

3.2 Content of the dataset

The ComputerScreen2023 dataset contains product specifications, but also metadata with additional information, such as product and shop names, as well as links to the original data source. Additionally, each file in the dataset contains an ID incorporated in its filename. The ID is always the first number, that occurs in a filename, and equal numbers refer to the same product.

3.2.1 Metadata allows clustering of equal computer screens

Files with matching ID all contain data for the same monitor. The link between the product offers and their common reference specification was solely gathered from Geizhals product detail pages, as shown in Figure 3.1. Thus, the dataset also provides ground truth data for clustering of computer screens. I assume this data is very accurate. Hence, the ComputerScreen2023 dataset is usable as training and validation dataset for a machine learning approach to product clustering and evaluating product matching algorithms.

3.2.2 Reference specifications and shop offers

Geizhals lists more than 2,500 computer screens together with their specifications. These specifications and product offers were collected from Geizhals product details page and are provided as unedited attribute-value pairs in JSON format, together with some metadata and merchant offers.

Each *reference file* contains the following data, which is solely gathered from Geizhals:

- Product name
- Geizhals link
- Geizhals product specifications as list of attribute-value pairs
- Shop offers with shop name, link to product landing page and price

Based on Figure 3.1 the attributes relate to the left column of the product specifications and the right column describes the related cell with one or multiple values. The latter are often separated within a cell via comma.

3.2.3 Shop offers

The shop offers provide the links to collect data from 45,460 product landing pages, served by 288 shops. On average, the same computer screen is offered by 17.8 shops on Geizhals. As the raw source code (HTML) of all these landing pages is huge, I provide the extracted, but unedited attribute-value pairs of the most common 30 shops as JSON.

Each *raw specification file* contains the following data, which is solely gathered from online shops:

- Product name
- Product price
- Product specifications as list of attribute-value pairs
- Shop name
- Shop link
- Name of the reference file, containing the reference specifications.

The attribute-value extraction from HTML is described in more detail in section 4.3.

3.2.4 Gold labeled dataset for HDMI and DisplayPorts

I provide gold labels for the HDMI and DisplayPort specifications of 200 computer screens. Label Studio, an open source tool for data labeling, was used for the manual annotation task. The program provides a graphical user interface (GUI) for a wide variety of labeling tasks for machine learning applications. Label Studio randomly selected the monitors based on all the shop offers. As the labels were solely created by the author as a single annotator, no inter-annotator agreement scores exists. An example of a manually annotated computer screen is given in Figure 3.2.

For each port, the following four labels are available:

- Port type, e.g. 'Mini DisplayPort'
- The port count, e.g. '2x'
- The version, e.g. '2.0'
- Additional details, e.g. '120Hz@2560x1440'

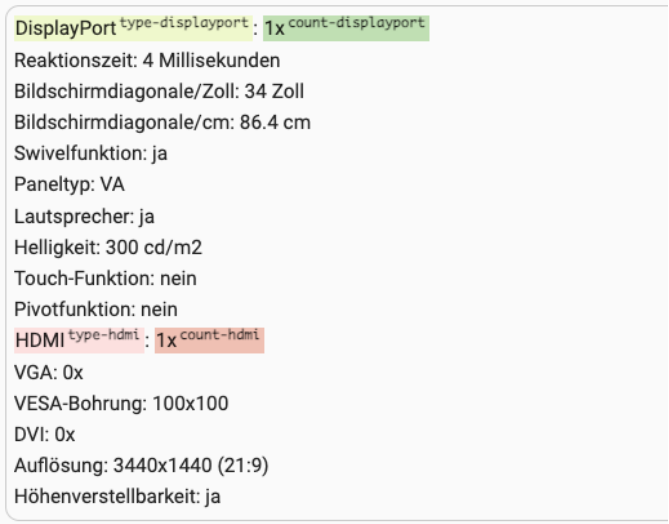


Figure 3.2: Gold labeled data as visualized in Label Studio

Be aware, that some shops also mention cables, which are supplied within the product box. Although, they include the port name, they are not assigned a label, as they are not a port. The labeled dataset is exported as CoNLL file, which is similar to the better known comma-separated value (CSV) format. Label Studio automatically applies a Word Piece tokenizer, so the exported data is split into tokens according to that. The granularity of the labels is therefore based on whole words and not individual characters.

Use case

These so-called gold labels or ground truth data is necessary for fine-tuning general purpose machine learning models like BERT on a specific task. Such transformer models are pre-trained on a huge dataset using powerful hardware, which is not feasible by individuals. Fortunately, some of these models are provided free of charge and can be further trained on specific tasks with much lower effort. Fine-tuning is possible with consumer hardware, and the tuned models aims to do noticeable better on the trained task. For this thesis, the labeled data is used for fine-tuning a BERT model and enhance the information retrieval pipeline (compare chapter 5).

CHAPTER 4

Information extraction pipeline

The pipeline for product data extraction from semi-structured data consists of two parts. First is the setup part, which is automated to a big extent. Second, the processing stages do the actual work and build the information extraction pipeline.

The proposed implementation creates structured and unified product specifications for computers screens by processing semi-structured data. The basic idea is based on the paper *Synthesizing Products for Online Catalogs* from Hoa Nguyen et al. [NFP+11]. The pipeline can bed fed with the HTML of computer screen landing pages from online shops or from already extracted, but unedited attribute-value pairs as provided in my ComputerScreen2023 dataset from chapter 3. Based on these semi-structured specifications, regular expression are used to extract specific properties and put them in a structured and unified format.

The hard part in this is the alignment of all the properties towards the predefined property names. The attribute names retrieved from the merchant's landing pages need to be aligned with my manually defined property names, which are the foundation of the unified structured data format. This problem is solved with field mappings, which link the predefined property names to the shop's property, which contains that information. The field mappings can be automatically created via fuzzy matching, but manual adoption leads to even better results. In addition, each predefined property is tight towards a manually crafted regular expression. This regex parses the property-value and extracts several groups of text. A width given as string '68.8cm' is split up into the parts 'number' and 'unit'.

The resulting structured data from one landing page is then fused with data from other shops to form a single instance of product specifications for each computer screen. The resulting structured and unified data is then stored in a JSON file per computer screen. All these files together build a database of uniform computer screen specifications, which

can be used for real world applications like product catalog websites with filter, search and product comparison functions.

Figure 4.1 gives a high level overview of all the stages. The preparation tasks are shown in green and processing stages in blue. The gray boxes represent the data flow through the pipeline and visualize the transformation of the data in each stage. The orange box represents real world use cases, for which the resulting database of computer screen specification can be used.

The implementation of the pipeline is solely written in Python.

4.1 Preparation

4.1.1 Define catalog properties for unified data structure

Shops supply their product specifications using various names for the same property, such as *Farbraum* or *Farbraumabdeckung* for the color gamut. As there are many synonyms in use, the property names for the unified database need to be defined. For each property, that should be gathered a German word like *Helligkeit* for the screen brightness has been chosen, and the expected data format has been defined. For the brightness, a number in cd/m^2 is expected. The pipeline supports extraction of more than 50 properties, an excerpt of them is shown in Table 4.1.

In addition, it can also extract the cables included in the box and the screen bezel sizes. I have disabled these properties for evaluation, as there is no reference data to compare them to. Moreover, this data is only supplied by a few shops.

4.1.2 Configure HTML product data parser

This pre-processing step is only necessary to extract semi-structured data as attribute-value pairs from HTML files. The ComputerScreen2023 dataset already provides the resulting data as JSON files (compare subsection 3.2.3).

Merchant landing pages contain product specifications in a table or list-like format (compare section 1.2). Hence, the relevant attribute-value pairs need to be extracted for further processing in the pipeline. I manually defined parser configurations for the web mining and data extraction tool minet[mS]. An example configuration using CSS descriptors to extract product specifications from the HTML of a specific shop is given in Figure 4.2.

The iterator targets the table rows, where each row contains a column for the property name and another one for the property value. It extracts these entries for each row of the gray highlighted box, shown in Figure 4.3. The CSS selectors then target the column tags inside the parent to gather name (highlighted in red) and value (highlighted in pink).

Such parser configurations have been set up for the 30 most common shops of the dataset, based on their product count. This results in approximately 70 % of the initially

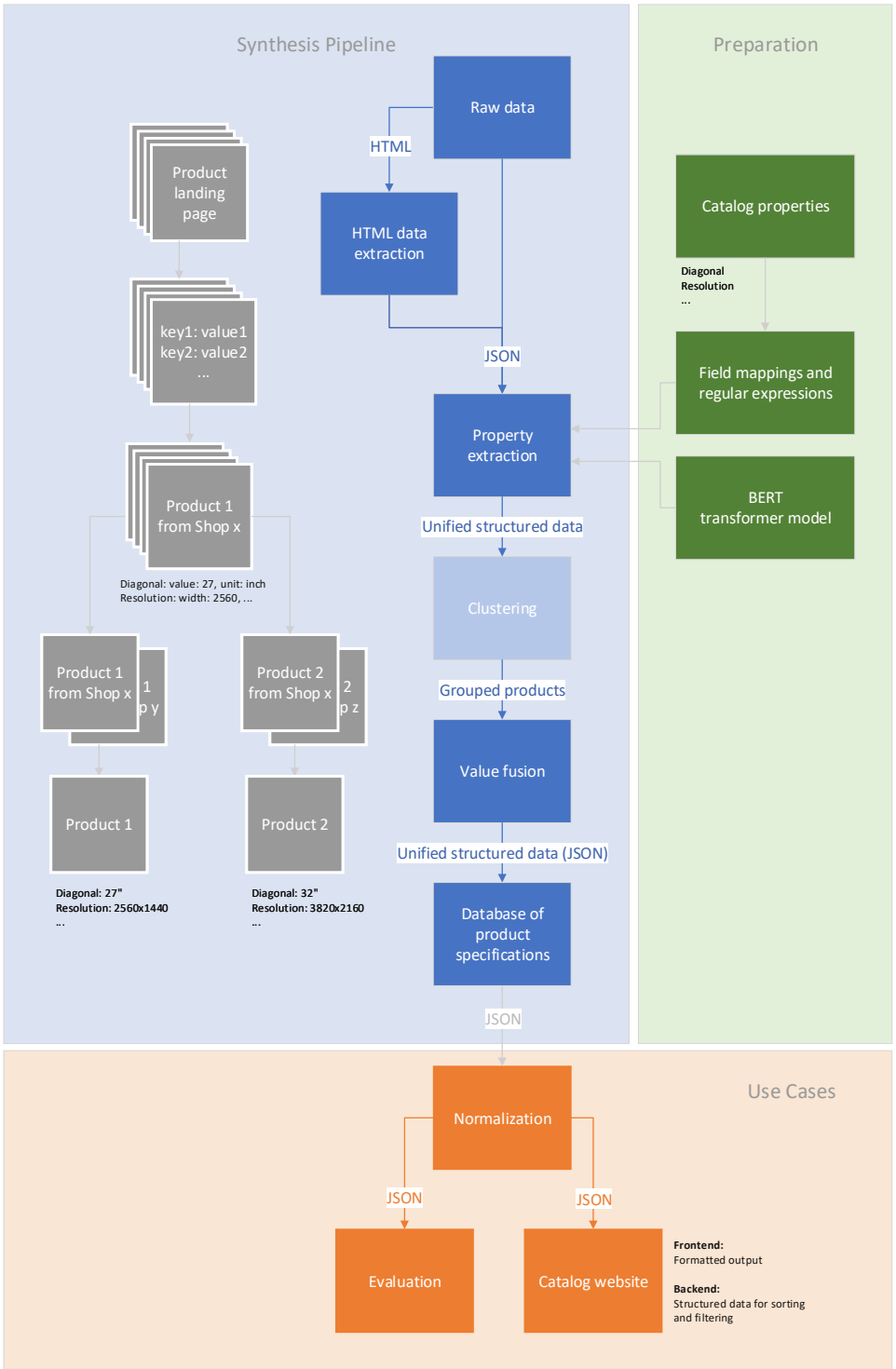


Figure 4.1: High level overview of the data extraction pipeline

Term in German	English term	Example value
Abmessungen	Dimensions	613.2x384.6x193.3mm
Anschlüsse DisplayPort	DisplayPort ports	1x DisplayPort 1.4
Anschlüsse HDMI	HDMI ports	1x HDMI 2.1
Anschlüsse USB-A	USB-A ports	1x USB-A 3.1
Auflösung	Resolution	2560x1440
Bilddiagonale (Zoll)	Screen diagonal (inch)	27"
Bildwiederholfrequenz	Refresh rate	60 Hz
Blickwinkel horizontal	Horizontal viewing angle	178°
Blickwinkel vertikal	Vertical viewing angle	178°
Farbraum DCI-P3	Color gamut DCI-P3	98%
Farbraum sRGB	Color gamut sRGB	100%
Farbtiefe	Color depth	10 Bit
Gewicht	Weight	2.98 kg
Herstellergarantie	Manufacturer's warranty	2 Jahre
Helligkeit	Brightness	350 cd/m^2
Kontrast	Contrast	1000:1
Leistungsaufnahme (SDR)	Power consumption (SDR)	28 W
Neigungswinkelbereich	Range of tilt angle	+21°/-5°
Panel	Panel	IPS
Reaktionszeit	Response time	5 ms
Seitenverhältnis	Aspect ratio	16:9
VESA	VESA	100x100

Table 4.1: Excerpt of extractable properties

```
iterator: div:is(.more-desc, .more-descs) tr
fields:
  name:
    sel: td:nth-child(1)
  value:
    sel: td:nth-child(2)
```

Figure 4.2: HTML parser configuration with CSS selectors

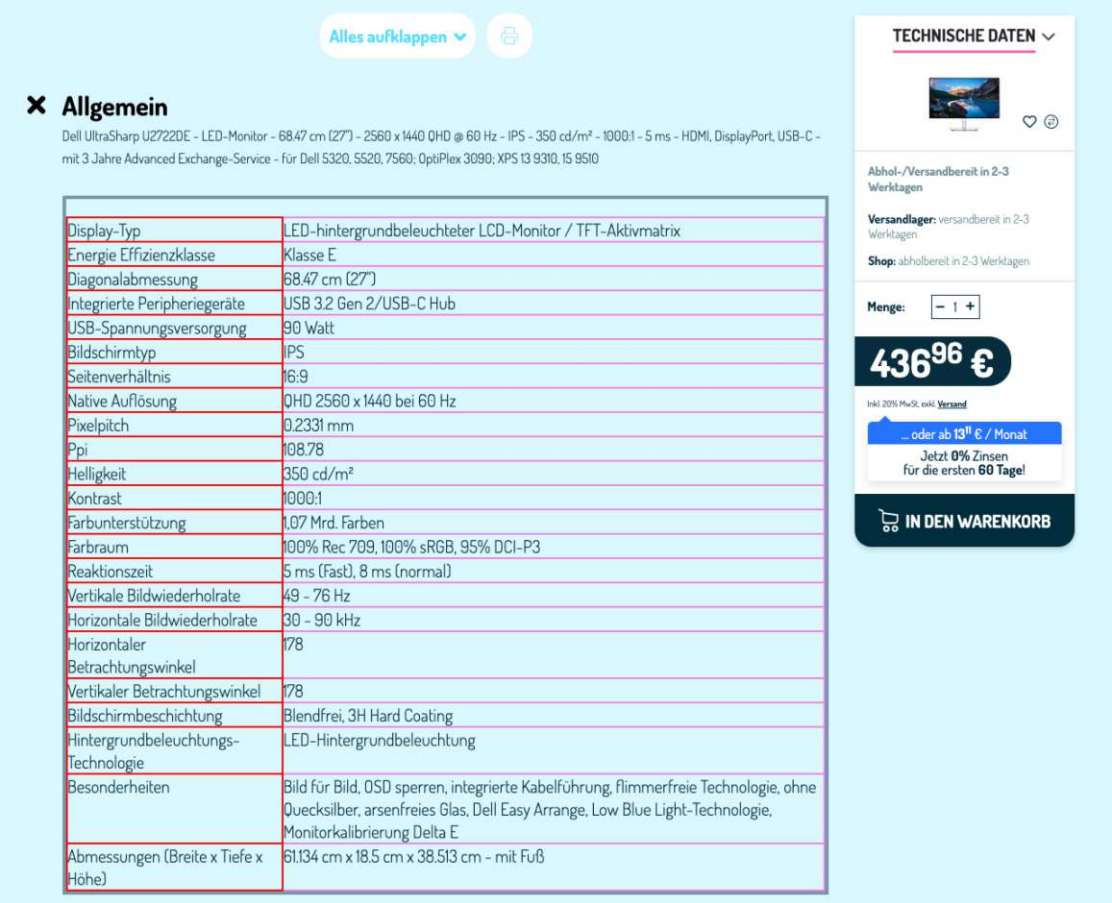


Figure 4.3: Semi-structured product data from an online shop

gathered data from computer screens landing pages (32,227 out of 45,460) being provided in the ComputerScreen2023 dataset and used for the evaluation of the pipeline. As filtering proper data from HTML is a research topic on its own [GK22] I solely considered extraction with manual parser configurations.

Extracted attribute-value pairs

The resulting data consist of a bunch of attribute title and value pairs, as shown in Figure 4.4.

4.1.3 Manually crafted regular expressions

In the base pipeline, the data extraction relies on regular expressions, which have been created by me based on Geizhals reference specifications and sample data from shops. In addition, the expressions have been generalized to support different spellings and deal with different use of whitespace. Each property defined in the catalog is linked to one

```
{
    ...
    "USB-Spannungsversorgung": "90 Watt",
    "Bildschirmtyp": "IPS",
    "Seitenverhältnis": "16:9,
    ...
}
```

Figure 4.4: Extracted JSON data from HTML

regular expression. They do the hard work and extract numbers, units, lengths, time spans and more from attribute value strings.

A simple example is the screen diagonal given as '68.6 cm', which is converted to a value string of '68.6' and a unit string of 'cm'. The related regular expression is shown in Figure 4.5. It extracts lengths into number and unit, when given in millimeter, centimeter or meter. The first round brackets define, what is taken as number and the second determines the characters gathered as unit.

```
(\d+[.,]*\d*)\s*(mm|cm|m)
```

Figure 4.5: Regular expression for screen diagonal extraction

The regular expression for the color depth is shown in Figure 4.6. A typical value for this is '10 Bit', which is split up into value and unit as well. The weight in kilogram or

```
(\d+)\s*(\D*[bB]it)
```

Figure 4.6: Regular expression for color depth extraction

gram is extracted similarly, as shown in Figure 4.7. It also results in a value and unit.

```
(\d+.?\d*)\s?(kg|g)
```

Figure 4.7: Regular expression for weight extraction

4.1.4 Field mappings align attribute names

The regular expressions are used to create structured data from the raw string values. An example of these values can be seen in Figure 4.3. The cells framed with a pink box

contain the attribute value. The regular expressions are applied to these text entries in the property extraction step of the processing pipeline.

At this time, it is unknown which regular expression should be applied to which row. A brute force approach, with every regular expression being applied on every row, seems possible. Nevertheless, even if a rule successfully extracts data, it is unknown to which catalog property it belongs to. That alignment step is crucial for the creation of a unified data structure, which is necessary for filter, sort and comparison functions. If you want to filter the products by screen brightness, the value needs to be accessible at a distinct place for all gathered products. That is why field mappings are necessary here. So, for data extraction with regular expressions, a mapping from a property name of the catalog properties to the related property name used by a shop (see red boxes in Figure 4.3) is necessary.

This mapping stores under which property name a distinct value is expected in the shop's raw data. Look at Figure 4.8 for an excerpt of a manually enhanced field mapping for a single shop. The catalog properties on the left link to the expected property name for this shop on the right. If there is no mapping for a specific key, then 'null' is shown, and a value for this property can not be collected for this shop via the regular expression approach.

```
"Besonderheiten": "Kennzeichnung",
"Bilddiagonale (Zoll)": "Bildschirmdiagonale",
"Bilddiagonale (cm)": "Bildschirmdiagonale",
"Bildwiederholfrequenz": "Maximale Bildwiederholrate",
"Blickwinkel horizontal": "Horizontaler Betrachtungswinkel",
"Blickwinkel vertikal": "Vertikaler Betrachtungswinkel",
"EAN": null,
"Energieeffizienzklasse": "Energie Effizienzklasse",
"Farbe": "Produktfarbe",
"Farbraum Adobe RGB": "Farbraum",
"Farbraum DCI-P3": "Farbraum",
"Farbraum NTSC": "Farbraum",
"Farbraum sRGB": "Farbraum",
"Farbtiefe": "Farbintensität",
"Form": "Bildschirmform",
"Gewicht": "Gewicht",
"HDR": "HDR-Format",
"Helligkeit": "Helligkeit (typisch)",
"Herstellergarantie": null,
```

Figure 4.8: Excerpt of field mappings for one shop

Automatic field mappings with fuzzy matching

I have introduced an automatic field mapper, which uses a given dataset, together with an example value for each property, as shown in Figure 4.9.

```
"Bilddiagonale (Zoll)": '27 "',
"Bilddiagonale (cm)": "68.6 cm",
"Marke": "Acer",
"Auflösung": "2560x1440",
"Helligkeit": "250 cd/m²",
"Reaktionszeit": "5 ms",
"Blickwinkel horizontal": "178",
"Blickwinkel vertikal": "178",
"Panel": "IPS",
```

Figure 4.9: Example values for some properties

For automatic creation of these links, an algorithm based on fuzzy matching with the Levenshtein distance has been implemented. The resulting score determines the similarity between two texts or single words. For each shop and property of the input data, the following procedure happens. First, the similarity scores between a property title from the input data and all catalog property names are calculated. If the best score is above a certain threshold, a possible mapping has been found, and it is added to the field mappings of that particular shop. The score related to such link is also stored. If a better mapping for the same shop and same attribute is found, then it overrides an existing mapping. This behavior aims to improve the field mapping quality.

One problem with value matching is, that they can perfectly match with similar, but incorrect properties. For example, the vertical and horizontal viewing angle are often given with a value of '178°', which is not distinguishable by its value alone. These entries require more context in terms of the property name. Thus, I have added a small penalty for value based mapping scores. A mapping found by similarity of property names can still be overridden by a mapping found via similarity of values.

The resulting JSON file contains mappings for each shop, and it is possible to fine-tune the mappings manually to enhance the pipeline results. An evaluation with one shop being enhanced with manual mappings is shown in section 6.4.

4.2 Data input

The implementation supports HTML pages from online shops or attribute-value pairs passed via JSON files.

4.3 HTML data extraction

To extract the attribute-value pairs from the HTML of merchant pages, a manually crafted and shop dependent parser is applied. Figure 4.2 shows such a configuration file for one shop. It converts the HTML data into a dictionary with the property name and value. An example of the resulting unedited shop data is given in Figure 4.4. For the 30 most common shops for computer screens at Geizhals, such parser configurations have been created. As websites are seldomly renewed, these configurations will be valid for a long time.

4.4 Extraction and alignment of properties

The pipeline consists of a baseline approach, that extracts values based on regular expressions and field mappings. An enhancement brings a machine learning model based on BERT, which has been fine-tuned on sequence labeling. It helps to retrieve port name, version and the number of ports for HDMI and DisplayPort. Further details are given in chapter 5.

4.4.1 Using regular expressions

Based on the property mappings shown in Figure 4.8 the schema matching component retrieves the corresponding merchant value. This plain text string is then parsed via a predefined regular expression.

The string provided for the screen brightness could look like '300 cd/m^2' and is one of the easier ones to get. The resulting pieces are the numerical value '300' and its unit 'cd/m^2'. From that point on, the brightness data is further processed in a structured format as shown in Figure 4.10.

```
"Helligkeit": {
    "unit": "cd/m^2",
    "value": "300"
}
```

Figure 4.10: Example of structured brightness data

This step is repeated for every attribute-value pair defined in the field mappings. At the end, the specifications are in a unified format and solely contains predefined property names. If a predefined known key, such as the German term 'Helligkeit' for the screen brightness is missing, then the brightness data was not available in the shop's data, or it could not be extracted with the given regular expression. To improve the situation, the regular expression can be relaxed or a more sophisticated approach is necessary.

4.4.2 Using a transformer model

For the latter, I trained a BERT transformer model to enhance the generalized extraction performance of the pipeline. The selected model is trained on *Named Entity Recognition* (NER) and fine-tuned on labeling port data using the gold labels included in the ComputerScreen2023 dataset. With limited resources in terms of time, processing power and training data, I set the focus on improving the quality of the extraction of ports, especially HDMI and DisplayPort. As data formats for these differ much from shop to shop, a regular expression is only of limited use for these properties. This approach does not require field mappings, but training on a prepared dataset for the learning process.

A detailed explanation of the machine learning approach is given in chapter 5.

4.5 Clustering products

Identical product offers e.g. with equal European Article Number (EAN) need to be matched by key attributes like EAN or other unique identifiers. However, the test data shows, that merchants in Austria and Germany seldom supply an EAN or a similar unique identification number in their specifications. That requires a more sophisticated approach, which is a whole research topic on its own [RPMP18, PB22]. Fortunately, I have initially gathered the data from Geizhals' product landing pages, and it provides the data clustered as necessary. This circumvents the need for a product matching implementation, and it is not addressed in this thesis. However, the ComputerScreen2023 dataset can be used for research on that topic, as it provides ground truth data for clustering. A more detailed description is given in subsection 3.2.1.

4.6 Value fusion

In this process step all extracted specifications related to the same product are merged into a single instance of unified, structured specifications. Common properties like the screen resolution and screen diagonal are supplied by various shops and need to be reduced to a single value. With a property being equal across data from all shops, that is trivial. However, filtering possible wrong values can be tricky when there are an equal number of competing values.

4.6.1 The strategy

I have chosen a fusion strategy, where the shop with most properties for this particular product wins and all its values are used. Then the next shop adds all properties, which are missing in the combined data. This continues with the priority of shops descending as they supply fewer specifications.

A voting is conducted for values which are delivered by multiple shops. If, by majority voting, more than half the of shops deliver the same value for an attribute, it overrules any other value.

Example Given 20 shops supply specifications for a product, and three out of five shops deliver the VESA mount property with '100x100' and the shop with most properties sets it to '75x75', then it is overwritten by majority voting with '100x100'.

At the end, a comprehensive product specification with data combined from all merchants is stored in the product catalog as structured and unified JSON data (compare Figure 4.11). The data structure is coherent among all computer screens present in the catalog. Filtering and sorting products by specifications, as well as in-depth property comparisons, are possible. The data is ready to build a product catalog listing with structured specifications like Geizhals.

4.7 Use cases

At this stage, the collection of unified computer screen specifications can be used for real world applications, such as a computer screen database with filter and search capabilities.

Two post-processing functions, which are likely necessary for any application, have been implemented.

4.7.1 Normalization

There is one caveat with the structured data produced by the pipeline - everything is stored as string. Even numbers and their units are kept as string, so further processing is left open for the application. However, normalization of numbers and units has been implemented for some simple cases and can be optionally applied.

With this additional step, numbers with a unit are converted into quantities, which allow sorting and filtering based on the data. Just consider sorting of two screens, one with 521 mm length and one screen with a length of 54 cm, ignoring the unit would not work. Additionally, time spans like 2 years and 24 months match after normalization. The automatic evaluation scores gathered in chapter 6 also apply this normalization before data comparison.

However, normalization is also beneficial for formatted and unified output of properties. On a product listing website, an attribute should be shown with the same unit and format for every product. In this case, normalized data is also helpful.

Apply synonyms

For some common keywords, a table with synonyms has been created. Synonyms, such as *schmaler Rahmen*, *Zero Frame* and *Slim Bezel* are replaced with the latter defined as the most common term by me. This allows better control of which keywords appear in the unified data.

```
{
    "Anschlüsse DisplayPort": {
        "count": "1",
        "value": "DisplayPort"
        "version": "1.2"
    },
    "Anschlüsse HDMI": {
        "count": "1",
        "value": "HDMI",
        "version": "1.4"
    },
    "Auflösung": {
        "height": "1440",
        "width": "2560"
    },
    "Beschichtung": "matt",
    "Bilddiagonale (Zoll)": {
        "unit": "Zoll",
        "value": "31.5"
    },
    "Bilddiagonale (cm)": {
        "unit": "cm",
        "value": "80.0"
    },
    "Bildwiederholfrequenz": {
        "unit": "Hz",
        "value": "75"
    },
    "Farbtiefe": {
        "unit": "bit",
        "value": "10"
    },
    "Form": "gerade",
    "Gewicht": {
        "unit": "kg",
        "value": "10.10"
    },
    ...
}
```


Figure 4.11: Computer screen specifications in JSON format

4.7.2 Formatted output

Printing the structured data in well human-readable way is a necessity for product databases as well as online shops.

The output format of my proposed pipeline is adaptable via a configuration. A collective term and the values summarized in one row are freely configurable. Figure 4.12 shows the formatted product specifications of a computer screen. It contains real data, as mined and extracted by the pipeline. The figure shows, that the collective term for the resolution, in German 'Auflösung', combines two properties in one field. The resolution and screen ratio are displayed as text separated by commas.

```
Diagonale:                      32Zoll, 79.8 cm
Auflösung:                      3840x2160, 16:9
Helligkeit:                     350 cd/m²
Kontrast:                       1000:1
Reaktionszeit:                  1 ms
Blickwinkel:                    178, 178
Panel:                          IPS
Form:                           gerade
Beschichtung:                   matt
HDR:                            HDR 400
Farbtiefe:                      10 bit
Bildwiederholfrequenz:          144 Hz
Variable Synchronisierung:      NVIDIA G-SYNC, AMD FreeSync ...
Anschlüsse:                     1x HDMI 2.1, 2x USB-A
Weitere Anschlüsse:             1x Gb LAN
Audio:                          1x Line-Out
USB-Hub In:                     1x USB-B 3.0
USB-Hub Out:                    2x USB-A 3.0
Ergonomie:                      17.6 cm, -15/15, 02/+13
Farbe:                          Schwarz
Gewicht:                        8.3 kg
VESA:                           100x100
Energieeffizienzklasse:         G
Leistungsaufnahme (SDR):        33 W
Leistungsaufnahme (Sleep):      33 W
Stromversorgung:                DC-In (externes Netzteil)
Abmessungen:                    71.46x31.11x60.26 cm
Besonderheiten:                 Spielmodus, ConnectShare, ...
Herstellergarantie:             24 Monate
```

Figure 4.12: Product specifications from one computer screen as formatted text

CHAPTER 5

Machine learning for enhanced data extraction

Product specifications provided from merchant landing pages differ in their structure and their provided properties from shop to shop. Well-structured properties, where all relevant values are contained entirely in one table row or list entry, form a good use case for extraction with regular expressions. But on the other hand, peculiar formats spread data from related properties like HDMI ports across multiple attributes, such as shown earlier in Figure 1.3 and Figure 1.4. Also, the details about USB-C spread across several attributes, with one even requiring knowledge of the previous attribute.

It is hard to link and combine such features for extraction with regular expressions. Thus, I introduce sequence labeling, a technique from Natural Language Processing, to improve the data extraction capabilities of the pipeline.

Figure 5.1 shows such sequence labels applied to an example sentence. The model used for this has been trained on labeling persons and locations.

Figure 5.1: Sequence labeling shown on an example sentence

Sequence labeling is used for several Natural Language Processing tasks. I use a BERT transformer model for Named Entity Recognition, which I fine-tune on HDMI and DisplayPort features. With that approach, text tokens related to port specifications should be recognized and properly tagged with labels. Based on such labels, the text is processed and used for an improved data extraction for difficult cases.

5.1 Selecting a machine learning model

Pre-trained machine learning models are state of the art for Natural Language Processing tasks. They are trained on huge datasets, which is not reproducible with your own model at home. Recent inventions are solely available as online service, which are not a good fit for this research, the results are not independently repeatable, and the services are apt to change any time.

Thus, the selection of a machine learning model was limited by the following conditions:

- Supports fine-tuning on local computer
- Inference runs on own machine, no connection to online server
- Ordinary consumer hardware fast enough for fine-tuning and inference
- Free and open source model, which is independent of any external servers, once it's downloaded

As BERT models [DCLT19] have been state-of-the-art in pre-trained models for Natural Language Processing over the last years and checked all points, I focused on them. Inventions, such as GPT 4 from OpenAI [Ope] or Gemini from Google [SP] will be more capable, but rely on remote servers. Furthermore, the results are apt to change over time and are not reliably repeatable as local models are. Thus, I did not take them into account and went on with **BERT**, an open source transformer model, which Google initially published in 2018.

I searched the Hugging Face library in the category of token classifications models and selected a well established and common model for Named Entity Recognition, called *dslim/bert-base-NER* [Fac]. It distinguishes between uppercase and lowercase letters and was fine-tuned on the English version of the CoNLL-2003 dataset for Named Entity Recognition. It achieves scores higher than 91% for precision and recall on their test dataset. Related NER-specific models were trained for niche categories or had no scores to back up how well they work. The model is trained to distinguish only between four classes, including location and person names.

5.2 Fine-tune a BERT model

I used my gold labeled dataset on HDMI and DisplayPort specifications (compare subsection 3.2.4) to fine-tune the *dslim/bert-base-NER* model on eight labels, including *COUNT-HDMI*, *VERSION-HDMI* and *TYPE-DISPLAYPORT*. The dataset is fully compatible with the machine learning library PyTorch, which is used in this thesis. A custom data loader adds compatibility between the labeled data exported from Label Studio and the PyTorch training library.

That data in conjunction with the Trainer API from PyTorch is the foundation to fine-tune the base NER model. It relies on a token size of 512 and cuts off text that is too long. This limitation on token size has one downside, as text below the fold is not taken into account for training or inference. To speed up the training and achieve faster convergence of the model, I also used the AdamW optimizer. The parameters for the optimizer are a *learning rate* of *2e-5* together with the default *weight decay* of *0.01*. The values were selected based on the default suggestions and some experimental evaluations based on the dataset.

5.3 Hardware requirements

Decent hardware with native support for machine learning is sufficient to run any calculation in this thesis. It has been conducted on an Apple M1 Chip in combination with 16 GB RAM. No discrete GPU was necessary for fine-tuning of the BERT model or inference on the ComputerScreen2023 dataset.

5.4 Inference

The fined-tuned model is fed with the raw specifications, gained from the HTML extraction stage, and outputs statistically calculated labels. The model receives its input as plain text, where the given attribute-value pairs are joined with a semicolon. This is the same separator that was used for model fine-tuning on computer screen data.

Applying the model on some text, is called inference. Given the text shown in Figure 5.2 as input, it results in labeled data shown in Figure 5.3. The results are shown in a CoNLL-like format, with the text token on the left and the label on the right of each line.

```
Paneltyp:VA;Höhenverstellbarkeit:nein;Energieeffizienzklasse:E;
VGA:1x;DVI:0x;HDMI:3x;VESA-Bohrung:100x100;
Swivelfunktion:nein;Pivotfunktion:nein;Lautsprecher:nein;
Reaktionszeit:4 Millisekunden;Helligkeit:250 cd/m2;
```

Figure 5.2: Product specifications formatted for inference

In this example, the name of the HDMI port and its number of occurrences is correctly labeled as expected. All other tokens end up with 'O', which denotes tokens, that do not belong to a named entity.

This leads to the relevant classifications *TYPE-HDMI: HDMI* and *COUNT-HDMI: 3x*. The values are parsed and structured, so they are fully compatible with the base pipeline using regular expressions. The evaluation scores of the pipeline with this machine learning extended extraction method is shown in section 6.3.

```
VGA O
: O
1x O
; O
DVI O
: O
0x O
; O
HDMI TYPE-HDMI
: O
3x COUNT-HDMI
; O
VESA-Bohrung O
: O
100x100 O
; O
```

Figure 5.3: Inference results visualized as token-label pair

CHAPTER 6

Experimental evaluation of the information extraction pipeline

All the following metrics are evaluated using data from more than 30,000 merchant landing pages provided by the ComputerScreen2023 dataset as input for my pipeline. The structured and unified data output is then compared with the reference specifications, initially gathered from Geizhals. It is worth to highlight, that the reference data is processed with the base pipeline to get the same data granularity and a compatible data format. The comparison is conducted based on strict equality of each property. The structured and unified data is handled as dictionary with a range of allowed keys, earlier defined as catalog properties. An example of such a dictionary is shown in Figure 4.11.

Normalization of values is applied on top, so one dimensional lengths and value-unit combinations with differing units can match. Be aware that normalization is not used on more complex data yet. Thus, the dimensions given as combination of width, length and height are not normalized, which likely decreases the evaluation scores.

In addition to that, the reference data is not perfect, as further investigated in the case study in section 6.5. All that taken into account, the scores build a baseline for further research and are a solid foundation for comparison of all three pipeline variants.

6.1 Definition of evaluation scores

The *reference data* refers to the Geizhals reference specification from the ComputerScreen2023 dataset. The *catalog data* refers to the output of the proposed data extraction pipeline, given the JSON shop offers from the same dataset as input. For the determination of evaluation score, we rely on the following definitions:

- *True positives*: The number of properties that exist in both (reference and catalog data) and the values match.
- *False positives*: The number of properties, where the attribute only exists in the catalog data or the values do not match.
- *False negatives*: The number of properties, where the attribute only exists in the reference data or the values do not match.

Attribute precision

The *attribute precision* (Equation 6.1) is defined as the ratio of *correctly extracted properties*, over the sum of *all extracted properties*. It represents how many of the gathered entries are correct, according to the reference data from Geizhals.

$$attribue\ precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{6.1}$$

The precision score emphasizes false positive values.

Attribute recall

The *recall score* (Equation 6.2) describes the ratio of *correctly gathered properties* over *all reference properties*, as supplied by Geizhals.

$$attribute\ recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{6.2}$$

The recall score accentuates false negatives.

Attribute F1

The *F1 score* (Equation 6.3) is a combination of precision and recall. In contrast to accuracy, this value is not biased for unbalanced data and the single value score of choice.

$$F1 = 2 * \frac{attribute\ precision * attribute\ recall}{attribute\ precision + attribute\ recall} \tag{6.3}$$

6.2 Base pipeline using regular expressions

The evaluation of the default pipeline, relies on automatic field mappings. It solely uses regular expressions to extract the data and achieves an attribute precision of 59.67 % and an attribute recall of 47.54 %, as calculated from the values in Table 6.1 and summarized in Table 6.2. Based on these numbers it can be concluded, that the extraction works, but there's room for improvement. Approximately a third of the extracted properties

do not match with the reference data. However, this could be caused by faulty data provided from online shops. In addition, the reference data can be incorrect or marginally different. The latter is often the case with free text entries, screen dimensions, widths or power consumption values, which happen to be rounded or slightly different. Thus, evaluation scores based on a selection of 15 attributes are shown in subsection 6.2.2. These scores better reflect the real world performance, as they focus on attributes with well comparable reference data.

True positives	33,837
False positives	22,874
False negatives	37,338

Table 6.1: Confusion matrix using base pipeline

Attribute precision	59.67 %
Attribute recall	47.54 %
Attribute F1	52.92 %

Table 6.2: Evaluation scores from base pipeline

Additional statistics of the ComputerScreen2023 dataset and the property retrieval pipeline are shown in Table 6.3. The base pipeline implementation achieves, an average extraction performance of 14.5 correct properties over more than 2,000 computer screens, while solely relying on more than 30,000 landing pages of 30 online shops as input.

Shop offers in dataset	32,227
Sum of extracted computer screens	2,330
Sum of all correct properties	33,837
Avg. correct properties/product	14.52

Table 6.3: Statistics from computations related to the ComputerScreen2023 dataset

6.2.1 Evaluation scores on the attribute level

Evaluation scores for each extractable property are given in Table 6.4. The recall and precision scores are calculated based on *true positives* (TP), *false positives* (FP) and *false negatives* (FN). These raw values provide valuable information about outliers and to suggest improvements and show shortcomings of the implementation or reference data. Attributes, which solely list *false positives*, indicate that they could only be extracted from the shop's product data. On the other side, attributes which solely contain *false negatives*, point out that they could only be extracted from the reference data. There are several attributes with no *true positives*, which hint at a problematic evaluation or a lack of normalization. These attributes are a good starting point for future work.

Attribute name	Prec. [%]	Rec. [%]	F1 [%]	TP	FP	FN
Abmessungen	0.12	0.09	0.10	2	1732	2308
Anschlüsse DVI	92.45	65.92	76.96	147	12	76
Anschlüsse DisplayPort	50.00	0.23	0.46	4	4	1709
Anschlüsse HDMI	91.30	3.03	5.87	63	6	2016
Anschlüsse Klinke	100.00	48.24	65.08	905	0	971
Anschlüsse LAN	100.00	73.37	84.64	135	0	49
Anschlüsse Mini DP	0.00	0.00	0.00	0	0	25
Anschlüsse USB-A	74.47	32.85	45.59	385	132	787
Anschlüsse USB-C	22.12	55.73	31.67	73	257	58
Anschlüsse VGA	97.27	67.71	79.84	499	14	238
Auflösung	91.82	75.67	82.96	1763	157	567
Ausgänge Display	0.00	0.00	0.00	0	0	124
Beschichtung	40.98	31.40	35.56	720	1037	1573
Besonderheiten	20.93	18.16	19.45	422	1594	1902
Bilddiagonale (Zoll)	96.00	62.79	75.92	1463	61	867
Bilddiagonale (cm)	32.05	25.36	28.32	591	1253	1739
Bildwiederholfrequenz	88.18	75.58	81.40	1761	236	569
Blickwinkel horizontal	84.36	73.13	78.34	1704	316	626
Blickwinkel vertikal	84.36	73.13	78.34	1704	316	626
Energieeffizienzklasse	83.85	78.29	80.97	1583	305	439
Farbe	0.10	0.09	0.09	2	1991	2328
Farbraum Adobe RGB	70.61	46.26	55.90	161	67	187
Farbraum DCI-P3	82.56	51.03	63.08	322	68	309
Farbraum NTSC	0.00	0.00	0.00	0	585	0
Farbraum REC 2020	0.00	0.00	0.00	0	0	33
Farbraum REC 709	0.00	0.00	0.00	0	45	152
Farbraum sRGB	82.02	60.01	69.31	830	182	553
Farbtiefe	95.33	58.51	72.51	1327	65	941
Form	84.57	40.47	54.75	943	172	1387
Gewicht	84.60	71.19	77.32	1643	299	665
HDR	82.29	56.43	66.95	395	85	305
Helligkeit	93.30	82.67	87.67	1923	138	403
Herstellergarantie	0.00	0.00	0.00	0	1354	0
Höhenverstellbar	22.71	28.94	25.45	432	1470	1061
Kontrast	14.60	12.59	13.52	292	1708	2028
Krümmung	99.01	52.62	68.72	201	2	181
Leistungsaufnahme (SDR)	52.73	42.87	47.29	974	873	1298
Leistungsaufnahme (Sleep)	30.48	23.50	26.54	534	1218	1738
Neigungswinkelbereich	0.00	0.00	0.00	0	1742	0

Table 6.4: Evaluation results per attribute from the base pipeline

Attribute name	Prec. [%]	Rec. [%]	F1 [%]	TP	FP	FN
Panel	86.07	78.14	81.91	1816	294	508
Reaktionszeit	85.11	75.81	80.19	1755	307	560
Schwenkwinkelbereich	0.00	0.00	0.00	0	763	0
Seitenverhältnis	98.92	86.14	92.09	2007	22	323
Stromversorgung	83.12	47.21	60.21	1098	223	1228
Thunderbolt	68.29	60.87	64.37	28	13	18
USB-Hub Ausgang	100.00	55.29	71.21	658	0	532
USB-Hub Eingänge USB-B	100.00	53.74	69.91	460	0	396
USB-Hub Eingänge USB-C	0.00	0.00	0.00	0	2	502
VESA	91.88	82.26	86.80	1822	161	393
Variable Synchronisierung	15.40	12.45	13.77	290	1593	2040

Table 6.5: Evaluation results per attribute from the base pipeline (cont.)

6.2.2 Evaluation scores of 15 selected attributes

The low scores based on over 50 attributes do not represent the full capabilities of the pipeline. Some attribute values need more work on the evaluation and normalization side in order to produce sensible evaluation scores, that better reflects the output of the pipeline. Therefore, I also provide scores for a selection of the following 15 attributes: *Auflösung*, *Bilddiagonale (Zoll)*, *Bildwiederholfrequenz*, *Blickwinkel horizontal*, *Energieeffizienzklasse*, *Farbraum sRGB*, *Farbtiefe*, *Gewicht*, *Helligkeit*, *Krümmung*, *Panel*, *Reaktionszeit*, *Seitenverhältnis*, *Stromversorgung* and *VESA*.

Based on the subset of scores given in Table 6.4, it results in new values for the confusion matrix, as shown in Table 6.6. The evaluation scores based on these 15 attributes, the precision drastically increases from 59.67 % to 89.13 % and the recall increases from 47.54 % to 72.01 % as shown in Table 6.7. Thus, the F1 score also improves noticeable from 52.92 % to 79.66 %.

True positives	22,696
False positives	2,768
False negatives	8,823

Table 6.6: Confusion matrix using the base pipeline with 15 selected attributes

Attribute precision	89.13 %
Attribute recall	72.01 %
Attribute F1	79.66 %

Table 6.7: Evaluation scores from the base pipeline with 15 selected attributes

6.3 Pipeline enhanced with machine learning model

The machine learning model and its fine-tuning is described in chapter 5. The evaluation results of the enhanced pipeline show a slight improvement of 0.5% in attribute recall, as calculated from the values in Table 6.8 and summarized in Table 6.9. On the other hand, the precision fell by more than 1 %. According to the data, the DisplayPort extraction scores did not change at all. This leads to the conclusion, that DisplayPort version numbers are seldomly provided and the four cases, where it was, were already covered by the base pipeline.

Based on the increased number of true positives, which went from 63 to 410 correctly extracted HDMI properties, it can be concluded, that the machine learning model is a good addition to gather data, that is hard to extract with regular expression alone. The flexibility of the machine learning enabled pipeline allows extraction of single port details, e.g. just the HDMI port count. Whereas the reference data and evaluation solely relies on the combination of port count, port name and port version. As there is no disposal algorithm for partly retrieved port data, the scores for the machine learning approach need to be taken with a grain of salt. The evaluation is strict, so even port data that is correct, counts as false positive and also as false negative in the statistics.

True positives	34,228
False positives	24,542
False negatives	36,947

Table 6.8: Confusion matrix using the base pipeline + machine learning

Attribute precision	58.24 %
Attribute recall	48.09 %
Attribute F1	52.68 %

Table 6.9: Evaluation scores from the base pipeline + machine learning

It is worth to note, that the machine learning enhancement only applies to the HDMI and DisplayPort related properties. With a larger scope of the machine learning model and improved training data, a higher positive impact will be possible.

6.4 Pipeline enhanced with manually improved field mappings

In addition to the base pipeline implementation, I also evaluate possible improvements with improved field mappings. They play a vital part for the extraction with regular expression. Invalid mappings directly reduce the number of extractable properties and imply a reduced number of correctly collected properties.

Due to approximately 2000 field mappings, I have manually enhanced the mappings for the most common shop from the ComputerScreen2023 database and compare the results with the base pipeline. The manually improved field mappings, for one out of thirty shops, already lead to a noticeable improvement by one percent in the F1 score. The attribute precision, as well as the attribute recall, have increased by roughly one percent as calculated from the values in Table 6.10 and shown in Table 6.11. That is as expected, because more valid mappings lead to more extractable values and the recall rises. Additionally, fewer incorrect mappings are supposed to lead to fewer falsely extracted property values, and the precision goes up as well.

True positives	34,678
False positives	22,777
False negatives	36,497

Table 6.10: Confusion matrix using the base pipeline + manual mapping of one shop

Attribute precision	60.36 %
Attribute recall	48.72 %
Attribute F1	53.92 %

Table 6.11: Evaluation scores from the base pipeline + manual mapping of one shop

A comparison of the attribute evaluation scores based on the pipeline variants is shown in Table 6.12.

Pipeline	**Attr. prec.**	**Attr. rec.**	**Attr. F1**	**TP**	**FP**	**FN**
Base	59.67 %	47.54 %	52.92 %	33,837	22,874	37,338
Base (15 attr.)	89.13 %	72.01 %	79.66 %	22,696	2,768	8,823
Base + ML	58.24 %	48.09 %	52.68 %	34,228	24,542	36,947
Base + Manual	60.36 %	48.72 %	53.92 %	34,678	22,777	36,497

Table 6.12: Comparison of evaluation results

6.5 Manual evaluation

The automatic performance evaluation with Geizhals as reference is only suitable as an approximation, as the reference specifications are likely based on additional data, which the dataset does not provide as input for the pipeline. There are data maintainers who enhance the data manually, according to their job descriptions [Gei]. Thus, my assumption is, that the Geizhals reference contains data from other sources like the website of the manufacturer and product datasheets.

The ComputerScreen2023 dataset consists of data from 1st October 2023. Only shops listed on that day - so shops which had the product ready for sale - are available in

that dataset. This means, that even a program with an ideal information extraction performance will not able to achieve a perfect score in the automatic evaluation.

To prove the assumption for the given dataset and get a clearer picture of the data mining performance and expected deviation from a perfect score, I pick a computer screen and investigate it. I automatically collect the properties of the computer screen and the reference data. In order to prove the claim of additional data being necessary, all the properties are also manually collected based on the input data for the pipeline.

I picked an example screen, for which the proposed pipeline supports data extraction for all shop offers of this product.

6.5.1 Case study of one computer screen

I have manually evaluated the raw data for the *Fujitsu E-Line E22-8 TS Pro (2021), 21.5"* computer screen, as available in my dataset. The underlying data can be found in my dataset under the ID 1115 and contains product data from three shops. Thus, there is data from three landing pages available to gather the specifications for this product.

I claim, that Geizhals uses more than data from these three websites to create their specification. The investigation shows, that even in theory it is not possible to extract and get all the specifications, that the reference contains, as properties are missing in the source data.

The automatic evaluation between the extracted data and the reference values shows low scores (see Table 6.13) compared to the average scores achieved with a base pipeline run using the ComputerScreen2023 dataset.

Attribute precision	50.00 %
Attribute recall	33.33 %
Attribute F1	40.00 %

Table 6.13: Automatic pipeline evaluation scores of a computer screen

The comparison of values in Table 6.14 and Table 6.15 shows, that the *reference* data from Geizhals contains some attribute values, which are not contained in the source data. The 'manual' column shows the hand-picked, apparently best values as selected by the author. Lastly, the *pipeline* column shows the property values as mined by the base pipeline. The pipeline, as well as the hand-picked reference values, are solely based on the data from the product landing pages of three online shops.

Data not taken into account

It can be seen, that the automatic field mapping algorithm did not find a mapping or the regular expression could not extract the relevant value for *Herstellergarantie* and *Bildwiederholfrequenz*. However, the manual evaluation brought up the properties, so they are in the dataset, but the pipeline could not collect them.

Property	Reference	Manual	Base pipeline
Diagonale	21.5"/54.6cm	21.5"/54.6cm	-
Auflösung	1920x1080, 16:9, 102ppi	1920x1080, 16:9	1920x1080, 16:9
Helligkeit	250cd/m^2	250cd/m^2	250cd/m^2
Kontrast	1.000:1	1.000:1	1000:1
Reaktionszeit	5ms	5ms	5ms
Blickwinkel	178°/178°	178°/178°	178°/178°
Panel	IPS	IPS	IPS
Form	gerade	-	-
Beschichtung	matt (non-glare)	Blendfrei, 3H Hard Coating	HDCP, ..., Low Blue Light Mode, sRGB Mode
Farbtiefe	8bit (16.7 Mio. Farben)	16,7 Millionen Farben	-
Farbraum	-	-	-
Bildwiederholfrequenz	60Hz	60Hz	76Hz
Anschlüsse	-	1x DisplayPort, 1x HDMI, 1x VGA, 1x DVI-D	-
Audio	Lautsprecher (2x 1.5W), 1x Line-In, 1x Line-Out	Lautsprecher	-
Ergonomie (neigbar)	+22°/-5°	-5°/+22°	-5°/+22°
Farbe	schwarz (Displayrahmen), schwarz ...	schwarz	schwarz
VESA	100x100	100x100	100x100
Leistungsaufnahme	20W (maximal), 14.6W (typisch), 0.12W (Standby)	26W (max.), 11.6W (Ein), 0.14W (Standby)	0.11W (Sleep)
Energieeffizienzklasse SDR (A bis G)	D	D	D
Energieverbrauch SDR	12kWh/1000h	11.6kWh/1000h	-

Table 6.14: Comparison of extracted computer screen specifications

Property	Reference	Manual	Base pipeline
Stromversorgung	AC-In (internes Netzteil)	-	-
Abmessungen (BxHxT)	500.5x343.45x212 mm	50.05x21.2x34.35 cm	50.05x21.2x34.35 cm
Gewicht	2.98kg	2.98kg	2.98 kg
Besonderheiten	EPEAT Silver, mechanische Tasten, Sicherheitsschloss (...)	EPEAT Bronze, ...	HDCP, ..., Low Blue Light Mode, sRGB Mode
Herstellergarantie	drei Jahre	drei Jahre	-

Table 6.15: Comparison of extracted computer screen specifications (cont.)

Data that is not retrievable

Then there are entries, which are not possible to extract right. These are the fields with a reference value, but an empty or only partly correct manual value, such as *Form* and *Stromversorgung*. The data is there at Geizhals, but the shop's landing pages did not contain them. These are the entries, that make it impossible to achieve a score of 100 % with the given input data.

Ambigious data

Lastly, there is similar, but not perfectly matching data, such as the ports with an additional 'DVI-D' port, listed on one of the show websites. At this point it is unknown, which party really has the correct information. Then there are the 'Abmessungen (BxHxT)' where just one differs by 0.05 mm, probably due to rounding to one decimal for millimeter values. And then there is the 'Energieverbrauch SDR' which also differs only in the last digit.

Due to humans not acting as strict and determined as computers do, I could not determine sensible real world performance scores. Many questionable cases arise when manually evaluating this computer screen.

Real world scores would mainly depend on the assumptions you make:

- When do lengths match?
- When do values match? Are they only correct when they are equal to the last digit?
- Is rounding okay?

In contrast to that, all shown scores are determined with a simple condition. If the pipeline does not get it, the property has not been retrieved and if a gathered value does not match, it is incorrect.

Due to these issues and humans being error-prone in such tasks, I do not present scores for the real world performance. But it would definitely result in better evaluation scores. The perfectly matching properties are already taken into account, so manual evaluation will only discover false positives and some false negatives could be marked as not retrievable from the source data. The amount of actually extractable properties shrinks and thus, the recall score increases. Additionally, it could be discovered, that the pipeline collects more perfectly fine values, but the properties are missing in the reference data.

CHAPTER 7

Conclusions

Summing up, the information retrieval and alignment pipeline works and extracts valuable specifications. Based on a selection of attributes with well comparable reference data, the pipeline shows its true potential with an attribute precision of 89.13 % and a recall of 72.01 % for these attributes. This leads to an F1 score of 79.66 %. The pipeline collects an average of 14.5 product properties based on the evaluation of more than 2,000 computer screens with data solely from merchant landing pages. There are many data processing steps, which can be improved in order to achieve better extraction scores for even more attributes. It can be concluded, that it is possible to create good unified product specifications from data supplied solely by online shops. The output is heavily dependent on the quality of the supplied raw data, and fewer on the number of shops, that provide it. Most online shops list the main properties, such as screen diagonal, resolution and panel type, but only few provide more detailed specifications for every aspect. Thus, few shops, that supply good data, are better than many shops providing the same basic properties over and over again.

According to the case study, Geizhals has more data to create their product specifications, than the web shops provided on October 1, 2023. That is no surprise, as the listed offers change regularly, based on the availability of the product in the related shop. However, Geizhals relies on data maintainers and user feedback. Thus, it can be assumed, that some properties are added from product data sheets and the manufacturer's official product pages as well. At the market launch of new products, it can be observed, that Geizhals provides very detailed product specifications from the beginning. This leads to the conclusion that they have access to more non-public channels, from which they get a large amount of good quality data. Based on these source, they likely unify and align the specifications to present uniform data across all products from many manufacturers.

All in all, the regular expressions are quite powerful and do a good job with pulling data from a text string. The need for field mappings is a disadvantage of the proposed implementation, as it requires real world data and at least a one time setup for each

new shop that the pipeline is used with. On the other hand, the implementation is certainly adoptable for other categories with dense and distinct product properties, such as smartphones or tablets. Another strength is, that expected properties and the resulting data format is configurable.

The fine-tuning of the BERT model resulted in a measurable increase of extracted data. However, to achieve better and more consistent results some preprocessing or a different machine learning approach is required. The shortcoming of BERT is the maximum token size of 512 tokens per run, which leads to cut off of long computer screen specification for training and inference. Taking care of these issues, will noticeably improve the machine learning model. Support for retrieval of more properties and details is another factor to achieve better results.

An overall learning is, that automatically aligning properties from various providers and unifying data is a hard task. While implementation of the basic things works fine, the issues start once it comes to the details of attribute extraction. Hidden characters appear and require clean up, or numbers are only given as text like 'one' and needs a conversion to Arabic numerals for further processing. Additionally, data can differ from shop to shop by some nit, and assumptions are necessary to deal with them and not get stuck by too much attention to one detail.

The supported input for the base pipeline is semi-structured data, in order to have attribute-value pairs to process. The approach is not suitable for plain text paragraphs with properties wrapped in full sentences. An extraction from plain text data is only applicable for the machine learning part, when it is trained on such input as well. The machine learning approach needs another thought in order to overcome the mentioned shortcomings and increase its helpfulness.

7.1 Future work

The proposed pipeline has not yet reached its full potential and there are many steps, that can be refined. Tackling any of the following suggestions likely leads to a noticeable gain in extraction performance:

- Support multiple regular expressions per property
- Enhance automatic mapping approach and validate found mappings
- Evaluate multiple value fusion algorithms
- Extend the normalization to support all values
- Rethink the machine learning approach, generalize and adapt it for the use with more attributes
- Evaluate the pipeline on a more fine-grained data level, rendering '1x HDMI' a correct subset of the reference data '1x HDMI 1.4'

Lastly, the thesis did not deal with a clustering strategy for equal products or a general purpose approach for data extraction from HTML, which leaves these open topics for further work as well.

List of Figures

List of Tables

Bibliography

[AG] Preisvergleich Internet Services AG. Mobiltelefone - handys ohne vertrag - preisvergleich geizhals Österreich. https://geizhals.at/?cat=umtsover.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[Fac] Hugging Face. dslim/bert-base-ner · hugging face. https://huggingface.co/dslim/bert-base-NER.

[Gei] Geizhals. Jobs - geizhals. https://unternehmen.geizhals.at/jobs/.

[GK22] Govind Krishnan Gangadhar and Ashish Kulkarni. Extraction of product specifications from the web - going beyond tables and lists. In *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*, CODS-COMAD 2022, page 19–27, New York, NY, USA, 2022. Association for Computing Machinery.

[Hag] Matthias Hagmann. Computer screen specification extraction. https://github.com/mathagmann/computerscreen-spec-extraction.

[KK22] Thomas Kehl and Pip Klöckner. *Folge 344 - Missbrauchen Tech-Giganten wie Google ihre Marktmacht? Pip Klöckner Im Interview (Teil 2)*. Finanzfluss, 2022.

[mS] médialab SciencesPo. medialab/minet: A webmining cli tool & library for python. contribute to medialab/minet development by creating an account on github. https://github.com/medialab/minet.

[NFP+11] Hoa Nguyen, Ariel Fuxman, Stelios Paparizos, Juliana Freire, and Rakesh Agrawal. Synthesizing products for online catalogs. *Proc. VLDB Endow.*, 4(7):409–418, apr 2011.

[Ope] OpenAI. Gpt-4. https://openai.com/gpt-4.

[PB17] Petar Petrovski and Christian Bizer. Extracting attribute-value pairs from product specifications on the web. In *Proceedings of the International Conference on Web Intelligence*, WI '17, page 558–565, New York, NY, USA, 2017. Association for Computing Machinery.

[PB22] Ralph Peeters and Christian Bizer. Cross-language learning for product matching. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 236–238, New York, NY, USA, 2022. Association for Computing Machinery.

[PPMB15] Petar Petrovski, Anna Primpeli, Robert Meusel, and Christian Bizer. Web data commons - gold standard for product matching. *Uni Mannheim*, 2015.

[QBD+15] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, Yanyan Shen, and Divesh Srivastava. Dexter: Large-scale discovery and extraction of product specifications on the web. *Proc. VLDB Endow.*, 8(13):2194–2205, sep 2015.

[RPMP18] Petar Ristoski, Petar Petrovski, Peter Mika, and Heiko Paulheim. A machine learning approach for product matching and categorization. *Semantic web*, 9(5):707–728, 2018.

[SP] Demis Hassabis Sundar Pichai. Gemini – unser größtes und leistungsfähigstes ki-modell. `https://blog.google/intl/de-de/unternehmen/technologie/gemini/`.