Fashion Product Entity Matching

Oliver Jundt
(s0213500)
3 May 2017

Master Thesis in fulfilment of the degree Master in Computer Science
Specialization: Information Systems Engineering (ISE)

Faculty/Department:
Electrical Engineering, Mathematics and Computer Science (EEMCS)
Chair Databases
University of Twente, The Netherlands

Graduation Committee:
Dr.ir. Maurice van Keulen (UT/Databases)
Dr.ir. Ferdi van der Heijden (UT/Robotics and Mechatronics)
Seppe Meinders (Fashion Evolution B.V.)

fashion. evolution

UNIVERSITY OF TWENTE.

ABSTRACT

Finding the same product at different webshops (entity matching) plays an important role for many product search engines like Google Shopping. Knowing which products are identical is essential for deduplicating search results and providing attractive price comparison features. In many product domains, the matching process is trivial as globally unique identifier (e.g. ISBN or EAN) can be used. However, for fashion products like clothing, shoes and accessories, globally unique identifiers are often missing or unreliable, making product entity matching much more challenging.

This thesis presents an entity matching approach for fashion products that is independent of globally unique identifiers. The basic idea is to utilize the combination of description, color, shape and texture features instead to compare and classify product pairs between webshops. However, for the approach to be viable in practice it has to be fast and scalable, robust against varying data quality and achieve near perfect accuracy. This research addresses these challenges based on a real-world example dataset of 1.5 million products from 250+ webshops active in the Netherlands.

In the first part, methods for normalizing and extracting product features are presented including a novel domain specific image segmentation approach in order to cope with varying data quality. For comparison of these product features, several measures are collected that are able to quantify the similarity between them.

In the second part, a typical three-step drilldown approach for entity matching is designed to fulfill the scalability and accuracy requirements. It consists of a fast preselection model as the first step to quickly cluster products using simple brand, category, target group and shop filters. This reduces the search space of possible product pairs from 1.15 trillion to 73 million. The second reduction step consists of a machine learning optimized classification model that uses the described similarity features to more thoroughly reduce the number of possible matches. The third and last step is the integration of human feedback to achieve near perfect accuracy.

8.000 differently configured similarity features and 18.000 labeled samples of product pairs are used to find the best performing classifier and feature subset for the refining reduction step. The experiments show that the best model can filter out 60% of incorrect matches while retaining 95% recall, effectively reducing the search space to 30 million product pairs. Unfortunately, this remaining number turns out to be far from feasible for integrating human feedback to solve all uncertainty in processing the whole dataset. However, the approach is shown to be feasible for smaller datasets and processing daily product updates. Its architecture also allows to easily add more reduction layers in future work to make it more scalable.

PREFACE

This thesis describes my research executed under the authority of Fashion Evolution and supervised by the University of Twente. It marks the end of an exciting journey with many up and downs that come with graduating at a startup company.

I came to the Netherlands in 2008 to study Computer Science at the University of Twente and it turned out to be a good choice. Through an interesting and challenging study, I was able to identify my passion for working on data and automation problems. I got especially intrigued by the area of information extraction, from simple normalization techniques to complex natural language and image processing approaches. Combined with machine learning and clever integration of user feedback a whole new world opened up for me.

When I started working at Fashion Evolution in 2013 as a summer part-time developer I did not expect what would happen in the next years. As someone whose fashion style can be better compared to the practical style of Mark Zuckerberg, the fashion domain was not particularly appealing to me. However, when I found out what technical challenges Fashion Evolution was confronted with I quickly saw the potential to fully live out my passion and apply everything I learned. Choosing a topic for my master's thesis became a no-brainer and within a few months I was busy researching and developing Fashion Evolution's most valuable core technologies, a very rewarding but also time consuming experience. With my focus on making Fashion Evolution a success, this thesis became less and less of a priority up to the point where I had to ask myself if I will ever finish writing it. However, you reading this document today means I eventually managed to do so. Yay!

In the end, I was not the only person who made this project possible. I want to thank Seppe Meinders who founded Fashion Evolution and gave me the opportunity to be part of his innovative visions and promising startup. Even though our dreams did not come true it was still a very exciting time. I learnt a lot about being a professional software and data engineer and an entrepreneur as well. I also want to thank Maurice van Keulen who I first met in 2009 for an interview about uncertain data - an interview that definitely influenced my specialization choice. As a great lecturer, research partner and supervisor Maurice continued to influence me and help me grow academically. Ferdi van der Heijden and Mannes Poel also played important roles in finding my passion thanks to their fascinating lectures and insights on image processing, computer vision and machine learning. Specials thanks goes to my new team at Moneybird (especially Susan Kamies) who motivated me to finish writing my thesis. Last but not least I want to thank my very patient wife Christine who supported me throughout the whole time.

INDEX

1 INTRODUCTION

Finding the same product at different webshops (entity matching) plays an important role for many product search engines like Google Shopping. Knowing which products are identical is essential for deduplicating search results and providing attractive price comparison features. Most product search engines already provide these features for many types of products like electronics, flights and books. In fact, deduplication and price comparison have become so ubiquitous that it is generally assumed to work for any kind of product. However, for fashion products like clothing, shoes and accessories it seems that product search engines have problems with matching and deduplicating identical entities.

Even established specialized fashion product search engines completely lack an entity deduplication feature (Figure 1) or only provide incomplete or faulty matching. As a consequence, an important incentive for users to actually use the search engine is missing. But why is it apparently so difficult to determine identical product entities in fashion retail?

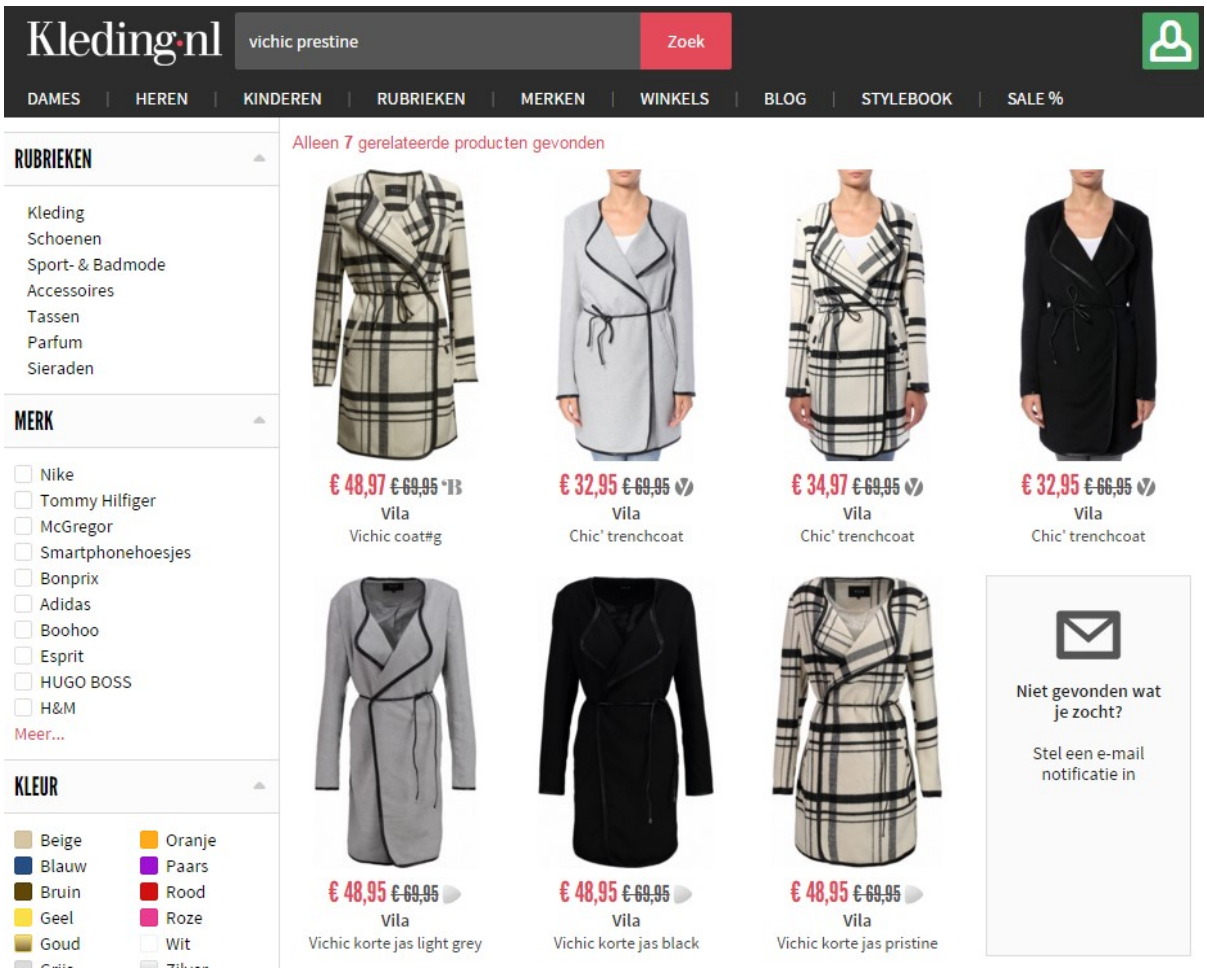

Figure 1 - Duplicate products at kleding.nl

1.1 Problem Statement

Finding identical products from different webshops is commonly based on matching globally unique product identifiers. For example, flights can be identified by their flight numbers, books by their ISBN and most retail products like electronics are assigned a so called Global Trade Item Numbers (GTIN).

Unfortunately, it appears that global identifiers are less commonly used for fashion products. Not all producers decide to assign them to their products and even when an identifier is assigned it does not imply that they can be readily matched. Some webshops deliberately decide against including global identifier information in their product information because they think it is unnecessary or they do not want to be compared with other webshops. Even more problems are introduced with the existence of different definitions of a product entity. In fashion, it is common to design a basic product model which is then sold in different variations of color, texture and size. Some producers and webshops choose the GTIN based on the model, ignoring all or some of the variations, while others treat any variation as a completely different entity with a separate GTIN. In this thesis, the most common definition is used which defines products as being identical if and only if the products are visually identical and at most differ in the offered size.

These problems with missing and unreliable product identifiers are probably the main reason why price comparison for fashion products is not a widespread feature.

Since global identifiers cannot be reliably used for fashion products, an alternative matching approach is required. In practice, we humans can often determine identical fashion products despite the lack of a GTIN. The intuitive approach behind that is simple. Instead of a single numeric identifier we use a combination of distinctive product features to identify a product. The more product features match, the more certain we are that two entities are identical. If the features clearly mismatch, it cannot be the same product. Unfortunately, automatization of this approach is not so simple.

First of all, fashion product data originates from different sources and widely varies in availability and quality (Table 1). Due to these differences matching product data is not as trivial as checking two integer or texts values for equality. Beside data quality issues there are also scalability challenges. The number of possible pairs grows quadratically with the number of products and there are millions of fashion products offered online and thousands of offers appear and disappear every day. Last but not least a very low error rate is also crucial. If consumers cannot trust that products are matched correctly it would render the whole approach useless.

1.2 Research Goal

The goal of this research is to design and evaluate a GTIN independent matching approach that can cope with the stated problems. In particular, the following requirements should be fulfilled:

R1 It matches product pairs despite data quality differences.

R2 It compares millions of products with each other within a feasible amount of time.

R3 It has nearly perfect accuracy.

Table 1 – Example data of two identical products from different Dutch fashion webshops

		
Webshop	Zalando	Otto
Brand	khujo	KHUJO
Name	JUPS - Winterjas	Kort jack Jups
Color	olive	<not available>
Price	€ 59,95	€ 189,00
Category	Dames / Outlet / Kleding / Jassen / Winterjassen	Damesmode > Kleding > Jassen > Korte jassen
Description	Bevat non-textiele bestanddelen lengte: normaal Pasvorm: normaal sluiting: ritssluiting Materiaal vulling: 100% polyester Voering: 100% polyester details capuchon: afneembare capuchon, gevoerde capuchon zakken: zakken met ritssluiting, klepzakken Rugbreedte: 37 cm bij maat S Lichaamslengte model: Ons model is 180 cm groot en draagt maat S patroon: geen (uni) Totale lengte: 64 cm bij maat S Halslijn / kraag: capuchon Mouwlengte: lange mouwen Materiaal buitenlaag: 100% katoen wasvoorschriften: niet geschikt voor de droger, machinewas tot 30°C, programma voor fijne was Artikelnr.: KH121O025-N11	Kort jack 'Jups' van KHUJO Afneembare capuchon Gedetailleerde uitvoering met studs en patches Met ribboorden Artikelnummer: 473814U Van KHUJO. Kort jack Jups Producttype: kort jack Pasvorm: aansluitend Sluiting: rits Onderhoudsadvies: machinewas Materiaalsamenstelling jas: buitenkant van puur katoen

1.3 Research Method and Questions

In order to fulfill the research goal, generally proven approaches will be followed for each of the requirements and applied to fashion product data. In particular, Fashion Evolution's dataset is used as a representative example of real world fashion product data throughout this thesis. The dataset contains about 1.5 million fashion products from 250+ (mainly Dutch) webshops and 16.500 brands, ranging from baby to adult fashion and a variety of categories like shirts, bags, shoes, dresses and hats.

A typical approach to the first requirement of handling quality differences (**R1**) is the usage of data normalization and feature extraction methods to obtain more robust product data. Candidates for these methods are collected from two different sources. Like other product data aggregators, Fashion Evolution already applies basic data normalization methods that have been proven useful in production. These methods are described in Chapter 2 answering the first research question:

Q1 How can product data quality differences be reduced?

As the second source, literature research is used in Chapter 3 to find methods for extracting additional robust and compact product features, answering the second research question:

Q2 What additional robust and compact product features can be extracted?

Also with literature research, methods for quantifying the similarity of product features are gathered in Chapter 4. Special attention is paid to the scalability of these methods, answering the question:

Q3 How can similarities between product features be quickly quantified?

The second requirement for fast comparison of millions of products (**R2**) is usually fulfilled by drilldown techniques that first quickly reduce the set of possible matches and then apply a more thorough check on the remaining candidates to determine a final selection [1]. The basic design for such a drilldown approach for fashion product data is described in Chapter 5 based on the previously presented product and similarity features. The guiding research questions are:

Q4 How can simple product features be used to quickly reduce the set of possible matches?

Q5 How can more computationally expensive similarity features be used to further reduce the set of possible matches?

As it is expected that the automatic classification performance alone will not be sufficient to reach the third requirement of near perfect accuracy (**R3**) this research will also discuss the feasibility of integrating human feedback for final verification. The central question for this is:

Q6 How much human interaction is required for reaching near perfect accuracy?

Performance of the drilldown approach and feasibility of user feedback integration under real world conditions is investigated using Fashion Evolution's dataset in Chapter 6. Labeled product pairs are collected and used to train and test different combinations of product similarity features and classifiers.

Final conclusions with suggested future research questions are given in Chapter 7.

1.4 Related Work

Related work on fashion product entity matching mainly falls into two categories: entity matching (in general or for other data domains) and approximate retrieval systems for fashion products.

Entity matching, also referred to as object matching, record linkage, entity resolution, duplicate identification or reference reconciliation, is a popular research topic with several surveys available [1] [2] [3]. Most entity matching research deals with the exact matching of textual data like documents, publication references and addresses of persons and companies. Product entity matching (with images) gets less attention [4] and is described as being especially challenging due to more variance in data quality [5]. Still, many of the common high level ideas in this research field are applicable to this thesis, e.g. the mentioned drilldown (blocking) approach [6] for quickly reducing the search space and the integration of user feedback to tackle uncertainty [7]. The frequently used textual data normalization methods and similarity measures (e.g. Jaro-Winkler) are also likely to be reusable for fashion product meta.

Approximate retrieval of fashion products is mainly focused on ranked suggestion systems using user-taken photos as input. In contrast to generally text based matching approaches, the focus of approximate retrieval clearly lies on images of products and their perceptual similarity. Instead of the name or description, visual features such as color, shape, pattern and texture are of more interest. However, many researchers choose only a subset of these visual features, leading to a variety of results.

Grana et al. [8] for example describe a purely color based retrieval model for fashion products trained on several user-suggested color classes. Their model outperforms simpler color histogram and signature approaches in terms of perceived color but fails at recognizing shape or pattern differences. In contrast, Tseng et al. [9] especially focus on the usage of shape contexts for finding similar clothing. They highlight the challenges of handling self-occlusion, folding and deformation that come naturally with non-rigid soft objects as clothing. Tsay et al. [10] and Chen et al. [11] focus on an improved clustering/bundling version of Lowe's well-known SIFT features [12] in order to measure similarity. Their methods work well for clothing with distinctive features like prints. Hsu et al. [13] did experimental research with a more complete feature set which includes SIFT features and color histograms but also texture features extracted with Gabor filters. Unfortunately, they missed the opportunity to apply machine learning to optimize their similarity. While the previously described approximate methods are purely based on images, Santos et al. [14] successfully improve their visual similarity ranking by including category information. Liu et al. [15] use local binary patterns for extracting texture features and RGB color histograms. Probably the best performance was recently achieved by Kiapour et al. [16]. They use novel deep image similarity features [17] to match clothing products from user-taken photos with webshop images. Although they claim to provide an exact matching the results are still too far from the high accuracy required to be considered exact. Furthermore, the scalability of the approach has yet to be investigated.

Beside the mentioned public research there also exist disclosed commercial fashion visual search projects like snapfashion.co.uk, like.com or shotnshop.com. Only like.com allows a small peak with their paper by Lin et al. [18] which describes a weighted and machine learning optimized combination of sophisticated local image features for color, shape and texture which in their user interface can also be further tuned in real-time.

In conclusion, approximate fashion product retrieval and (general) exact entity matching have been extensively researched but research on the **combination** of both as proposed in this thesis seems to be largely unexplored. Especially the focus on pragmatic requirements and the consideration of text and image data at the same time appears to be unique.

2 NORMALIZATION

In general, product search engines do not have direct access to the database of their affiliated webshops. Instead they have to work with the product data as it is published by the webshops. Crawling or scraping the product pages is one possibility to collect product data from the webshops [19] but in practice the most popular publishing method for commercial goals are so called product feeds. Product feeds are simple data files (CSV or XML formatted) containing multiple rows/nodes where each row/node consists of multiple attribute fields and describes a single product. A product feed originates either directly from the webshop or is passed through an affiliate network which acts as mediator between the webshop and the publisher advertising the products.

Unfortunately, the collection of data from the different webshops comes with one major issue that needs to be tackled before the data becomes comparable: the quality varies greatly between webshops. The main reason for this quality problem is the lack of standards for distributing fashion product data. Webshops have chosen their own set of attributes and their own data formats. If affiliate networks are involved the situation is better because the networks usually require the webshops to fill in certain attributes and assign fixed attribute names. However, the rules are usually not very strict. In combination with the often-lacking technical understanding of the people creating the product feeds or configuring the generating tools this freedom of choice leads to a great variation of availability and quality of the data.

The following subsections analyze in more detail the attribute data as it is currently provided in practice using Fashion Evolution's dataset of 1.5 million products from 250+ webshops as a representative example. Each subsection also describes methods to normalize the data. The general main challenge here is to find normalization methods that are robust against quality differences on the one hand, but on the other hand are also sensitive enough to preserve useful information for product comparison.

2.1 Mapping Attribute Fields

As a first step into interpreting and normalizing the product data it is necessary to know which attribute fields contain what kind of data. In practice there exist many variations and synonyms for field names which actually describe the same attribute, e.g. *'product_name'*, *'ProductName'*, *'title'* to identify the product name field or *'BRAND'* and *'vendor'* to identify the brand field. Luckily there appears to be a limited number of variations so using a mapping table is one possible solution to this problem. Adding a layer of simple string normalization (e.g. downcasing) and using regular expressions for matching helps to keep the mapping table small (Table 2).

Table 2 - Typical attribute field mapping

Attribute	Aliases
Name	name, title
Brand	brand, vendor, manufacturer
Target group	target, gender, age
Category	category, categories, type
Price	price, current_price, price_now
Description	description, text
Colors	color, colour
Materials	material, fabric
GTIN	gtin, ean, upc, european article number
Images	image, img, image_url, largeimage, image_medium

2.2 Name

In general, all webshops assign names to their products. Those names can be seen as very short product descriptions, usually consisting of less than 50 characters. Their information content varies strongly. On the one end, they can be very general, describing only the type of clothing, e.g. *'winter jacket'*. On the other end, they can be very specific, containing the clothing type, model name, color and brand, e.g. *'Sneakers Chuck Taylor All Star Fashion Washed Ox M by Converse'*.

Sometimes the names even contain identifiers or distinctive substrings like *'SW13M051'* that appear to be globally unique identifiers however this is rarely the case. Those identifiers are also very hard to distinguish from internal identifiers used by webshops that are not globally unique.

In order to normalize product names it makes sense to start with the removal of redundant data that is already part of other product attributes such as the brand name or color names. For further normalization it is usually safe to replace any word spacing with simple spaces. Additional removal of non-word characters and applying the same letter casing are even more aggressive but safe options.

Table 3 - Example name normalization results

Raw name	Normalized name
white Adidas sneakers ZX flux	sneakers zx flux
Nike - AIR MAX 1 - black	air max 1

2.3 Brand

Product brand information is also commonly available. It is probably the attribute with the lowest noise level. The only striking problems with brand names are different name variations due to different usage of letter casing, spacing, hyphenation or accentuation, e.g. *'S. Oliver'* and *'s.Oliver'*. Only in very rare cases webshops do not assign the correct brand. The most common error is the usage of parent brands instead of more specific daughter brands like *'Esprit'* instead of *'edc by Esprit'* or *'van Haren'* instead of *'Graceland'*. Most of the variations for brand names can be normalized to a single representation using a simple alias table containing all known variations of a brand. Most string variations such as the letter casing, non-word characters and spacing can be safely ignored for brands and help to keep the alias table compact.

Table 4 - Example brand aliases

Brand	Aliases
Tommy Hilfiger	hilfiger, tommy sportwear
G-Star RAW	gstar
Hugo Boss	boss, hugo, boss by hugo boss

2.4 Target Group

It is common to provide target group information with the product data. The target group is usually a combination of gender and age group like *'male adults'*, *'female teens'*, *'kids unisex'*. Sometimes gender and age are not explicitly given but combined in a single term like *'women'* or *'boys'*. In other cases, only abbreviations are used like *'w'* for women and *'m'* for men. Fortunately, the possible variations are quite limited and using case insensitive regular expressions and alias tables makes it relatively simple to normalize the target group data. If only the age is explicitly mentioned, then usually unisex can be assumed as gender. On the other hand, if only the gender is given, then in practice the products are very likely to be designed for adults.

Table 5 - Target group aliases and their associated (*assumed) target gender and age labels

Aliases	Target gender	Target age
adult	Unisex*	Adult
child, children, kid, infants		Child
female	Female	Adult*
woman, women		Adult
girl		Child
male	Male	Adult*
man, men		Adult
boy		Child

2.5 Category

For filtering purposes webshops normally organize their products in a category tree. However, webshops specify their own category trees which again introduces a lot of variety. A typical category descriptor given in product feeds reflects a path in the category tree, e.g. *'Shoes > Sneakers'*. The descriptors can range from general like *'Clothing'* to specific like *'Clothing > Tops > T-Shirts > T-Shirts with print'*. Especially webshops which specialize in a certain type of fashion products tend to have more detailed category descriptors. Sometimes category paths are also ambiguous like *'Jeans & Trousers'*.

In order to normalize the category data, it is common to use mapping approaches that map category paths from one webshop to a central unified category descriptor. One disadvantage of this approach is that mappings have to be created for every category path that is used by webshops. Luckily most webshops tend to have similar category trees. The more webshops are known the higher the chance that category paths of a new webshop have already been seen before. With basic string normalization methods and keyword-based suggestion systems it is possible to reduce the number of different category paths to a manually manageable amount.

Another challenge of this approach is to find a unified category descriptor list or tree that covers all interesting product types but is still compact. It is not really beneficial to include different types of soccer shoes just because a single soccer fashion webshop provides this information when most other webshops are never more specific than *'soccer shoe'*.

Table 6 - Example category path mappings

Raw path	**Mapped to**
Clothing / Sweaters & Jumpers / Knitted jumpers	knitted jumper
WINTERJACKETS	winter jacket
Outlet > Boots > High	high boots
ACC | EARRINGS | GOLD	earring

2.6 Price

The price is one of the most essential product attributes for online shopping and hence can be assumed to be always present for any valid and interesting offer. The price may also be useful for comparing products but it is important to consider that the price may change over time. Sales can let a price drop by a discount of up to 70% or even more. Therefore, one should distinguish between the original retail price and possible sale prices. Some shops provide this information separately while for other shops the sale and retail prices could be derived from monitoring the price history. The latter is also more trustworthy because some webshops just make up high before-prices to trigger sales.

Normalizing prices is usually not a problem. The prices are given in one of a few different number formats in use that can relatively easily be distinguished (e.g. US format or Dutch format). Beside that prices can be given in different currency which might have to be converted to a common currency first.

Table 7 - Example price normalization

Raw price	Normalized price
300 Euro	€ 300.00
€ 1.234	€ 1234.00
23.50 Eur	€ 23.50
2,345 USD	$ 2345.00 ≃ € 2171.60

2.7 Description

Often product data also comes with a description text. Product descriptions are probably also the attribute with the highest variety in data quality. The length varies widely (Figure 2) and like product names the content ranges from general to specific. Still, it can contain interesting information for comparing products. For example, a description might contain more information about the materials used and the size and colors of different parts of the product. They might highlight certain details and features of a product like 'golden logo' or 'detachable belt'. However, description might also contain irrelevant data (advertisements, slogans) or data that is subjective to the creator of the description, e.g. 'these shoes go well with blue jeans'. Spelling and grammar mistakes are another noise as well as the fact that descriptions may be written in different languages.

All these problems make normalization of descriptions very challenging. Like product names it is possible to apply string normalization techniques like normalizing word spaces, letter casing and removing non-word characters. Different languages could be tackled using automatic translation. Nevertheless, extracting and comparing the contained information remains hard.

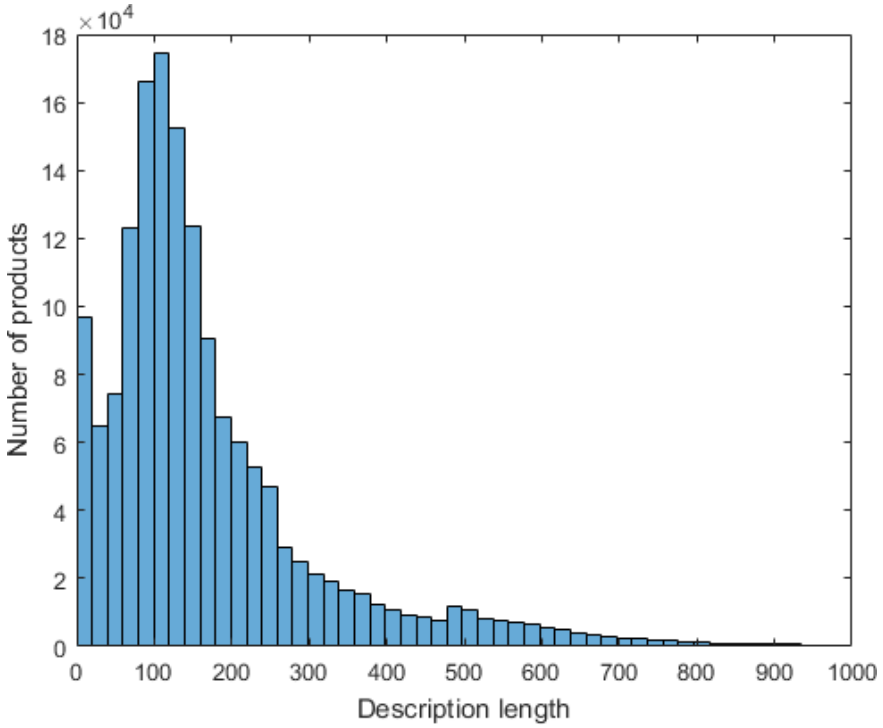

Figure 2 - Distribution of description length

2.8 Colors

The majority of shops provide color descriptions for their products. It is usually a list of one or more common color names like *'green'* which fit a wide spectrum of colors. Only in rare cases more specific or even exotic color descriptions are used such as *'burlywood'*. Similar to brands and target groups, a simple mapping approach can be used to map most of those color names to a manageable and unified color collection. However, beside issues interpreting single color names, it is also unclear how to interpret color information containing multiple labels. The descriptions vary from mentioning only the most prominent color to a list of all occurring colors with their ratio being unclear.

Table 8 - Typical color aliases

Color	Aliases
black	coal, anthracite
gray	grey, stone, melange
white	ivory, cream
red	ruby, bordeaux, cherry, copper
green	emerald, mint, lime
blue	sapphire, indigo, cyan
yellow	banana, lemon, mustard
pink	rosé, magenta
orange	peach
multicolor	bicolor, tricolor, combi

2.9 Materials

About a fourth of the products comes with material information. Similar to colors, the material names may be given in different languages and can range from commonly used (e.g. *'cotton'*, *'canvas'*) to less known variants like *'ecoleather'*. Similar to color names, this problem might be partially solvable by a mapping normalization using patterns or alias tables. Contrary to colors, material data occasionally contain percentages to describe the composition of a product. However, those percentages are often missing or only the major material component is mentioned.

Table 9 - Typical material compositions

Raw materials	Normalized composition
Cotton, 40% Polyester	60% cotton, 40% polyester
Denim	100% denim
Ecoleather	100% leather

2.10 GTIN

As stated in the beginning most products do not have a Global Trade Item Number (GTIN) available and if they do, the information is not necessarily trustworthy. Nevertheless, if an identifier is present it might as well be utilized but normalization is required as well. Depending on the region and product type different GTIN subsets are in use, e.g. EAN-13 and UPC. These subsets were designed to not overlap and can therefore be easily normalized. They even contain a check digit to verify whether a number is actually a correct GTIN.

Table 10 - Example GTIN normalization

Value	GTIN subset	Normalized GTIN
0096385074	EAN-8	96385074
8 717 886 496 200	EAN-13	8717886496200
1234-5678-9999	UPC	123456789999

2.11 MPN

Manufacturer Part Numbers are similar to GTIN but are even less common. In combination with brand information MPN can be considered globally unique identifiers but since every manufacturer has its own numbering scheme there is no specific format one can normalize to. They may be simply numeric like '*476841*' or as complex as *'ADU 1231-12/13'*. MPN are also easily confused with SKU identifiers (Stock Keeping Unit) which only have a meaning within a certain webshop and are therefore not suitable for global matching.

2.12 Images

Each product should also come with a download link to an image depicting the product. Most commonly those images are given in JPEG or PNG format and their resolution may vary widely, from as low as 0.03 megapixels up to 10 megapixels. For most of the product images the background is an even white color with high contrast to the depicted product. In some cases however, the background is a non-white color and can have shadow or gradient effects or the contrast between background and product is so low that there is no visible difference between background and parts of the product. There is also a small percentage of images with even more complex background, including logos or short texts. On the positive side, the products are usually well illuminated.

Figure 3 - Typical product images of shoes/boots and upper clothing

The products themselves are photographed from different perspectives and in different arrangements/poses, though there appears to be agreement on the most attractive perspective and pose within certain categories. Shoes for example are usually photographed from the side with a slight angle while upper and lower clothing is usually depicted showing the front (Figure 3). The biggest variation can be found within accessories and categories with products that consist of multiple parts (Figure 4). In general, it seems to hold that the more flexible the material and shape of a certain kind of product is, the more variation can be found in the perspectives and poses.

Figure 4 - Typical product images of accessories

Furthermore, products are occasionally worn by models or mannequins. Especially in categories like bikinis this means that the actual product often only covers a small portion of the image (Figure 5). Sometimes they are also depicted together with other products.

Figure 5 - Product images with (partial) models/mannequins

Last but not least, there may be multiple images assigned to a single product, usually showing different perspectives or zooming in on certain details of the product.

2.12.1 Segmentation

Normalizing fashion product images is more challenging than normalizing the previously mentioned textual data. For comparing products, the images should ideally consist of an all-around view of the product taken from a fixed viewpoint in a well-lit environment and without any other objects in the picture. Unfortunately, this ideal standard deviates widely from the previously described image quality. There are certain auto-adjusting techniques and efforts to produce 3D models from 2D images but the results are mediocre at best [20], especially since most products come with just a single image, making it hard to estimate depth.

However, some degree of normalization can be achieved by removing image segments that contain irrelevant information that distort the comparison process. This normalization step can also be called image segmentation. During the segmentation process the product image pixels are segmented into product related and unrelated pixels. In particular, the following irrelevant segments are observed:

- Evenly colored background segments with possible gradient effects applied
- Model/mannequin segments like face, hair and skin
- Product segments that are not the motive of the image
- Logos and labels

A basic image segmentation could be achieved by applying a flood fill algorithm starting from the edges of the product image and consider each flooded pixel as irrelevant. This would work for many images due to the uniform background color, sufficient contrast and common central placement of the products (e.g. Figure 3, Figure 4). However, there is still a substantial number of images which do not fulfill these conditions. For example, background segments enclosed by the object and other non-product pixels like skin and hair would be incorrectly segmented (e.g. Figure 5). Therefore, a more sophisticated image segmentation approach is required.

One possible approach for a more sophisticated image segmentation is to use a top-down approach which detects unrelated parts step by step until only product related segments are left. In contrast, a bottom-up approach starts with detecting related segments first.

In the following sections a novel top-down approach is described that facilitates a popular segmentation technique called graph cuts [21] which in itself can be seen as a combined top-down and bottom-up approach. Based on certain and possible pixel samples for the foreground and background it estimates two pixel classification models and segments the image based on cuts determined by energy minimization. The challenging part for using graph cuts is to choose the pixel samples for the foreground and background. Initially graph cut was designed to be used iteratively by users that manually select samples. However, as shown in the following sections it is possible to fully automate the pixel sampling with a few general pixel selection rules that exploit the previously described image characteristics. Each of these rules were derived and fine-tuned in a trial and error process using manual inspection of the segmentation results on roughly a hundred real world images from different product categories with a variety of unrelated parts.

2.12.2 Cropping & Resizing

As a basic normalization step each image should be cropped and resized. The cropping and resizing has the advantage that it reduces the differences in background padding and resolution between webshops and it also reduces the number of pixels that have to be processed in all subsequent steps hence speeding up the image processing process.

Simple cropping techniques would remove only pixel rows/columns where all pixels are exactly of the same color. Evenly colored segments are usually a safe indicator for background pixels but this indicator would fail for images with e.g. gradient effects or JPEG artifacts. A smarter cropping approach is to use edge maps to determine areas without perceptible changes.

Canny edge detection [22] may be used for this with a threshold chosen that is sensitive enough to detect edges between product and background but is not affected by the mentioned smooth gradient effects or JPEG artifacts (Figure 6). One challenge for this cropping approach are webshops that put borders around their images or put unicolored padding around images with different backgrounds. These changes introduce strong edges that would lead to incorrect cropping results. Fortunately, those edges are straight vertical and horizontal lines that can be easily detected and ignored during cropping.

Figure 6 - Input image with gradient → grayscale image → Canny edge map → cropped image

After cropping the images can be resized. For comparison purposes, it is beneficial to retain the aspect ratio during resizing to prevent further image distortion. The target size can be chosen based on different approaches, e.g. resize-to-fit or resize-to-fill or resize by the number of pixels. Resize-to-fit and resize-to-fill would ensure a maximum or minimum width/height respectively but have the disadvantage that the resulting total pixel count can vastly vary, resulting in a high chance of fluctuating processing times and information content. Therefore, it is preferable to calculate the target width and height for each image individually to ensure a similar number of pixels for all images.

2.12.3 Background

The segmentation of background pixels is the first segmentation step. Similar to the smart cropping approach the basic idea is to use edge maps to determine the foreground boundaries. However, edge pixels themselves are rarely representative for the rest of the foreground and do not cover sufficient area to provide informative samples for the graph cut algorithm. They also do not necessarily fully enclose a region that could be assumed as foreground. These problems can be partially solved by using the convex hull of all edges as possible foreground area and it can be further improved by taking the background color into account.

In almost all cases of fashion product images the background color can be estimated by analyzing a small sample of pixels from the top upper corners. Often this is plain white but this simple preparation step makes the background segmentation more dynamic. With the estimated background color, it is possible to better define the graph cut samples. All edge-free border-touching regions outside the convex hull that are colored like the estimated background color (taking possible gradient effects into account) are a safe guess for the background. On the other hand, all strong edges and regions that have a safe color distance to the background color are used as certain foreground pixels. Everything in-between is assumed to be possible foreground.

Figure 7 – Initial foreground graph cut segments → result after applying graph cut → final mask

Crucial for this background segmentation approach is a proper choice of color distance and edge detection thresholds. Those thresholds should be chosen based on a good balance between robustness against noise and sensitivity for foreground boundaries. The boundaries are clear when there is sufficient contrast between product and background but if for example a white t-shirt is depicted on a white background the boundaries can become fuzzy (Figure 8). A good indicator of contrast is the ratio of edges that are found within areas that are colored similar to the estimated background color. The lower the ratio the higher the contrast and thus more robust edge detection and color matching can be used with higher thresholds. This indicator can be dynamically estimated and used individually for each image.

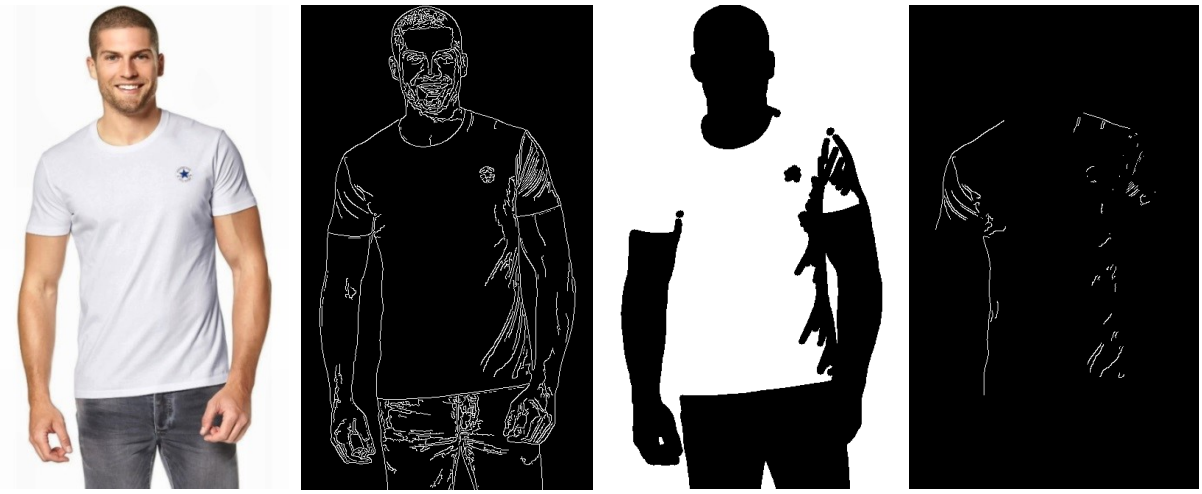

Figure 8 – Low contrast example with significant amount of edges within possible background

Note that it is useful to keep only the biggest/most informative segment as foreground during this step. That way most logos and labels can be filtered out. Of course this fails if the segments are not distinct. Filtering segments based on information content (e.g. using entropy estimates) produces better results than filtering based on pixel mass as logos and text overlays tend to have lower information content and there may be cases where the logos are bigger than the depicted product.

Figure 9 - Image with logo → segmentation after graph cut → final mask

2.12.4 Head and Hair

The second segmentation step aims at head and hair segments that can occur in the upper half of a product image. Detecting hair can be very challenging due to many possible hair colors and forms but detecting heads is less complex. One approach for head detection is to utilize face detection methods, e.g. fast but accurate haar wavelets [23]. When a face can be detected it is natural to assume that the rest of the head and the hair can be found in the area near it.

These simple assumptions allow to collect the required graph cut samples as follows. All pixels of the previously detected background and lower image half are used as certain non-head/hair pixels (foreground). Skin-colored areas where faces have been detected are used as certain head pixels (background). By simply adding pixels from above the face as certain head/hair pixels it is very likely that the remaining hair pixels will also be correctly segmented using graph cut. As further possible head/hairs pixels one can add skin colored pixels that touch the face area, e.g. to also cover neck areas.

Figure 10 - Initial head/hair graph cut segments → result after applying graph cut → final mask

2.12.5 Skin

After faces/heads and hair have been segmented there are still other skin segments possible like arms, hands and legs. There exist many approaches for skin detection [24] and most of them are purely color based, suffering from high false positive rates. To reduce the false positive rate, it is beneficial to require the absence of strong edges in possible skin areas based on the assumption that most skin areas have a smooth texture. It also helps to work with a limited color palette of skin colors (Figure 11). In simple approaches, pixels are classified as skin when their color vector is within a certain distance of an average skin color but due to the vast variety of skin colors a broad color distance is required to cover them all. Using a color palette of multiple average skin colors allows for a more precise color matching, especially when used with a color space like CIELAB where the distance between color vectors more accurately reflects the perceived color distance [25] (Figure 12). The distance thresholds for the luminance value (L*) and color values (a* and b*) can be tuned separately to better control the allowed shades of skin colors.

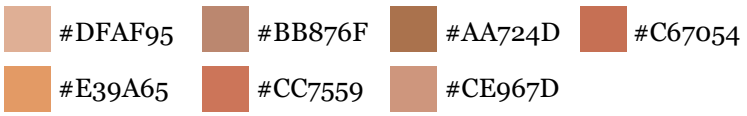

Figure 11 - Possible RGB skin color palette

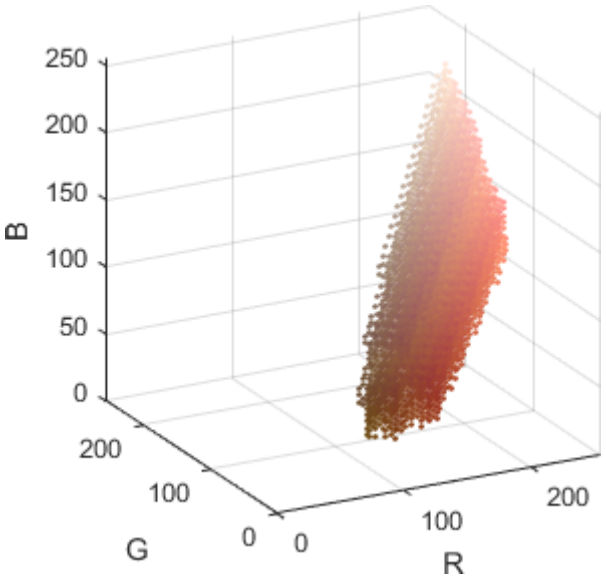

Figure 12 - RGB samples matching the skin color palette with ΔL* < 20 and Δa* < 10 and Δb* < 10

When the color and edge based skin detection approach is subsequently combined with graph cuts it leads to satisfying results. It also helps to preprocess the image with a bilateral smoothing filter to further smoothen skin texture while retaining stronger edges, allowing graph cuts to create smoother segmentations.

Based on the color and edge based skin detection one can define the following sample classifications for a graph cut. Everything that is far from being skin colored is assumed to be certain foreground. All edge-free areas that are colored close to one of the known skin colors are presumed to be certain skin areas. Simple morphological operations like dilation and erosion can be used to further improve the sample blobs for a graph cut by closing gaps and removing noise. As possible skin area one can use the same requirements but with less strict edge and skin color distance thresholds.

Figure 13 - Initial skin graph cut segments → result after applying graph cut → final mask

As a final check, it makes sense to determine what ratio of pixels have been classified as skin because there is still the possibility of confusion with leather and other skin colored smooth areas. Depending on the product category only a certain maximum ratio of skin pixels should be allowed depending on what is realistic. Shoes for example are rarely shown on a model but can contain leather parts. A good strategy would therefore be to allow only a small percentage of skin segments. At the opposite, beach wear is often shown with models and is rarely leather or skin-colored. For this case, it makes sense to allow a high ratio of skin segments. If the detected skin ratio is exceeding the defined threshold, there is high chance of a false positive classification and it becomes safer to skip this step.

2.12.6 Incomplete Products

The last segmentation step of this approach aims at product segments that are not the motive of the image. These segments are hard to distinguish from the desired product segments but it is possible to exploit a domain specific characteristic to detect and handle at least some of the cases. Often decorative products are only depicted incompletely because they are cut off at the image border. This cutoff at the borders can be detected by searching for groups of foreground pixels touching the border. Furthermore, decorative products are often chosen in contrasting colors to the actual motive which leads to edges between both products that should facilitate a good graph cut.

Based on this observation the sample classes for graph cut can be defined as follows. If a group of border-touching foreground pixels is found, then a narrow strip of pixels at the border is used as certain samples for the background. A dilated version of the certain incomplete pixels is then used as possible background pixel samples. These pixels are a hint for graph cut that the product segment might continue further into the image. Unfortunately, it is hard to decide where the incomplete product stops and the desired product begins. Therefore, all remaining pixels are classified as possible foreground pixels. No certain foreground pixels are set. This might lead to complete products being detected as being unrelated pixels therefore it is important to check the ratio of incomplete products afterwards. If more pixels have been classified as background than foreground, then the incomplete product segmentation is probably faulty.

Figure 14 - Initial incomplete product graph cut segments → result after applying graph cut → final mask

2.12.7 Flipping

For images of shoes there exists a simple but effective additional image normalization step. Most shoes are depicted from the side but some webshops use an angle from the left while other use an angle from the right. To simplify the comparison process it is possible to exploit the typical shape and the often-symmetrical design of shoes. If one half of a segmented shoe image contains more foreground pixels than the other half, it is very likely that the half with more pixel mass depicts the back/shaft of a shoe. By flipping the image horizontally such that a certain half of the image always contains the most pixel mass it becomes less complex to compare shoes (Figure 15).

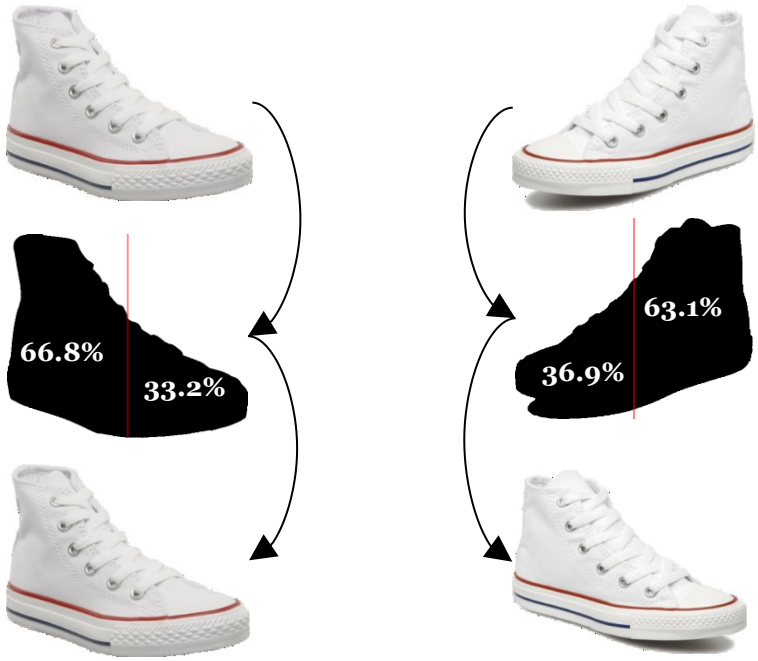

Figure 15 - Pixel mass based horizontal flip for shoes

2.12.8 Runtime Performance and Accuracy

The whole segmentation process is computationally expensive but as the segmentation has only to be done once per image, the impact on the overall comparison performance is small. Still, simple category dependent optimizations can be used to speed up the process. For example, shoes are almost never depicted with other products or mannequins so the steps for removing these segments can be safely skipped without influencing the segmentation accuracy.

To estimate the segmentation accuracy of the proposed algorithm, it has been implemented in a software prototype using library functions from OpenCV 2.4 [26] and applied to 1 million product images. Manual inspection of random samples from the results shows that the quality of the segmentation is satisfactory most of the time. Figure 16 presents three examples of successful segmentation. Figure 17 shows examples of incomplete/erroneous segmentation. These errors tend to happen in cases where the background contains strong gradient/shadow effects or other distinct artifacts. Exact measurements of the segmentation errors have not been done as it goes beyond the scope of this research. Any improvement on the selection of relevant pixels compared to a simple flood-fill approach is beneficial but it is not expected to be of critical impact to the rest of this research.

Figure 16 - Successful segmentation examples

Figure 17 - Incomplete segmentation examples

2.13 Conclusion

This chapter gave an overview of the data quality of fashion product data as it is currently observed at Fashion Evolution. Several pragmatic methods for normalizing data are presented and often basic string normalization and mapping approaches are shown to be sufficient to overcome most of the webshop differences. However, not only the quality but also the availability is important. Meta information such as the color and material is valuable information but was found to be rather noisy. A lot of information about the color, shape and texture of a product is only consistently available in the product images. As preparation for extracting this information, a novel segmentation approach has been proposed that is able to remove unrelated information from product images like background, body and incomplete product segments. This normalized product data is the base for the following chapters.

3 FEATURE EXTRACTION

The previous chapter described the available product data and how it can be normalized to make it more robust and better comparable. However, product descriptions and images still contain more information. Especially the product images are a rich source of color, shape and pattern information that is not readily available. This section aims at investigating how that extra information can be extracted using well-established methods found in the literature.

3.1 Name and Description Tf-Idf

For a computer, the name and description of a product is just a series of words without knowing which parts of the text are important and interesting. This makes a smart comparison of these types of unstructured texts very challenging.

One approach to estimate the key information of texts are bag-of-words models that disregard the structure of a text and only focus on collection a set of words/terms that distinctively represent the text. A popular approach for finding distinctive terms within a text is called tf-idf (term frequency - inverse document frequency). Tf-idf assigns weights to terms based on their frequency in the text and the frequency in the whole corpus. In case of this application the corpus could be the collection of all product names and descriptions. Tf-idf will assign higher weights to terms that occur often within a text and rarely within the whole corpus. That way tf-idf highlights distinctive key terms of a text.

Note that terms can in fact be different substrings of a text. One can for example count single words or n-grams of words or even n-grams of letters. Each type of term has its own advantages and disadvantages, e.g. counting n-grams of letters is less prone to spelling errors than counting whole words but n-grams of words are required for capturing cohesive phrases.

Table 11 - Most frequent n-grams of words in fashion product names and descriptions (Dutch market)

Words	TF	Bigrams	TF	Trigrams	TF
de	2.72%	solid colour	1.13%	bij zalando nl	0.77%
een	2.05%	no appliqués	0.74%	nu zonder verzendkosten	0.39%
van	1.97%	bij zalando	0.57%	zonder verzendkosten bij	0.39%
en	1.94%	zalando nl	0.57%	verzendkosten bij zalando	0.39%
met	1.53%	no pockets	0.53%	zalando nl bestellen	0.39%
is	1.19%	met een	0.47%	outlet bij zalando	0.23%
in	1.12%	long sleeves	0.46%	outlet nu zonder	0.23%
no	0.98%	in de	0.42%	is voorzien van	0.20%
solid	0.84%	aan de	0.41%	our model wears	0.20%
colour	0.84%	heeft een	0.35%	model wears a	0.19%
het	0.84%	logo detail	0.33%	voorzien van een	0.19%
voor	0.83%	round collar	0.31%	is gemaakt van	0.17%
bij	0.60%	op de	0.31%	van het merk	0.16%
appliqués	0.54%	mid rise	0.30%	to be dressed	0.13%
closure	0.53%	button closing	0.30%	wears a uk	0.12%
sleeves	0.50%	plain weave	0.29%	in de kleur	0.12%
pockets	0.50%	verzendkosten bij	0.28%	true to size	0.11%
zip	0.50%	nu zonder	0.28%	kleding bij zalando	0.11%
model	0.49%	zonder verzendkosten	0.28%	kleding nu zonder	0.11%
op	0.48%	nl bestellen	0.28%	verkrijgbaar in damesmaat	0.10%

3.2 Color Histogram and Signature

As a replacement for the low quality and sometimes missing color attribute it is more reliable to extract the colors from the images. This can be done in different ways, with the most popular ones being color histograms and color signatures [27] .

For histograms, the color space is split into a fixed number of bins and for each bin it is counted how many pixels of the images fall into that color bin.

For signatures one defines a fixed number of clusters that have to be found (e.g. using k-means) in the color space and the centroids of those colors define the most prominent average colors (Figure 18).

The counting or clustering can also be done in different color spaces. While images are given in RGB it is possible to convert them to other color spaces like HSV and to Cie L*a*b*. Especially the latter one was designed to be closer to human perception in terms of distance between color vectors [25].

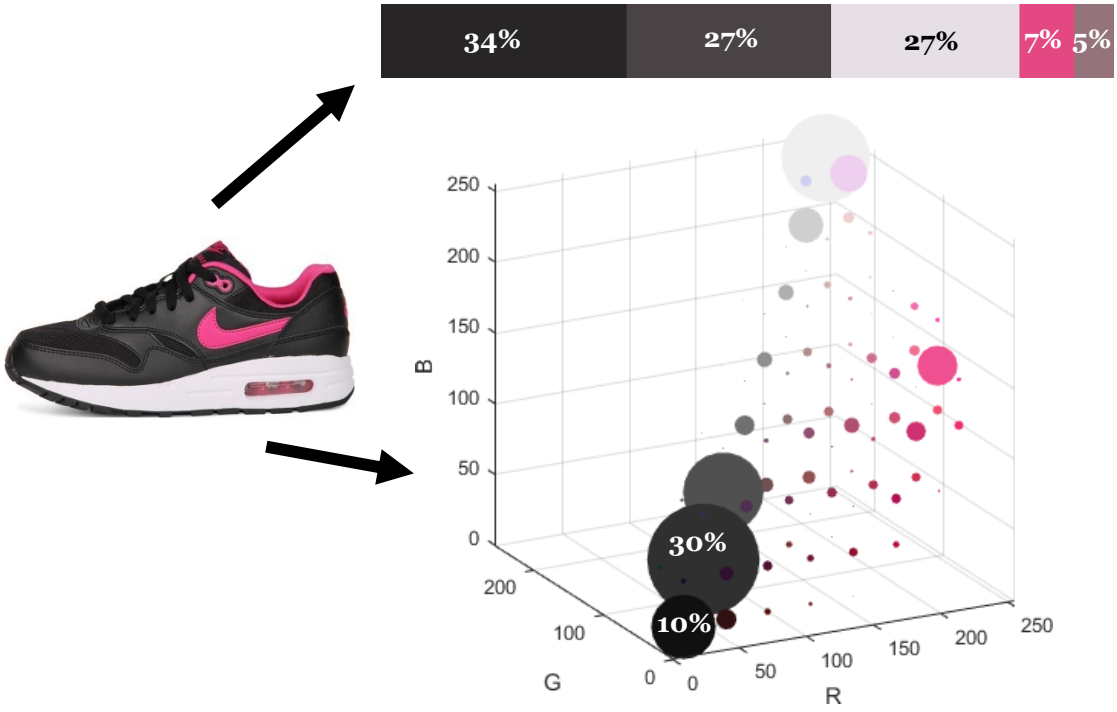

Figure 18 - RGB signature with 5 centroids and RGB histogram with 8x8x8 bins

3.3 Grayscale Statistics

Grayscale variants of a color image can be seen as a rough estimate of the shades in an image. These shades depend on the shape and texture of a product and hence may be a valuable indicator for these aspects. It is recommended to use a colorimetric (luminance preserving) conversion when calculating the grayscale variant to get optimal results [28] (Figure 19)

Figure 19 - Grayscale variant of colored image with luminance preserved

Since a grayscale image is just a two-dimensional array of numerical values its characteristics can be described with basic statistical measures, though care has to be taken that these measures are scale and translation invariant to allow for a meaningful comparison between images. The mean gray value and (standard) deviation are such measure. However, there also exist more complex descriptors such as the so called central normalized image moments which allow to capture the skewness and kurtosis of a numerical distribution [29].

3.4 Gradient Histogram and Signature

Histograms of oriented gradients (HOG) are based on the grayscale variant of an image and have shown to work well for shape based object detection and thus might also be useful for comparing fashion products by their shape [30].

The basic idea of HOG is to calculate the gradient at each grayscale pixel (e.g. using a Sobel or Scharr operator [31], Figure 20) and then accumulate the gradient orientations in a histogram to get a compact feature. The gradients capture the local steepness and orientation of luminance changes within an image and therefore give valuable information about the shape and texture of an object. For example, it can help to differentiate shirts with vertical and horizontal stripes.

When calculating the histograms, the sign of orientation is usually ignored so that the bins spread between 0° and 180°. For each bin, one can simply count the number of matching pixels or one could also take the magnitude of the gradients into account such that strong gradients are counted multiple times (Figure 21). The latter option tends to give better results when the resulting histograms are compared for similarity [30].

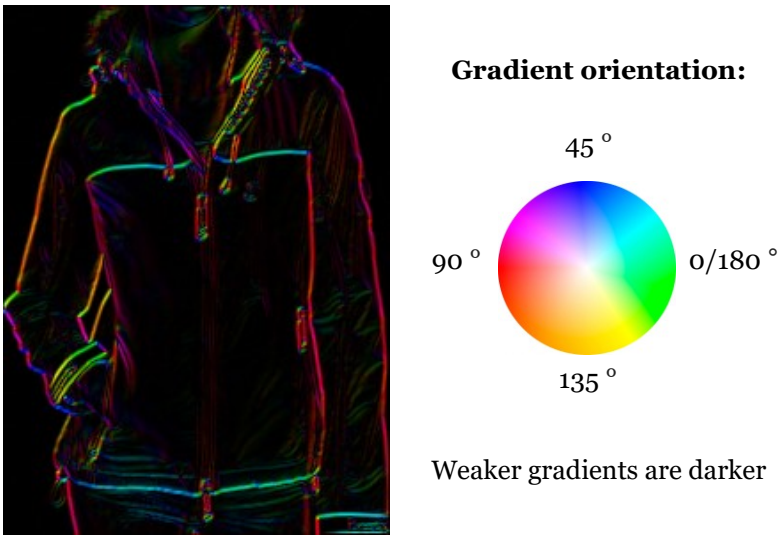

Figure 20 - Example gradient map

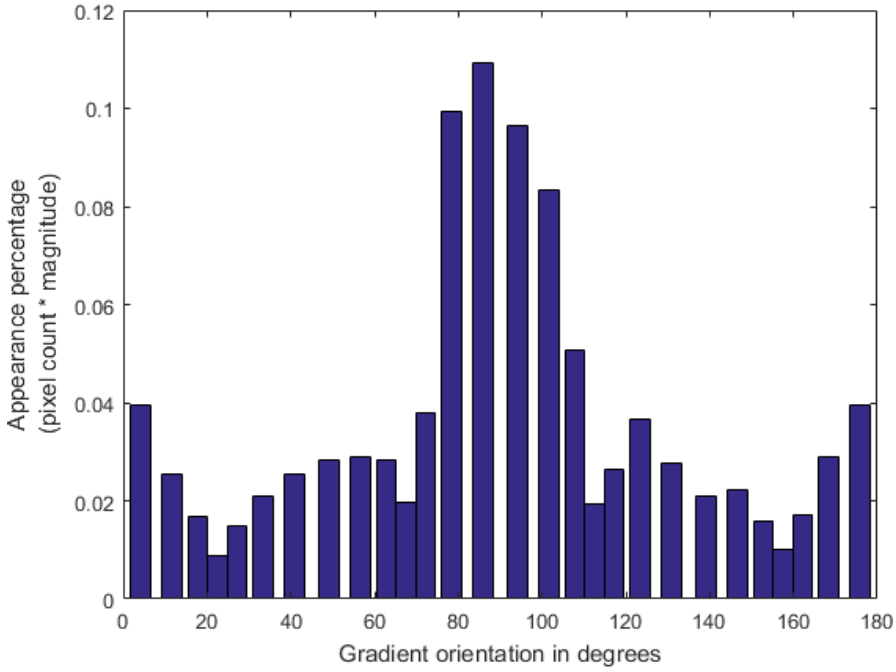

Figure 21 - Histogram of gradients

Beside histograms it is also possible to accumulate the gradient data in signatures similar to color signatures. However, instead of color vectors each pixel translates to a vector containing the gradient orientation. Again, pixels may be sampled multiple times depending on the gradient magnitude to get more accurate clustering results. Note however, that for clustering the samples it is important to use the correct angle representation. The clustering utilizes a vector distance measure like Euclidian to find similar samples. Unfortunately, this gives incorrect results when comparing angles in degree or radians notation. For example, angles of 0° and 179° would be considered to be very different when using Euclidian distance although they are almost the same. As a solution, orientations can be mapped to Cartesian coordinates on a circle of a fixed radius, resulting in a two-dimensional angle vector that can be meaningfully compared using vector distance measures.

3.5 LBP Histogram

Local Binary Patterns (LBP) are another popular way to describe the texture or pattern of an image [32]. Based on the grayscale variant of an image the basic approach is to compare each pixel with its neighbors and assign a binary number to the pixel reflecting which neighbors have a greater or lower value. The resulting numbers of all pixels are then accumulated in a histogram (Figure 22). The approach can be customized by specifying the number of neighbors that are taken into account and the radius, resulting in different feature sensitivity and data size.

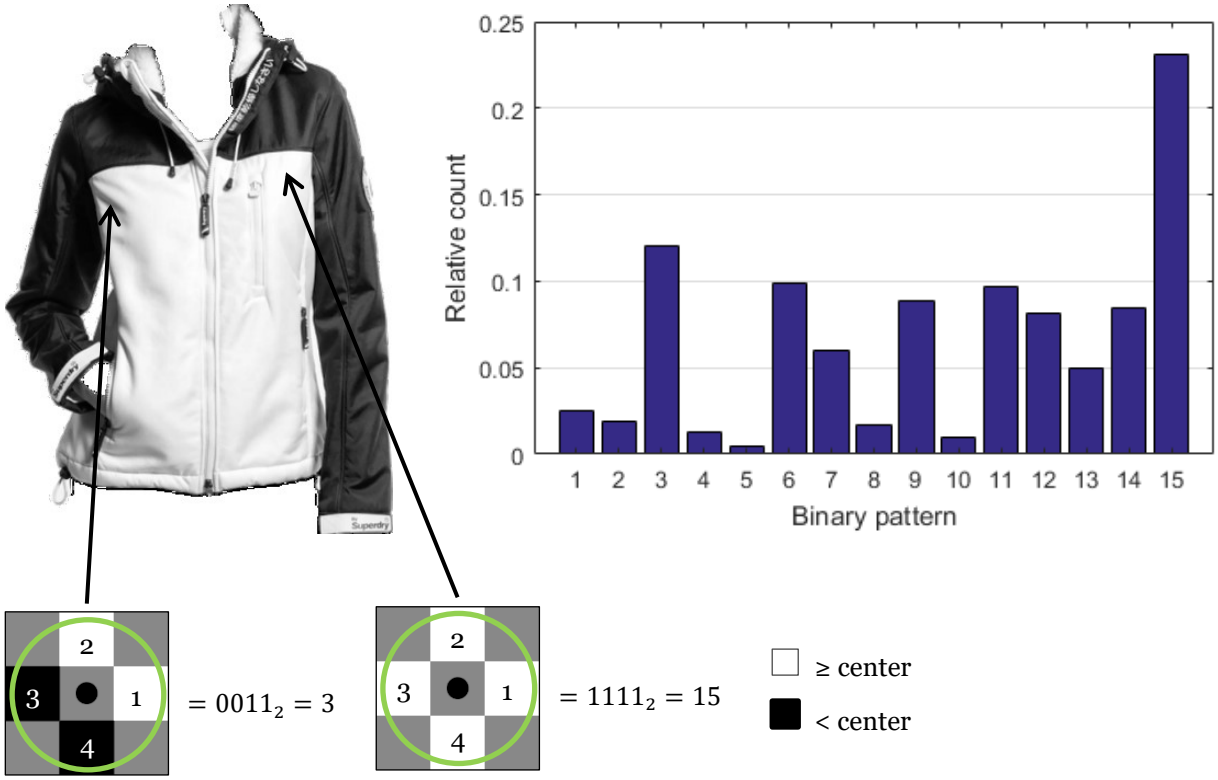

Figure 22 - Example LBP histogram with 4 local samples in a radius of 1 pixel

3.6 Gabor Filter Bank Responses

Another approach for extracting texture features is converting grayscale variants of an image from the spatial domain to the frequency domain using Discrete Fourier Transform. In the frequency domain, it is simpler to find the frequency and orientation of prominent patterns in an image (Figure 23). This information can for example help to distinguish shirts with thin stripes from shirts with thick stripes.

Figure 23 - Grayscale image in its frequency domain representation.

Since storing and comparing the complete frequency domain version of an image is not compact enough it is common to use a collection of filter bank responses instead. A good filter bank consists of filters for all interesting frequencies and orientations. Gabor filter banks have shown to work particularly well for this [33]. Each filter extracts the response to a certain group of frequencies and orientations and for each response measurements can be taken like the mean, variation, entropy and energy of the magnitude and phase information to quantify the response [27]. The exact filter bank design can be configured by choosing the number of orientations and scales to cover (Figure 24).

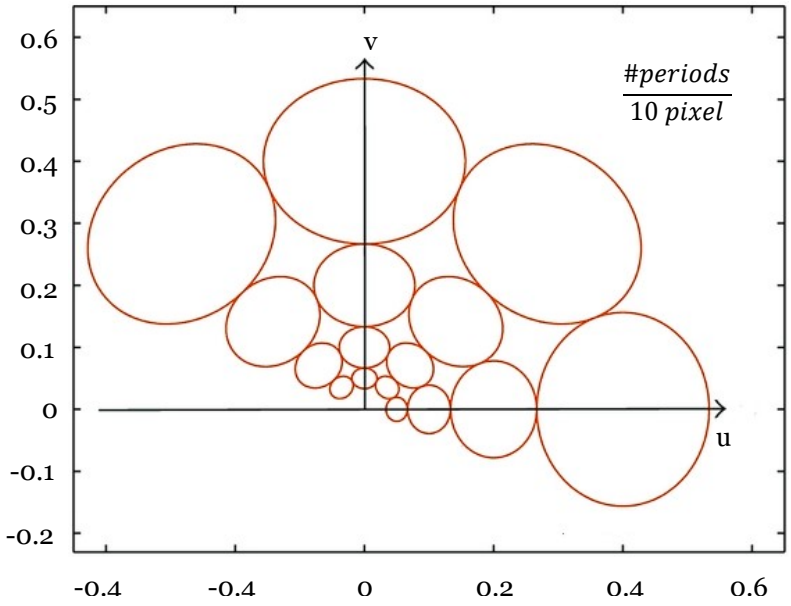

Figure 24 - Example Gabor filter bank design with 4 scales and 4 orientations

3.7 Contours

The contours of a product are another feature for describing the complex shape of a product. Contours can be extracted as series of two dimensional points found by detecting and tracing edges (Figure 25). The extracted contour points themselves can be treated as a feature but it can also be further abstracted. For example, one can calculate the ratio between height and width of the object. This might for example help to distinguish between long and short boots. Furthermore, several statistical descriptors like the deviation, skewness and kurtosis of the contour points distribution can be calculated, similar to the grayscale statistics.

Figure 25 - Example edge-based contour

3.8 Local Sensitivity

The above-mentioned image feature extraction methods were described as global features that are calculated and aggregated for the whole product image. However, related work has shown that adding local sensitivity to image features can significantly improve the quality of the features [27]. There are different approaches for adding local sensitivity depending on whether the feature is a histogram or signature feature.

Signature features for example can be localized relatively simple by including the x and y coordinates during clustering. Though for clustering and comparison it is important to normalize the coordinates and make them invariant to scaling and translation. One solution for this is to surround the image with a square grid ranging for -1 to 1 at each axis with the image center being at (0,0). Each pixel coordinate can then be translated to a normalized grid coordinate (Figure 26).

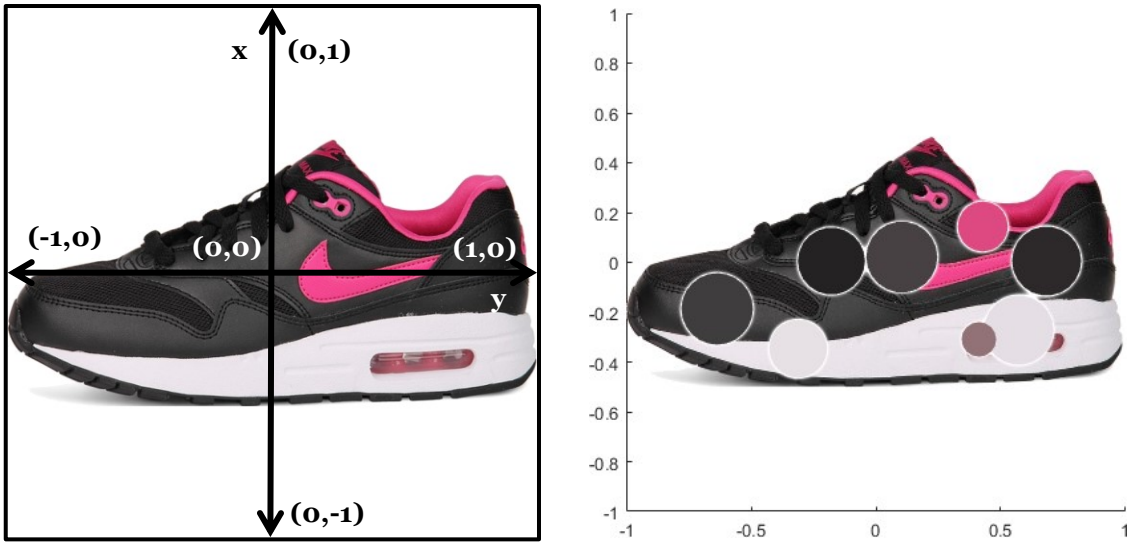

Figure 26 - Image centered within normalization grid and localized RGB signature example

For local histograms, the image can be split into smaller parts, e.g. by dividing the image into equally sized regions. Since product images can be of different aspect ratio dividing images into equally sized regions is not always straightforward. The image has to be resized to simplify comparison. Just like for localized signatures the easiest solution is to center the image within a square canvas image and treat any added pixels as irrelevant background pixels that are not counted (Figure 27).

Figure 27 - Image divided into 8x8 = 64 regions

The local features are calculated separately for each region and the resulting feature vectors are simply concatenated. However, this approach has the disadvantage that matching pixels between two images may end up in different regions when the compared images do not align properly. Especially pixels located close to the dividing grid are likely to end up in different regions which might lead to reduced similarity scores.

Another approach to localized features is based on using only regions around interesting key points with interesting being defined as having high information density and invariance to rotation, scaling and illumination. SIFT [34] and SURF [35] are two popular approaches that have methods for detecting interesting key points. While they produce potentially better localized features they are also much more difficult and expensive to compare. The key points have to be matched before the local features can be compared and unfortunately finding a match between key point sets is unreliable with deformable objects such as fashion products, rendering the key point approach unfeasible for this application (Figure 28).

Figure 28 - Matching SIFT key points of two shoe images calculated with fast nearest neighbor search

3.9 Perceptual Hash

Another image based feature is the perceptual hash. The idea behind perceptual hashes is to digest a complete image into a small value that can be used for duplicate image detection. This is also very handy for matching products because some webshops actually use the same images and if two products are depicted using identical images then it is very certain that the products are also identical.

A trivial hash for images could be the URL where it is located. However, this would not really help since webshops might use the same images but are likely to host them under a different address. Another hashing approach might be to calculate the MD5 or SHA1 hash of the files but this fails when the images are modified, e.g. when the images have been resized. For this scenario hashing techniques were developed which assess the perception of an image and are less prone to image.

Perceptual hashing methods vary in their robustness and accuracy. In general, the bigger the hash in bytes, the lower the error rate becomes but the slower the comparison gets. One well performing perceptual hashing technique is pHash MH [36]. It compresses edge information into a 72-byte hash that can be compared using the Hamming distance.

However, using pHash MH to scan all images for duplicates would be too slow. To counteract this drawback one can add a preselection step for duplicate candidates based on simple image features that filter out many true negatives without losing any true positive duplicates.

In particular, the mean and standard deviation of the image RGB values and the image aspect ratio are distinctive features that can be quickly calculated and compared. After preselection, only the remaining duplicate candidates have to be checked using the more complex perceptual hash. While this approach will result in a well-balanced accuracy it is not optimal for this application. To further reduce the false positive rate, a third pixel based verification step should be applied (Figure 29). When both images are resized to the same size and aligned using 2D correlation methods [37] it is possible to apply pixel based comparison measures as the peak signal-to-noise ratio (PSNR) or the structural similarity index (SSIM) [38]. Those methods are relatively slow but much more accurate.

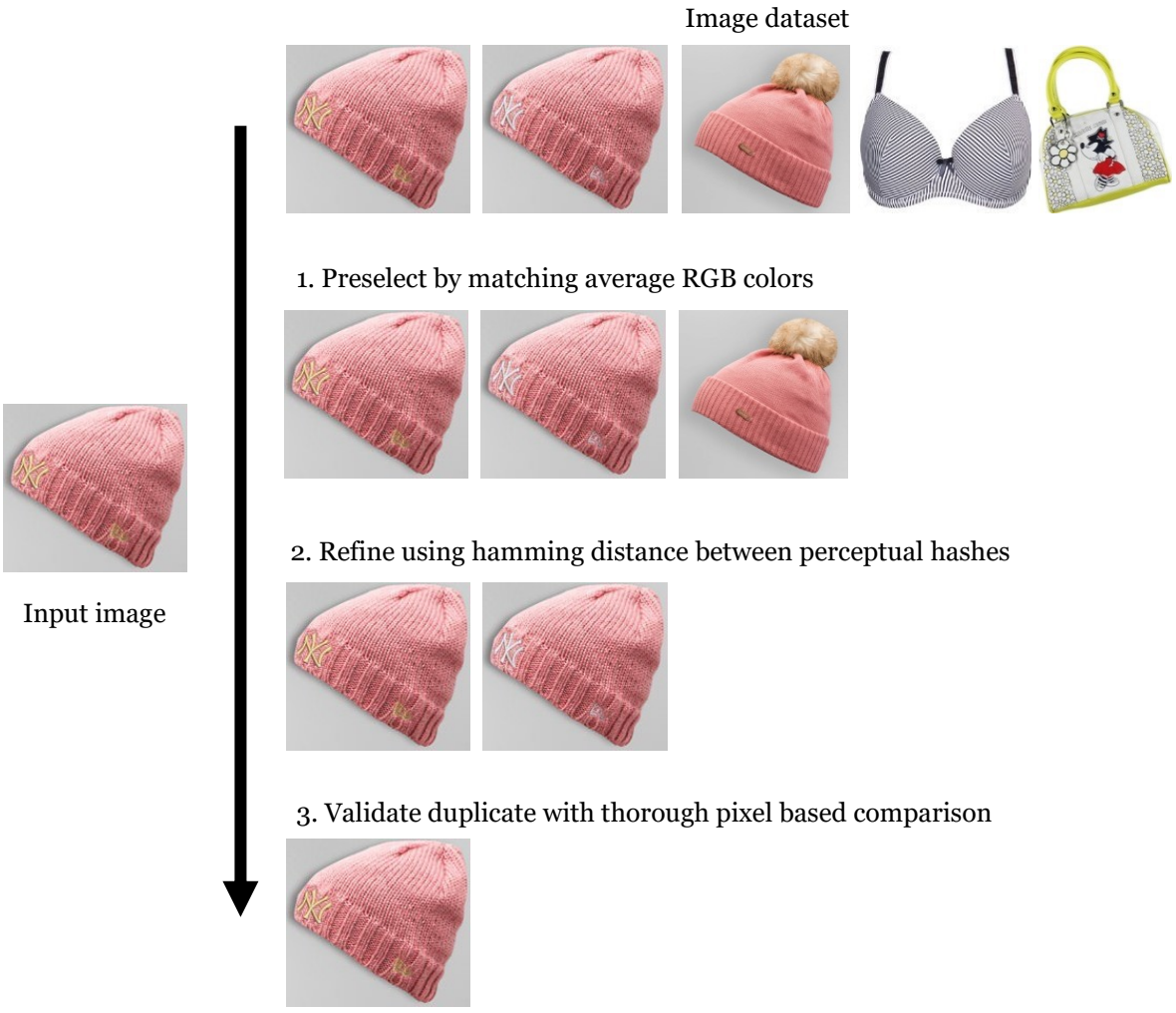

Figure 29 - Duplicate image detection approach

3.10 Runtime Performance and Space Requirements

For the scalability of the approach it is important that all steps can be executed in a feasible amount of time. This requirement also applies to the product feature extraction process although its impact is small compared to the calculation of similarity features. Contrary to the similarity features, the extracted features have to be calculated just once per product instead for each possible match. Furthermore, the calculation times of image features can be reduced by downscaling the images without affecting the results too much. Still, the time complexity of the presented feature extraction methods is an important selection criterion in the feature selection process later on.

As part of the runtime performance for the whole approach, it is also important to consider the retrieval of product features and hence their space requirements. It is desirable that all features can be stored in RAM for fast random access. Assuming that all features are stored on a single powerful machine with 512 GB of memory, the total feature size per product should not exceed more than a couple of hundred kilobytes in case of the Fashion Evolution dataset ($\frac{512\ GB}{1.500.000} \cong 340\ kB$).

To get an overview of the runtime performance and space requirements, all presented product feature extraction methods have been implemented in a software prototype and were applied to a set of 100 products. Whenever possible, well established and tested libraries were used to ensure correctness of the results. Most of the basic image processing is based on functions available in OpenCV 2.4 [26]. Only if the algorithm was very simple or no library function was available (e.g. for Gradient or LBP calculation) it has been implemented in Fashion Evolution's own library and thoroughly tested using extensive specifications and several months of production use.

Note, that the runtime and space requirements of the extraction methods depends on certain parameter settings, e.g. number of (local) histogram bins, n-gram length and number of signature centroids. To get representative benchmark results, ~1500 product feature variants with different parameter settings were chosen and analyzed. The complete list of features, including their benchmark results can be found in Appendix 9.1 and Appendix 9.2. Benchmarks were run on a Intel i7-3610QM CPU @ 2.30 GHz with 8GB memory and SSD storage. Images were scaled to a pixel mass of ~512 x 512 pixels before calculating image features.

A summarized overview of the runtime results can be seen in Figure 30. It shows that most of the chosen product features can be calculated within a feasible amount of time. As expected, text based features can be calculated the fastest. Image based feature are more expensive. In particular, the calculation of Gabor filter based features are the most time consuming due to the complex Discrete Fourier Transform. The slowest Gabor filter based features (> 900 ms) will not be used in further steps.

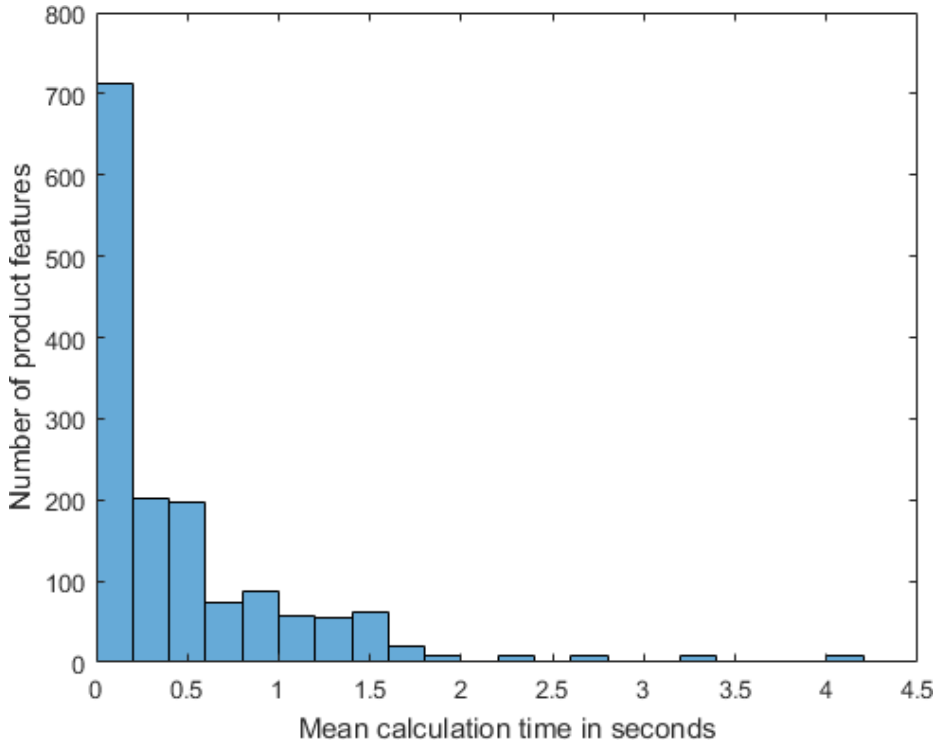

Figure 30 - Histogram of mean calculation time for product features

A summarized overview of the space requirements can be seen in Figure 31. Most of the product features fall within the range of less than 340 kB. Only very detailed histograms and long descriptions exceed this limit.

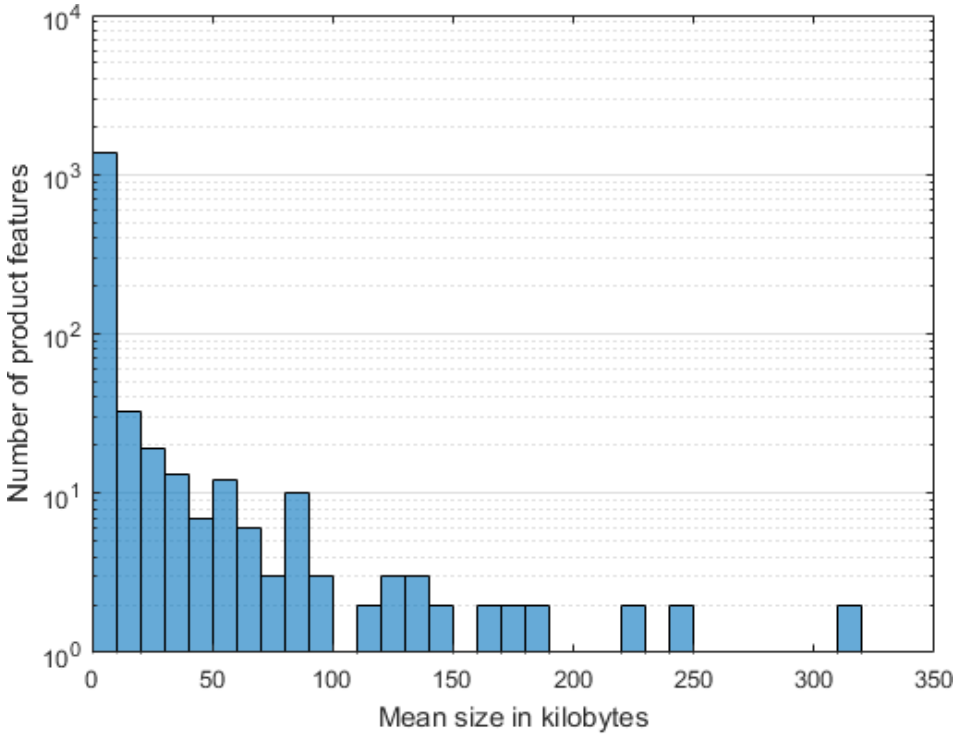

Figure 31 - Histogram of mean feature sizes for product features

3.11 Conclusion

This chapter focused on the extraction of additional robust product features that were not yet readily available. For textual metadata, the additional information was rather limited. Tf-idf models were found to help identify key terms in product names and descriptions but most additional information can be extracted from the product images. Several methods for extracting color, shape and texture features were found in the literature. Furthermore, the possibility to add local sensitivity to the image features for even more distinctiveness has been shown.

Due to different parameter options for each feature extraction method, an arbitrary number of product features with different characteristics can be generated. A selection of ~1.500 product features has been made and analyzed for runtime performance and space. Most of them are found to be suitable for usage in a scalable comparison strategy.

Last but not least, an accurate but also scalable approach for assigning perceptual identifiers to product images has been proposed. Despite different URLs and image resolutions, it allows to find identical images using a combination of drilldown selection, perceptual hashing and pixel based similarity measures.

4 SIMILARITY FEATURES

The previous sections presented several fashion product features, ranging from single textual or numerical values to multidimensional data. Given these product features, the following step is to find ways to quantify their similarity. This allows to later on decide whether two sets of product data can be considered identical. The next sections will each address a specific type of product feature and use literature research to find methods for comparing them.

4.1 Single Values

Product features such as the normalized brand, perceptual hash and possibly available MPN can be seen as single value features that can be easily compared. They are either identical or different so a simple equality check suffices.

However, there are also other single value numerical features such as the price and aspect ratio that should be treated with more flexible similarity measures. Instead of checking two product prices for equality it makes sense to measure the numerical distance, e.g. the Manhattan or Euclidian distance between them because prices may naturally vary due to discounts or currency conversion. Note however, that matching based on price might contradict the goal application of finding interesting savings. If only products with very similar prices are matched, then the whole point of a price comparison feature might get lost. Therefore, price similarity features are excluded in this research.

Similar to prices it might also be beneficial to apply numerical distances to GTIN. In practice when producers assign different GTIN to different size variants of a product it can often be observed that those numbers are consecutive. Therefore, a small distance between two GTIN can be a valuable similarity hint.

4.2 Texts

Comparing the name and description attributes can be treated as a string or document similarity problem. A lot of research has been done on this field. For comparing shorter strings (e.g. product names) Hamming, Levenshtein, Jaro, Jaro-Winkler, Longest Common Substring, Longest Common Subsequence and Pair Distance (based on Jaccard) are popular measures which are also widely available in string similarity libraries [39].

For longer documents comparing the suggested tf-idf vectors is more conventional. The frequency vectors can be compared with histogram similarity measures as described further on, assuming that the terms mark the bins and the tf-idf score represent the bin count. There are also measures which try to combine the advantages of the tf-idf measure and the above-mentioned algorithms like Jaro-Winkler into so called soft tf-idf similarity measures [40].

4.3 Category Paths

For assessing the similarity between two category paths one can apply tree similarity measures. For example, the 'depth of common ancestor' measure finds the lowest common ancestor and takes the depth of it as similarity measure. The deeper the lowest common ancestor, the more similar they are. The shortest path distance on the other hand measures the lowest number of edges to be traversed until one category node can be reached from the other. The shorter the distance, the more similar they are (Figure 32). Furthermore, there are measures by Wu et al. [41] and Nguyen [42] which combine those measures. Li et al. [43] developed an even more complex category similarity which also takes the semantic similarity of concepts into account.

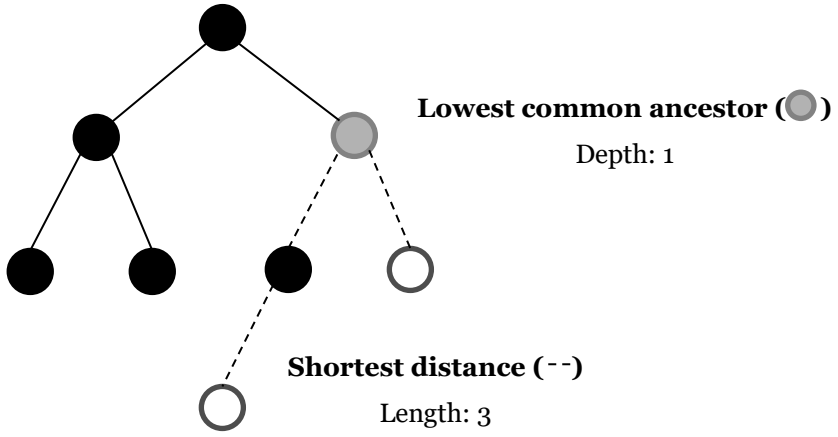

Figure 32 - Example tree with lowest common ancestor and shortest path illustrated

4.4 Histograms

The color, gradient and LBP histogram features are histograms in the common sense, i.e. they consist of an array of binned counts. However, tf-idf vectors, grayscale moment values and Gabor filter bank responses can also be treated as histograms. As such, several similarity/distance measures exist that can be applied to those features, e.g. the Manhattan distance (L1 norm), Euclidian distance (L2 norm), angular, cosine angle distance, intersection distance, Hellinger and Chi^2 test.

4.5 Signatures

Signatures such as color and gradient signatures consist of weighted feature centroids. Since those centroids can be located anywhere in the multidimensional feature space it is very likely that the centroids between two signatures do not align as in histograms which have fixed bin ranges. Hence a more flexible similarity measure is required and the Earth Movers Distance (EMD) [44] is a popular choice for this. The name originates from the idea that the weighted centroids represent piles of dirt and that the distance between two signature vectors can be represented by the amount of dirt and the distance it has to be moved to transform one signature into the other. Internally the Euclidian distance is usually used to measure vector distance but other vector distance measures are possible as well.

4.6 Contours

Assessing the similarity of a series of contour points requires another type of similarity feature. Both shapes have to be matched and a distance has to be established. Basic approaches as the sum of distance between nearest neighbors come to mind but lead to unsatisfactory results if the contours do not align well. Better results can be achieved with more advanced but also much slower methods like Shape Context Matching [45] which use the distribution of contour points in the neighborhood of each point to find matches between two points and then calculate the distortion that is needed to get from one contour to the other using affine or thin plate spline (TPS) transformations (Figure 33).

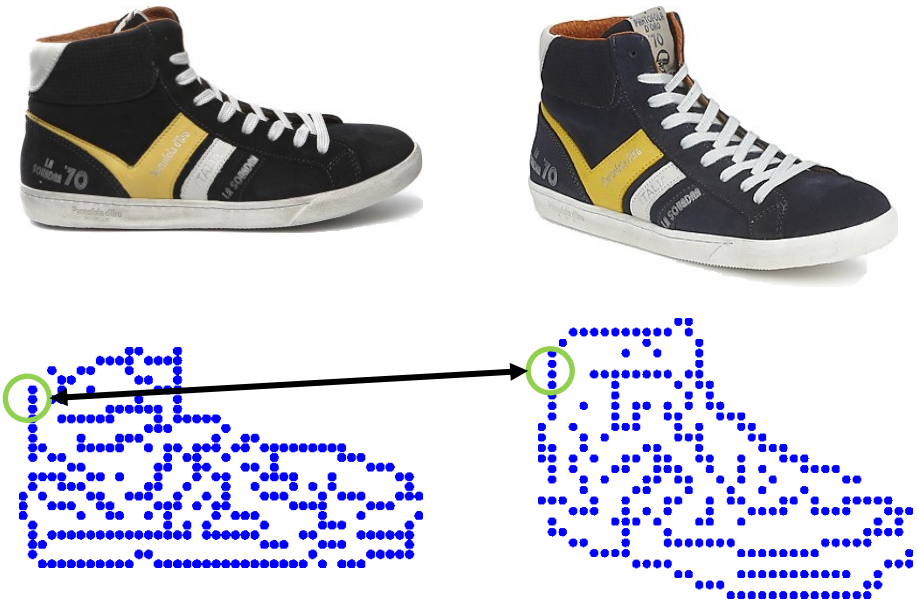

Figure 33 - Shape context matching example

4.7 Runtime Performance

The costs of calculating feature similarity depends on the type and data size of the input product features. Single values such as the image hash can be checked for equality in a neglectable amount of time but other similarity features for signatures and contours have to investigate multiple possible solutions and naturally take more time. Since these calculations are done for each possible match, it is a critical hotspot that needs to be well optimized to allow for a scalable comparison approach.

All suggested similarity measures have been implemented in a software prototype and applied to 100 random product pairs. Again, whenever possible, well established and tested libraries were used to ensure correctness of the results. String similarities were calculated using the amatch ruby library [46]. EMD and most of the histogram distance measures are used from OpenCV 2.4 [26].

Combining the previous list of ~1.500 product features with the different similarity features from this chapter results in a set of ~11.000 similarity features. The full list of similarity features and their runtime performance can be found in Appendix 9.3.

Figure 34 shows that most of the similarity features perform within an acceptable amount of time. Only shape context calculation is a significant outlier with more than 500 ms calculation time even with images scaled down to a pixel mass of 32*32 pixels. It is therefore excluded from further steps.

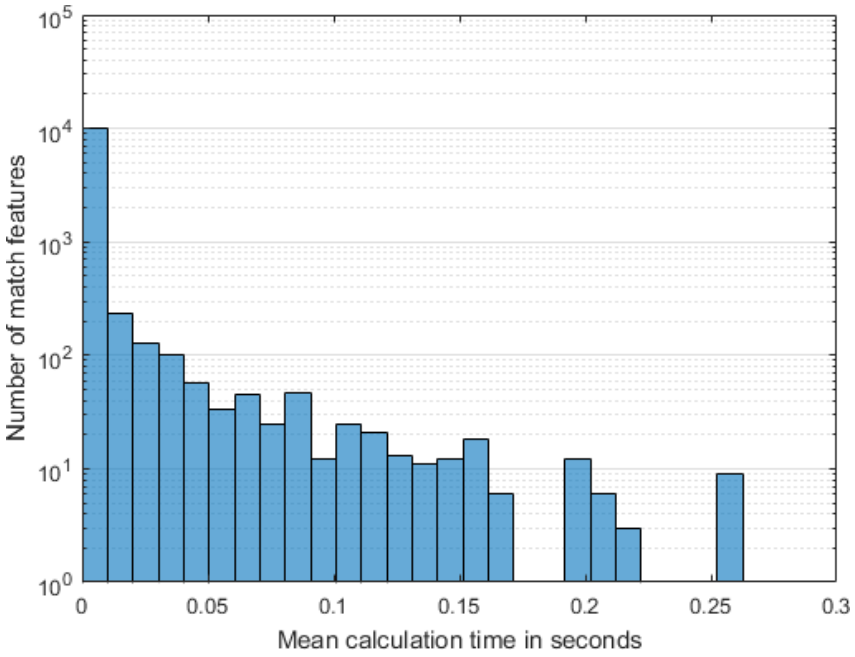

Figure 34 - Histogram (log scale) of mean calculation time for match features

4.8 Conclusion

This chapter presented multiple similarity and distance measures found in the literature that are applicable to the presented product features. Most of the product features are histograms or signatures, and for those especially many options were found to compare them.

Combining the selected product features from Chapter 2 and Chapter 3 with all different similarity measures results in a total of ~11.000 similarity features. Only a few features had to be rejected due to excessive computational costs. The rest was found to scale sufficiently well for later steps in the entity matching approach.

5 DRILLDOWN MATCHING

Based on the previously described similarity features it becomes possible to create a classification model that, theoretically, could be applied to all possible product pairs. However, as the number of possible pairs grows quadratically with the number of products, it is not feasible to process all of them when millions of products are available.

Fortunately, many product pairs can be quickly ruled out without having to compare them pair by pair. When we reflect on how we would manually try to find identical products most people would probably proceed as follows. If the desired product has a distinctive name, the quest will usually start with a keyword search. When the name turns out to not be distinctive enough, most people will then start to add more search filters to narrow down the results, e.g. by selecting a specific category, brand and/or color. In many cases this leads to a manageable number of remaining products which then can be more thoroughly compared and hopefully result in identical matches.

Of course, by hand this procedure is very cumbersome but it reflects a typical three-step approach for automatic entity matching [6] [5] (Figure 35).

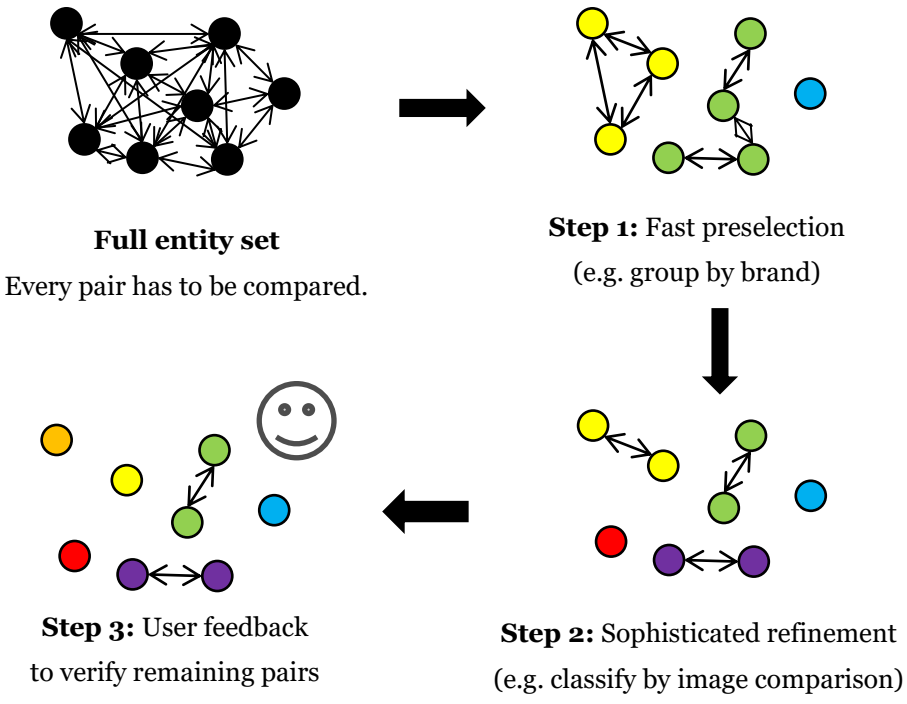

Figure 35 - Typical drilldown approach for entity matching

Step 1 (fast preselection) mainly deals with the scalability of the approach while **step 2** (sophisticated refining) focuses more on the actual problem of product entity matching. The purpose of **step 3** (user feedback) is to ensure correct matches as output.

The first two steps optimally rule out as many non-matches as possible (low false positive rate) while retaining near 100% of the matches (high true positive rate). Unfortunately, the costly user feedback cannot be left out because the refining classification model will probably not achieve the required near-perfect accuracy. However, if the automatic first two steps succeed in sufficiently reducing the set of final candidates the whole approach can be viable and fully automatic when combined with cloud services like Amazon Mechanical Turk [47]. Integrating user feedback has also the advantage that it provides valuable test data that can be used to iteratively improve the system in the future.

5.1 Fast Preselection

The first drilldown step is to quickly reduce the set of possible matches using a simple clustering approach. In a dataset with 1.5 million entities there are up to 1.125 trillion possible pairs. Luckily there exist a few trivial product data filters to quickly reduce this amount without excluding true matching pairs.

5.1.1 Filter by Brand

First of all, only products of the same normalized brand need to be compared. If the brand differs between two products, they cannot be identical. Assuming that the brand information can been normalized almost perfectly as described in Chapter 2.3 the brand information can be used to cluster the products into much smaller disjoint sets. In case of Fashion Evolution's dataset for example, the sets have an average size of about 100 products with only a few brands having up to 20.000 products (Figure 36).

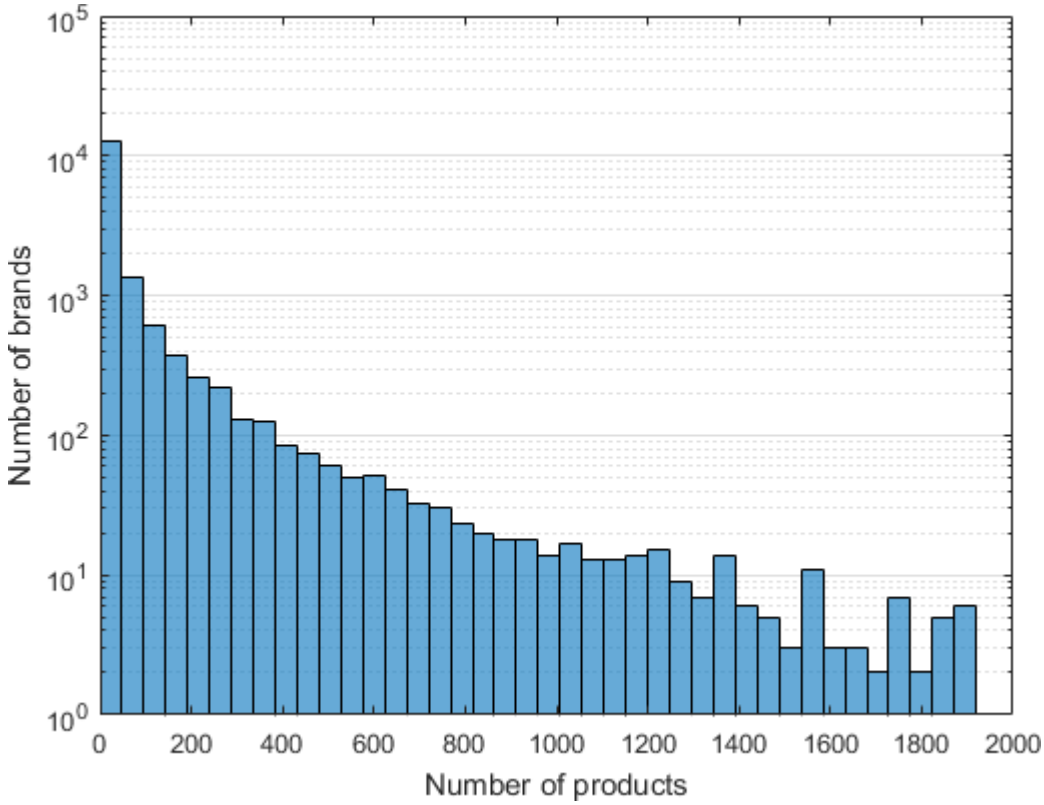

Figure 36 - Distribution of product count per brand

5.1.2 Filter by Category

The second trivial yet very efficient filter consists of matching only products of the same category. It does not make sense to compare jackets with jeans for example. However, this filter has to be less strict than the brand filter. Products may actually be placed in multiple categories, e.g. a winter sport jacket may be categorized as *'winter jacket'* and *'sport jacket'*. Furthermore, products may not always be categorized as specific as possible, e.g. a bra might be categorized as *'bra'*, while its identical partner might be categorized as *'strapless bra'*. These details require a more flexible filter where some deviation in completeness and specificity is tolerated.

5.1.3 Filter by Target Group

The third basic filter condition is based on the normalized target group of products. For most fashion products, it does not make sense to compare products for adults with products for children for example. The same applies to the targeted gender. Female-only products will not match with male-only products. However, in practice it may occur that one shop marks a product as unisex while another shop labels the same product as being (fe)male. Age specificity may also vary between shops, e.g. one shop communicating children as target group while another uses the more specific teenager or baby label. To cope with these cases, it makes sense to allow some slack for the target group matching as well.

In case of Fashion Evolution's dataset all products can be clustered into ~200 sets with a unique combination of category and target group. Note however, that these sets are not completely disjoint as sometimes products may be present in multiple sets due to ambiguous categorization. Nevertheless, category and target group also allow for efficient clustering as can be seen in Figure 37. Most clusters contain less than 3.000 products, however there are a few large clusters with up to 130.000 products (e.g. women shirts). On average, this results in 8.800 products per set and with each product being a member of ~1.17 sets.

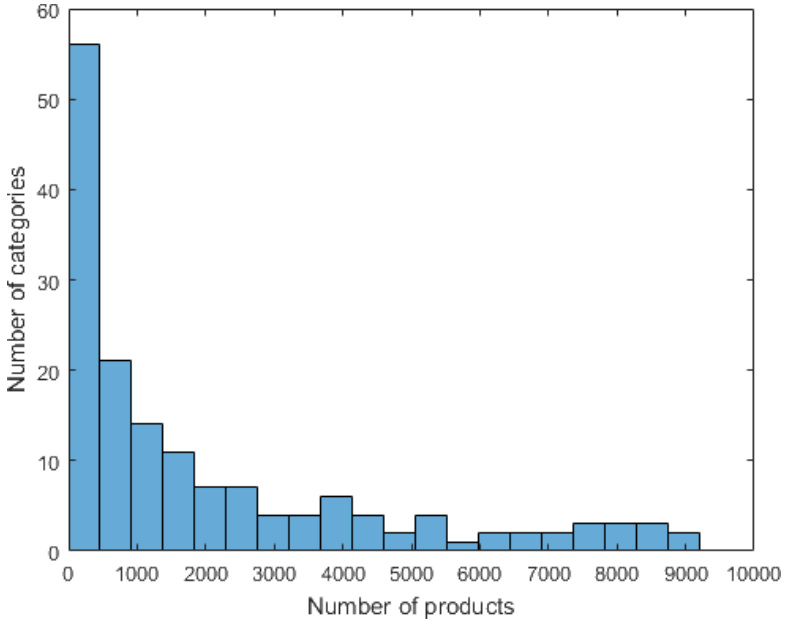

Figure 37 - Distribution of product count per category and target group combination

5.1.4 Combining with Shop Filter

Another preselection filter exploits the assumption that webshops do not have duplicate entities in their own dataset. Hence pairs of products which originate from the same webshop can be skipped. This rule is especially efficient at filtering out pairs between products of exclusive brands that are only available at a single shop.

Combining brands with category and target group filters and ignoring clusters with only a single shop results in the most efficient preselection model. When applied to the dataset of Fashion Evolution the products are distributed over ~20.400 mostly disjoint sets. Most of these clusters have less than 1000 products. There are a few outliers with up to 6.000 products, tough, e.g. women shirts by ONLY. These outliers contain less than 10% of all products but are responsible for bumping the average counts up to 55 products and 8.900 possible pairs per cluster. A total of ~180 million product pairs remains to be checked for the next drilldown step. If the outliers with more than 1.000 products are ignored, assuming that the outliers can be further split using deeper categorization or other filters, the average counts drop to 49 products and 3.580 pairs, a total of ~73 million possible product pairs.

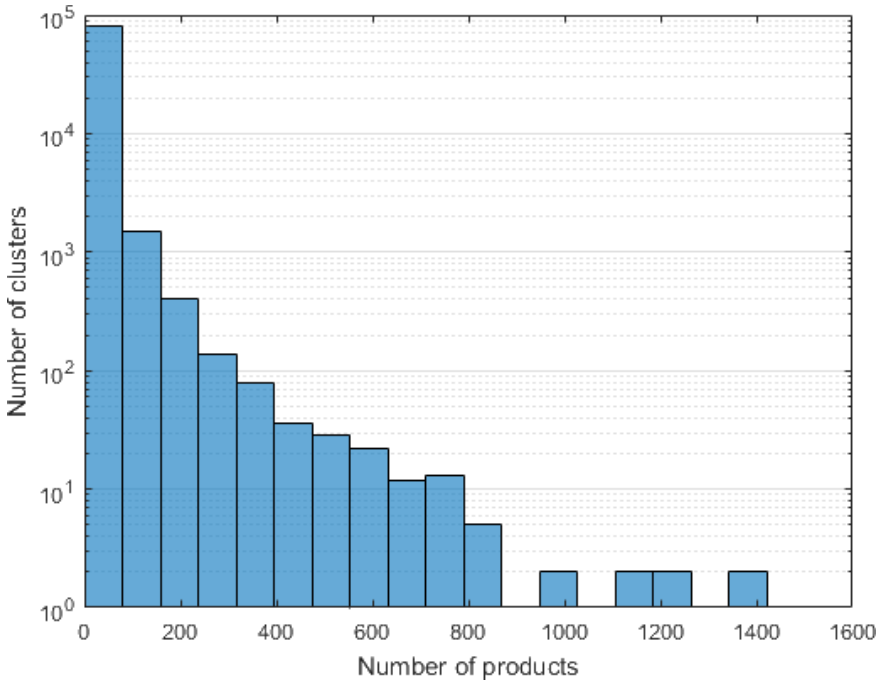

Figure 38 – Distribution of product count per preselected cluster

Last but not least some of the more complex similarity features may also be considered useful for a fast preselection. For example, calculating the Levenshtein distance between product names or calculating the Earth Mover Distance of color signatures can happen relatively fast as was shown in a benchmark (Appendix 9.2). However, it is crucial to design these filters such that (almost) no true positive matches are ruled out in order to not affect accuracy. This vastly limits the effectiveness of these features for preselection. It also adds the complexity of pairwise comparison to the preselection process compared to the much simpler described product based filters. Therefore, more complex similarity features are left to the more sophisticated refining model in the next section.

5.2 Refining

The second step after the initial preselection aims at further filtering out identical matches using more thorough comparison methods. In total four refining strategies are proposed. The first three strategies are based on matching image hashes, GTIN and MPN data. These strategies have the advantage of being simple and fast and hence are optimal for the initial comparison of a new dataset. The disadvantage of these strategies however is that they are not applicable to all product pairs due to the limited availability of the attributes they are based on. Therefore, the focus will be on a generic strategy applicable for any product pair that depends only on consistently available data.

5.2.1 Optimal Comparison Strategy

To prevent redundant comparisons of products during refining, a clear comparison strategy has to be followed. The simplest solution is to assign a numeric identifier from an increasing sequence to each product and pair products in a fixed order. Products should only be compared with other products that have an identifier that is less than its own identifier (Figure 39). Newly discovered products should be assigned greater identifiers such that they will be automatically compared with existing products (Figure 40). Updates of existing and already compared products are an interesting edge case. In those cases, all products with smaller identifiers need to be considered as well as newer products which themselves were already compared (Figure 41). Following these strategies guarantees that only the exact number of possible pairs $\left(\frac{n(n-1)}{2}\right)$ is checked.

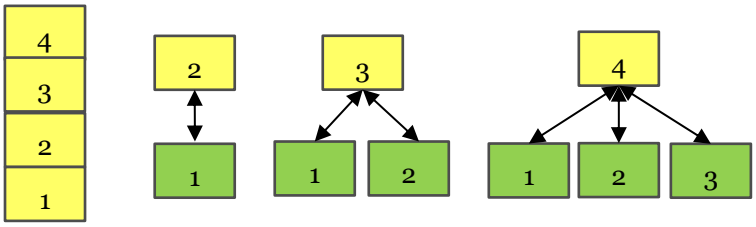

Figure 39 - Case A: Possible pairs to check when all products are uncompared

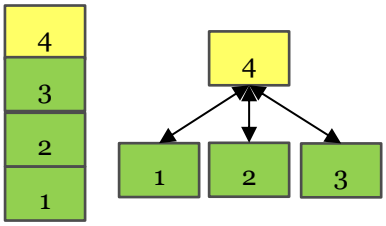

Figure 40 - Case B: Possible pairs to check when new uncompared product is added

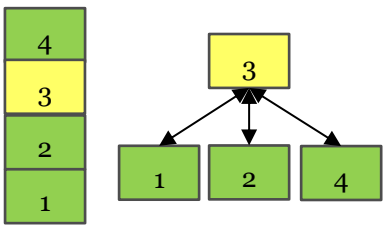

Figure 41 - Case C: Possible pairs to check when previously compared is updated

5.2.2 Using Image Hash, GTIN and MPN

The **first** refining strategy exploits the fact that some webshops use the same images and that identical images are a very reliable indicator for identical products. When the product images are hashed and deduplicated as described in Chapter 3.9 it becomes easy to check for an identical match when comparing two products. If both products have at least one shared image the match can be classified as identical (Figure 42).

Figure 42 - Matches based on shared identical images

The **second** refining strategy simply matches based on a product's GTIN. Of course, this is only applicable for the limited number of products that actually are published with one or more GTIN but it would be an unused opportunity if available GTIN data is ignored. Yet it requires more care than the image hash based matching.

If both products have exactly one GTIN assigned and those numbers match exactly, it turns out to be quite likely that the products are actually identical. Only if one or both products have more than one GTIN assigned, experience has shown that the match is significantly more uncertain. Valid multiple GTIN for a single product are possible due to different GTIN for size variants but some webshops mix in GTIN from other color variant that would lead to incorrect matches. When matching products with multiple GTIN, the uncertainty should be reflected in the matching score.

Additionally, GTIN matching can be extended with the GTIN similarity feature (Chapter 4.1) that exploits the fact that variant GTIN are often assigned as successive numbers. Hence if two GTIN only differ slightly it could be very well a match. However, the difference and resulting uncertainty should be reflected in the matching score with bigger differences resulting in lower scores, e.g. using the L1 or L2 distance.

The **third** refining strategy can be applied to matches where both products have MPN data. For comparing MPN it is required that both products are actually from the same manufacturer but this requirement is already fulfilled due to the brand filtering preselection step. Beside that the MPN strategy works the same as the GTIN strategy.

5.2.3 General Fallback

In order to handle any product pair, a general fallback strategy is required that relies solely on consistently available similarity features. The findings from the previous chapters have shown that brand, category, target group, name, description and image based similarity features fall into that category. Brand, category and target group are already used in the preselection step. In combination with the benchmark from Chapter 3.10 and Chapter 4.7 the list of useable similarity features for a general classification model is further reduced (Table 12).

Table 12 - List of possible similarity features for refinement

Product feature	Similarity measure
• Name (raw, normalized) • Description (raw, normalized)	• Levenshtein • Hamming • Jaro • Jaro-Winkler • Longest Subsequence • Longest Substring • Pair Distance
• Category	• Common ancestor • Shortest path • Wu et al. • Nguyen et al.

• Color histogram
• Gradient histogram
• Name/description tf-idf
  ○ (Stemmed) word n-grams
  ○ Letter n-grams
• Grayscale statistics
• Edge map statistics
• Gabor filter response statistics
	• Manhattan
	• Euclidian
	• Intersection
	• Chi^2
	• Cosine
	• Angular
	• Hellinger

• Color signature • Gradient signature	• EMD

These similarity features have to be combined in a classification model which can decide whether a product pair is identical or different. Such a model could output a simple yes/no answer but preferably it results in a matching score that is related to the certainty of two products being identical. This would allow to apply user feedback more efficiently, starting with those matches which have the highest chance of being correct. It would also allow to control the false positive/negative rate by simply adjusting the classification threshold.

Furthermore, any model needs to be fast at calculating the matching score or it would be a threat to the scalability of the approach.

Last but not least, it should be possible to build the model automatically using example data. Manually building and tuning a model as done by Hsu et al. [13] is very cumbersome and likely to perform far from optimal. A better choice are models that can be optimized with machine learning, especially when training costs are less of a concern as in this application.

Considering these requirements of (1) confidence estimates, (2) fast classification and (3) machine learning, there are a few model types that fit:

- Naive Bayes
- Random Forests
- Support Vector Machines (SVM)
- Logistic Regression
- Neural Networks

Which of these models actually performs best for refining depends on the nature of the data and will be determined using experiments with real world data in Chapter 6.

5.3 Conclusion

This chapter described how a typical drilldown approach in entity matching problems can be applied to the fashion product domain using the previously presented product and similarity features.

The first step of the drilldown approach, preselection, was designed to exploit domain specific knowledge in order to quickly reduce the search space. It was found that a simple clustering on a combination of brand, category, target group and webshop is very efficient. It reduces the number of pairs to be processed by the next drilldown step from 1.125 trillion to ~73 million in a typical dataset of 1.5 million products.

For the second drilldown step (refining) multiple approaches were described. Matching based on GTIN, identical or shared image hashes and MPN are briefly discussed as refining options. These methods can only be applied to all small subset of possible pairs but allow to quickly find positive matches in new datasets. The main focus was the development of a general refining model that only uses consistently available name, description and image based similarity features. Five machine learning optimizable classification models are suggested which are able to combine these features and calculate a matching score. These classifier options will be used in the experiments of the next chapter.

6 VALIDATION

This chapter is dedicated to the validation of the proposed matching approach using real world data. It builds upon the results from the fast preselection method in Chapter 5.1 to build and assess a classification model for refining, using different combinations of similarity features and classifiers. Based on the results of the best performing model, it will estimate the number of remaining uncertain matches and investigate the feasibility of integrating user feedback to solve this uncertainty.

6.1 Dataset

The dataset of labeled product pairs for supervised learning of a classification model has been created from scratch as there appears to be no existing benchmark dataset for this problem. The fashion domain datasets from related work (e.g. [16]) is not usable as they lack required meta data and contain fashion images for other applications (e.g. magazines) that differ significantly from the presented typical product photos.

It has been decided to collect product pairs from real world data available at Fashion Evolution. The data originates from product feeds provided by more than 250 webshops active in the Netherlands with about 1.5 million products. The products belong to 16.500 different brands and range from baby to adult fashion and a variety of categories like shirts, bags, shoes, dresses and hats. Most of the products have Dutch descriptions but some products come with English descriptions.

Different strategies have been applied to collect positive (identical) and negative (different) product pairs in order to ensure a variety of samples, ranging from obviously mismatching pairs to challenging edge cases of visually (very) similar products. All sampled pairs fulfill the preselection conditions as described in Chapter 5.1, i.e. all positive and negative samples are matches between products with the same brand, similar category and target group (age and gender). Images with identical perceptual hash as determined in Chapter 3.9 are also excluded. This allows to focus on the harder refining classification of pairs that cannot be classified otherwise and subsequently make valid statements about the feasibility of user feedback integration.

As a starting point for finding positive samples it was chosen to look for popular products that are known to exist in multiple webshops. However, this strategy was very cumbersome and therefore quickly dismissed. Nevertheless, in combination with a random selection of non-matching pairs the found samples were sufficient to train a simple model that could be used to speed up the process [48]. In order to prevent the initial model to influence future models too much, more than half of the positive samples were collected using alternative ways such as matching based on GTIN, shared images and MPN as described in Chapter 5.2.1.

For negative samples a selection of randomly paired products were made (within the preselection conditions) as well as similar yet mismatching product pairs that were found during manual verification of all samples. Manual verification was done with an expert group of at least 2 people and when in doubt samples were not included in the dataset. The product pairs with small difference form the most interesting edge cases for the classification problem.

The final dataset consists of 18.028 sample pairs, half of them belonging to identical product pairs and the other half consisting of non-matching product pairs. Category-wise there are 45% clothing pairs, 37% shoes and 18% accessories. Target group distribution is 36% men, 39% women, 12% unisex adult and 13% children. The paired products originate from 233 different shops and 947 different brands. Similarity features with a variance of less than 1% across all samples were removed as useless, resulting in a reduced feature set of 8.069 features.

Note, that the class ratio in this dataset is not representative for the real world as the actual percentage of identical pairs is much smaller. There are for example many brands that only sell at a single shop and therefore cannot possibly have a match with another shop. Still, it is preferable to train on a dataset with equal class sizes. Hulse et al. [49] has shown that training with balanced class sets instead of simulated real world ratio leads to better classification performance.

One drawback of this approach is that it is impossible to calculate real world accuracy or precision since these measures are dependent on the ratio between both classes. When the ratio of non-matching pairs increases, the precision/accuracy will naturally decrease as the number of false positives increases. However, precision and accuracy are of less relevance for the refining step. The most important measure for the refining classification model is the recall (true positive rate). The desired model has to exclude as many non-matching pairs as possible while retaining (almost) all true positives. ROC curves include this information by plotting the true positive rate against the false positive rate at different threshold choices of a model. They also allow for an easy visual comparison of model performances and will therefore be used as the preferred measure of classification performance.

For testing the final model(s) 20% of the dataset is excluded during any training and intermediate testing. The dataset is standardized by subtracting the mean and dividing by the standard deviation for each similarity feature before training and testing any model.

6.2 Classifier Selection

For brevity and focus of this research it is first determined which classifier produces the best models using a random stratified subset of 1.000 samples. This is necessary as the subsequent steps like feature selection depend on the classifier choice and executing all steps with all possible classifiers on the full dataset would be too extensive too present and too time consuming to calculate.

The classification models are compared using ROC curves after 10-fold cross validation. The model with the highest area under the curve (AUC) is assumed to perform the best. Each model is trained and tested using three sets of features. The first feature set is the complete set of ~8.000 similarity features. This gives us a baseline for the possible performance that can be achieved. The second and third feature sets are subsets reduced with Principal Component Analysis (PCA) to assess the classifier behavior on smaller and more feasible feature sets. In particular, the second feature set is reduced to 1.000 features with linear-kernel-based PCA [50] and then further reduced to the third subset using common PCA with eigenvalue decomposition while retaining 95% of the variance [51]. The intermediate kernel-based PCA step is necessary as calculating the eigenvalue decomposition on the full feature set could not be calculated within a feasible amount of time.

All machine learning is done with RapidMiner 7.4 [52] and Weka 3.8 [53]. Used settings are stated in Table 13. The resulting ROC curves can be seen in Figure 43 to Figure 45.

Table 13 - Classifier settings

Classifier	**Implementation**	**Settings**
Naïve Bayes	RapidMiner Naive Bayes	Default
Random Forest	RapidMiner Random Forest	50 trees
SVM	RapidMiner SVM	Linear (dot) kernel
Logistic Regression	Weka SimpleLogistic	Default
Neural Network	RapidMiner Deep Learning	Default (2 hidden layers with 50 nodes each)

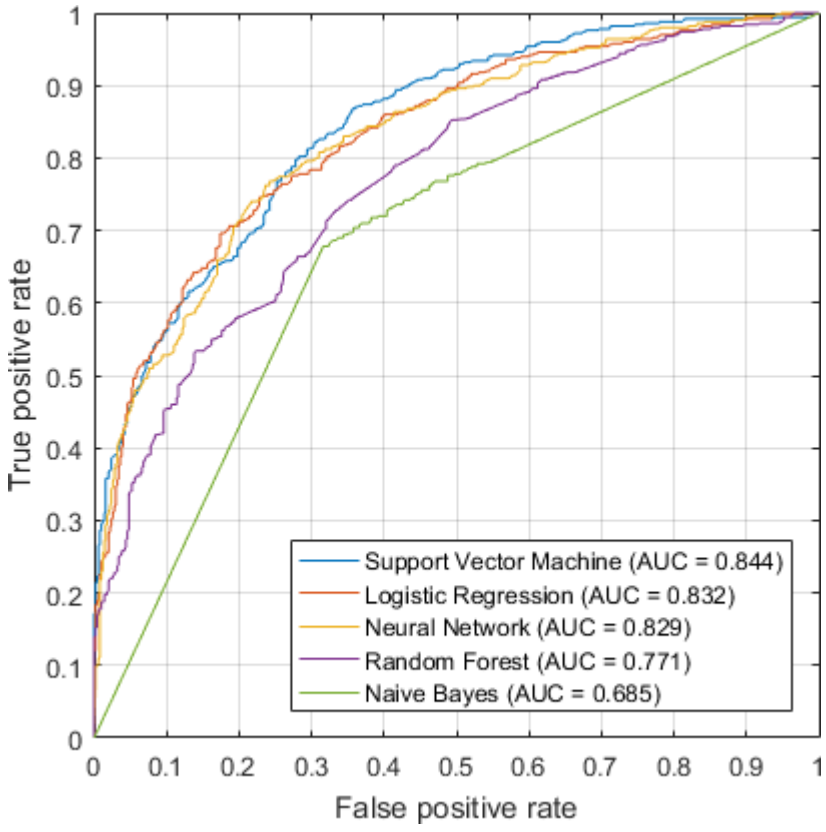

Figure 43 – ROC curves for different classifiers on dataset with all features

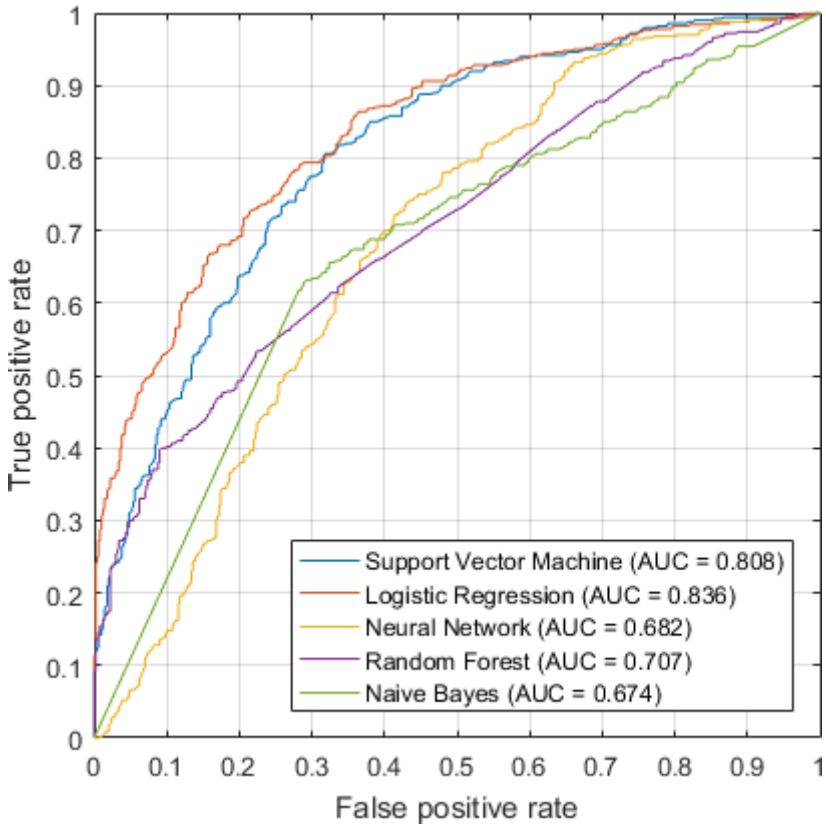

Figure 44 - ROC curves based on feature set reduced with kernel-based PCA

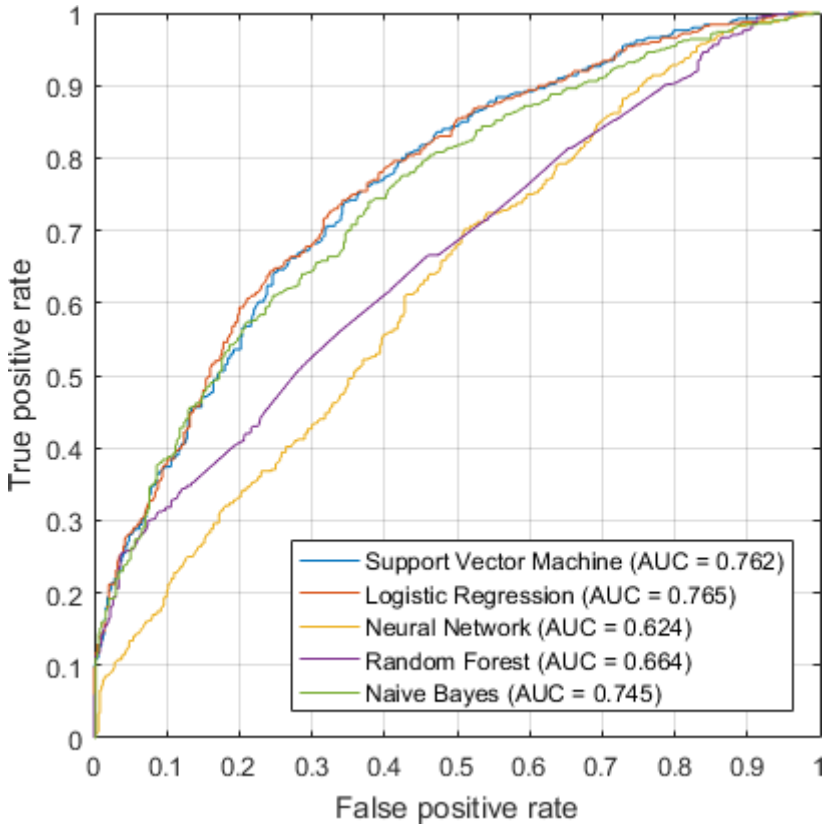

Figure 45 - ROC curves based on feature set reduced with standard PCA

Figure 43 shows that logistic regression, support vector machines and neural networks perform almost equally well when using the full feature set. In the reduced datasets however, the neural network performs significantly worse while logistic regression and SVM are able to keep a good performance. While not shown, changing the number of hidden layers and nodes per layer or the number of training epochs did not help to restore the performance of neural networks for the reduced datasets.

Random forest is clearly not a good choice for the classification problem at hand as it consistently performs in the lower range. Naïve Bayes also performs consistently lower than other classifiers, especially in the high dimensional data. Only on the smallest dataset, Naïve Bayes is able to perform well. This observation is probably the result of the limitation of Naïve Bayes of handling only independent distributions well. In the larger feature subsets, there are many correlating features.

Based on these results logistic regression and support vector machines appear to be both good choices. Logistic regression however has a slightly more consistent top ranking performance and is also faster to execute. Therefore, logistic regression will be used as the classifier of choice for the following steps.

6.3 Feature Selection

In Chapter 4 ~11.000 differently configured similarity features have been selected for potential usage in the refining classification model. This was reduced to ~8.000 features after removing features with a variance of less than 1%. While in theory it would be possible to use all these features in the chosen classification model, it is neither feasible nor benefits the classification performance. An estimate of maximum feature size per product in Chapter 3.10 has shown that only a handful of features may eventually be used if those features ought to be stored in fast RAM. However, memory is not the only limit for feature selection. Using only a small subset of the features has also the advantage that the resulting models are easier to interpret, can be trained and executed faster and are less prone to overfitting.

Many of the features strongly correlate as was shown with the PCA during the model selection. While keeping 95% of the variance it was possible to reduce the full attribute list down to 12 features. However, these 12 principal components are not a useable feature subset for a scalable entity matching approach because calculation of the principal components requires to first calculate all the underlying features. Instead, it is necessary to find a subset of raw features that performs similarly well.

In order to find such a subset, the space of all possible feature subsets has to be searched. However, an exhaustive search is unfeasible, thus an approximate generalizing feature selection method is required. A common approximation method for finding smaller feature subsets is forward feature selection [54]. It is a greedy method that only uses the subset of the top-k highest ranking features assuming that the most important features are part of that well-performing subset. The method than builds increasingly bigger subsets, always using the top-k best performing feature subsets from the previous round until the performance increase falls below a certain threshold. The subset with the best error-rate is returned.

The evaluation of the feature subsets, including the initial ranking of each single feature, can be done in different ways e.g. using information gain, correlation measures or wrapper methods [55] which use the classification performance of a model as a scoring measure. Particularly the latter approach tends to give better results and is also chosen for this research.

While the forward feature selection method has been shown to be less prone to overfitting, it is possible to further test the generalizability of the feature selection by repeating the selection process using cross validation. Each fold will result in its own feature subset and if a certain feature is repeatedly part of the best subset then this is an indicator for a well generalizing feature.

For the feature selection, a random stratified subset of 1.000 samples is used instead of the full dataset. Again, this decision has been made due to resource limitations. The initial ranking of features is calculated using Weka's ranker search method. One logistic regression model is trained and tested for each feature separately. The resulting average AUC after 5-fold cross validation is used to rank the features. The whole ranking process is additionally done within a 10-fold cross validation process to estimate the mean rank and ranking for each feature.

The similarity features can be grouped by their based-on product features (e.g. color signatures, as within each group the results naturally strongly correlate. The top 25 ranking features for each of the 12 feature groups are given in Table 22 in the appendix.

The strongest but also largest features with the best classification performance are color histogram based similarity features. Even standalone they achieve striking AUC scores up to 0.76, similar to the best performing models using the PCA reduced set from the previous section. This proves that more features are not always better. It also strikes that the best performing color histogram similarity features all use the HSV color space and the Hellinger distance. Apparently, this is a combination that works well together.

The color histogram features are followed by color signature, Gabor histogram, gradient signature/histogram and name based similarity features. The best performing features from these groups all perform equally well with standalone AUC scores around 0.7. Description based features perform slightly worse than name based features with AUC scores around 0.69. This is probably due to the higher noise level and because descriptions are not always present. The best LBP histogram, grayscale and edge map statistics based features achieve AUC scores of less than 0.67, indicating that matching based on texture/shape is less robust than using color or text data. Aspect ratio and more fine-grained category similarity features provide the smallest merit with AUC scores of less than 0.58. This might indicate that the simple category filter used in the preselection step is already close to optimal.

In general, it can be observed that tf-idf based comparison gives better results than direct string comparison, with n-grams of stemmed words performing the best. In most cases, basic string normalization such as downcasing tends to improve the performance of the string comparison features. Also gradient signature similarity performs slightly better than gradient histograms. This is opposite to color based features where signatures perform worse than histograms. The reason for this might be the dimensionality of the data with colors having more dimensions than gradient data.

It also strikes that Cie L*a*b* tends to be a more suitable color space for clustering while HSV is better suited for counting in histograms. This might be due to Cie L*a*b* being optimized for measuring color distances but not having possible values in each bin across the channels.

It is expected that using a combination of one or more features from each feature group should result in a similar performance as using the 12 principal components. Therefore these $12 * 20 = 240$ highest ranking features are selected for further subset calculations using Weka's linear forward selection method with the same evaluation settings as used for the initial ranking.

The resulting best subsets consisted between 18 and 33 features with AUCs between 0.887 and 0.896. Table 14 shows all of the features that were a member of the best performing subset in at least 50% of the validation folds.

Table 14 – Most frequent members of best feature subset

Feature	**Percentage of folds with membership in best subset**
aspect_ratio_euclidian	90%
categories_common_ancestor	50%
color_histogram_hsv_5_local_5_hellinger	70%
description_normalized_letter_2_grams_tfidf_angular	80%
description_normalized_word_stemmed_3_grams_tfidf_angular	90%
description_pair_distance	50%
description_word_3_grams_tfidf_angular	70%
gabor_histogram_6_2_stddev_correlation	90%
gabor_histogram_9_2_stddev_euclidian	50%
lbp_histogram_3_12_local_3_chi_square	50%
name_longest_subsequence	80%
name_normalized_pair_distance	60%
name_normalized_word_stemmed_1_grams_tfidf_angular	90%
name_pair_distance	50%

It strikes that almost none of the best performing standalone features from each feature group is also a consistent member of the best subset. Other, slightly lower ranking features are found instead. This is to be expected, however, as high standalone performance is not a guarantee for high performance in combination with other features but just an indicator.

Also note, that some of these features still have strong correlation, hence unnecessarily bloating up the feature vector. For descriptions there are two frequently appearing similarity features based on 3-grams of words. The feature based on normalized and stemmed words occurs more often than the one using raw word data. Gabor histograms are also represented twice. The variant with 6 Gabor filter orientations seems to be more stable than the variant with 9 orientations. Last but not least, the pair distance feature appears twice for names, once for normalized and once for unnormalized name data with the normalized one slightly performing better. Removing the three less stable features reduces the number of features down to 11 with a total size of less than 256 kb per product. Only these 11 features will be used for further experiments.

6.4 Performance on Full Dataset

With a massively reduced feature set it becomes feasible to train and test the logistic regression classifier on the full dataset of ~18.000 samples. This set is split into 16.000 samples for training and 2.000 samples for testing. The 1.000 samples used previously for classifier and feature selection are included in the training set. The left-out dataset does not include any previously used samples to ensure completely unseen data.

First the classification performance on the 16.000 sample dataset is measured using 10-fold cross validation. Finally, a logistic regression model is trained using all 16.000 samples and applied to the left-out test set. Both resulting ROC curves, are given in Figure 46 .

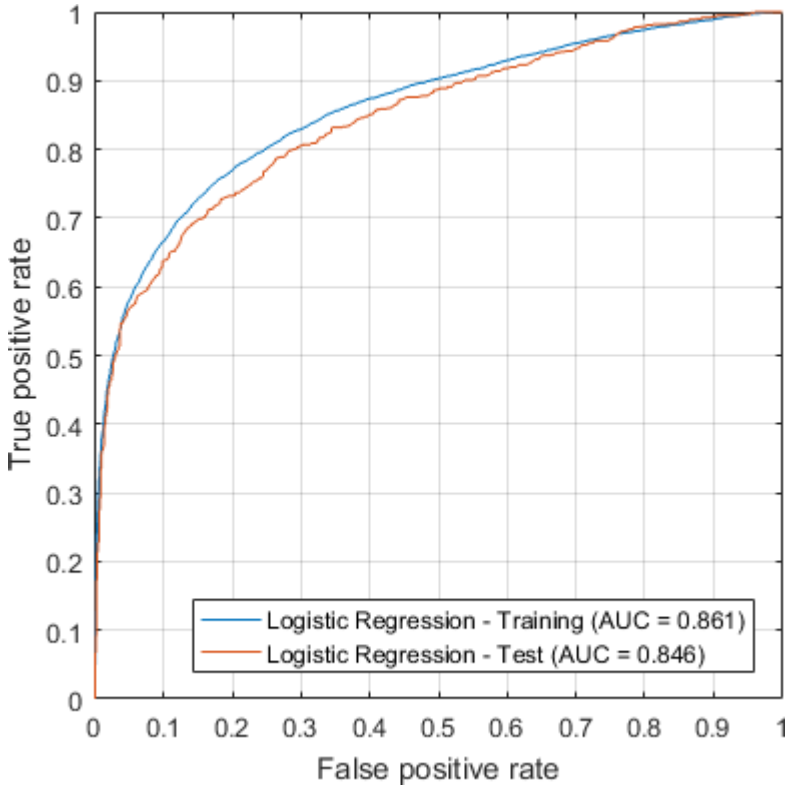

Figure 46 – ROC performance on left-out test set compared to training set

The resulting ROC curve from the left-out test set closely resembles the ROC curve of the cross validation on the training set, indicating that the trained model indeed generalizes well. It also shows that the reduced feature set of just 11 features is sufficient to reach and even surpass the performance of a classification model using all 8.000 similarity features.

In order to assess the abilities and shortcomings of the final model, Table 15 to Table 18 show (mis)classification results with high confidence scores.

Table 15 - False positive examples

Shirt Lange Mouw Vingino hanteert de volgende maten: 2=92; 3=98; 4=104; 5=110; 6=116; 8=128; 10=140; 12=152; 14=164; 16=176. Fabrikantkleur: Dark Grey. 	**Shirt Lange Mouw** De opdruk heeft een vintage-look. Vingino hanteert de volgende maten: 2=92; 3=98; 4=104; 5=110; 6=116; 8=128; 10=140; 12=152; 14=164; 16=176. Fabrikantkleur: Dark Grey. 	0.948
CW1006 Riem CW1006 Riem Cowboysbelt Riem 	**Leren Riem** Blauwe leren riem met stoere 'damaged' zilveren studs van Cowboysbelt 	0.907

NU KORTING Zijden Jurk Geraffineerd doorgestikt! Met carmenhals. Ca. 98 cm lang. Aansluitend model. Zuivere zijde. Voering: 100% acetaat. Wasadvies: stomerij. 	**Zijden Jurk** Geraffineerd doorgestikt! Met carmenhals. Ca. 98 cm lang. Aansluitend model. Zuivere zijde. Voering: 100% acetaat. Wasadvies: stomerij. 	0.906

The first false positive example shows two children sweaters with the same basic gray color and shape and very similar name and descriptions. Only the color accents and prints differ. The main reason why these were incorrectly classified as identical are matches on the size mapping terms in the description. Those are recognized as key terms in both tf-idf vectors.

The second false positive example shows two belts with similar name and description and the same design but different coloring. The many round buttons are strongly reflected in the Gabor filter responses which nudged the model into classifying this sample as identical.

The last false positive example might actually not be a clear false positive. Name, description and model of the product match nearly 100%. Only the coloring is slightly different which might also be the result of different lighting. These are edge cases which probably will remain uncertain unless more information is available about the products.

Table 16 - False negative examples

Rio-Slip Sierkraaltjes Voor en achter van transparante kant. Kralendetails opzij. Achter met sexy hals en kralendetail. 	**Slip Black/Creme** LASCANA Slip black/creme Kleding bij Zalando NL | Materiaal buitenlaag: 79% polyamide, 12% elastaan, 9% katoen | Kleding nu zonder verzendkosten bij Zalando NL bestellen! 	0.961
T-Shirt Vermont 2-Pack Set van twee T-shirts van Alan Red, model Vermont. Rechte pasvorm, v-hals met smalle boord en korte mouwen. 	**2-Pack Oklahoma Stretch** Alan Red 2-Pack Oklahoma Stretch 	0.957

Horloge Dames horloge, typenummer 358SGSCD, van Skagen Denmark met een zilverkleurige mesh horlogeband en een goudkleurige horlogekast. Het horloge heeft een, voor dit merk kenmerkende, ultra-dunne horlogekast.  	**Horloge Freja 358SGSCD** Skagen Denmark Freja Dameshorloge met steentjes. Dit Skagen Denmark dameshorloge heeft een zilverkleurige mesh horlogeband. Het horloge heeft een, voor dit merk kenmerkende, ultra dunne goudkleurige horlogekast. Het horloge wordt geleverd in een luxe bewaarverpakking. Model: 358SGSCD  	0.933

The first false negative example shows the effect of incomplete image segmentation and difference in image perspective. Name and description also have very little in common. Thus, there are no clear hints, that this is the same product. Yet, as a human one can be quite certain that these are the same product by recognizing the very similar design and imagining how the pictured parts fit together.

The second false negative example actually reveals a small mistake in the training data. The color, form, brand and even the number of products match completely but one is the model ‘Vermont’ while the other is called ‘Oklahoma’.

The third false negative example is the result of different lighting and zoom. In this case the color seems different but especially for metals this is often the result of different light reflection. In combination with the zoom the local binning of colors will not match. Based on the model name mentioned in the description it is however obvious that both products are identical.

Table 17 - True positive examples

Shirt Korte Mouw Op de linkerschouder zitten twee drukknoopjes. Fabrikantkleur: White. 	**Shirt Korte Mouw** Op de linkerschouder zitten twee drukknoopjes. Fabrikantkleur: White. 	0.997
Horloge Een dames horloge van Danish Design uit de Brandfield collectie. Het model is uitgevoerd met een horlogeband van titanium, een ronde zilver- en goudkleurige kast van titanium met mineraalglas. 	**Horloge** Een dames horloge van Danish Design uit de Brandfield collectie. Het model is uitgevoerd met een horlogeband van titanium, een ronde zilver- en goudkleurige kast van titanium met mineraalglas. 	0.996

STAN SMITH Sneakers Laag White/Green adidas Originals STAN SMITH Sneakers laag white/green | materiaal buitenlaag: leer, voering: imitatieleer/ textiel, zool: kunststof, binnenzool: textile 	STAN SMITH Sneakers Laag White/Green adidas Originals STAN SMITH Sneakers laag white/green | materiaal buitenlaag: leer, voering: imitatieleer/ textiel, zool: kunststof, binnenzool: textiel 	0.993

Most true positives with high confidence show the same characteristics like example 1 and 2. They are matches between webshops that use the same product image and same name and description. These pairs where supposed to be excluded from the training data using the proposed image identifier approach for deduplication from Chapter 3.9. However, in the depicted cases the images were slightly tempered with, e.g. by increasing brightness or adding shadow effects which lead to different image ids.

The third true positive example also has the exact identical name and description but a slightly different image and is also correctly recognized as a strong true positive match.

Table 18 - True negative examples

Slip 95-5 Super Mini De Schiesser 95-5 Super mini slip zwart is een comfortabele herenslip met een elastische taileband. Deze slip van de 95-5 serie is gemaakt van hoogwaardig stretch katoen voor een prettige pasvorm.Stofsamenstelling:95% katoen 5% elastan Artikelnummer:205421-000 	**ACTIE 2-Pack Wijde Boxershorts** Heren Boxershort van Schiesser kleur RoodWit materiaal Katoen 	0.995

Colshirt U kunt kiezen uit een veelvoud aan basic modellen - in 15 trendy kleuren. Voor uw actuele garderobe zit er zeker een juist exemplaar bij! Verwerkt in een prettige, gemakkelijk te onderhouden katoenmix, met Elastan voor een perfecte pasvorm! Van 95% katoen, 5% Elastan. Lengte ca. 64 cm. Aansluitend model. Wasbaar. 	**Colshirt** Iets getailleerd, perfect passend. Aansluitend model. Shirts van Tactel voelen zacht aan, zijn kreukvrij, kleurecht en zeer gemakkelijk in onderhoud. Lengte ca. 60 cm. Aansluitend model. Van 90% polyamide, 10% Elastan (Tactel). Wasbaar. 	0.987
Tricot-Kniekousen Tricot-kniekousen 'Wool Balance' van Falke. Warme pluchezool. Bijzonder fijne scheerwol in de schacht zorgt voor aangenaam draagcomfort. Drukvrij, met de hand afgehecht teenstuk. 60% scheerwol, 40% polyamide. 	**PURE SHINE Panty Black** Falke PURE SHINE Panty black Kleding bij Zalando NL | Materiaal buitenlaag: 93% polyamide, 7% elastaan | Kleding nu zonder verzendkosten bij Zalando NL bestellen! 	0.983

The first true negative example shows a clear mismatch between two types of man underwear. The reason why these two products were compared in the first place is ambiguity in the categorization and could be prevented with better categorization.

The second true negative example shows two turtlenecks with identical names and slightly similar description but clearly different coloring and texture. This is also correctly reflected in the color and Gabor filter based similarity features.

The third true negative example shows the advantage of considering textual and image data at the same time. Purely based on the image, both products are actually quite similar. However, the name and descriptions have clear distinctive terms that do not match, hence resulting in a classification as different.

6.5 Scalability and Feasibility

To assess the scalability and feasibility of the whole entity matching approach, including human verification, the results of all drilldown steps have to be taken into account.

As shown in Chapter 5.1 the preselection method has the most beneficial impact on the scalability of the approach. The simple clustering splits all possible products into much smaller sets and with it, the number of possible pairs. In the exemplary case of Fashion Evolution with 1.5 million products, the dataset is split into 20.400 clusters with an average of 49 products and 3.580 possible pairs each (73 million pairs in total). As each set can be treated separately, it is also possible to distribute the comparison over multiple machines and process them concurrently.

In each of the clusters, the similarity features for the ~3.600 possible pairs can be quickly calculated. Each of the suggested similarity features takes only a few milliseconds to calculate on an average machine as shown in Chapter 4.7 and retrieval of the product features should also not be a problem as the selected features of 49 products should fit easily in fast RAM.

The only possible bottleneck is the human verification step to reach near-perfect accuracy. From the manual dataset collection phase at Fashion Evolution, it has been estimated that humans achieve an accuracy of around 99% when a product pair is unanimously classified by two people. Experience has also shown that on average it takes a person only 2 seconds to decide if two products are identical or different. The learning period for this classification problem is also particularly short. Usually a handful of example edge cases is sufficient to prepare someone for this task.

If we carefully assume that 25% of all offered fashion products (375.000) are sold at more than one webshop, those 375.000 products should be distributed over the preselected clusters. Assuming that products are sold on average in 3 different webshops, there are about 125.000 unique product entities. With on average 3 matching pairs each, there are 375.000 identical product pairs in the total set of 73 million pairs after preselection - a 0.5% chance of randomly selecting a true positive match.

If 95% of all true positive matches ought to be found, the ROC curve of the final model (Figure 46) shows that this choice comes with the trade-off of selecting 70% false positives. Note however, that it is not necessary to find all true positive pairs. If it's found that A is identical to B and B is identical to C than by transition A is also identical to C and has not to be verified separately. So, a lower recall that ensures a high chance of finding at least 2 out of the 3 matches is acceptable as well. For example, in order to reach at least 95% recall for products that are sold at 3 or more webshops, a pair-wise recall of 86.5% is sufficient:

$$p^3 + 3p^2(1-p) \geq 0.95$$

$$p \geq \mathbf{0.865}$$

Note, that this decision reduces the recall for products sold at only 2 webshops (i.e. 1 matching pair) to 86.5%. However, the lower recall comes with the advantage of a lower false positive rate of 40%. This has a significant impact of the total number of pairs to be verified by humans. Applied to the dataset of Fashion Evolution with 73 million preselected product pairs and an estimated number of 375.000 matching pairs, the total number of pairs to be verified by humans can be estimated as follows:

0.375 million matching pairs * 86.5% recall = 0.324 million true positive pairs

(73 million – 0.375 million) non-matching pairs * 40% fpr = 29.05 million false positive pairs

0.324 + 29.05 = **29.374 million positive pairs to be verified**

Assuming that actually 2 people are required for proper verification of each pair, the actual number of verifications doubles to ~60 million. With an average classification time of 2 seconds, it would take about 60 million * 2 seconds = 3.8 years of non-stop work to classify the whole dataset. Even with services such as Amazon Mechanical Turk [47] this is not a feasible option.

In order for human interaction to be feasible, it should not exceed more than something in the order of 300.000 pairs over the time span of 1 month. With prices around 1 cent per verification in the cloud, this amounts to 3.000$ per month. However, this limit would require a false positive count of less than 150.000 or 0.2%. This would cause the recall to drop to almost zero as well.

Note however, that these calculations are done for the onetime problem of classifying the whole dataset. Assuming the initial classification has been done, only new products and product updates that affect one of the relevant product features have to be (re)compared. Within the budget of 150.000 pairs per month or 5.000 pairs per day, the maximum supported number of daily new/updated products can be estimated as follows using the same assumptions as above:

> Let n be the maximum supported number of daily new/updated products.
>
> On average one third of the products has no possible pairings after preselection. For the rest each product has on average 49 possible partners in one of the preselected clusters. This is true as long as there is a good chance that all new products are distributed evenly among the preselected clusters, i.e. when n is much smaller than the number of clusters. Together this results in $\frac{2}{3} * 49 * n = 32.67\, n$ product pairs that are estimated to be introduced with the daily updates.

25% of the products are expected to exist on average at three webshops. Assuming the other two matching product entities are not part of the update, this results in $25\% * 2 * n = 0.5\, n$ identical matches.

With a desired true positive rate (recall) of 86.5% and consequently 40% false positive rate, this results in $0.5\, n * 86.5\% + (32.67\, n - 0.5\, n) * 40\% = p$ pairs per day to check.

Rearranging this formula results in $\boldsymbol{n \approx \frac{p}{13.3}}$.

Hence, in case of Fashion Evolutions dataset and a daily budget of p = 5.000 pairs the proposed method is estimated to be able to process about $n = 375$ new or significantly updated products per day. Depending on the time of year, e.g. during season changes where many new seasonal products are introduced, this limit is still likely to be insufficient for Fashion Evolution. For smaller datasets, however, the limit will be much greater due the number of possible pairs per preselected cluster dropping quadratically. Also there is always the possibility to reduce the number of pairs to check by choosing a higher classification threshold, i.e. reducing the desired recall. For example a recall of 75% (20% false positive rate) or 65% (10% false positive rate) can increase n by a factor of 2 and 4 respectively.

6.6 Conclusion

8.000 differently configured similarity features and 18.000 labeled samples of product pairs collected from real world data of Fashion Evolution were used to find the best performing classifier and feature subset for the refining reduction step. Throughout multiple experiments, logistic regression models have been shown to perform the best and most stable throughout different feature subsets. In particular, the feature set could be drastically reduced in size to just 11 features while at the same time reducing overfitting of the model.

On the left-out testset the best refining model has been shown to filter out 60% of incorrect matches while retaining 95% recall, effectively reducing the search space after preselection to 30 million product pairs. Unfortunately, this remaining number is far from feasible for integrating human feedback to solve all uncertainty. However, it is shown that the approach can still be feasible for smaller datasets and processing daily product updates. For larger datasets it is also useful when finding only some of the identical matches is acceptable as well.

7 CONCLUSION

This research focused on matching identical fashion product entities between different webshops without depending on often missing or unreliable globally unique identifiers. Based on the intuitive human approach to finding identical fashion products, it was suggested to use a combination of distinctive name, color, shape and texture features instead. However, several concerns existed for this approach, such as varying data quality, scalability issues and high accuracy expectations. In the course of this research all these concerns were addressed step by step, following generally proven methods in entity matching approaches.

Q1: *How can product data quality differences be reduced?*

In the first part of this thesis several normalization approaches have been presented that are used in practice by Fashion Evolution for dealing with data quality issues in the fashion retail domain. Often basic string normalization and mapping approaches are shown to be sufficient to overcome most of the differences between webshops. However, not only the quality but also the availability is important. Meta information such as the color and material is valuable information but was found to be rather noisy. Most of the product features are only consistently available in the product images. A novel image segmentation approach was specifically designed as a first step normalization step towards extracting those image features.

Q2: *What additional robust and compact product features can be extracted?*

Knowing what data is available and what quality can be achieved using basic normalization methods, the focus was shifted to extracting additional data that was not yet readily available. For textual metadata, the additional information was rather limited. Tf-idf models were found to identify key terms in product names and descriptions but most additional information can be extracted from the product images. Several methods for extracting color, shape and texture features were found in the literature. Furthermore, the possibility to add local sensitivity to the image features for even more distinctiveness has been discussed. About 1.500 variants of product features extraction methods are proposed and analyzed in terms of runtime performance and space requirements to assess their suitability for a scalable approach. Last but not least, an accurate but also scalable approach for assigning perceptual identifiers to images has been proposed, allowing to match identical images despite different URLs and image resolutions.

Q3: *How can similarities between product features be quickly quantified?*

In order to quantify the similarity or distance between the product features, literature research has been done to find applicable measures. Most of the product features are histograms or clustered feature signatures, and for those especially many options were found to compare them, ranging from simple Euclidian distance to more dynamic measures like EMD. In total ~11.000 different variants of similarity features were proposed and also analyzed in terms of runtime performance.

Q4: *How can simple product features be used to quickly reduce the set of possible matches?*

Based on the research of product and similarity features the first step of a typical drilldown matching approach was designed. By exploiting domain specific knowledge, it was found that a simple clustering on a combination of brand, category, target group and webshop is very efficient. It reduces the number of pairs to be processed by the next drilldown step from 1.125 trillion to ~73 million in a typical dataset of 1.5 million products. Other filters such as similar names and color features were also discussed but found to add little improvement while significantly increasing the time complexity of the preselection.

Q5: *How can more computationally expensive similarity features be used to further reduce the set of possible matches?*

After simple preselection, the second more sophisticated drilldown step was investigated. Matching based on GTIN, identical or shared image hashes and MPN are briefly described as refining options for quickly finding positive matches for completely new datasets but since they are not applicable to all product pairs the main focus was put on the development of a general refining model. Knowing that the remaining candidates are already matched by brand, category and target group the only reliable and distinctive similarity features left for a more thorough classification were color, shape and texture information extracted from the images as well as the product name and description. Five machine learning optimizable classification models are suggested which are able to combine these features.

Q6: *How much human interaction is required for reaching near perfect accuracy?*

8.000 differently configured similarity features and 18.000 labeled samples of product pairs collected from real world data of Fashion Evolution were used to find the best performing classifier and feature subset for the refining reduction step. The experiments show that the best model can filter out 60% of incorrect matches while retaining 95% recall, effectively reducing the search space after preselection to 30 million product pairs. Unfortunately, this remaining number turns out to be far from feasible for integrating human feedback to solve all uncertainty. However, the approach can still be feasible for smaller datasets and processing daily product updates.

7.1 Discussion

The results of the experiments and data quality analysis in this research are mainly based on the dataset from Fashion Evolution so it might be possible that the results are not representative for other fashion product datasets. Unfortunately, there seems to exist no benchmark data to compare the results with. However, most of the assumptions made are also expected to be true for other datasets.

For example, most fashion product search engines have to process the same set of brands, categories and target groups as these are internationally shared concepts. The only language specific feature used in this research are the text similarity features but with appropriate document frequency dictionaries in other languages it is not expected to cause significant performance differences. Most of the impact comes from the image features and these are also similar around the world.

Furthermore, the size of the dataset is representative for most general fashion product search engines. Specialized services for certain fashion categories or small countries are naturally much smaller but most European services range between 500.000 and 1.500.000 products. Probably the largest fashion search engine in Europe is fashiola.de with 2.5 million products in their database, 1.66 times as many, hence probably also $1.66^2 = 2.8$ times as many possible matches to deal with.

7.2 Future Work

While the current proposed approach has been shown to be only feasible for smaller datasets, there are still a few ways left to further improve the proposed matching approach and make it more feasible in the future.

First of all, most of the presented subsolutions for feature extraction and feature similarity were chosen because they were well established and tested. However, new and alternative ways are likely to exist. For example meta classification techniques such as boosting (AdaBoost [56], LogitBoost [57]) have not been investigated yet. Especially the recent progress of deep convolutional neural networks (CNN) for image analysis and comparison [17] is certainly worth to explore as another drilldown layer before user integration, although it might require a larger dataset to train with. Automatic clustering methods for better preselection also have potential to significantly reduce the search space and hence the human work load.

Furthermore, there are steps not yet optimized for the fashion product domain. For example, more category specific feature extraction methods might lead to better results. Imagine matching two shoes where different parts of the products, e.g. the sole, laces and shaft, are identified beforehand and compared separately. This would require the development of more sophisticated segmentation methods.

Additional category specific classification models are also worth exploring. For example, color and shape is probably less important for jeans since they vary little in those aspects. The name and description might be much more important. On the other hand, color and shape is very important for shoes. These differences can be better handled by separate models instead of a single generic model although care has to be taken. A classification model for every single subcategory requires a lot of data and work and increases the risk of overfitting

As stated in the validation chapter there does not yet exist a benchmark dataset for this research area. Making the dataset from this research available to other researchers is likely to open up completely new insights and ideas.

It might also be worth to investigate if the proposed approach can be applied to other markets where global identifiers are missing or unreliable as well.

8 REFERENCES

[1] H. Köpcke and E. Rahm, "Frameworks for entity matching: A comparison," *J. Data and Knowledge Engineering,* vol. 69, no. 2, pp. 197-210, 2010.

[2] W. E. Winkler, "Overview of record linkage and current research directions," *US Bureau of the Census,* 2006.

[3] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE Trans. Knowledge and Data Engineering,* vol. 24, no. 9, pp. 1537-1555, 2012.

[4] M. Bilenko, S. Basil and M. Sahami, "Adaptive product normalization: Using online learning for record linkage in comparison shopping," *5th IEEE Int. Conf. Data Mining,* p. 58–65, 2005.

[5] H. Köpcke, A. Thor and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proc. VLDB Endowment,* vol. 3, no. 1-2, pp. 484-493, 2010.

[6] R. Baxter, P. Christen and T. Churches, "A Comparison of Fast Blocking Methods for Record Linkage," *ACM SIGKDD,* no. 3, pp. 25-27, 2003.

[7] M. v. Keulen and A. d. Keijzer, "Qualitative effects of knowledge rules and user feedback in probabilistic data integration," *Int. J. Very Large Data Bases,* vol. 18, no. 5, pp. 1191-1217, 2009.

[8] C. Grana, D. Borghesani and R. Cucchiara, "Class-based color bag of words for fashion retrieval," *IEEE Int. Conf. on Multimedia and Expo,* pp. 444-449, 2012.

[9] C.-H. Tseng, S.-S. Hung, J.-J. Tsay and D. Tsaih, "An efficient garment visual search based on shape context," *WSEAS Trans. on Computers,* vol. 8, no. 7, pp. 1195-1204, 2009.

[10] J.-J. Tsay, C.-H. Lin, C.-H. Tseng and K.-C. Chang, "On Visual Clothing Search," *IEEE Int. Conf. Applications of Artificial Intelligence,* pp. 206-211, 2011.

[11] Q. Chen, J. Li, G. Lu, X. Bi and B. Wang, "Clothing retrieval based on image bundled features," *2nd IEEE Int. Conf. Computing and Intelligent Systems,* vol. 2, pp. 980-984, 2012.

[12] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[13] E. Hsu, C. Paz and S. Shen, "Clothing image retrieval for smarter shopping," *EE368,* 2011.

[14] J. M. dos Santos, J. M. Cavalcanti, P. C. Saraiva and E. S. de Moura, "Multimodal Re-ranking of Product Image Search Results," *Proc. 35th European Conf. IR Research,* pp. 62-73, 2013.

[15] S. Liu, Z. Song, G. Liu, C. Xu, H. Lu and S. Yan, "Street-to-Shop: Cross-Scenario Clothing Retrieval via Parts Alignment and Auxiliary Set," *IEEE Conf. Computer Vision and Pattern Recognition,* pp. 3330-3337, 2012.

[16] M. H. Kiapour, "Where to Buy It: Matching Street Clothing Photos in Online Shops," *Proc. IEEE Int. Conf. Computer Vision,* pp. 3343-3351, 2015.

[17] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen and Y. Wu, "Learning fine-grained image similarity with deep ranking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1386-1393, 2014.

[18] X. Lin, B. Gokturk, B. Sumengen and D. Vu, "Visual search engine for product images," *Proc. SPIE 6820,* 2008.

[19] O. Jundt and M. v. Keulen, "Sample-based XPath Ranking for Web Information Extraction," *Proc. 8th Conf. European Society for Fuzzy Logic and Technology,* 2013.

[20] Q. Wei, "Converting 2d to 3d: A survey," 2005.

[21] V. Kwatra, A. Schödl, I. Essa, G. Turk and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," *ACM Trans. Graphics,* no. 22, pp. 277-286, 2003.

[22] J. Canny, "A Computational Approach To Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, p. 679–698, 1986.

[23] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Computer Vision,* no. 57, pp. 137-154, 2004.

[24] N. Hasan, A. Hamouda, T. Deif, M. A. El-Saban and R. Shahin, "Using Skin Segmentation to Improve Similar Product Recommendations in Online Clothing Stores," *VISAPP,* no. 1, pp. 693-700, 2013.

[25] R. S. Hunter, "Photoelectric color difference meter," *Josa,* pp. 985-993, 1958.

[26] OpenCV. [Online]. Available: http://opencv.org/. [Accessed April 2016].

[27] Y. Rubner and C. Tomasi, Perceptual metrics for image database navigation, Springer, 2000.

[28] C. A. Poynton, "Rehabilitation of gamma," *Photonics West'98 Electronic Imaging,* pp. 232-249, 1998.

[29] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Information Theory,* vol. 8, no. 2, pp. 179-187, 1962.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 886-893, 2005.

[31] H. Scharr, Optimal Filters for Extended Optical Flow, Berlin, Heidelberg: Springer, 2007.

[32] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 971-987, 2002.

[33] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Proc. IEEE Int. Conf. Systems, Man and Cybernetics,* pp. 14-19, 1990.

[34] D. G. Lowe, "Object recognition from local scale-invariant features," *Proc. 7th IEEE Int. Conf. Computer Vision,* vol. 2, pp. 1150-1157, 1999.

[35] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, "Speeded-up robust features (SURF)," *J. Computer Vision and Image Understanding,* vol. 110, no. 3, pp. 346-359, 2008.

[36] C. Zauner, *Implementation and benchmarking of perceptual image hash functions,* 2010.

[37] R. Brunelli, Template Matching Techniques in Computer Vision: Theory and Practice, Wiley, 2009.

[38] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Processing,* vol. 13, no. 4, pp. 600-612, 2004.

[39] W. Cohen, P. Ravikumar and S. Fienberg, "A comparison of string metrics for matching names and records," *KDD Workshop Data Cleaning and Object Consolidation,* vol. 3, pp. 73-78, 2003.

[40] E. Moreau, F. Yvon and O. Cappé, "Robust similarity measures for named entities matching," *Proc. 22nd Int. Conf. Computational Linguistics,* no. 1, pp. 593-600, 2008.

[41] Z. Wu and M. Palmer, "Verbs semantics and lexical," *Proc. 32nd Ann. Meeting Assoc. Computational Linguistics,* pp. 133-138, 1994.

[42] H. Nguyen and H. Al-Mubaid, "New ontology-based semantic similarity measure for the biomedical domain," *IEEE Int. Conf. Granular Computing,* pp. 623-628, 2006.

[43] Y. Li, Z. A. Bandar and D. McLean, "An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources," *IEEE Trans. Knowledge and Data Engineering,* vol. 15, no. 4, pp. 871-882, 2003.

[44] Y. Rubner, C. Tomasi and L. J. Guibas, "The Earth Mover's Distance as a Metric for Image Retrieval," *Int. J. Computer Vision,* vol. 40, no. 2, pp. 99-121, 2000.

[45] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 4, pp. 509-522, 2002.

[46] amatch, "Approximate Matching Extension for Ruby," [Online]. Available: http://flori.github.io/amatch/doc/index.html. [Accessed April 2016].

[47] Amazon Mechanical Turk, [Online]. Available: https://www.mturk.com/. [Accessed March 2017].

[48] O. Jundt, "Fast Candidate Selection for Fashion Product Data Matching," *Unpublished,* 2014.

[49] J. V. Hulse and T. M. Khoshgoftaar, "Experimental perspectives on learning from imbalanced data," *Proc. 24th Int. Conf. Machine learning,* pp. 935-942, 2007.

[50] B. Schölkopf, A. Smola and K. R. Müller, "Kernel principal component analysis," *International Conference on Artificial Neural Networks,* pp. 583-588, 1997.

[51] S. Wold, K. Esbensen and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems,* no. 2.1-3, pp. 37-52, 1987.

[52] RapidMiner, [Online]. [Accessed March 2017].

[53] University of Waikato, [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed April 2016].

[54] M. Gutlein, E. Frank, M. Hall and A. Karwath, "Large scale attribute selection using wrappers," *IEEE Symp. Computational Intelligence and Data Mining,* pp. 332-339, 2009.

[55] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *J. Artificial intelligence,* vol. 97, no. 1, pp. 273-324, 1997.

[56] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting," *J. Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[57] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: A statistical view of boosting.," *Ann. Statist.,* vol. 28, no. 2, pp. 337-374, 2000.

9 APPENDIX

9.1 Runtime Performance of Feature Extraction Methods

For brevity feature names were combined using regular expressions:

- Histogram features follow the format "<*name*>_<*m*>_local_<*l*>" where *m = #bins per region* and *l = #bins per axis*.
- Signature features follow the format "<*name*>_<*#centroids*>".
- Gabor features follow the format "gabor_(signature|histogram)_<*#orientations*>_<*#scales*>_…"
- LBP histograms follow the format "lbp_<*pixel radius*>_<*#samples*>_…"
- "*statistics*" is a placeholder for mean, standard deviation and normalized 2D image moments ($\nu_{11}, \nu_{20}, \nu_{02}, \nu_{12}, \nu_{21}, \nu_{30}, \nu_{03}$).

Table 19 – Runtime benchmark for product features

Time	Feature
< 100 ms	aspect_ratio (name|description)_letter_(1|2|3)_grams_tfidf (name|description)_word_(1|2)_grams_tfidf (name|description)_word_stemmed_1_grams_tfidf (name|description)_normalized (name|description)_normalized_letter_(1|2|3)_grams_tfidf (name|description)_normalized_word_(1|2)_grams_tfidf (name|description)_normalized_word_stemmed_1_grams_tfidf color_histogram_(rgb|lab|hsv)_(1|3|5|7|9|11|13|15) color_histogram_(rgb|lab|hsv)_(1|3|5|7|9|11|13|15)_local_(2|3|4|5|6|7|8) color_signature_(rgb|lab|hsv)_(2|4|6) color_signature_(rgb|lab|hsv)_(2|4)_local edge_histogram_statistics edge_histogram_statistics_local_(2|3|4|5|6|7|8) gabor_histogram_3_1_statistics_local_(2|3|4|5|6|7|8) gradient_histogram_(3|6|9|12) gradient_histogram_(3|6|9|12)_local_(2|3|4|5|6|7|8) gray_histogram_statistics gray_histogram_statistics_local_(2|3|4|5|6|7|8) lbp_histogram_(1|2|3)_(3|6|9|12) lbp_histogram_(1|2|3)_(3|6|9)_local_(2|3|4|5|6|7|8) lbp_histogram_(1|2|3)_12_local_(2|3)

< 200 ms	(name|description)_word_3_grams_tfidf (name|description)_word_stemmed_(2|3)_grams_tfidf (name|description)_normalized_word_3_grams_tfidf (name|description)_normalized_word_stemmed_2_grams_tfidf color_signature_(rgb|lab|hsv)_(8|10|12|14|16|18) color_signature_(rgb|lab|hsv)_(6|8|10|12|14)_local gabor_histogram_3_(1|2)_statistics gabor_histogram_3_(1|2)_statistics_local_(2|3|4|5|6|7|8) gradient_signature_(2|4|6) gradient_signature_(2|4)_local lbp_histogram_(1|2|3)_12_local_(4|5|6|7|8)
< 300 ms	(name|description)_normalized_word_stemmed_3_grams_tfidf color_signature_(rgb|lab|hsv)_(20|22|24|26|28) color_signature_(rgb|lab|hsv)_(16|18|20|22|24)_local gabor_histogram_3_3_statistics gabor_histogram_3_3_statistics_local_(2|3|4|5|6|7|8) gradient_signature_(8|10) gradient_signature_(6|8)_local
< 400 ms	color_signature_(rgb|lab|hsv)_(30|32) color_signature_(rgb|lab|hsv)_(26|28|30|32)_local gabor_histogram_6_1_statistics gabor_histogram_6_1_statistics_local_(2|3|4|5|6|7|8) gabor_histogram_9_1_statistics gabor_histogram_9_1_statistics_local_(2|3|4|5|6|7|8) gradient_signature_(12|14) gradient_signature_(10|12|14)_local
< 500 ms	gabor_histogram_6_2_statistics gabor_histogram_6_2_statistics_local_(2|3|4|5|6|7|8) gabor_histogram_12_1_statistics gabor_histogram_12_1_statistics_local_(2|3|4|5|6|7|8) gradient_signature_(16|18) gradient_signature_(16|18)_local
< 600 ms	gradient_signature_(20|22) gradient_signature_(20|22)_local

< 700 ms	gabor_histogram_6_3_statistics gabor_histogram_6_3_statistics_local_(2|3|4|5|6|7|8) gabor_histogram_9_2_statistics gabor_histogram_9_2_statistics_local_(2|3|4|5|6|7|8) gradient_signature_(24|26|28) gradient_signature_(24|26)_local
< 800 ms	gradient_signature_(30|32) gradient_signature_(28|30)_local
< 900 ms	gradient_signature_32_local
< 1 s	gabor_histogram_9_3_statistics gabor_histogram_9_3_statistics_local_(2|3|4|5|6|7|8)
≥ 1 s	gabor_histogram_12_2_statistics gabor_histogram_12_2_statistics_local_(2|3|4|5|6|7|8) gabor_histogram_12_3_statistics gabor_histogram_12_3_statistics_local_(2|3|4|5|6|7|8)

9.2 Space Requirements of Extracted Features

Table 20 - Mean product feature sizes in bytes

Size	**Feature**
4 bytes	aspect_ratio
8 bytes	color_histogram_(rgb|lab|hsv)_1 edge_histogram_statistics gray_histogram_statistics
< 32 bytes	name name_normalized color_histogram_(rgb|lab|hsv)_1_local_2 edge_histogram_statistics_local_2 gray_histogram_statistics_local_2 gabor_histogram_3_(1|2)_statistics gabor_histogram_6_1_statistics
< 64 bytes	name_word_(1|2|3)_grams_tfidf name_word_stemmed_(1|2|3)_grams_tfidf name_normalized_word_(1|2|3)_grams_tfidf name_normalized_word_stemmed_(1|2|3)_grams_tfidf

	color_signature_(rgb|lab|hsv)_2 color_signature_(rgb|lab|hsv)_2_local color_histogram_(rgb|lab|hsv)_3 color_histogram_(rgb|lab|hsv)_1_local_3 lbp_histogram_(1|2|3)_3 gradient_histogram_3 gradient_signature_2 gradient_signature_2_local edge_histogram_statistics_local_3 gray_histogram_statistics_local_3 gabor_histogram_3_3_statistics gabor_histogram_6_2_statistics gabor_histogram_(9|12)_1_statistics gabor_histogram_3_1_statistics_local_2
< 128 bytes	name_letter_1_grams_tfidf name_normalized_letter_1_grams_tfidf color_signature_(rgb|lab|hsv)_(4|6) color_signature_(rgb|lab|hsv)_4_local color_histogram_(rgb|lab|hsv)_5 color_histogram_(rgb|lab|hsv)_1_local_(4|5) edge_histogram_statistics_local_(4|5) gabor_histogram_6_3_statistics gabor_histogram_(9|12)_2_statistics gabor_histogram_3_2_statistics_local_2 gabor_histogram_6_1_statistics_local_2 gradient_histogram_6 gradient_signature_(4|6) gradient_signature_4_local gray_histogram_statistics_local_(4|5)
< 256 bytes	name_letter_(2|3)_grams_tfidf name_normalized_letter_(2|3)_grams_tfidf description_letter_1_grams_tfidf description_normalized_letter_1_grams_tfidf color_signature_(rgb|lab|hsv)_(8|10|12) color_signature_(rgb|lab|hsv)_(6|8)_local color_histogram_(rgb|lab|hsv)_7 color_histogram_(rgb|lab|hsv)_1_local_(6|7) color_histogram_(rgb|lab|hsv)_3_local_2 edge_histogram_statistics_local_(6|7|8)

	gabor_histogram_(9|12)_3_statistics gabor_histogram_3_1_statistics_local_(3|4) gabor_histogram_3_3_statistics_local_2 gabor_histogram_6_2_statistics_local_2 gabor_histogram_(9|12)_1_statistics_local_2 gradient_histogram_9 gradient_histogram_3_local_2 gradient_signature_(8|10|12|14) gradient_signature_(6|8)_local gray_histogram_statistics_local_(6|7|8) lbp_histogram_(1|2|3)_3_local_2
< 512 bytes	description description_normalized description_word_1_grams_tfidf description_word_stemmed_1_grams_tfidf description_normalized_word_1_grams_tfidf description_normalized_word_stemmed_1_grams_tfidf color_signature_(rgb|lab|hsv)_(14|16|18|20|22|24) color_signature_(rgb|lab|hsv)_(10|12|14|16)_local color_histogram_(rgb|lab|hsv)_1_local_8 color_histogram_(rgb|lab|hsv)_3_local_3 color_histogram_(rgb|lab|hsv)_5_local_2 gabor_histogram_3_1_statistics_local_(5|6) gabor_histogram_3_2_statistics_local_(3|4) gabor_histogram_3_3_statistics_local_3 gabor_histogram_6_1_statistics_local_(3|4) gabor_histogram_6_3_statistics_local_2 gabor_histogram_9_1_statistics_local_3 gabor_histogram_9_2_statistics_local_2 gabor_histogram_12_2_statistics_local_2 gradient_histogram_12 gradient_histogram_3_local_3 gradient_histogram_6_local_2 gradient_signature_(16|18|20|22|24|26|28|30) gradient_signature_(10|12|14|16|18)_local lbp_histogram_(1|2|3)_6 lbp_histogram_(1|2|3)_3_local_(3|4)

< 1 kB	description_letter_2_grams_tfidf description_word_2_grams_tfidf description_word_stemmed_(2|3)_grams_tfidf description_normalized_letter_2_grams_tfidf description_normalized_word_2_grams_tfidf description_normalized_word_stemmed_(2|3)_grams_tfidf color_signature_(rgb|lab|hsv)_(26|28|30|32) color_signature_(rgb|lab|hsv)_(18|20|22|24|26|28|30|32)_local color_histogram_(rgb|lab|hsv)_(9|11) color_histogram_(rgb|lab|hsv)_3_local_4 gabor_histogram_3_1_statistics_local_(7|8) gabor_histogram_3_2_statistics_local_(5|6) gabor_histogram_3_3_statistics_local_(4|5) gabor_histogram_6_1_statistics_local_(5|6) gabor_histogram_6_2_statistics_local_(3|4) gabor_histogram_6_3_statistics_local_3 gabor_histogram_9_1_statistics_local_(4|5) gabor_histogram_9_2_statistics_local_3 gabor_histogram_9_3_statistics_local_2 gabor_histogram_12_1_statistics_local_(3|4) gabor_histogram_12_3_statistics_local_2 gradient_histogram_3_local_(4|5) gradient_histogram_6_local_3 gradient_histogram_9_local_2 gradient_signature_32 gradient_signature_(20|22|24|26|28|30|32)_local lbp_histogram_(1|2|3)_3_local_(5|6)
< 2 kB	description_letter_3_grams_tfidf description_word_3_grams_tfidf description_normalized_letter_3_grams_tfidf description_normalized_word_3_grams_tfidf color_histogram_(rgb|lab|hsv)_13 color_histogram_(rgb|lab|hsv)_3_local_(5|6) color_histogram_(rgb|lab|hsv)_5_local_3 color_histogram_(rgb|lab|hsv)_7_local_2 gabor_histogram_3_2_statistics_local_(7|8) gabor_histogram_3_3_statistics_local_(6|7) gabor_histogram_6_1_statistics_local_(7|8) gabor_histogram_6_2_statistics_local_(5|6)

	gabor_histogram_6_3_statistics_local_(4|5) gabor_histogram_9_1_statistics_local_(6|7) gabor_histogram_9_2_statistics_local_(4|5) gabor_histogram_9_3_statistics_local_(3|4) gabor_histogram_12_1_statistics_local_(5|6) gabor_histogram_12_2_statistics_local_(3|4) gabor_histogram_12_3_statistics_local_3 gradient_histogram_3_local_(6|7|8) gradient_histogram_6_local_4 gradient_histogram_9_local_3 gradient_histogram_12_local_2 lbp_histogram_(1|2|3)_9 lbp_histogram_(1|2|3)_3_local_(7|8) lbp_histogram_(1|2|3)_6_local_2
< 4 kB	color_histogram_(rgb|lab|hsv)_15 color_histogram_(rgb|lab|hsv)_3_local_(7|8) color_histogram_(rgb|lab|hsv)_5_local_4 color_histogram_(rgb|lab|hsv)_9_local_2 gabor_histogram_3_3_statistics_local_8 gabor_histogram_6_2_statistics_local_(7|8) gabor_histogram_6_3_statistics_local_(6|7) gabor_histogram_9_1_statistics_local_8 gabor_histogram_9_2_statistics_local_(6|7) gabor_histogram_9_3_statistics_local_(5|6) gabor_histogram_12_1_statistics_local_(7|8) gabor_histogram_12_2_statistics_local_(5|6) gabor_histogram_12_3_statistics_local_(4|5) gradient_histogram_6_local_(5|6|7) gradient_histogram_9_local_4 gradient_histogram_12_local_3 lbp_histogram_(1|2|3)_6_local_3
< 8 kB	color_histogram_(rgb|lab|hsv)_5_local_(5|6) color_histogram_(rgb|lab|hsv)_7_local_(3|4) color_histogram_(rgb|lab|hsv)_11_local_2 gabor_histogram_6_3_statistics_local_8 gabor_histogram_9_2_statistics_local_8 gabor_histogram_9_3_statistics_local_(7|8) gabor_histogram_12_2_statistics_local_(7|8) gabor_histogram_12_3_statistics_local_(6|7)

	gradient_histogram_6_local_8 gradient_histogram_9_local_(5|6|7) gradient_histogram_12_local_(4|5) lbp_histogram_(1|2|3)_6_local_(4|5)
< 16 kB	color_histogram_(rgb|lab|hsv)_5_local_(7|8) color_histogram_(rgb|lab|hsv)_7_local_(5|6) color_histogram_(rgb|lab|hsv)_9_local_(3|4) color_histogram_(rgb|lab|hsv)_11_local_3 color_histogram_(rgb|lab|hsv)_13_local_2 color_histogram_(rgb|lab|hsv)_15_local_2 gabor_histogram_12_3_statistics_local_8 gradient_histogram_9_local_8 gradient_histogram_12_local_(6|7|8) lbp_histogram_(1|2|3)_12 lbp_histogram_(1|2|3)_6_local_(6|7|8) lbp_histogram_(1|2|3)_9_local_2
< 32 kB	color_histogram_(rgb|lab|hsv)_7_local_(7|8) color_histogram_(rgb|lab|hsv)_9_local_(5|6) color_histogram_(rgb|lab|hsv)_11_local_4 color_histogram_(rgb|lab|hsv)_13_local_3 color_histogram_(rgb|lab|hsv)_15_local_2 lbp_histogram_(1|2|3)_9_local_(3|4)
< 64 kB	color_histogram_(rgb|lab|hsv)_9_local_(7|8) color_histogram_(rgb|lab|hsv)_11_local_(5|6) color_histogram_(rgb|lab|hsv)_13_local_(4|5) color_histogram_(rgb|lab|hsv)_15_local_(3|4) lbp_histogram_(1|2|3)_9_local_(5|6) lbp_histogram_(1|2|3)_12_local_(2|3)
< 128 kB	color_histogram_(rgb|lab|hsv)_11_local_(7|8) color_histogram_(rgb|lab|hsv)_13_local_(6|7) color_histogram_(rgb|lab|hsv)_15_local_(5|6) lbp_histogram_(1|2|3)_12_local_(3|4) lbp_histogram_(1|2|3)_9_local_(7|8)
< 256 kB	color_histogram_(rgb|lab|hsv)_13_local_8 color_histogram_(rgb|lab|hsv)_15_local_(7|8) lbp_histogram_(1|2|3)_12_local_(5|6)
≥ 256 kB	lbp_histogram_(1|2|3)_12_local_(7|8)

9.3 Runtime Performance of Feature Similarity Methods

Same hardware applies for this benchmark as in Appendix 9.1.

- "*hist*" is a placeholder for histogram similarity measures:
  (correlation|cosine|euclidian|manhattan|hellinger|angular|chi_square|intersection)
- "*stringsim*" is a placeholder for string similarity measures:
  (hamming|jaro|jaro_winkler|levenshtein|longest_subsequence|longest_substring|pair_distance)

Table 21 - Runtime benchmark for match features

Time	Feature
< 10 ms	aspect_ratio_hist categories_(wu|nguyen|common_ancestor|path) (name|description)_stringsim (name|description)_normalized_stringsim (name|description)_letter_(1|2|3)_grams_tfidf_angular (name|description)_word_(1|2|3)_grams_tfidf_angular (name|description)_word_stemmed_(1|2|3)_grams_tfidf_angular (name|description)_normalized_letter_(1|2|3)_grams_tfidf_angular (name|description)_normalized_word_(1|2|3)_grams_tfidf_angular (name|description)_normalized_word_stemmed_(1|2|3)_grams_tfidf_angular color_histogram_(rgb|lab|hsv)_(1|3|5|7|9|11|13|15)_hist color_histogram_(rgb|lab|hsv)_(1|3|5)_local_(2|3|4|5|6|7|8)_hist color_histogram_(rgb|lab|hsv)_7_local_(2|3|4|5|6)_hist color_histogram_(rgb|lab|hsv)_9_local_(2|3|4)_hist color_histogram_(rgb|lab|hsv)_(11|13)_local_2_hist color_histogram_(rgb|lab|hsv)_11_local_3_(correlation|cosine|euclidian|manhattan|hellinger) color_histogram_(rgb|lab|hsv)_15_local_2_(correlation|cosine|euclidian|manhattan) color_signature_(rgb|lab|hsv)_(2|4|6|8|10|12|14|16|18|20|22|24|26|28|30|32)_emd color_signature_(rgb|lab|hsv)_(2|4|6|8|10|12|14|16|18|20|22|24|26|28|30|32)_local_emd edge_histogram_statistics_hist edge_histogram_statistics_local_(2|3|4|5|6|7|8)_hist gabor_histogram_(3|6|9|12)_(1|2|3)_statistics_hist gabor_histogram_(3|6|9|12)_(1|2|3)_statistics_local_(2|3|4|5|6|7|8)_hist gradient_histogram_(3|6|9|12)_hist gradient_histogram_(3|6|9|12)_local_(2|3|4|5|6|7|8)_hist gradient_signature_(2|4|6|8|10|12|14|16|18|20|22|24|26|28|30|32)_emd gradient_signature_(2|4|6|8|10|12|14|16|18|20|22|24|26|28|30|32)_local_emd gray_histogram_statistics_hist gray_histogram_statistics_local_(2|3|4|5|6|7|8)_hist lbp_histogram_(1|2|3)_(3|6|9|12)_hist lbp_histogram_(1|2|3)_(3|6)_local_(2|3|4|5|6|7|8)_hist lbp_histogram_(1|2|3)_9_local_(2|3|4|5)_hist

< 20 ms	color_histogram_(rgb|lab|hsv)_7_local_(7|8)_hist color_histogram_(rgb|lab|hsv)_9_local_(5|6)_hist color_histogram_(rgb|lab|hsv)_11_local_3_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_11_local_4_(correlation|cosine|euclidian|manhattan|hellinger) color_histogram_(rgb|lab|hsv)_11_local_5_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_13_local_3_(correlation|cosine|euclidian|manhattan|hellinger) color_histogram_(rgb|lab|hsv)_15_local_2_(hellinger|angular|chi_square|intersection) lbp_histogram_(1|2|3)_9_local_6_hist lbp_histogram_(1|2|3)_9_local_7_(correlation|cosine|euclidian|manhattan|hellinger) lbp_histogram_(1|2|3)_9_local_8_(correlation|cosine|euclidian|manhattan) lbp_histogram_(1|2|3)_12_local_2_hist
< 30 ms	color_histogram_(rgb|lab|hsv)_9_local_(7|8)_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_9_local_7_hellinger color_histogram_(rgb|lab|hsv)_11_local_4_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_11_local_5_(correlation|cosine|euclidian|manhattan|hellinger) color_histogram_(rgb|lab|hsv)_11_local_6_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_13_local_3_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_13_local_4_(correlation|cosine|euclidian|manhattan|hellinger) color_histogram_(rgb|lab|hsv)_15_local_3_(correlation|cosine|euclidian|manhattan|hellinger) lbp_histogram_(1|2|3)_9_local_7_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_9_local_8_(correlation|cosine|euclidian|manhattan|hellinger) lbp_histogram_(1|2|3)_12_local_3_(correlation|cosine|euclidian|manhattan|hellinger)
< 40 ms	color_histogram_(rgb|lab|hsv)_9_local_7_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_11_local_5_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_11_local_6_hellinger color_histogram_(rgb|lab|hsv)_13_local_4_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_13_local_5_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_15_local_3_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_15_local_4_(correlation|cosine|euclidian|manhattan) lbp_histogram_(1|2|3)_9_local_8_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_3_(angular|chi_square|intersection)
< 50 ms	color_histogram_(rgb|lab|hsv)_11_local_6_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_11_local_7_(correlation|cosine|euclidian|manhattan|hellinger) color_histogram_(rgb|lab|hsv)_13_local_5_hellinger color_histogram_(rgb|lab|hsv)_13_local_6_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_15_local_4_hellinger color_histogram_(rgb|lab|hsv)_9_local_8_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_4_(correlation|cosine|euclidian|manhattan|hellinger)

< 60 ms	color_histogram_(rgb|lab|hsv)_11_local_8_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_13_local_5_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_13_local_6_hellinger color_histogram_(rgb|lab|hsv)_15_local_4_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_15_local_5_(correlation|cosine|euclidian|manhattan)
< 70 ms	color_histogram_(rgb|lab|hsv)_11_local_7_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_11_local_8_hellinger color_histogram_(rgb|lab|hsv)_13_local_7_(correlation|cosine|euclidian|manhattan) lbp_histogram_(1|2|3)_12_local_4_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_5_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_15_local_5_hellinger
< 80 ms	color_histogram_(rgb|lab|hsv)_15_local_6_(correlation|cosine|euclidian|manhattan) lbp_histogram_(1|2|3)_12_local_5_hellinger
< 90 ms	color_histogram_(rgb|lab|hsv)_11_local_8_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_13_local_6_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_13_local_7_hellinger color_histogram_(rgb|lab|hsv)_13_local_8_(correlation|cosine|euclidian|manhattan) color_histogram_(rgb|lab|hsv)_15_local_5_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_6_(correlation|cosine|euclidian|manhattan)
< 100 ms	color_histogram_(rgb|lab|hsv)_15_local_6_hellinger
< 110 ms	color_histogram_(rgb|lab|hsv)_13_local_7_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_13_local_8_hellinger color_histogram_(rgb|lab|hsv)_15_local_7_(correlation|cosine|euclidian|manhattan) lbp_histogram_(1|2|3)_12_local_5_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_6_hellinger
< 120 ms	color_histogram_(rgb|lab|hsv)_15_local_6_(angular|chi_square|intersection)
< 130 ms	color_histogram_(rgb|lab|hsv)_15_local_7_hellinger lbp_histogram_(1|2|3)_12_local_7_(correlation|cosine|euclidian|manhattan)
< 140 ms	color_histogram_(rgb|lab|hsv)_13_local_8_(angular|chi_square|intersection) color_histogram_(rgb|lab|hsv)_15_local_8_(correlation|cosine|euclidian|manhattan)
< 150 ms	lbp_histogram_(1|2|3)_12_local_6_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_7_hellinger
< 175 ms	color_histogram_(rgb|lab|hsv)_15_local_8_hellinger color_histogram_(rgb|lab|hsv)_15_local_7_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_8_(correlation|cosine|euclidian|manhattan)

< 200 ms	lbp_histogram_(1|2|3)_12_local_7_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_8_hellinger
< 250 ms	color_histogram_(rgb|lab|hsv)_15_local_8_(angular|chi_square|intersection) lbp_histogram_(1|2|3)_12_local_8_(angular|chi_square|intersection)
≥ 500 ms	shape_context

9.4 Separate Feature Ranking

Table 22 - Top 20 ranking similarity features per feature group

Feature	Mean AUC	AUC Stddev	Mean Rank	Rank Stddev
color_histogram_hsv_11_local_8_hellinger	0.76	0.004	3.4	2.42
color_histogram_hsv_11_local_7_hellinger	0.76	0.004	3.8	2.93
color_histogram_hsv_13_local_7_hellinger	0.759	0.004	5.6	3.77
color_histogram_hsv_11_local_6_hellinger	0.758	0.004	8.6	3.2
color_histogram_hsv_11_local_5_hellinger	0.758	0.004	8.8	4.35
color_histogram_hsv_11_local_8_intersection	0.757	0.005	10.4	5.0
color_histogram_hsv_13_local_6_hellinger	0.757	0.004	10.6	6.56
color_histogram_hsv_11_local_8_manhattan	0.757	0.005	11.0	5.55
color_histogram_hsv_5_local_5_hellinger	0.757	0.006	12.2	10.59
color_histogram_hsv_5_local_8_hellinger	0.757	0.006	12.8	13.12
color_histogram_hsv_13_local_5_hellinger	0.757	0.004	13.4	9.2
color_histogram_hsv_5_local_7_hellinger	0.757	0.006	13.4	10.65
color_histogram_hsv_11_local_7_intersection	0.757	0.005	13.8	6.62
color_histogram_hsv_11_local_7_manhattan	0.757	0.005	14.4	6.83
color_histogram_hsv_5_local_6_hellinger	0.757	0.006	16.6	11.77
color_histogram_hsv_11_local_3_hellinger	0.756	0.004	16.6	4.76
color_histogram_hsv_13_local_3_hellinger	0.755	0.004	19.2	10.57
color_histogram_hsv_11_local_6_intersection	0.755	0.005	19.4	5.57
color_histogram_hsv_11_local_6_manhattan	0.755	0.005	19.6	6.31
color_histogram_hsv_13_local_7_manhattan	0.755	0.005	20.4	4.03
color_signature_lab_2_emd	0.711	0.005	311.0	44.97
color_signature_lab_4_emd	0.709	0.004	343.6	16.6
color_signature_lab_32_emd	0.707	0.003	354.0	15.9
color_signature_lab_30_emd	0.707	0.003	356.2	15.59
color_signature_lab_32_local_emd	0.706	0.003	359.8	25.01
color_signature_lab_30_local_emd	0.706	0.003	363.2	22.01
color_signature_lab_26_local_emd	0.706	0.003	365.4	23.5
color_signature_lab_28_emd	0.706	0.003	367.2	16.49

color_signature_lab_28_local_emd	0.706	0.003	369.4	19.12
color_signature_lab_24_emd	0.705	0.003	369.6	15.34
color_signature_lab_24_local_emd	0.705	0.003	371.6	22.33
color_signature_lab_26_emd	0.705	0.003	372.2	16.92
color_signature_lab_18_emd	0.705	0.003	374.8	17.78
color_signature_lab_22_emd	0.704	0.003	381.2	17.74
color_signature_lab_22_local_emd	0.704	0.003	381.8	24.45
color_signature_lab_20_emd	0.704	0.003	382.0	16.11
color_signature_lab_20_local_emd	0.704	0.003	391.2	22.75
color_signature_lab_14_emd	0.703	0.003	398.0	19.6
color_signature_lab_16_local_emd	0.703	0.003	398.8	21.54
color_signature_lab_16_emd	0.703	0.003	400.0	18.69
gabor_histogram_6_1_stddev_local_2_manhattan	0.707	0.012	362.0	134.06
gabor_histogram_6_3_mean_local_2_correlation	0.703	0.009	392.0	72.62
gabor_histogram_6_3_mean_local_5_euclidian	0.702	0.006	401.8	53.83
gabor_histogram_9_2_stddev_local_2_manhattan	0.703	0.01	404.4	98.36
gabor_histogram_9_2_stddev_correlation	0.701	0.009	419.0	88.08
gabor_histogram_6_2_stddev_correlation	0.7	0.008	437.0	86.9
gabor_histogram_6_1_stddev_local_2_euclidian	0.701	0.011	442.8	158.1
gabor_histogram_6_3_stddev_local_2_manhattan	0.698	0.008	453.8	99.76
gabor_histogram_9_2_stddev_euclidian	0.699	0.007	455.6	105.45
gabor_histogram_9_2_stddev_local_2_euclidian	0.699	0.009	456.4	118.87
gabor_histogram_9_2_stddev_angular	0.698	0.009	458.6	99.48
gabor_histogram_6_3_stddev_local_2_euclidian	0.697	0.008	472.8	106.39
gabor_histogram_6_2_mean_local_5_euclidian	0.696	0.006	480.4	50.04
gabor_histogram_6_2_stddev_angular	0.697	0.01	481.0	115.2
gabor_histogram_9_2_stddev_intersection	0.697	0.01	481.6	116.77
gabor_histogram_6_3_mean_local_7_euclidian	0.696	0.009	484.0	76.59
gabor_histogram_6_1_mean_local_2_manhattan	0.7	0.013	485.0	203.63
gabor_histogram_6_1_mean_local_7_euclidian	0.696	0.007	490.4	78.59
gabor_histogram_9_2_stddev_hellinger	0.696	0.009	492.8	109.1
gabor_histogram_6_2_stddev_local_2_manhattan	0.696	0.008	494.0	115.36
name_normalized_word_stemmed_1_grams_tfidf_angular	0.707	0.012	374.6	159.56
name_normalized_word_1_grams_tfidf_angular	0.706	0.012	397.8	174.9
name_normalized_letter_3_grams_tfidf_angular	0.699	0.012	539.8	298.61
name_normalized_pair_distance	0.691	0.011	720.6	444.09
name_normalized_letter_2_grams_tfidf_angular	0.688	0.01	798.4	476.83
name_pair_distance	0.687	0.01	824.0	476.44
name_word_stemmed_1_grams_tfidf_angular	0.685	0.01	856.2	384.75

name_word_1_grams_tfidf_angular	0.684	0.009	865.2	372.82
name_letter_3_grams_tfidf_angular	0.68	0.01	1045.4	529.65
name_normalized_longest_substring	0.676	0.012	1200.6	665.96
name_letter_2_grams_tfidf_angular	0.676	0.01	1219.4	586.23
name_longest_substring	0.671	0.009	1388.4	620.82
name_letter_1_grams_tfidf_angular	0.671	0.01	1421.0	716.69
name_normalized_letter_1_grams_tfidf_angular	0.668	0.013	1561.8	807.03
name_jaro	0.659	0.015	2047.0	908.18
name_longest_subsequence	0.658	0.01	2077.4	738.07
name_normalized_jaro	0.657	0.015	2121.2	939.86
name_jaro_winkler	0.656	0.016	2169.8	972.73
name_normalized_jaro_winkler	0.654	0.016	2246.4	982.94
name_normalized_longest_subsequence	0.653	0.011	2391.0	670.39
gradient_signature_32_local_emd	0.704	0.007	383.0	63.12
gradient_signature_18_emd	0.702	0.008	401.4	75.75
gradient_signature_20_local_emd	0.701	0.008	417.0	71.55
gradient_signature_28_emd	0.702	0.009	419.2	95.24
gradient_signature_24_emd	0.701	0.009	431.2	106.09
gradient_signature_30_local_emd	0.7	0.007	432.0	67.99
gradient_signature_24_local_emd	0.7	0.009	433.2	80.71
gradient_signature_28_local_emd	0.7	0.008	434.8	75.91
gradient_signature_32_emd	0.699	0.009	447.4	102.34
gradient_signature_14_local_emd	0.699	0.008	449.2	78.55
gradient_signature_30_emd	0.699	0.01	451.2	105.11
gradient_signature_26_emd	0.698	0.009	468.8	104.28
gradient_signature_22_local_emd	0.697	0.009	479.8	87.44
gradient_signature_20_emd	0.697	0.009	481.8	103.56
gradient_signature_16_local_emd	0.693	0.007	539.2	80.29
gradient_signature_18_local_emd	0.693	0.009	545.8	90.55
gradient_signature_26_local_emd	0.691	0.007	575.8	87.47
gradient_signature_22_emd	0.69	0.009	601.6	117.99
gradient_signature_16_emd	0.689	0.008	640.2	118.0
gradient_signature_14_emd	0.687	0.007	672.4	103.17
gradient_histogram_3_hellinger	0.699	0.008	451.4	72.13
gradient_histogram_3_manhattan	0.698	0.008	453.2	63.62
gradient_histogram_3_intersection	0.698	0.008	453.8	64.37
gradient_histogram_3_angular	0.693	0.009	531.6	82.35
gradient_histogram_3_local_2_manhattan	0.693	0.008	548.8	113.31
gradient_histogram_3_local_2_intersection	0.693	0.008	549.0	113.55

gradient_histogram_3_euclidian	0.692	0.008	550.8	75.08
gradient_histogram_3_local_3_angular	0.69	0.008	606.0	125.33
gradient_histogram_3_local_5_manhattan	0.69	0.009	614.4	127.35
gradient_histogram_3_local_5_intersection	0.69	0.009	614.6	128.26
gradient_histogram_6_intersection	0.689	0.007	623.0	87.36
gradient_histogram_6_manhattan	0.689	0.007	623.6	87.14
gradient_histogram_3_local_3_intersection	0.689	0.009	639.8	147.23
gradient_histogram_3_local_3_manhattan	0.689	0.009	640.0	147.52
gradient_histogram_3_chi_square	0.689	0.009	640.8	139.44
gradient_histogram_3_local_3_cosine	0.689	0.008	649.2	134.19
gradient_histogram_6_hellinger	0.688	0.007	650.4	95.01
gradient_histogram_3_local_3_correlation	0.688	0.008	655.6	121.82
gradient_histogram_3_local_6_manhattan	0.687	0.008	672.2	132.5
gradient_histogram_3_local_6_intersection	0.687	0.008	672.4	132.66
description_normalized_word_stemmed_2_grams_tfidf_angular	0.689	0.012	680.8	218.3
description_normalized_word_stemmed_1_grams_tfidf_angular	0.687	0.007	706.8	199.85
description_normalized_word_2_grams_tfidf_angular	0.686	0.01	743.6	211.86
description_normalized_word_1_grams_tfidf_angular	0.682	0.006	874.8	228.35
description_normalized_letter_3_grams_tfidf_angular	0.68	0.009	956.0	335.55
description_letter_3_grams_tfidf_angular	0.676	0.008	1134.0	383.32
description_normalized_word_stemmed_3_grams_tfidf_angular	0.674	0.012	1219.0	425.67
description_normalized_letter_2_grams_tfidf_angular	0.674	0.008	1244.6	445.5
description_normalized_word_3_grams_tfidf_angular	0.672	0.012	1335.4	407.84
description_word_stemmed_1_grams_tfidf_angular	0.669	0.005	1464.0	322.55
description_word_1_grams_tfidf_angular	0.666	0.005	1629.2	342.22
description_letter_2_grams_tfidf_angular	0.665	0.008	1725.2	596.78
description_word_stemmed_2_grams_tfidf_angular	0.664	0.006	1746.8	244.51
description_normalized_letter_1_grams_tfidf_angular	0.663	0.008	1854.4	605.23
description_word_2_grams_tfidf_angular	0.661	0.005	1932.0	214.62
description_word_stemmed_3_grams_tfidf_angular	0.657	0.009	2159.2	381.58
description_word_3_grams_tfidf_angular	0.656	0.01	2225.4	412.31
description_normalized_longest_substring	0.653	0.009	2398.4	349.38
description_pair_distance	0.653	0.007	2404.6	435.81
description_normalized_pair_distance	0.652	0.007	2482.4	482.45
lbp_histogram_3_12_local_3_manhattan	0.679	0.005	955.2	172.98
lbp_histogram_3_12_local_3_intersection	0.679	0.005	955.8	172.27
lbp_histogram_3_12_local_3_chi_square	0.674	0.005	1198.8	210.34
lbp_histogram_3_12_local_4_manhattan	0.673	0.005	1255.8	190.89
lbp_histogram_3_12_local_4_intersection	0.673	0.005	1256.2	191.31

lbp_histogram_3_9_local_3_manhattan	0.67	0.006	1373.8	200.53
lbp_histogram_3_9_local_3_intersection	0.67	0.006	1374.4	199.83
lbp_histogram_3_12_local_3_hellinger	0.67	0.004	1392.0	232.25
lbp_histogram_2_12_local_3_intersection	0.669	0.005	1433.8	333.0
lbp_histogram_2_12_local_3_manhattan	0.669	0.005	1434.0	333.18
lbp_histogram_3_9_local_5_manhattan	0.669	0.007	1470.0	163.62
lbp_histogram_3_9_local_5_intersection	0.669	0.007	1470.2	164.22
lbp_histogram_2_9_local_3_intersection	0.668	0.006	1491.0	288.37
lbp_histogram_2_9_local_3_manhattan	0.668	0.006	1491.2	287.89
lbp_histogram_3_12_local_4_chi_square	0.668	0.004	1517.4	252.98
lbp_histogram_2_9_local_7_intersection	0.667	0.007	1540.6	160.62
lbp_histogram_2_9_local_7_manhattan	0.667	0.007	1540.6	159.95
lbp_histogram_2_9_local_6_intersection	0.667	0.007	1553.6	164.58
lbp_histogram_2_9_local_6_manhattan	0.667	0.007	1554.2	164.03
lbp_histogram_2_9_local_5_intersection	0.667	0.006	1583.4	101.95
gray_histogram_stddev_local_3_manhattan	0.676	0.007	1103.6	258.48
gray_histogram_nu20_local_7_correlation	0.67	0.009	1413.4	344.32
gray_histogram_stddev_local_3_euclidian	0.669	0.007	1451.0	264.6
gray_histogram_stddev_local_6_manhattan	0.664	0.005	1761.4	214.19
gray_histogram_stddev_local_6_intersection	0.664	0.008	1768.4	229.19
gray_histogram_stddev_local_4_manhattan	0.66	0.006	1962.0	246.33
gray_histogram_stddev_local_2_manhattan	0.659	0.008	2017.2	274.26
gray_histogram_stddev_local_2_euclidian	0.658	0.008	2101.8	304.49
gray_histogram_nu02_local_7_correlation	0.658	0.011	2143.4	514.55
gray_histogram_stddev_local_6_angular	0.657	0.009	2148.8	223.91
gray_histogram_mean_local_3_euclidian	0.656	0.005	2224.4	200.58
gray_histogram_stddev_local_6_euclidian	0.656	0.005	2226.6	161.2
gray_histogram_stddev_local_5_manhattan	0.655	0.008	2265.4	513.52
gray_histogram_stddev_local_5_euclidian	0.655	0.007	2285.0	444.34
gray_histogram_nu11_local_8_angular	0.654	0.007	2343.8	354.24
gray_histogram_nu11_local_8_cosine	0.654	0.007	2356.0	354.27
gray_histogram_stddev_local_6_cosine	0.653	0.008	2404.2	151.53
gray_histogram_stddev_local_7_manhattan	0.652	0.006	2436.8	154.08
gray_histogram_stddev_local_4_euclidian	0.652	0.007	2450.2	237.81
gray_histogram_nu03_local_5_angular	0.652	0.012	2460.0	577.31
edge_histogram_stddev_x_local_3_manhattan	0.636	0.005	3288.6	134.99
edge_histogram_nu02_local_3_correlation	0.634	0.007	3336.0	325.3
edge_histogram_stddev_x_local_3_angular	0.633	0.007	3401.0	176.14
edge_histogram_stddev_x_local_3_euclidian	0.633	0.005	3425.8	148.27

edge_histogram_stddev_x_local_3_intersection	0.632	0.008	3450.2	188.85
edge_histogram_nu02_local_3_angular	0.629	0.005	3568.8	92.11
edge_histogram_stddev_y_local_6_euclidian	0.628	0.009	3586.2	224.66
edge_histogram_nu02_local_6_intersection	0.627	0.009	3623.6	326.9
edge_histogram_mean_y_local_4_manhattan	0.626	0.009	3672.2	224.66
edge_histogram_nu02_local_6_correlation	0.625	0.009	3677.6	313.61
edge_histogram_nu02_local_3_cosine	0.626	0.004	3679.2	82.56
edge_histogram_nu02_local_3_intersection	0.626	0.006	3683.8	42.06
edge_histogram_nu20_local_3_correlation	0.624	0.01	3689.2	307.09
edge_histogram_mean_y_local_6_manhattan	0.625	0.009	3693.0	261.63
edge_histogram_stddev_y_local_6_manhattan	0.625	0.008	3696.4	167.28
edge_histogram_stddev_x_local_3_cosine	0.624	0.008	3724.8	177.9
edge_histogram_mean_x_local_2_hellinger	0.624	0.008	3757.6	119.22
edge_histogram_stddev_y_local_5_euclidian	0.623	0.007	3762.4	219.46
edge_histogram_nu20_local_3_angular	0.623	0.01	3771.2	187.43
edge_histogram_stddev_y_local_6_angular	0.623	0.009	3776.8	190.73
aspect_ratio_manhattan	0.583	0.01	4889.6	332.92
aspect_ratio_euclidian	0.583	0.01	4890.2	332.47
categories_common_ancestor	0.582	0.007	4929.8	296.34
categories_shortest_path	0.567	0.008	5395.2	292.14
categories_nguyen	0.567	0.008	5402.8	296.99
categories_wu	0.567	0.008	5413.8	293.61