Knowledge and Information Systems (2021) 63:2277–2312
https://doi.org/10.1007/s10115-021-01595-z

REGULAR PAPER

On augmenting database schemas by latent visual attributes

Tomáš Grošup[1] · Ladislav Peška[1] · Tomáš Skopal[1]

Received: 23 September 2020 / Revised: 30 June 2021 / Accepted: 3 July 2021 /
Published online: 12 August 2021

Abstract
Decision-making in our everyday lives is surrounded by visually important information. Fashion, housing, dating, food or travel are just a few examples. At the same time, most commonly used tools for information retrieval operate on relational and text-based search models which are well understood by end users, but unable to directly cover visual information contained in images or videos. Researcher communities have been trying to reveal the semantics of multimedia in the last decades with ever-improving results, dominated by the success of deep learning. However, this does not close the gap to relational retrieval model on its own and often rather solves a very specialized task like assigning one of pre-defined classes to each object within a closed application ecosystem. Retrieval models based on these novel techniques are difficult to integrate in existing application-agnostic environments built around relational databases, and therefore, they are not so widely used in the industry. In this paper, we address the problem of closing the gap between visual information retrieval and relational database model. We propose and formalize a model for discovering candidates for new relational attributes by analysis of available visual content. We design and implement a system architecture supporting the attribute extraction, suggestion and acceptance processes. We apply the solution in the context of e-commerce and show how it can be seamlessly integrated with SQL environments widely used in the industry. At last, we evaluate the system in a user study and discuss the obtained results.

Keywords Relational databases · Multimedia retrieval · Multi-modal retrieval · Visual attribute discovery · Similarity search · Database modelling

1 Introduction

People are surrounded by large volumes of data in their everyday lives. Much of it is unstructured in nature—images, videos, sounds, or sensory data in general. However, such data is rarely present without a context. Instead, it is usually combined with structured attributes and exists in a heterogeneous form, for example, with prices and categories of products in

✉ Tomáš Skopal
tomas.skopal@matfyz.cuni.cz

[1] SIRET Research Group, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

e-commerce context, with names and tags in social networks, or with time and location in photograph albums.

The most common tools for information retrieval are based on the relational data model which expects a fixed structure of data and its attributes, i.e. a database schema. Although the binary data of a multimedia object might be considered as an attribute as well, it is rarely useful for retrieval tasks. The reason is that standard techniques for searching relational data are based around atomic values and often expect an exact matching or at least a linear ordering within the search space. Although it is possible to do exact search for binary data (like pixel-to-pixel matching) in theory, it is rarely useful in practice. On the other hand, approximate search of multimedia requires similarity queries for which the relational database systems are inappropriate.

Research communities have been trying to address the problem of revealing internal structure of multimedia data in the last decades with ever-improving results [17]. A common approach to enable searching of multimedia data is the usage of distance-based methods, which utilize some kind of pairwise distance function. The more similar the objects, the lesser the distance. But how can a user search with it? Different query paradigms, e.g. query-by-example, have to be used.

This problem is addressed by multimedia exploration, which admits that query formulation is not always possible [3]. Even in cases where a user cannot provide an example, they are able to distinguish between relevant and irrelevant objects once the results are presented to them. The goal of multimedia exploration software systems is to suggest intuitive exploratory steps in the dataset and to enable drilling down to particular examples once a relevant object is found. Instead of having an explicit query, implicit feedback is used to automatically expand queries that are used within the database (automatic query expansion).

In this paper, we address the problem of closing the gap between visual information retrieval and relational database model. In order to stick to a real-world use-case, we assume a running example of fashion products retrieval within an e-shop.[1] Simply said, the goal is to provide discovery of new visual attributes from product photographs (such as "high heel" or "floral pattern") and their easy integration with the pre-defined relational attributes (like price, size). However, our approach is extensible to other domains where visual features and traditional alphanumeric attributes need to be integrated. In Sect. 3, we propose and formalize a model for extracting candidates for new attributes based on visual inputs. In Sect. 4, we describe a design and implementation of a system architecture supporting the attribute extraction, suggestion and acceptance processes. Then in Sect. 5 we evaluate the solution in the context of a fashion e-shop and show how it can be seamlessly integrated with SQL environments widely used in the industry.

1.1 Motivation

There have been addressed problems related to effective search in heterogeneous data by the means of multimedia exploration systems [7,18,35]. That involved similarity-driven (e.g. force-directed) layout to present results on computer screen, exploratory operations that allow to navigate within distance-based spaces, and efficient data structures to execute them. Although these tools had good evaluation results, the results are difficult to transfer to every day's search needs. One of the reasons is that multimedia exploration is not established as

[1] Other examples that combine relational with multimedia attributes could be movies database (trailer), biologic/medical records (mesh/tissue photograph), archaeology artefacts (artefact photograph), etc.

an retrieval model in existing tools (nor unified or even standardized) when compared to the relational model (and SQL) that is well known, simple and ubiquitous.

One of the major challenges the information retrieval domain facing nowadays is the ability to work with multiple modalities (e.g. numerical attributes, text and images) at the same time. Such so-called multi-modal search tools need to operate with different querying mechanisms, but also multiple search models (e.g. relational, graph-based and similarity-based). In earlier approaches to multi-modal searching, there have been utilized multiple modalities by mapping them all into a similarity space by creating a distance function that considers all known data. However, the search behaviour is sometimes counter-intuitive for the end user and cannot be simply incorporated into existing software applications.

Let's formulate an example from the domain of fashion and decompose the following search intention: "I am looking for a flower-pattern dress, around 100$, for a party".

- "Flower-pattern" information is a texture in the photograph
- "Price" is a relational attribute
- A "party" relates to an abstract (emotional) visual category of the same photograph, detectable by higher layers of neural networks used for computer vision.

In this case, any single-model query wouldn't be able to answer the search precisely, while a proper multi-model query could provide an exact answer. Please note that such decomposition is not universal and is subject to knowledge available for a given domain. There could be a database ready to answer the outlined search need with a single SQL query, because all the data are already available in the desired form. However, for most cases, the visual information is only contained in an unstructured image and must be processed—either by software or by the end user. In this example, both party and floral pattern are subjects to be "somehow" turned into attributes.

An important related task that attracted a considerable amount of research recently is the object classification task. Classification is a function from an unbound domain of input objects like images, to a fixed set of classes. The goal is to learn a mapping between object's representation and corresponding class that generalizes well on not-yet-seen data. Some variants of the classification methods also focus on detecting and classifying parts of objects, but the underlying limitation is the pre-defined set of classes. In practical implementations, it also requires multiple examples of each class as training data.

However, let us consider an inverse problem. What if we know what our data is and we can control it, but we cannot define all possible class labels upfront and we also do not have any training data for them? We would like to iteratively discover new class labels based on patterns and usage of the data and integrate the labels into existing database schema as new attributes. We denote this task as *attribute discovery*. Our main research question is whether and how we could build on the knowledge from objects' classification methods to solve such a different task as attribute discovery. The comparison in Table 1 suggests that there is a duality with respect to several axes when comparing attribute discovery to common image classification.

The key differences are as follows:

- Classification requires precisely labelled training data in order to classify unbound inputs into a fixed schema, delivering an assignment of any object into a well-known class. The target can be generic, including domain-specific classes as well as general imagery.
- Multimedia exploration based on similarity search works with unbound inputs, without any training phase or schema. The result is an interactive process with human controlling the loop.

Table 1 Table highlighting main differences between image classification, multimedia exploration and visual attribute discovery

	Classification	Exploration	Attr. discovery
Inputs	Unbound	Unbound	Known upfront
Training	Supervised	None	Unsupervised
Schema	Fixed	None	Dynamic
Examples	Manual labels	None	Equal to real data
Target	Generic	Generic	Domain-specific
Delivers	Assignment	Interaction	Schema extension

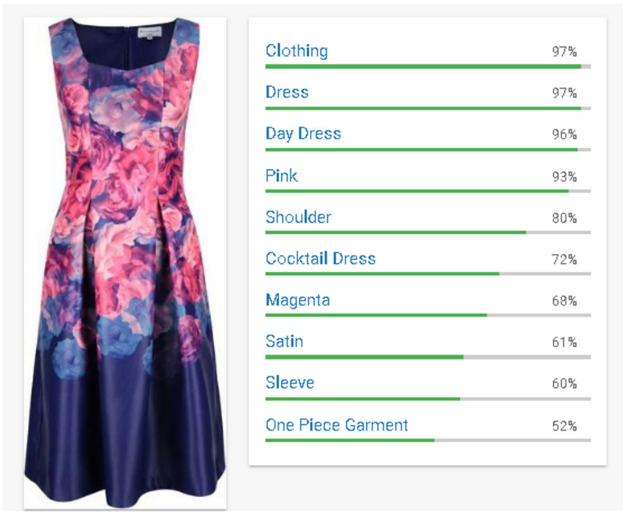
– Attribute discovery is in many aspects dual to classification approaches. It works with a well-known dataset, does not differentiate between training and real data and does not need any supervision at design time. Supervision at design time is replaced by acceptance at run time. It gradually delivers an extension of database schema by an unbound number of new attributes, out of which none were known before.

1.2 Database schema augmentation and attribute discovery

It is important to note that the described techniques originally target different tasks and operate on different levels of granularity. If certain attributes are known at design time, a specialized and pre-trained classifier (e.g. for stripe pattern, or for happy-emotion-clothes) will always do a better job than an unsupervised technique. In such case, the classes map to binary attributes (true/false classification) or numeric attributes (probability/weight of classification or number of occurrences). Similarly, if the domain has the capability to use or generate thousands of N-tuples of similar and dissimilar matching entities, supervised methods like PatternNet [27] will provide more specialized and better results. The important property of the environment our method tries to target is lack of any training data and no or limited existence of a specialized network for that particular domain.

For real-world purposes, we recommend the following orthogonal strategies to be evaluated in order to augment/extend database schemas:

1. Search for additional data in open data repositories, for example Linked Open Data. Linked Data already has a form than can be easily transformed to relational attributes or connected using SPARQL queries and does not require any pre-processing phase.
2. Classify image data using existing models of state-of-the-art image classifiers. As we can see in Fig. 1, Google Vision API can detect many useful properties about real-world objects. This again does not require any prepossessing step and can be accomplished using existing cloud solutions.
3. Use noisy text data from public communities and social networks. If there is a large dataset of image–text pairs in the given domain, relevant concepts can be extracted from it and linked to database entities. The associations can be learned, for example, via visual feature descriptors or neural network activations, as is shown by Berg et al. [4] and Vittayakorn et al. [55], respectively.
4. Use the visual attribute discovery following the methodology of this work. Our proposed method for visual attribute discovery targets information that is not found using existing generic classifiers. It is targeting domains where visual data is an important part of information retrieval process, and the ability to provide it in a structured way provides good benefits. As the example in Fig. 2 demonstrates, visual attribute discovery can propose a



Clothing	97%
Dress	97%
Day Dress	96%
Pink	93%
Shoulder	80%
Cocktail Dress	72%
Magenta	68%
Satin	61%
Sleeve	60%
One Piece Garment	52%

Fig. 1 Example of state-of-the-art image labelling provided by Google Vision API. As an example, “floral pattern” aspect is missing (https://cloud.google.com/vision/)

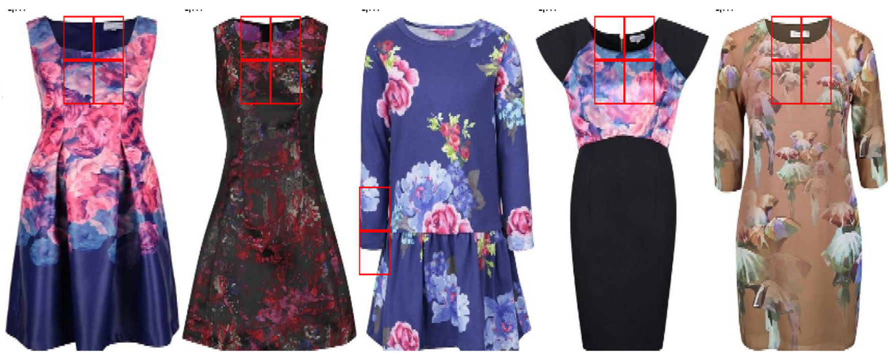


Fig. 2 Candidate for a new visual attribute proposed by the system based on common image patches. One possible label for this attribute is “floral pattern dress”. Images are a property of https://www.zoot.cz

set of clothes with floral pattern, which is information that the generic classifier did not provide and which describes a possibly interesting feature for products in arbitrary fashion store. This approach requires some effort from the human domain expert; however, the required task is rather a simple confirmation that the proposed attribute is relevant. The confirmation task is considerably less complex than constructing the attributes from scratch. (We give more evidence on this in Sect. 5.4.) Furthermore, visual attribute discovery did also provide attributes that are relevant, yet the domain expert did not think of them in advance.

1.3 Contribution

Our motivation outlines the goal toward database schema augmentation via latent visual attributes. This work is a comprehensive full research paper synthetizing and developing partial results published in conference papers [19,46,48]. The papers [48] (demo paper) and [46] (short paper) addressed some narrow-scope aspects of discovering shared visual features in product photographs within a fashion e-shop system (i.e. missing the application-agnostic extension to database scheme augmentation). The works aimed rather at a multi-modal product recommender system than at a database-oriented research. In the short vision paper [19], the challenges and roadmap for a more general framework were discussed, with no actual methodology, implementation and experiments. As planned/envisioned in [19], in this manuscript the whole model is developed into a complete data management pipeline (including the human in the loop). Over the previous works, this manuscript includes richer related work (Sect. 2), formal methodology for the process of database schema augmentation (Sect. 3), system architecture and implementation of the proposed methodology (Sect. 4) and finally a user study and experiments demonstrating the outcome of the proposed methodology (Sect. 5).

2 Related work and background

In this section, we present background topics our research builds on—the relational data management, similarity search, multi-modal search and deep learning. We continue with related topics and existing state-of-the-art solutions published for them. We discuss in what ways are the problems different and what possible synergies we can imagine for future work.

2.1 Relational data management

Relational data model was first coined by Codd [16] as a proposal to manage large data banks. Each relation consists of a heading and a body. The heading is an un-ordered set of named attributes with data types. As per the Cambridge dictionary,[2] an attribute is *"a quality or feature of a person or thing, esp. one that is an important part of its nature"*. An attribute could be represented by a simple data type (number, string, date) or by a complex object, often serialized as BLOB type. The body of a relation is a subset of the Cartesian product of all attributes, a set of n-tuples following the heading. Each n-tuple can be uniquely identified using a key, a subset of all attributes marked using underline.

$$\begin{aligned} &Relation\ Products(\underline{ID}, Name, Price, Image) \\ &Products \subseteq \mathbb{N} \times String \times \mathbb{Q} \times BLOB, \\ &\text{where BLOB denotes binary large object.} \end{aligned}$$

Industry is dominated by SQL databases derived from the relational model, family of data management solutions called after the query language SQL. Despite novel approaches to data management (nosql, hybrid databases, multi-model databases), SQL remains the language with widest adoption across industry professionals thanks to it's ease of use and declarative nature [20]. The declarative power of the relational model is further built upon by newer additions to the language such as window functions or pivoting coming with SQL : 2003 standard [15].

In our work, SQL environments are the integration target. We define data pipeline for visual information which lands its result in normalized data tables. We offer projections

[2] https://dictionary.cambridge.org/dictionary/english/attribute.

on the data using standard SQL features, which allow different views on captured visual information and can provide benefit to different application-specific use-cases.

2.2 Similarity search

Similarity-search concept provides a general model for content-based search in unstructured data such as multimedia. Given a dataset of descriptors of objects (e.g. images) and a query example descriptor, the most similar objects are returned. To avoid expensive similarity calculations, metric access methods [59] such as the M-Tree [11,52] were designed for problems satisfying the metric postulates. Most forms of metric access methods involve pivots, selected objects from the dataset which are used for pre-calculation of similarity values, and prune large portions of the database at query time [36]. The pivot information can be also used to partition the dataset into a Voronoi space [12], which can be utilized by both exact and approximate search queries [38]. In our work, similarity search is the basic building block to find visual patterns in a dataset of (product) images. Due to size constraints, a large-scale technique for approximate similarity joints based on Voronoi partitioning was used [8].

A field which leverages similarity search is the multimedia exploration [3]. Admitting that perfect query formulation is not always possible, novel user interfaces and techniques to navigate multimedia and multi-modal datasets were researched in the past [34,35]. In selected domains and use-cases, our proposed work aims to make multimedia exploration redundant by solving multi-modal search via relational schema augmentation.

A standard operator used in similarity search is the kNN query for single-input queries and kNN similarity join for set operations. Following text provides basic definitions for them.

Definition (*kNN query*) For a dataset $\mathcal{DS} \subseteq \mathbb{U}$, a query q $\in \mathbb{U}$ and a distance function $\delta : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ (where $\mathbb{U}$ is the descriptor universe), the k nearest-neighbour query is defined as:

$$kNN(q, \mathcal{DS}) = \{X \subset \mathcal{DS}; |X| = k \wedge \forall x \in X, \forall y \in \mathcal{DS} - X : \delta(q, x) \leq \delta(q, y)\}$$

Definition (*kNN approximate query*) An approximate kNN query for an object $q \in \mathbb{U}$ is labelled as $kNN_a(q, \mathcal{DS})$ and defined as an ϵ-approximation of the exact kNN: $kNN_a(q, \mathcal{DS}) = \{X \subset \mathcal{DS}; |X| = k \ \wedge \ \max_{x \in X} \delta(q, x) \leq \epsilon \cdot \max_{x \in kNN(q, \mathcal{DS})} \delta(q, x)\}$, where $\epsilon \geq 1$ is an approximation constant.

Definition (*kNN similarity join*) For two sets, query set $Q \subseteq \mathbb{U}$ and a dataset $\mathcal{DS} \subseteq \mathbb{U}$, we define the k nearest-neighbour similarity join: $Q \bowtie \mathcal{DS} = \{(q, o) \mid q \in Q, o \in kNN(q, \mathcal{DS})\}$

Analogously we define the approximate k nearest-neighbour similarity join: $Q \bowtie_a \mathcal{DS} = \{(q, o) \mid q \in Q, o \in kNN_a(q, \mathcal{DS})\}$

When running a similarity join with the query set and dataset being equal, we refer to it as self-join. It produces sets of objects close to each other in the distance space, i.e. identifies near-duplicate or duplicate objects.

2.3 Multi-modal search

Modality is a way in which something is experienced by humans—such as vision, text, sound, taste or smell. The possibility of combining multiple modalities in information retrieval has been researched in the past decades and is referred to as multi-modal search [2,6]. The basic categories of combination are early and late fusion of multiple modalities. Furthermore, cross-modal search [56] is defined by taking one modality as an input query and retrieving relevant data of different modalities.

Our work aims to satisfy both multi-modal and cross-modal search paradigms. By extracting complex information into relational attributes, standard query mechanisms can be used to accomplish both.

2.4 Deep learning

In the last years, many research topics were experiencing the deep learning revolution. Techniques based on deep convolutional neural networks (DCNNs) became the state of the art for many problems, and it was one of the driving forces for our vision of visual attribute discovery. The gradually increasing semantic levels [58] of different layers of the network can detect patterns previously not imaginable with analytical feature descriptors like MPEG-7 or SIFT [9,37]. One of the research topics lately dominated by deep learning is also image segmentation [31], one of the building blocks of our proposed methodology.

The pioneer DCNN architecture AlexNet [25] contains five convolutional layers, three fully connected layers, and max-pooling layers in between. Newer networks are usually comprised from much more layers and include additional techniques and architectures to further optimize its effectiveness [24,43]—ResNet, Inception and ResNeXt are just a few examples. The different layers of various networks have been shown to hold different levels of visual information [58], starting with pixels and edges and ending with semantic classes.

DCNN networks are commonly optimized for an image classification task with a fixed set of generic classes. However, the trained models have been successful also as generic feature extractors [14,60] creating feature vectors by collecting neuron activations after an inference (forward-pass). This makes it possible to use a pre-trained model of a network to solve a problem different than the original static classes. This technique was adopted in our research as well. In general, this technique is part of transfer learning methodologies [49]. There was validated the positive impact of using a model pre-trained on general imagery (ImageNet) on a visually different domain [48]. Further analyses of the impact when using the same model for patches of images shown further improvements of search results [41,46].

2.5 Visual pattern discovery

The problem of identifying commonly occurring visual aspects within an image dataset is called "Visual Pattern Discovery". This problem was defined in order to solve many tasks in computer vision, e.g. content-based image search, object localization [53] or visual pattern recognition [57]. An example of recent visual pattern recognition architecture is PatternNet [27]. PatternNet analyses filters on the last convolutional layer aiming to find locally consistent visual patches for preselected categories. The network specifically stresses on finding patterns that are both representative and discriminative. Although such solution could be used as a building block in identifying new visual attributes, it does require a domain-specific learning phase and a number of training example-sets. At the same time, the expected number of visual patterns must be preset.

In our use-case of discovering latent visual attributes, we face the problem of an unbounded size which is impossible to estimate at design time and is always varying over time due to discrepancies in human interpretation. At the same time, we are targeting use-cases without a supervised training step and aim to eliminate the need for a dedicated training dataset. Nonetheless, we adopted the need for representative and discriminative patterns in the proposed framework.

2.6 Product and fashion classification

The standard problem of image classification exists also in specialized variants for products. This is driven by needs for efficient product image classification methods, and state-of-the-art solutions are typically based on supervised methods [30]. Like in generic image classification, convolutional neural networks have lead to largest advances. With the increased usage of e-commerce for fashion, specialized techniques for fashion item retrieval have gained research attention. The domain brings several additional challenges.

Cross-domain retrieval between street-captured photographs and cleaned product photographs represents one of such challenges. Product photographs typically have a standard background, orientation and scale. Novel approaches have been created to target different variants of the problem, such as Street2Shop, Shop2Street or Video2Shop [10,23]. For supervised approaches, an established dataset DeepFashion [32,33] provides training data with over 800,000 labelled images, classified into 50 categories and 1000 attributes. Another fashion-specific challenge is the outfit completion problem [29,50,54]. Here, the task is to recommend suitable complements to a partially constructed outfit, so that the visual style of the proposed product match with the rest of the outfit. These tasks can be considered in a pipeline, where Street2Shop-like algorithms can be utilized as data constructors for the outfit completion problem.

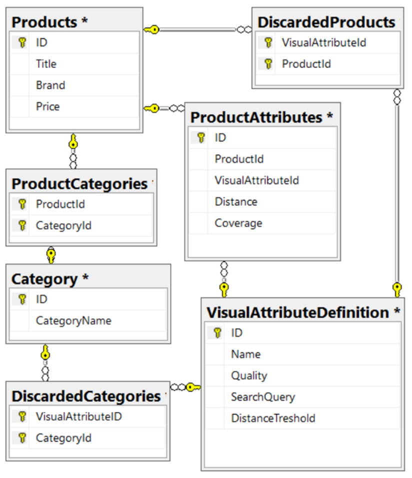
We see possible synergies between the work of attribute discovery and cross-domain fashion retrieval. Namely, techniques for fashion style generation based on street-level image analysis could be another source of implicit feedback for visual patterns in the fashion domain. That is, the fact that certain items are often spotted together on the street might be a relevant signal for a common attribute that is yet to be discovered. Existing trained models for product classification would be also a natural pre-processing step to extend a new dataset with attributes, thus reducing the scope for attribute discovery. On the other hand, the goal of outfit completion algorithms is mostly orthogonal with our use-case. Instead of focusing on recommending items based on their latent style descriptors, we aim on disclosing the visual (style) patterns themselves and making them transparent for users.

3 Augmenting database schema by latent visual attributes

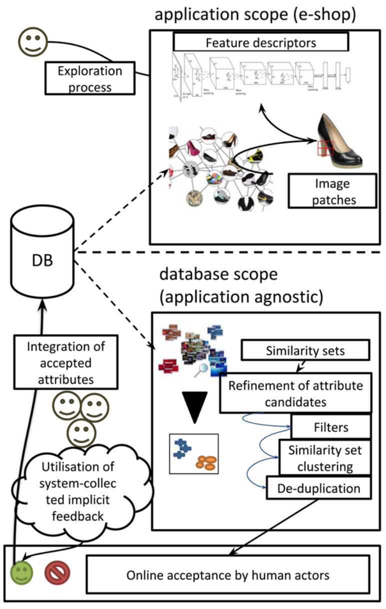
Based on our motivation, this section defines the problems of visual attribute discovery and database schema augmentation. We propose a methodology for the process and define the essential entities, actors and steps needed to accomplish it.

In short, the proposed methodology can be decomposed into an offline pre-processing stage, which aims on proposing candidate visual attributes and an online step involving a human judge (i.e. domain administrator) that could refine proposed attributes and confirm or reject them.

The pre-processing stage requires some visual feature descriptor (e.g. a deep convolutional network) and comprises from a pipeline of image patches generation, image patches' similarity sets construction and several attribute candidate refinement and filtering techniques. The overview of the proposed methodology is depicted in Fig. 3b.



(a) Database schema for capturing information about accepted attributes.


application scope (e-shop)
Feature descriptors
Exploration process
Image patches
DB
database scope
(application agnostic)
Similarity sets
Refinement of attribute candidates
Filters
Similarity set clustering
De-duplication
Integration of accepted attributes
Utilisation of system-collec ted implicit feedback
Online acceptance by human actors

(b) Methodology for visual attribute discovery.

Fig. 3 Diagrams for entity relationships and data flow of the proposed methodology

3.1 Formal task definition

We assume the existence of a **database** modelling the entities and their known attributes as named tuples, and an **image** of the entity represented as a bitmap of pixels. As for the image properties, we assume to have canonical product photographs that are common for e-commerce vendors, i.e. centred objects oriented in the same direction with similar product size ratio and uniform background. Premise of our work is that this image inherently contains other semantic features which are not yet covered as attributes in the existing database.

$$Entities(\underline{ID}, A_1 \dots A_n, Img),$$

where ID represents the unique identifier of each entity, $A_1 \dots A_n$ are existing known attributes and Img is a serialized binary representation of an image. Images are typically represented as a (possibly compressed) matrix of pixels. Each pixel (with i, j denoting its coordinates) carries the value of a colour from a chosen colour model, with RGB being the dominant one for storage purposes.

The aim of attribute discovery is to have a process transforming the original set of attributes into an extended set of attributes, while utilising visual information contained in the image. We split the process into two core parts:

1. First, in the offline prepossessing stage, the process of **discovering** attribute candidates is performed. The aim is to yield new attribute candidates from the existing relation entities and image representations:

$$Discovery := Entities \mapsto candidate_1, \dots, candidate_m$$

The data type of new attribute candidates can have more possible forms, such as binary flag referring to a category (class, tag) if just entity identification (e.g. classification) is used in the pipeline, or numerical and more complex data types if further aggregations are used on top of the classification/discovery results. In Sect. 4.8, we elaborate more on this topic. The process of candidates construction is initiated by the pre-processing of source images into image patches (Sect. 3.2). The next step is to mine sets of visually similar patches (Sect. 3.3) followed by several filtering techniques, removing, e.g. near-duplicate sets or sets too similar to already existing attributes (Sect. 3.4).

2. In the second stage, the active acceptance of attribute candidates and **augmentation** of database schema is performed (Sect. 3.5). We consider the schema augmentation as a (conditional) acceptance of some of the existing attribute candidates:

$$Augmentation := (Entities, candidate) \mapsto (\underline{ID}, A_1 \ldots A_n, Img, candidate)$$

The goal is to augment the schema in a way that maintains a meaningful mapping between the attributes and human understanding of the real-world objects which the entities represent.

With this definition of the problem, we propose a formal methodology to systematically discover new attribute candidates and augment database schemas using integration of these attribute candidates. The process is a pipeline consisting of several steps, illustrated in Fig. 3b. The following text describes each of the process steps in detail.

3.2 Image patches

To support multiple levels of granularity for new attributes, we can segment the original image into many possibly overlapping **patches**, out of which each can yield separate proposals for new attributes. More precisely, the segmentation process assigns each pixel of the image to 1 or more patches. A patch is a subset of the original image, and the union of all patches re-creates the original image. We denote a set of all possible patches as $Patches$ and a single instance as $Patch_{Img,k}$, defined by its original image and a sequential number.

$$Segmentation := Img \mapsto \{Patch_{Img,k} | Patch_{Img,k} \in Patches, k \in \mathbb{N}\}$$
$$\forall Img, k : Patch_{Img,k} \subseteq Img$$
$$\forall Img : \bigcup_{k} Patch_{Img,k} \equiv Img$$

With this terminology, the image itself can be considered as one of the patches, and when writing about patches, we will implicitly assume that one of them is also the full original image. In simplistic scenarios, image patches can be restricted to be continuous and rectangular. In that case, each patch could be defined just by a pair of image pixel locations defining the upper left and lower right corners of the rectangle. But in the general case, the segmentation problem allows patches to be of any shape and not necessarily continuous.

3.3 Similarity sets

In our model, proposals for new attributes are suggested using **visual patterns** detected across patches from the entire dataset. Similar patterns are defined as patches being close in a distance space which is constructed using a feature extraction tool and a patch-distance function δ. A feature extractor takes a single patch as input and produces a feature descriptor as output. The universe of feature descriptors is denoted as $\mathbb{U}$. The visual pattern then acts as a constructor for a multi-query distance function, which is to be evaluated against the entire dataset and materialize a new attribute. We define the initial proposals for visual patterns as **similarity sets**.

$$\begin{aligned} Feature\ extraction &:= Patches \mapsto \mathbb{U} \\ \delta(x, y) &:= \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R} \\ Similarity\ set &:= \{S \subset \mathbb{U} |\ \forall_{x \in S, y \in S, z \in \mathbb{U} \setminus S} : \delta(x, y) \leq \delta(x, z)\} \end{aligned}$$

In a practical setup, $\mathbb{U}$ would usually be a vector space, where vectors of individual patches are obtained as neuron responses from a DCNN layer during forward pass. Also, the distance function should usually satisfy the properties of metric spaces. However, these requirements are not enforced by the general model but specific to an implementation. For the same dataset, multiple feature extraction methods and possibly also multiple distance functions can be used at the same time, resulting in more proposals for attribute candidates. In the area of deep learning networks, each layer of the neural network can produce an independent feature extractor for example.

For efficient creation of the similarity set, approximate algorithms for distributed and highly scalable similarity self-join and kNN-join were used [8]. This was needed to reduce the computation cost of calculating all pairwise distances across the entire dataset and is an approximation which uses the triangle inequality property of metric spaces to avoid majority of distance calculations. The results of the kNN-join can be used as a starting point for refinement to obtain candidates for approximate similarity sets, after several steps of filtering and clustering operations. The following text describes the refinement steps needed to reduce the amount and size of results produced by a similarity self-join on patch features from the entire dataset. It also shows how these results are clustered together to produce unique and de-duplicated sets.

3.4 Refinement of attribute candidates

Having the similarity sets, we have observed that not each of them represents what humans understand as an attribute and what brings value in information retrieval scenarios. More specifically, we are looking at the following properties to be satisfied:

- A new attribute should be **discriminative** within the dataset. Although a new attribute set to the same value for every entity in the database is possible, it would not bring any benefit to end users of the system (e.g. discovering a binary attribute SHOE=true within a dataset of only shoes). In terminology of information retrieval, that attribute would be a stop word (visual stop word, or visual noise in our case). As defined by Li et al. [27], visual patterns in one attribute should be significantly different from patterns found in other attributes.
- A new attribute should be **represented** within the dataset. Inversely to the discriminative property, attributes discovered in one or just a few entities would infest the schema and decrease usability of information retrieval. As seen by Li et al. [27], visual patterns of representative attributes should appear frequently among images (but not as frequent to become stop words).
- A new attribute should be **unique** within all attributes, both pre-existing and new.

We propose a sequence of heuristics to support these three properties and to reduce too frequent patches (visual noise) in the similarity sets. Example of noisy sets include background-only patches, mono-colour patches or other patches which are equal or near-equal in the distance space. The heuristics are based on observations of real data and governed by hyperparameters which were tuned in the offline and online evaluation phases.

- **Near-duplicate filter:** Reducing the number of similarity sets by excluding any sets based on duplicates or near duplicates. This is filtered out by comparing threshold $T_{nearDuplicate}$ value against the distance values of the smallest ten distances of the similarity set.
- **Large sets filter:** Reducing the number of similarity sets by excluding all similarity sets which would have too many close members. This is done by comparing the maximal distance value against $T_{maxValue}$.
- **Distance derivative filter:** The size of similarity sets is reduced using distance derivative as a filter. All distances within a set can be ordered from lowest to greatest, and objects cut-off based on a steep increase of the distance value within the distance distribution. A $T_{distanceDerivative}$ is compared against the ratio of two consecutive distance values.
- **Similarity set clustering:** After executing all previous filtering steps, the number of duplicate sets can be reduced using clustering. The remaining similarity sets can be viewed as edges in a graph of patches, and these sets can be merged together using independent component graph analysis [21].

  Let $G(P, S)$ be a graph where P is a set of patches and S is a set of similarity sets

  $$\{G^1, G^2, \dots G^n\} := connectedComponentAnalysis(G)$$

  then removing of too small/large sets leads to the final set of similarity sets

  $$S' := \{\, G^i(V^i, E^i) |\, T_{clusterMin} \leq |V^i| \leq T_{clusterMax}\}$$

  As an alternative, frequent pattern mining algorithms like Market basket analysis [1] can be used to perform a similar task.
- **De-duplication:** The set of similarity sets S' can be turned into a function which scores all entities in the database (by evaluating a multi-query search in the distance space). This means we could compare them by the rankings/permutations they produce. However, this would not work for pre-existing attributes in the schema, which are not distance-based or ranking-based. We can employ the point-biserial correlation r_{pb} [51] as an correlation indicator between a pre-existing dichotomous variable/attribute and a distance-based continuous variable

  $$r_{pb} = \frac{M_1 - M_0}{s_n}\sqrt{\frac{n_1 n_0}{n^2}}$$

  where n is the total number of entities, n_1 is the number of entities having the flag and M_1 is the mean value of their distance, n_0 is the number of entities not having the flag and M_0 is their mean distance. s_n represents the standard deviation of the distance values, and it is a member to normalize the results. We can find the most correlated existing attribute for each similarity set and ignore the similarity set if there is a perfect correlation. On top of that, correlation to existing attributes can be further used as a mechanism to rank similarity sets based on what value they could provide to an existing database. The exact same principle and calculation can be done and compared against n-tuples (pairs, triples) of existing attributes. As an example, Fig. 4 illustrates a proposed attribute for blue jeans. The original dataset does not have any definition of that, but it does know a category “jeans” and a tag “blue”. The visual information correlates with that pair better than with any of the pair’s individual components.

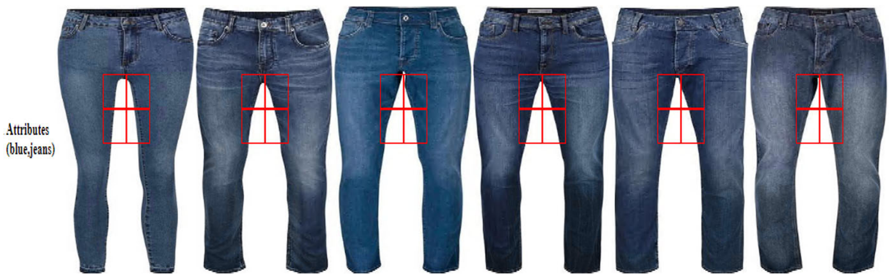


Fig. 4 Example of a discovered visual attribute correlated with two existing attribute values—“blue” and “jeans”

After these five steps, we end the offline pre-processing phase and have a list of what we call **attribute candidates**.

The complexity of the pre-processing phase is dominated by the similarity self-join calculated on all patches from the entire database. This is a super-linear operation whose efficiency depends on the distribution of values in the distance space [8]. These candidates are meant to be reviewed in a system-supported workflow, accepted, and easily integrated into existing database schema.

3.5 Online acceptance by human actors

In our model, human is an essential last step to complete the workflow. For the complete workflow, we define the following actors:

- **Database architect** is the data professional responsible for database schema modelling and application usage of the new attributes which the system will generate. They are also expected to select appropriate feature extractors and distance functions from the available palette given his/her knowledge of the application domain and initial trial runs to fine-tune hyperparameters.
- **Domain administrator** (also denoted as domain expert) understands the domain of the database and the entities it contains, and they do not need to have technical knowledge about the system’s software. Their task is to go trough proposed attribute candidates, evaluate them and accept or reject them. Part of the acceptance is also appropriate naming of new attributes.
- **End user** utilizes new attributes in a transparent and integrated way, not having to differentiate between pre-existing attributes and the ones provided by the system. Usages of the attributes are application-specific and can, for example, cover searching, comparison or recommendation. By using the application, end user is also implicitly providing feedback for the system, e.g. by search history, basket contents, page operations or time spent on different pages. In Sect. 4.7, we elaborate how this information can be utilized as crowd-based evidence to further improve the process and reduce the workload of the domain administrator.

3.6 Integration of accepted attributes

It is the domain administrator who contributes to the last step completing the workflow. Discriminative power of human vision can be used to quickly filter out noisy or irrelevant attribute candidates and to review promising ones. Detailed review of an attribute candidate evaluates the before-mentioned multi-query search across the entire database to show how the attribute applies to it. It is left to the domain expert to filter out noise entities, blacklist/whitelist entire categories, find a distance threshold for the proposed attribute candidate and provide a new name for it. For a given similarity set and an entity $e(id, a_1..a_n, img)$, $e \in Entities$, the **attribute distance** f calculates the average value between the entity's image img and each of the members in the similarity set. The value for entity's image and one patch is then calculated as the minimal patch-distance δ between that query patch and any of the extracted features from all patches of the image. The attribute distance f between an entity e and an attribute candidate s is defined as follows:

$$f(S_i, e(id, a_1..a_n, img)) := \frac{1}{|S_i|} \times \sum_{\forall q \in S_i} \min_{\forall p \in Patch_{img}} \delta(q, extraction(p))$$

where $S_i \in S'$ is the similarity set representing the attribute candidate $candidate_i$, q are all its features obtained from patches, $extraction$ is the selected feature extractor and δ is the patch-distance function.

The accepted attribute is then persisted in the database schema and made available in different forms for application uses. In Fig. 3a we show the E-R diagram for capturing information about accepted attributes. The entities for $Products$, $ProductCategories$ and $Category$ are expected to already exist in the database. The other tables are filled by the system at the time of attribute acceptance with the following semantics and example instances:

– **VisualAttributeDefinition** represents the new visual attribute. It contains a domain-expert-provided name, their subjective quality evaluation, the original attribute candidate as a compact and repeatable search query, and a distance threshold that was selected by the domain expert.

ID	Name	Quality	Candidates	DistanceTreshold
42	Flower pattern	9	Img:mpn961-6078.jpg;Patch:6x8@27	1.7464

– **ProductAttributes** is a linkage table between the new attribute and the database entities. It also provides the calculated distance for ranking purposes and a coverage within the image calculated based on the entity image's patches matching the attribute.

Product	Attribute	Distance	Coverage
143933	42	1.7270	20

– **DiscardedProducts, DiscardedCategories** represent manual filtering done by the domain expert before accepting the attribute.

Attribute	DiscardedProduct
42	227680

Attribute	DiscardedCategory
265	accessories

The described schema not only allows to define different views for application usage, but also contains necessary information to re-apply the attribute in the context of dynamic databases, where new products are being added. This is described in more detail in Sect. 4, including SQL-based VIEWs that utilize this schema.

4 System architecture

In this section, we focus on the software engineering aspect of the solution. We present the system architecture, describe different components involved and present specific implementations used in our experiment. We also highlight what components of system's architecture are designed to be easily replaced by different solutions. It emphasizes that core of the work is done on the database level, which is an application agnostic concept. By extending the database schema, a range of applications can benefit from it at the same time. The following text refers to application components as they are illustrated in Fig. 3b.

4.1 Extraction of multimedia descriptors

As described in Sect. 1, our motivation for attribute discovery comes from recent advances in computer vision community. The advances have been boosted greatly by the deep learning revolution in recent years and delivered many solutions to different tasks. These solutions are not only black boxes approximating a specialized function, but also have a valuable internal structure. As various authors have shown [14,48,58], data from the internal structure can be extracted and used as high-dimensional feature descriptors. Specifically, for deep learning-based techniques, a standard mechanism to generate feature descriptors is to extract values of neuron activations from a certain layer of the network after a forward pass of input data. For the sake of simplicity, we have selected AlexNet [25] in our experimental setup. AlexNet is a pioneer architecture for deep convolutional neural networks (DCNNs) developed in 2012 for the classification task at the ImageNet competition [13]. Since then, it was shown many times that the generalizing effect of DCNN architectures allows to utilize the activations of the internal DCNN layers also for different tasks, such as image retrieval [39]. Recently, it was shown [45] that deep models pre-trained on generic imagery (such as AlexNet) could be successfully used even for representation of artificial images being very different from images used for training (such as visualizations of non-visual data). This observation is rooted in the convolution operation that gradually aggregates lower-level concepts into higher-level ones, enabling thus options for visual-semantic feature extraction.

As new models are created, this component of the system architecture can be easily replaced. Both by new models pre-trained on different image datasets, and by novel architectures and computer vision solutions, as long as there is a possibility to extract feature descriptors. Although analytical solutions like SIFT [37,60] or PCT signatures [26] could be used as an implementation for this step, it is the convolution operation which delivers the ability to reveal and describe concepts of higher level of abstraction.

4.2 Generation of image patches

To support different levels of attribute granularity, our solution operates not just on entire images of entities, but also on segmented patches. As shown in [46], the usage of image patches improves retrieval quality in various search tasks and makes use of them in attribute discovery as well. In our experiment, we have started with a naive approach of segmenting input images using regular grids. Expanding on our previous work [46], segmentation was done using two regular grids, 3×4 and 6×8 in size. This has produced 60 fine-grained patches per input image in total, and offering two different levels of size granularity. This naive approach was possible as the selection of segmentation algorithm is orthogonal to the other parts of the processing pipeline and the underlined dataset contains canonical product photographs.[3] For more complex scenarios (e.g. street-level photographs), the segmentation algorithm can be simply replaced by some more advanced technique (e.g. the ones surveyed by Liu et al. [31]).

During our experiments, we also evaluated the usage of inner AlexNet's overlapping windows in convolutional layers as a possible source of image patches and their descriptors. However, the demonstrated quality of the results was too low to justify this choice.

For each of the created patches, feature descriptors were extracted using the same mechanism as in our previous works [46,48]. Based on preliminary results, the neuron activations of convolutional layers 3, 4 and 5 followed by a max-pooling step were chosen for the experiments with domain experts. Other configurations (e.g. initial convolutional layers and fully connected layers) were excluded due to the poor quality of generated attribute proposals. This resulted in 384-dimensional vectors for the third and fourth layers, and 256-dimensional vector for the fifth convolutional layer of AlexNet.

4.3 Constructing similarity sets

As the next step, we ran the approximate kNN self-join operation to initialize the approximate similarity sets construction. In our experiment, we have worked with 19,172 product images. With 60 patches per image, we generated 1,150,320 intermediate images. A non-optimized similarity-based retrieval would have to calculate distance between all possible pairs—1,323,236,102,400 distance calculations. A full similarity graph is computationally not a feasible solution, and we had to trade off full precision for pre-processing time. As similarity sets only contain patches that are similar to each other, we chose the restriction of retrieving 512 nearest neighbours for each patch and calculated a kNN-self-join using MapReduce on the Hadoop environment [8]. Note that the restriction of 512 neighbours only affects the construction of similarity sets, which later defines an attribute candidate. The volume of nearest neighbours was chosen as the largest value we are able to calculate and store results for. The attribute candidate can then be turned into a multi-query and executed against the entire dataset as needed.

[3] All images were centred, oriented in the same direction, maintained the same product size ratio and had flat white background.

4.4 Implementing noise removal

We have defined attribute candidates using sets of image patches, acting as a constructor for multi-example query. This produces a function which can score entire dataset using a distance measure. However, not every produced candidate follows what humans understand as an attribute, and intermediate results after applying approximate similarity self-kNN-join contain too much noise and duplicates.

To reduce the noise, we implemented a pipeline of filtering steps based on distance values, distance distribution within a kNN result, and based on symmetrical property enforced using reverse-kNN-lookup. These heuristical filtering steps allowed us to reduce the number of proposed candidates by 85% and the average number of patches contributing to a candidate by 99%.

To eliminate duplicates and further decrease the number of intermediate results, we employed clustering using graph decomposition into independent connected components. To maintain the discriminative and representative properties of attribute candidates, the resulting clusters were further refined using threshold hyperparameters based on the size of a cluster, $T_{clusterMin}$ and $T_{clusterMax}$. Depending on the selected configuration, the final number of proposed attribute candidates was on average 0.3% of the original size of kNN-self-join.

At last, existing attributes already present in the dataset were cross-checked against all attribute candidates using Point-Biserial correlation measure. This acts both as a filtering measure to remove information already contained in existing data, and a ranking mechanism to sort candidates by additional value they could provide over existing schema.

4.5 Extracting frequent patterns

Within our candidates for visual attributes, same database entity might be covered using multiple candidates at the same time. Implicitly, this builds a relationship graph between entities using co-appearance in an attribute candidate as edges between the nodes. If an end user is looking at entities A and B, can we infer any new knowledge using this graph? Not surprisingly, this question is very similar to problem definitions in the recommender systems community and there are existing tools to solve it. One class of solutions for rule inference is called “frequent pattern mining” [28] and Market Basket Analysis [1] is one of its representatives. In order to fit the technique, we had to transform our data. Attribute candidates become baskets, and entities covered in it (not their image patches) become items of the basket. The output of the algorithm then produces association rules in the form of functions

$$(antecedent) \mapsto (consequent, confidence \in \mathbb{R})$$

where both antecedent and consequent are sets of items (database entities in our transformation).

This generates an additional insight from visual data on top of database schema extension, with applicability in application-specific use-cases like product recommendation.

4.6 Interfaces for domain administrator

In section 3, we have defined the basic actors of our methodology. For accurate integration of new attributes, domain administrator is the most important actor. For larger databases, the domain awareness (the knowledge of domain expert) would become a scalability bottleneck. To overcome it, a grouping of the database into sub-domains based on pre-existing information would be needed. The methodology assumes domain knowledge, and different actions are supported by the software architecture. We have developed an administration user interface where domain administrators can:

– See an initial view with proposed attribute candidates together with the patches defining them.
– Reject candidates from further considerations, as an explicit feedback to the system.
– Expand proposed candidates to see how they are applied to other entities in the dataset.
– Accept an expanded attribute up to a certain threshold while using available mechanism to filter out possible noise.

The process for accepting attribute candidates by the domain administrator begins with an initial display. Pre-processed attribute candidates are loaded from persistent storage (candidates that were accepted or rejected in the previous sessions are filtered out). Displaying of the attribute candidates is done using thumbnails of their defining images and highlights of all relevant patches. This display usually carries sufficient information to assess an attribute candidate and quickly reject it if needed. The most common reasons for rejection were:

– Not meaningful as an attribute.
– Already covered as an existing category.
– Too trivial, e.g. just a colour feature.

In other cases, the attribute candidate can be expanded. This shows application of the attribute to the dataset, being ranked via a multi-query distance function as described in Sect. 3.6. The display also highlights patches that caused the attribute to be applied. The task of the domain administrator is to find a threshold for applicability of the attribute in the ranked list. In order to exemplify the task, human judge is simply asked to select the last (most distant) item that still complies with the intended attribute description. The distance of this item is considered as the threshold value. Additionally, to filter out possible noise, human judge has the following tools at hand:

– Filter out individual images.
– Black-list entire entity categories using existing relational data.
– White-list some of the existing categories and hide everything else.

When an attribute is accepted, the acceptance information is immediately persisted and integrated into the database. On top of the ranked list of entities from the dataset falling below the threshold, this also includes calculated distance value, coverage of matched image patches for each image, and information about applied filters. This information is then utilized in different views operating on the data, as well as during ingestion of new entities to the database and determining relevance of already applied attributes.

During ingestion of new entities to the system, the first part of the pipeline is executed as is—images need to be segmented into patches, and feature descriptors extracted. After that, extracted descriptors can be evaluated against all already accepted attributes. The system remembers the distance threshold per each attribute and can therefore use it as a decision criterion after calculating the distances between descriptors belonging to the new entity and the descriptors of patches defining the attribute. On top of that, whitelist and/or blacklist applied by the domain administrator can be also applied against known information about the new entity.

For large relative changes to the dataset, entire pipeline should be repeated as new entities can reveal new patterns in the data. This does not drop previous work done by the domain administrator; it is a mere generation of additional attribute candidates. The same de-duplication technique based on correlation ranks is to be used to reduce the amount of attribute candidates.

4.7 Utilisation of system-collected implicit feedback

Our model so far has been modelled around unsupervised generation of attribute candidates and final acceptance by domain administrators. In the context of long-running web enterprises, however, a lot of knowledge about relations between items can be exposed in collected implicit user feedback (i.e. historical co-consumption data) . This implicit feedback can take many forms—search history, shopping basket contents, stream of visited objects, images displayed in full size to end users and many others.

The aggregated implicit feedback is often utilized, for example, in collaborative recommender systems [22] to suggest additional potentially relevant items to the users. The working hypothesis of collaborative recommender systems is that users tend to pursue their interests consistently to some extent, and therefore, if users shared interests (e.g. mutually visited objects) in the past, they should share them in the future as well. In our use-case, we follow a similar hypothesis by considering that a portion of user's visits reflect his/her interest in some visual attribute. Although such information tends to be very noisy in individual instances, several techniques for frequent patterns extraction, e.g. Market Basket Analysis [1,44], were quite successful in de-noising this type of data.

While the collected frequent feedback patterns does not necessarily contain visually related objects, we assume that a portion of them would possess such property, and in addition, there is a certain level of guarantee that users took interest in this particular collection of items. Therefore, we can use the same attribute suggestion pipeline, but limit the set of possible attributes to the pre-filtered collection of items that frequently go together. That is, instead of using a full KNN-self-join over the entire database to generate similarity sets, we can restrict the search to operate just on prepared subsets of data. Rest of the data processing pipeline will ensure that the images are pushed through feature extraction, segmented, patterns of image patches are identified, cleaned and aggregated. This will again produce attribute candidates, but this time further supported by a crowd-based evidence.

4.8 Integrating schema augmentation into SQL environment

The last step of our schema extension pipeline is the integration of new attributes into the original schema of the database. This happens in real time when the domain administrator accepts a new attribute by filling in data structure presented in Fig. 3a. The normalized schema is optimized for data maintenance, but is not optimal for attribute querying. For the purpose of data querying, we defined views that present the new attributes in a way that can be better consumed from various applications such as recommendation, search or display functionalities of a system.

The information captured during database schema augmentation goes beyond binary tags (named labels that are either present or missing). The acceptance process transparently captures additional information that is able to provide numerical information for querying purposes. The first additional information is the number of patches matched during attribute acceptance phase, expressing the coverage of a certain attribute. A database user is then able to rank or filter objects according to their ratio, such as "retrieve all objects ranked by their floral pattern coverage". A second indicator is the distance between the attribute definition and the object itself, which is a decimal number. For normalisation purposes, we provide a view which turns the distance values into a percentile. This can be used to define, for example, "TOP N attributes for an entity" using the distance percentile as a ranking criteria across different attributes. In the following text, we demonstrate the integration of collected attributes into an existing SQL database environment. The examples are written for Microsoft SQL Server 2017, and the query constructs work with minor changes in any database supporting the SQL:2003 language standard.

First query shows how to normalize distance values using distance percentile within each attribute instead of using the raw distance value between an object and a visual attribute definition. The distance percentile is calculated using the $PERCENT\_RANK()$ window function. This is used as a basis for other queries and views.

```
CREATE VIEW [dbo].[VW_Attribute_Distance_Percentiles] AS
SELECT
    p.ID as ProductId
    ,va.Name
    ,PERCENT_RANK( ) OVER (
        partition by va.ID order by pa.Distance DESC
    ) as DistancePercentile
  FROM Products p
  JOIN ProductAttributes pa ON pa.ProductId = p.ID
  JOIN VisualAttributeDefinition va ON va.ID = pa.AttributeId
  WHERE pa.Distance <= va.DistanceTreshold ;
```

The following query uses the normalized distance information to pick top 5 attributes for each product in the database. Since each relation between attribute and product is originally represented as a separate row in the database, the $PIVOT$ clause is used to turn top 5 attributes into individual columns instead. Figure 5a illustrates section of possible results.

```
CREATE VIEW [dbo].[VW_Top5_Attributes] AS
WITH BestAttributes AS (
  SELECT attr.ProductId, attr.Name,
    ROW_NUMBER() OVER (
      PARTITION BY attr.ProductId
      ORDER BY MAX(DistancePercentile) DESC
    ) as #OrderWithinProduct
  FROM VW_Attribute_Distance_Percentiles attr
  GROUP BY attr.ProductId,attr.Name
)
SELECT * FROM BestAttributes
PIVOT(
MIN(NAME)
FOR #OrderWithinProduct IN ([1],[2],[3],[4],[5])) ba;
```

The last code snippet shows how all attributes can be queried at the same time, in the form of a matrix of products and attributes. In the result set, each row represents 1 product and each column 1 attribute. This is achieved using dynamic SQL, where all attributes are first queried from the attribute definitions and then turned from rows into columns using the SQL *PIVOT* clause. The end result is closest to standard database representation of an attribute and makes it natural to express search intentions like “Find me converse-style shoes that are colourful, ordered by price”. A part of the result is illustrated in Fig. 5b.

```
BEGIN
    DECLARE @cols AS NVARCHAR(MAX),
        @query  AS NVARCHAR(MAX);

    SET @cols = (
            SELECT STRING_AGG('['+NAME+']',',')
            FROM VisualAttributeDefinition
        );

    set @query = '
    select * from VW_Attribute_Distance_Percentiles
    pivot(
    max(DistancePercentile)
    for Name in ('+@cols+')
    ) p';
    execute(@query);
END
```

5 Evaluation

In this section, we describe the evaluation process of the proposed model and discussed the observed results. First, we briefly introduce the underlying dataset, we use as a base for the evaluation; then, we focus on the offline evaluation, which could be seen as a pre-processing while selecting model’s hyperparameters. Based on the offline evaluation, we selected several variants of the model, which were further evaluated in a user study.

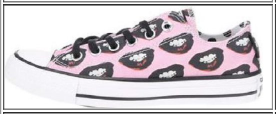
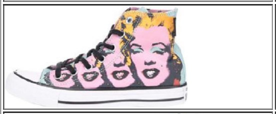
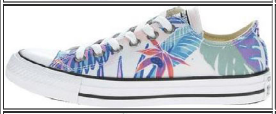
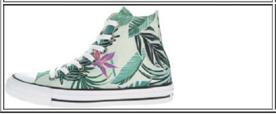
5.1 Dataset

The evaluation dataset consists of the objects available in two major Czech retail shops,[4] both focusing on the fashion domain. The range of the dataset is relatively broad. Besides for the major categories of shoes and clothes, there are also some members of accessories (e.g. watches) or home appliances categories present in the dataset. In total, the dataset contains **19,172** objects, organized into **215** categories and further described by **711** tags. For the purpose of the research, we do not distinguish between tags and categories (as they were primarily distinguished for the navigation purposes in the original websites) and denote them uniformly as original attributes. Figure 6 displays a histogram of attribute-product coverage. We can see that there are a few densely covered attributes and a large volume of long-tail attributes with only a handful of products.

[4] Bata and Zoot.

ImgUrl	1	2	3	4	5
	women jeans - all styles	womens skinny jeans	women jeans	blue jeans	Jeans
	womens dress - black color	womens dresses - black color	black dresses	Dark dress	black dress - all kinds
	Dark dress	womens dress - black color	black dress - all kinds	black	Short sleeve ('d black woman dress)

(a) Query results for top attributes of three selected entities.

IMG	Price	Ala-converse	Colorful
	1539 CZK	p1	p0.72
	1609 CZK	p0.91	p0.93
	1989 CZK	p0.5	p0.33
	2089 CZK	p0.59	p0.25

(b) Section of the tabular view on products times attributes, with percentile ranks as cell values.

Fig. 5 Graphical results of queries using our provided database views combining product information and collected visual attributes

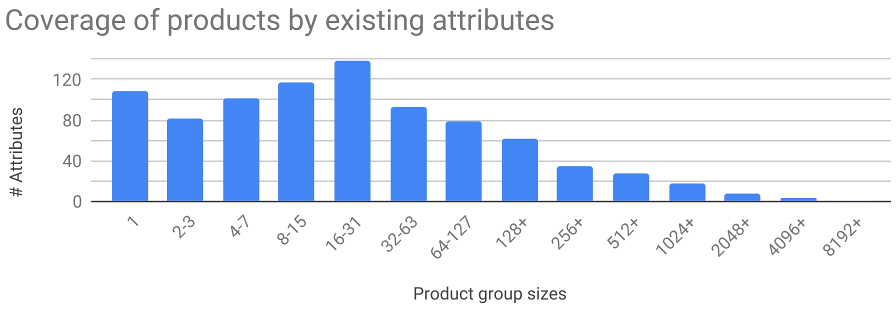


Fig. 6 Frequency histogram showing pre-existing categories/tags and how many products they typically cover. The number of covered products is on the *x* axis, the *y* axis shows how many attributes (tags, categories) fall to that respective coverage range. Vast majority of such attributes covers less than 1% of the database

5.2 Offline model selection

In Sect. 3, we introduced several hyperparameters of the considered model, those were the $feature\ extraction$ method and the thresholds for near-duplicate filters $T_{nearDuplicate}$, large sets filter $T_{maxValue}$, distance derivative filter $T_{distanceDerivative}$ and finally minimal and maximal similarity set cluster sizes $T_{clusterMin}$ and $T_{clusterMax}$. This induces a relatively broad parameter space, which is impossible to be evaluated directly by domain administrators. Therefore, we introduced an offline evaluation step in order to pre-process the suitable variants of the model parameters. In the offline pre-processing phase, we performed a grid-search evaluation as follows. As a feature extraction method, we utilized activations of one of the last three convolution layers of the underlying DCNN (denoted as $conv3$, $conv4$ and $conv5$) and the cosine distance metric. The $T_{nearDuplicate}$ and $T_{maxValue}$ hyperparameters were held constant at 0.3 and 0.95, respectively. These constants were defined manually by observation of the distance distribution in the space of descriptors created by $AlexNet$ layers and the Euclidean distance, with the intention to filter out clear near-duplicates. $T_{distanceDerivative}$ was selected from (0.76, 0.99) range, $T_{clusterMin}$ was selected from {2, 4, 6, 8, 10, 12} set and $T_{clusterMax}$ from {25, 50, 100, 200, 400, 800, 1600} set. In the rest of the paper, the hyperparameter settings of the model will be denoted as $M(fe, T_{distanceDerivative}, T_{clusterMin}, T_{clusterMax})$, where fe denotes the feature extraction method. It is important to note that this setting only affects the search for attribute candidates, i.e. sets of patches used as a pattern across the entire database. The ranges of possible values were chosen based on preliminary results which excluded lower/higher values because of degenerated results.

Usually, the offline evaluation would consider some variants of maximization of user (domain administrator) engagement per model variant, such as the ratio of accepted attributes per model or similar. However, as we did not have any such feedback upfront, we resorted to evaluate some of the aggregated properties of the proposed attribute sets. To be more specific, for each model settings $M(fe, t_1, t_2, t_3)$ we considered the volume of proposed attributes a_M and the overall coverage of items from the dataset c_M.

First, we expect a good model to reasonably cover the whole dataset (i.e. that there is at least one attribute candidate for most of the items). Second, in order to decrease the workload of domain administrators, we wanted to minimize the total volume of candidate attributes.

Therefore, while evaluating the suitable candidates, we kept only the model variants, which had a coverage $c_M > 10000$ items, i.e. 50% of the dataset. Out of the remaining models, we performed a greedy selection based on the c_M/a_M ratio, while maintaining a sufficient level of diversity among the selected models. The last condition was introduced because the offline evaluation metric is provisional at best, and therefore, we opted for the chance to evaluate a broader range of hyperparameters in the user evaluation phase. To guide with this subjective selection, several other metrics calculated offline were considered: average size of attribute candidates, average number of matching patches per image, distance variance within attribute candidates or maximal distance within candidates.

Finally, Table 2 depicts the model variants that were selected for the user evaluation.

5.3 User study

In order to evaluate the usability of the suggested visual attributes, we conducted a user survey focused on confirmation or rejection of the candidate attributes proposed by individual variants of the model.

Table 2 List of model variants evaluated in the user study

$feat.extr.$	$T_{dist.Der.}$	$T_{clusterMin}$	$T_{clusterMax}$	$\#candidates$	$coverage$
conv3	0.88	8	400	1531	10,584
conv3	0.91	10	400	1251	10,086
conv3	0.96	10	50	1517	10,032
conv4	0.89	8	800	1776	11,901
conv4	0.91	12	800	983	10,418
conv4	0.94	12	400	1096	10,235
conv5	0.76	6	50	2176	10,719
conv5	0.80	8	800	1270	10,024
conv5	0.92	8	200	1725	10,071

5.3.1 Evaluation protocol

First, the participants were informed about the general mission of the project and their role as the domain administrators. Participants were instructed that their task is to find visual attributes that contain relevant information for the end-users. As the means of utilization, content-based recommendation and "showing more of the same attribute" use-cases were mentioned. Participants first familiarized themselves with the GUI of the system on some sample data. Once the preliminary steps were done, participants were asked to evaluate suggested attributes of 9 different variants of the model selected based on the offline evaluation.

For each of the model settings, a list of 75 candidate attributes were shown to the users. For the set of candidates, the system checked they did not correlate with existing attributes and then selected the displayed ones at random. New set of candidates was provided upon the page refresh. Each candidate attribute was represented by the corresponding similarity set (Fig. 7A).

Participants were instructed to inspect whether the candidate is potentially relevant or not. They were given several suggestions based on which to reject the candidate, e.g. the candidate merely contains visual noise, it cannot be properly labelled, it is not comprehensible for a human or that the patches are trivial or not visually similar (from human perspective). Based on their evaluation, users could either explicitly *reject* the attribute, explore the detail of the attribute and then *accept* it, or (in the case of uncertainty) ignore the attribute. Ignored attributes would eventually re-appear to other domain administrators, who can make the final decision about them.

While exploring the detail of an attribute (Fig. 7B), the system presents the user with top-k items closest to the candidate attribute as described in Sect. 3.6. Participants were instructed to set the distance threshold, where items still possess the desired feature and optionally restrict the set of eligible item categories or manually remove irrelevant items. Finally, if users were satisfied with the resulting attribute, they were asked to provide a label for it and state the expected usability of the attribute (on a [0, 10] scale).

During the evaluation, we mainly focused on the following questions: Did users consider (some of) the proposed attributes relevant? How difficult is to find some relevant suggestions and what is their quality? Were there any significant differences between individual model settings?

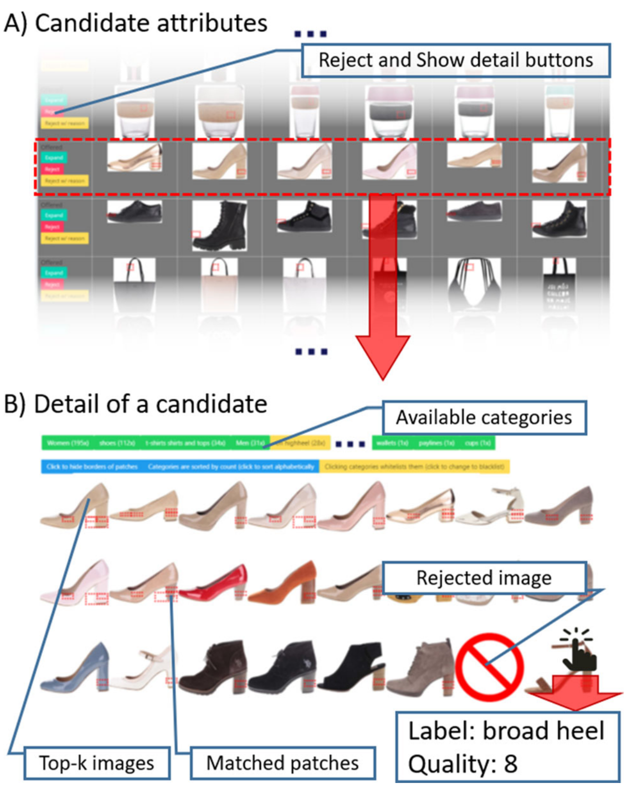


Fig. 7 Screenshot of the administrator tool for attributes confirmation

5.3.2 Results

During the evaluation process, in total 9 users declared in total 218 attributes as relevant and rejected 163 attribute candidates. Some examples of confirmed attributes with high subjective quality are depicted in Fig. 8. The ratio between rejected and confirmed attributes is lower than expected; we supposed that much more attributes will be rejected. However, upon inquiring the participants, they often mentioned that either they were unsure whether there is something relevant on the attribute proposal or not, or that they did not want to loose time with explicit rejections. Therefore, we plan to introduce some implicit rejection model based on observing, but ignoring the attribute [42] in the future.

While considering time span between two consecutive attribute proposals per user and per session, the mean time span is 101.8 s, while the median is 75.5 s. Together with the total volume of attributes, this indicates that users are capable to enrich the database schema via newly defined attributes on a regular basis. Also, at least one new attribute was defined for 6901 items, representing 36.0% of the source database. The mean of user-perceived quality of confirmed attributes was 7.3, while the median was 7 (on the 0–10 scale). To sum up, we may conclude that our approach in general is a viable strategy to enhance the underlying database and it has a sufficient impact on the information retrieval process.

We further focused on several properties of the confirmed attributes, which are depicted in Fig. 9. First, it can be seen that users mostly managed to discover less populated attributes

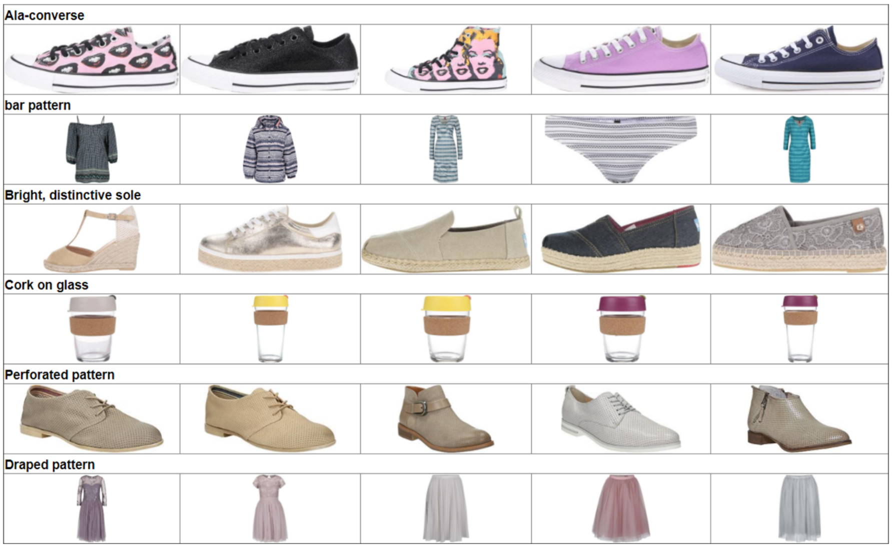


Fig. 8 Examples of confirmed visual attributes and top-5 attribute members

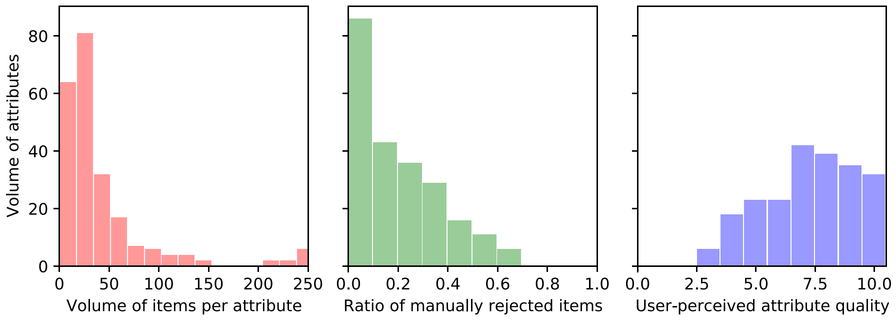


Fig. 9 Histograms of three properties of confirmed attributes. Left: size of the attribute, middle: ratio of rejected items, right: assigned attribute quality score

(the mean of the attribute size is 43.6, while the median is 28). On the other hand, users only rarely removed extensive volume of items manually (mean removal rate is 0.20, median is 0.15 and furthermore for 17% of attributes, no manual removals were made); therefore, we can suppose that the addition of novel items to these attributes can be done with only a minor supervision.

The distribution of the user-perceived quality of attributes is skewed towards better rated ones. We interpret this in the way that users behave in a cost-effective way and do not waste time on defining attributes that are not very helpful for the retrieval tasks. Based on this observation, we may also hypothesize that simple features like the volume of confirmed attributes or the ratios of confirmed and rejected attributes are reasonable proxy metrics of usefulness, while evaluating different model settings individually.[5]

[5] To further support this observation, let us note that we found a relatively large Pearson's correlation ($r = 0.78$) between mean user-perceived attribute quality and the ratio of confirmed attributes, while considering each model setting individually.

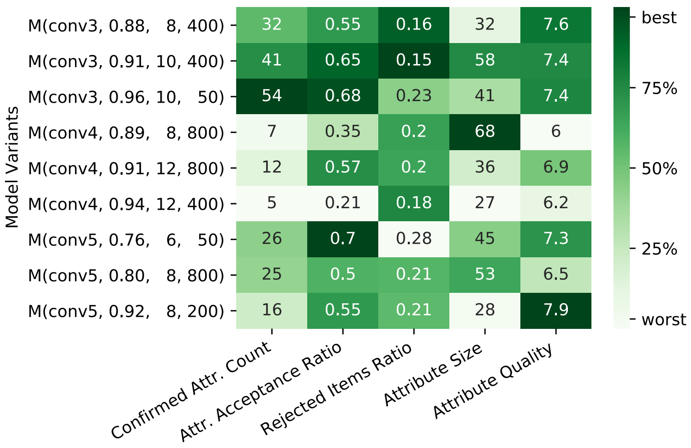


Fig. 10 Evaluation of model variants individually

Finally, we observed that the volume of novel attributes is rather large with rather modest volumes of member items per attribute. Therefore, we investigated the attribute labels provided by the users. We saw some evidence of entanglement with item's category, e.g. "mens underware with distinctive label" or semantically similar labels, e.g. "hood"/"fur hood". For some usage, e.g. displaying "more of the same feature" or content-based recommendations, such entanglement should not be a serious problem, but we plan to explore some options to disentangle the attributes as a part of our future work.

As for the individual models, Fig. 10 depicts results of individual model settings w.r.t. the mean values of several evaluation metrics. For the sake of clarity, we arrange the values in the form of heatmap, where dark green represents the best values per metric and white represents the worst ones. The scale corresponds to the linear min–max scaling of the feature values into the [0, 1] interval.

Seemingly, the results heavily depend on the set of visual features (i.e. $conv3$, $conv4$, $conv5$ layers); therefore, during the evaluation of statistical significance, we mainly focused on the overall effect of utilized DCNN layer. First, both $conv3$- and $conv5$-based models have significantly higher ratio of accepted attributes than $conv4$ (according to Fisher exact test, p values 0.0003 and 0.009, respectively). Similar tendency was found for the mean perceived quality of attributes; however, only the differences between $conv3$ and $conv4$ were statistically significant (t test p value: 0.028).

We also focused on the elapsed time between two consecutive attribute confirmations. The mean values were 94s, 99s and 120s, while median values were 70s, 82s and 98s for $conv3$, $conv4$ and $conv5$ models, respectively. There were several outliers in the data, so we did not observe any significant differences w.r.t. mean values. Nonetheless, the median time between two consecutive attribute confirmations was significantly lower for $conv3$ than for $conv5$ (Kruskal–Wallis test p value: 0.0008). Finally, the difference of the mean ratios of rejected attributes between $conv3$ and $conv5$ was on the edge of statistical significance (t test p value:0.054).

As for the other hyperparameters, the results w.r.t. different metrics were rather contradictory. For instance, while model variants with small clusters ($T_{clusterMax} = 50$) were significantly inferior w.r.t. rejected items ratio, they were superior w.r.t. attributes acceptance ratio compared to other model variants.

To sum up, the evaluation of the individual model variants does not have a clear winner as five out of eight methods are on the Pareto front. Nonetheless, models based on $conv3$ layer, especially $M(conv3, 0.91, 10, 400)$, performed consistently well w.r.t. all evaluated metrics and therefore are good candidates for further experiments. As there was no single best-performing method, we would like to implement some ensemble method, e.g. [5,40], to combine attribute proposals coming from different model variants.

5.4 Comparison with manually created attributes

The user study depicted in the previous section left one important question open. If we let the domain expert to create attributes manually, could the process be more efficient? In order to clarify this point, we employed several domain experts and let them work with a VADET tool [41] designed for manual visual attributes construction. VADET tool allows to filter items based on categories and search for visually similar items based on multiple query items. It also allows to focus the similarity search to certain regions of the queried items. Please refer to [47] for a detailed description of the tool.

The tool was populated with the same dataset as we utilized for the user study in the previous section. Domain experts were first asked to familiarize themselves with the tool and the dataset.[6] Subsequently, users were asked to manually construct new visual attributes that are not yet present in the database. In the rest of this section, we will refer to such constructed attributes as *manual* attributes as opposed to the *semi-automatic* attributes constructed by the pipeline proposed in this paper.

In total, five domain experts participated during the evaluation. Overall they interacted with the system for 4 h and 24 min and constructed 41 visual attributes. The mean time to construct a *manual* attribute was 6 min and 27 s, which is considerably higher than 1 min 42 s required in average for constructing an attribute in a *semi-automatic* manner. *Manually* created attributes also contained less items than *semi-automatically* created attributes (19.65 vs. 43.6 in average). Therefore, we can conclude that the proposed semi-automatic pipeline is more efficient than the manual constriction of visual attributes.

Next, we assume that the typical domain expert would first like to manually construct those attributes that are already in his/her mind and only then browse through the visual attributes suggested by the proposed pipeline for additional ideas. For such use-case, it is important that the suggested attributes are sufficiently diverse from the manually created ones. In order to evaluate this feature, we focused on the level of similarity between *manually* and *semi-automatically* created visual attributes.

Specifically, for each *semi-automatic* attribute, we selected the most similar *manual* attribute w.r.t. Jaccard similarity. Histogram of these similarities is depicted in Fig. 11. It can be seen that although several of the attribute pairs have relatively high similarity (up to 0.54), the vast majority of *semi-automatic* attributes does not correspond to any *manually* created attribute. We can therefore conclude that the proposed pipeline provides sufficiently novel visual attributes as compared to the ones created manually.

[6] A separate instance of VADET tool was utilized here; interactions from this step are not included in the evaluation.

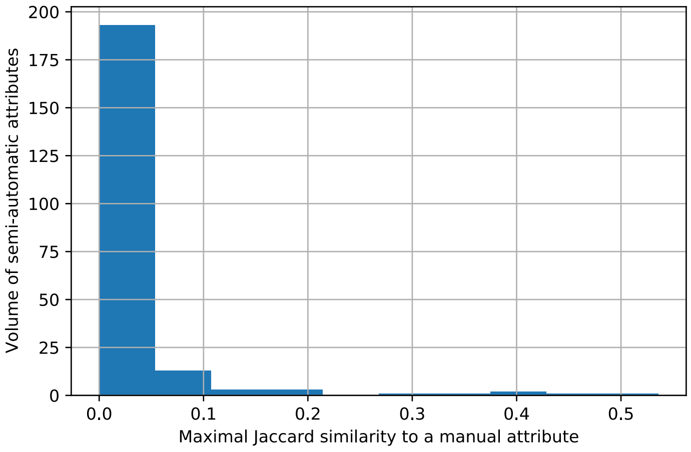


Fig. 11 Histogram of Jaccard similarities between *semi-automatic* attributes and their most similar *manually* created counterparts

5.5 Attributes assignment for novel products

Apart from defining visual attributes, it is also important to maintain the product–attribute relations up to date in the dynamic databases such as e-commerce. Therefore, in a follow-up user study, we evaluated whether the created visual attributes can be applied to the newly added products as well. In order to do so, we constructed an updated version of the products dataset corresponding to the available products as of November 2019 and identified novel products compared to the previous dataset's version (May 2019). Then, for all visual attributes and new products, we evaluated whether the attribute fits for the product (i.e. attribute distance f is within the threshold value and white/black list conditions are satisfied). In this fashion, we constructed a list of attribute–product candidates and asked users of the previous study to either confirm or reject these suggestions. In evaluation, we focused on the ratio between confirmed and rejected suggestions as well as on conditions affecting the confirmation probability.

First of all, we need to note that the evaluation was negatively impacted by the fact that the list of categories is largely incompatible for both dataset versions. This included renaming existing categories, introducing some new ones and also some hierarchy modifications. In an online settings, this problem can be solved during the category construction/modification process, but in our evaluation settings, this condition hindered the applicability of black/white list filtering considerably (e.g. a significant portion of attributes with white-list conditions did not match to any new items and more than a half of the rejected suggestions could have been filtered out by a black-list condition, if the categories were not changed). Despite the reduced impact of the evaluation, we were still able to draw some conclusions.

In total, 122 suggestions were accepted and 101 rejected. We also identified several conditions significantly affecting the acceptance probability. First we focused on how distant were new items from the source similarity set (distances were normalized by the threshold defined during attribute construction). Accepted product–attribute pairs had in average lower normalized distance f than rejected ones (0.967 vs. 0.980; t test p value: 0.0002). Furthermore, attributes corresponding to the accepted suggestions had in average less manually discarded objects in the past (6.9 vs. 10.8; t test p value: 0.0007). Finally, the chance of

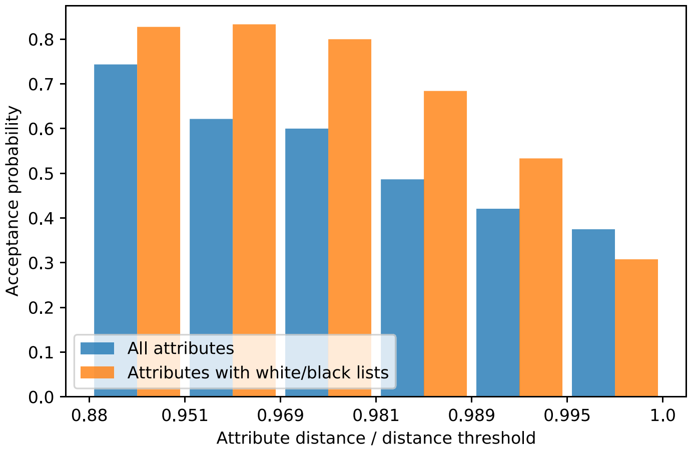


Fig. 12 Aggregated acceptance ratios w.r.t. relative attribute distance

acceptance was higher, if the attribute was restricted by some white/black list (Fisher's exact test p value:1.5e−5). Surprisingly, user-perceived quality of an attribute had no significant impact on the acceptance probability.

Figure 12 depicts the dependence of acceptance ratio on relative product–attribute distance. It can be seen that the further away from the threshold distances are, the higher is the acceptance ratio. Nonetheless, the probability is still far away from values that would allow us to automatically confirm the product–attribute assignment without the human supervision. On the other hand, the confirmation task was relatively easy with a median time to decision equal to 1.98 s.

To sum up, attribute assignment evaluation for novel products showed that although the visual attributes are not distinguishing enough to allow automated assignment, majority of suggestions are accepted and the acceptance probability can be well modelled through several descriptive features of product–attribute relations. Therefore, visual attribute assignment can be approached as an attribute recommendation component incorporated into the new product insertion process (i.e. list of recommended visual attributes will be displayed for each new product, where the most probable attributes may be preselected).

6 Conclusion

In this work, we have presented the problem of schema augmentation and attribute discovery. We have described a formal methodology to achieve it, implemented a system architecture and evaluated the results in an user study situated in the context of e-commerce fashion products. The results show that domain administrators were able to quickly pick up proposed attribute candidates and extend original database schema by many attributes in a relatively short time. We believe that an important contribution is applicability to established SQL databases and the ability to start without any training data or labelling phase, as well as the options to leverage advances in novel computer vision architectures and trained models. The modularity of the proposed architecture allows for the system to grow as individual components can be replaced, such as the image patching or feature extraction solutions.

6.1 Future work

As our future work, we would like to further research on the applicability of the proposed methodology.

First axis of the applicability are problem domains. Our work is best suited for domains with strong visual element, which also represents an important deciding factor in information retrieval. For example, furniture, travel, art, decorations or posters could be interesting candidates for further investigation and experiments. On the other side of the spectrum, there are domains where visual data exists, but either does not carry much information for user's decision-making process or such information cannot be clustered into visual attributes. Although such distinction is subjective to some extent, we would not recommend our work, for example, for the domains of books, music records, software products, real estates or most areas of computer components.

Second axis would comprise from evaluating the added value of constructed visual attributes for end-users. Attributes may be directly disclosed to users via some searching or browsing GUI, or utilized, for example, in inner models of recommender systems. We would like to target both the changes in user's navigational patterns and/or changes in overall consumption statistics via some A/B testing.

As third axis, we would like to explore a possible transition to other forms of multimedia beyond images. The deep learning revolution did not affect just image processing; it significantly improved state-of-the-art techniques in many other disciplines. With feature descriptors extracted from data sources like videos, sounds or texts, similar methodology could be applied to extract common patterns out of the data and turn it into structured attributes.

Acknowledgements This research has been supported by Czech Science Foundation (GAČR) Projects Nr. 17-22224S and 19-22071Y.

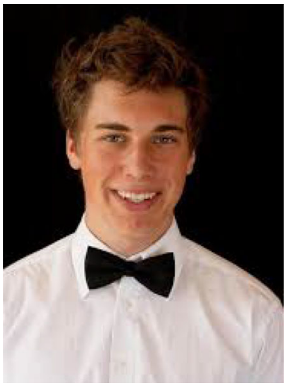
References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th international conference on very large data bases, VLDB '94, pp 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA. http://dl.acm.org/citation.cfm?id=645920.672836
2. Baltrušaitis T, Ahuja C, Morency L (2019) Multimodal machine learning: a survey and taxonomy. IEEE Trans Pattern Anal Mach Intell 41(2):423–443
3. Beecks C, Skopal T, Schoeffmann K, Seidl T (2011) Towards large-scale multimedia exploration. In: Das G, Hsristidis V, Ilyas I (eds) Proceedings of the 5th international workshop on ranking in databases (DBRank 2011). VLDB, Seattle, WA, pp 31–33
4. Berg TL, Berg AC, Shih J (2010) Automatic attribute discovery and characterization from noisy web data. In: Daniilidis K, Maragos P, Paragios N (eds) Computer vision—ECCV 2010. Springer, Berlin, pp 663–676

5. Brodén B, Hammar M, Nilsson BJ, Paraschakis D (2018) Ensemble recommendations via thompson sampling: an experimental study within e-commerce. In: 23rd International conference on intelligent user interfaces, IUI '18, pp 19–29. ACM, New York, NY. https://doi.org/10.1145/3172944.3172967
6. Budikova P, Batko M, Zezula P (2017) Fusion strategies for large-scale multi-modal image retrieval. Springer, Berlin, pp 146–184. https://doi.org/10.1007/978-3-662-55696-2_5
7. Čech P, Grošup T (2015) Comparison of metric space browsing strategies for efficient image exploration. In: 2015 13th International workshop on content-based multimedia indexing (CBMI), pp 1–6. https://doi.org/10.1109/CBMI.2015.7153631
8. Čech P, Maroušek J, Lokoč J, Silva YN, Starks J (2017) Comparing mapreduce-based K-NN similarity joins on Hadoop for high-dimensional data. In: Advanced data mining and applications. Springer, pp 63–75
9. Chang SF, Sikora T, Purl A (2001) Overview of the mpeg-7 standard. IEEE Trans Circuits Syst Video Technol 11(6):688–695. https://doi.org/10.1109/76.927421
10. Cheng ZQ, Wu X, Liu Y, Hua XS (2017) Video2shop: exact matching clothes in videos to online shopping images. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 4169–4177. https://doi.org/10.1109/CVPR.2017.444
11. Ciaccia P, Patella M, Zezula P (1997) M-tree: an efficient access method for similarity search in metric spaces. In: Proceedings of the 23rd international conference on very large databases, VLDB '97. Morgan Kaufmann Publishers Inc., San Francisco, CA, pp 426–435. http://dl.acm.org/citation.cfm?id=645923.671005
12. Dehne F, Noltemeier H (1987) Voronoi trees and clustering problems. Inf Syst 12(2):171–175. https://doi.org/10.1016/0306-4379(87)90041-X
13. Deng J, Dong W, Socher R, Li L, Kai L, Li F-F (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE, pp 248–255. https://doi.org/10.1109/CVPR.2009.5206848
14. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2014) Decaf: a deep convolutional activation feature for generic visual recognition. In: Proceedings of the 31st international conference on international conference on machine learning, ICML'14, vol 32, pp I–647–I–655
15. Eisenberg A, Melton J, Kulkarni K, Michels JE, Zemke F (2004) Sql:2003 has been published. SIGMOD Rec 33(1):119–126. https://doi.org/10.1145/974121.974142
16. Codd EF (1970) A relational model of data for large shared data banks. Commun ACM 13:377–387. https://doi.org/10.1007/978-3-642-48354-7_4
17. Georgiou T, Liu Y, Chen W, Lew M (2020) A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision. Int J Multimed Inf Retr 9(3):135–170. https://doi.org/10.1007/s13735-019-00183-w
18. Grošup T, Čech P, Lokoč J, Skopal T (2015) A web portal for effective multi-model exploration. In: He X, Luo S, Tao D, Xu C, Yang J, Hasan MA (eds) MultiMed Model. Springer International Publishing, Cham, pp 315–318
19. Grosup T, Peska L, Skopal T (2019) Towards augmented database schemes by discovery of latent visual attributes. In: Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019, pp. 670–673. https://doi.org/10.5441/002/edbt.2019.83
20. Harrington JL (2010) 3–introduction to SQL. In: Harrington JL (ed) SQL clearly explained, edition edn. The Morgan Kaufmann series in data management systems. Morgan Kaufmann, Boston, pp 65–74
21. Hopcroft J, Tarjan R (1973) Algorithm 447: efficient algorithms for graph manipulation. Commun ACM 16(6):372–378. https://doi.org/10.1145/362248.362272
22. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE international conference on data mining, pp 263–272. https://doi.org/10.1109/ICDM.2008.22
23. Jiang S, Wu Y, Fu Y (2018) Deep bidirectional cross-triplet embedding for online clothing shopping. ACM Trans Multimed Comput Commun Appl 14(1):5:1–5:22. https://doi.org/10.1145/3152114
24. Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev. https://doi.org/10.1007/s10462-020-09825-6
25. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, vol 25, pp 1097–1105. Curran Associates, Inc
26. Kruliš M, Lokoč J, Skopal T (2016) Efficient extraction of clustering-based feature signatures using GPU architectures. Multimed Tools Appl 75(13):8071–8103. https://doi.org/10.1007/s11042-015-2726-y
27. Li H, Ellis JG, Zhang L, Chang SF (2019) Automatic visual pattern mining from categorical image dataset. Int J Multimed Inf Retr 8(1):35–45. https://doi.org/10.1007/s13735-018-0163-1

28. Li Y, Liu L, Shen C, van den Hengel A (2015) Mid-level deep pattern mining. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 971–980
29. Lin Y, Tran S, Davis LS (2020) Fashion outfit complementary item retrieval. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3308–3316
30. Liu T, Wang R, Chen J, Han S, Yang J (2018) Fine-grained classification of product images based on convolutional neural networks. Adv Mol Imaging 08:69–87. https://doi.org/10.4236/ami.2018.84007
31. Liu X, Deng Z, Yang Y (2019) Recent progress in semantic image segmentation. Artif Intell Rev 52(2):1089–1106. https://doi.org/10.1007/s10462-018-9641-3
32. Liu Z, Luo P, Qiu S, Wang X, Tang X (2016) Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 1096–1104. https://doi.org/10.1109/CVPR.2016.124
33. Liu Z, Yan S, Luo P, Wang X, Tang X (2016) Fashion landmark detection in the wild. In: European conference on computer vision (ECCV). Springer International Publishing, Cham, pp 229–245
34. Lokoč J, Grošup T, Skopal T (2012) Sir: the smart image retrieval engine. In: Navarro G, Pestov V (eds) Similarity search and applications. Springer, Berlin, pp 240–241
35. Lokoč J, Grošup T, Skopal T (2012) Image exploration using online feature extraction and reranking. In: Proceedings of the 2Nd ACM international conference on multimedia retrieval, ICMR '12. ACM, New York, NY, pp 66:1–66:2. https://doi.org/10.1145/2324796.2324871
36. Lokoč J, Moško J, Čech P, Skopal T (2014) On indexing metric spaces using cut-regions. Inf Syst 43(C):1–19. https://doi.org/10.1016/j.is.2014.01.007
37. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94
38. Novak D, Batko M (2009) Metric index: an efficient and scalable solution for similarity search. In: 2009 Second international workshop on similarity search and applications. IEEE, pp 65–73. https://doi.org/10.1109/SISAP.2009.26
39. Novak D, Batko M, Zezula P (2015) Large-scale image retrieval using neural net descriptors. In: Baeza-Yates R, Lalmas M, Moffat A, Ribeiro-Neto BA (eds) Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, Santiago, Chile, 9–13 Aug 2015. ACM, pp 1039–1040. https://doi.org/10.1145/2766462.2767868
40. Peska L, Balcar S (2019) Fuzzy d'hondt's algorithm for on-line recommendations aggregation. In: Proceedings of the 2nd workshop on online recommender systems and user modeling, proceedings of machine learning research, vol 109. PMLR, pp 2–11
41. Peska L, Grosup T, Kovalcik G, Lokoc J, Skopal T (2017) Vadet: visual attributes exploration and discovery tool. http://herkules.ms.mff.cuni.cz/vadet-merged
42. Peska L, Vojtas P (2017) Using implicit preference relations to improve recommender systems. J Data Semant 6(1):15–30. https://doi.org/10.1007/s13740-016-0061-8
43. Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes M, Shyu ML, Chen S, Iyengar S (2018) A survey on deep learning: Algorithms, techniques, and applications. ACM Comput Surv. https://doi.org/10.1145/3234150
44. Ramya S, Beham R (2016) Comparative study on algorithms of frequent itemset mining. Int J Comput Sci Mob Comput 5:271–275
45. Skopal T (2020) On visualizations in the role of universal data representation. In: Gurrin C, Jónsson BT, Kando N, Schöffmann K, Chen YP, O'Connor NE (eds) Proceedings of the 2020 on international conference on multimedia retrieval, ICMR 2020, Dublin, Ireland, 8–11 June 2020. ACM, pp 362–367. https://doi.org/10.1145/3372278.3390743
46. Skopal T, Peska L, Grošup T (2018) Interactive product search based on global and local visual-semantic features. In: 11th international conference on similarity search and applications, SISAP 2018, Lima, Peru, pp 87–95
47. Skopal T, Peška L, Grošup T (2018) Interactive product search based on global and local visual-semantic features. In: Marchand-Maillet S, Silva Y, Chávez E (eds) Similarity search and applications. Springer International Publishing, Cham, pp 87–95. https://doi.org/10.1007/978-3-030-02224-2_7
48. Skopal T, Peška L, Kovalčík G, Grošup T, Lokoč J (2017) Product exploration based on latent visual attributes. In: Proceedings of the 2017 ACM on conference on information and knowledge management, CIKM '17. ACM, pp 2531–2534. https://doi.org/10.1145/3132847.3133175
49. Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C (2018) A survey on deep transfer learning. In: Kůrková V, Manolopoulos Y, Hammer B, Iliadis L, Maglogiannis I (eds) Artificial neural networks and machine learning—ICANN 2018. Springer International Publishing, Cham, pp 270–279
50. Tan R, Vasileva MI, Saenko K, Plummer BA (2019) Learning similarity conditions without explicit supervision. In: 2019 IEEE/CVF international conference on computer vision (ICCV) pp 10372–10381

51. Tate RF (1954) Correlation between a discrete and a continuous variable point-biserial correlation. Ann Math Stat 25(3):603–607
52. Uhlmann JK (1991) Satisfying general proximity / similarity queries with metric trees. Inf Process Lett 40(4):175–179. https://doi.org/10.1016/0020-0190(91)90074-R
53. Uijlings JR, Sande KE, Gevers T, Smeulders AW (2013) Selective search for object recognition. Int J Comput Vis 104(2):154–171. https://doi.org/10.1007/s11263-013-0620-5
54. Veit A, Belongie SJ, Karaletsos T (2017) Conditional similarity networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 1781–1789
55. Vittayakorn S, Umeda T, Murasaki K, Sudo K, Okatani T, Yamaguchi K (2016) Automatic attribute discovery with neural activations. CoRR arXiv:1607.07262
56. Wang K, Yin Q, Wang W, Wu S, Wang L (2016) A comprehensive survey on cross-modal retrieval. CoRR arXiv:1607.06215
57. Wang Z, Meng J, Yu T, Yuan J (2017) Common visual pattern discovery and search. In: 2017 Asia-Pacific signal and information processing association annual summit and conference (APSIPA ASC), pp 1011–1018. https://doi.org/10.1109/APSIPA.2017.8282178
58. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) European conference on computer vision—ECCV 2014. Springer, pp 818–833
59. Zezula P, Amato G, Dohnal V, Batko M (2006) Similarity search—the metric space approach, vol 32. Advances in database systems. Springer, Berlin. https://doi.org/10.1007/0-387-29151-2
60. Zheng L, Yang Y, Tian Q (2018) Sift meets CNN: a decade survey of instance retrieval. IEEE Trans Pattern Anal Mach Intell 40:1224–1244. https://doi.org/10.1109/TPAMI.2017.2709749



Tomáš Grošup obtained his PhD in 2019 from Charles University in Prague, Czechia. He worked at the SIRET research group focusing on similarity analytics and multimedia exploration systems. He has published 15 papers with H-index 5 (GS). He now works as a Vice President at Barclays investment bank for the risk, finance and treasury department.

Ladislav Peška obtained his PhD in 2016 from Charles University in Prague, Czechia. He is currently an assistant professor at Department of Software Engineering, Charles University, Prague. He is a member of the SIRET research group, where his main research interests are recommender systems and multimedia retrieval. L. Peska has published over 50 papers in journals and conference proceedings, with 350+ citations and H-index 12 (GS).

Tomáš Skopal obtained his PhD in 2004 from Technical University in Ostrava, Czechia. He is a full professor of Computer Science at Faculty of Mathematics and Physics, Charles University, Prague, Czechia, where he leads the SIRET research group focusing at similarity search in unstructured data, database systems and multimedia retrieval and analytics. T. Skopal has published over 120 papers in journals and conference proceedings with 1700+ citations and H-index 24 (GS). His ORCID profile can be found at https://orcid.org/0000-0002-6591-0879.