Dynamic Facet Ordering for Faceted Product Search Engines

Damir Vandic, Steven Aanen, Flavius Frasincar, and Uzay Kaymak

Abstract—Faceted browsing is widely used in Web shops and product comparison sites. In these cases, a fixed ordered list of facets is often employed. This approach suffers from two main issues. First, one needs to invest a significant amount of time to devise an effective list. Second, with a fixed list of facets it can happen that a facet becomes useless if all products that match the query are associated to that particular facet. In this work, we present a framework for dynamic facet ordering in e-commerce. Based on measures for specificity and dispersion of facet values, the fully automated algorithm ranks those properties and facets on top that lead to a quick drill-down for any possible target product. In contrast to existing solutions, the framework addresses e-commerce specific aspects, such as the possibility of multiple clicks, the grouping of facets by their corresponding properties, and the abundance of numeric facets. In a large-scale simulation and user study, our approach was, in general, favorably compared to a facet list created by domain experts, a greedy approach as baseline, and a state-of-the-art entropy-based solution.

Index Terms—Facet ordering, product search, user interfaces

1 Introduction

Studies from the past have shown that other factors than the price play a role when a consumer decides to choose where to buy a product online [1]. Therefore, online retailers pay special attention to the usability and efficiency of their Web shop user interfaces. Nowadays, many Web shops make use of the so-called *faceted navigation* user interface [2], which is in literature also sometimes referred to as 'faceted search' [3]. Facets are used by some users as a search tool, while others use it as a navigation and/or browsing tool [4], [5]. One of the reasons why faceted search is popular among Web shops is that users find it intuitive [6], [7]. The term 'facet' has a rather ambiguous interpretation, as there are different types of facets. In this work, we refer to facets as the combination of a property and its value, such as `WiFi:true` or `Lowest price (€):64.00`. Furthermore, facets are usually grouped by their property in user interfaces, in order to prevent them from being scattered around, and, thereby, confusing the user. In other words, the facet properties, such as `Color`, are shown first, and each property presents the actual values (e.g., `Red`, `Green`, and `Blue`). Figure 1 shows an example of a faceted search user interface, where the same concepts apply (e.g., the 'Featured Brands' property with its values 'Samsung', 'Motorola', 'Nokia', etc.).

Faceted search is primarily helpful in situations where the exact required result is not known in advance. As opposed to product search using keyword-based queries, facets enable the user to progressively narrow down the search results in a number of steps by choosing from a list of query refinements. However, one of the difficulties with faceted search, especially in e-commerce, is that a large number of facets are available. Displaying all facets may be a solution when a small number of facets is involved, but it can overwhelm the user for larger sets of facets [9].

Currently, most commercial applications that use faceted search have a manual, 'expert-based' selection procedure for facets [10], [11], or a relatively static facet list [8]. However, selecting and ordering facets manually requires a significant amount of manual effort. Furthermore, faceted search allows for interactive query refinement, in which the importance of specific facets and properties may change during the search session. Therefore, it is likely that a predefined list of facets might not be optimal in terms of the number of clicks needed to find the desired product.

In order to deal with this problem, we propose an approach for dynamic facet ordering in the e-commerce domain. The focus of our approach is to handle domains with sufficient amount of complexity in terms of product attributes and values. Consumer electronics (in this work 'mobile phones') is one good example of such a domain. As part of our solution, we devise an algorithm that ranks properties by their importance and also sorts the values within each property. For property ordering, we identify specific properties whose facets match many products (i.e., with a high impurity). The proposed approach is based on a facet impurity measure, regarding qualitative facets in a similar way as classes, and on a measure of dispersion for numeric facets. The property values are ordered descending on the number of corresponding products. Furthermore, a weighting scheme is introduced in order to favor facets that match many products over the

- *Damir Vandic and Flavius Frasincar are with the Erasmus School of Economics, Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands. Email: {vandic,frasincar}@ese.eur.nl*
- *Steven Aanen is the co-founder of Grible.co, Coolsingel 104, NL-3011 AG Rotterdam, the Netherlands. E-mail: aanen@grible.co*
- *Uzay Kaymak is with the Information Systems IE&IS, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, the Netherlands. Email: u.kaymak@tue.nl*

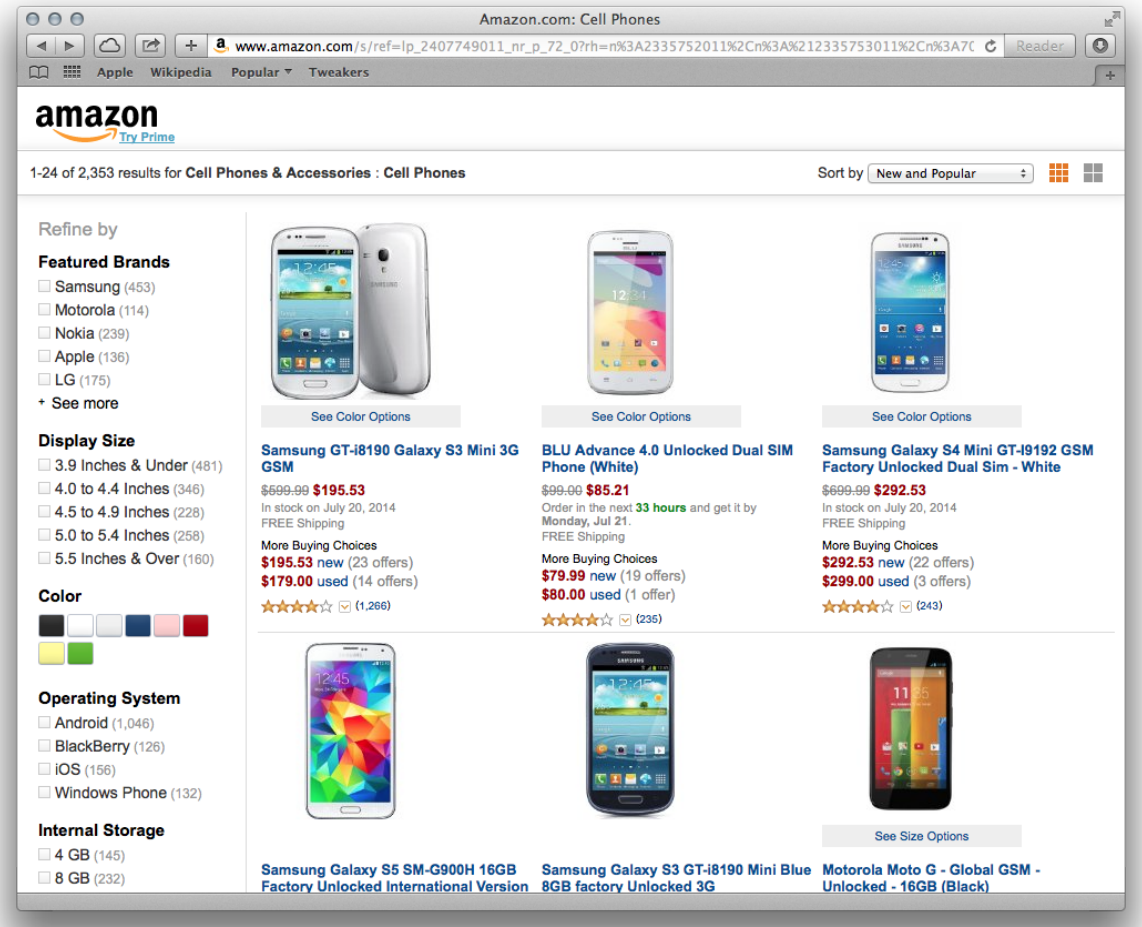


Fig. 1. A screenshot of Amazon.com [8], showing a typical faceted search user interface in e-commerce.

ones that match only a few products, taking into account the *importance* of facets. Similar to existing recommender system approaches [12], our solution aims to learn the user interests based on the user interaction with the search engine.

2 Related Work

We can find approaches in the literature that focus on personalized faceted search [13], [14], [15]. However, we do not discuses these, as, unlike our approach, they require some sort of explicit user ratings. Therefore, we only consider related work that does not require any explicit user input other than the query.

The faceted search system proposed in [16] focuses on both textual and structured content. Given a keyword query, the proposed system aims to find the *interesting* attributes, which is based on how surprising the aggregated value is, given the expectation. The main contribution of this work is the *navigational* expectation, which is, according to the authors, a novel interestingness measure achieved through judicious application of p-values. This method is likely not to be suitable for the domain of e-commerce, where also small data sets occur and statistically deriving interesting attributes is not possible.

In [17], a framework for general-domain facet selection is proposed, with the aim to maximize the rank promotion of desired documents. There are many aspects in the proposed approach that make it not applicable in an e-commerce environment. First, two main assumptions are made: (1) the search process is initiated using a keyword-based query, and (2) the result is a ranked list of documents. These are serious limitations, as many Web shop users start with a facet selection instead of a keyword-based search, and product ranking is often not supported. Therefore, the framework we propose does not use these two assumptions. Second, the proposed solution does not consider multiple iterations of the search process (i.e., multiple drill-downs). Third, the authors do not differentiate between facet types. Consequently, numeric facets are treated in the same way as qualitative facets

(discussed in Section 3), thereby losing their ordinal nature. Fourth, the authors assume that a user can only perform a drill-down using only conjunctive semantics. In our study, we use the common disjunctive semantics for values and conjunctive semantics for properties and take into account the possibility of drill-ups. This means that result set sizes are expected to both increase and decrease during the search session, either by deselecting a facet or choosing an addition facet in a property (e.g., selecting 'Samsung' when 'Apple' is already selected). Fifth and last, the authors do not distinguish in their approach between values (e.g., `Samsung`) and properties (e.g., `Brand`), instead, they only consider the combination of values and properties.

In [18] the approach of [17] was extended and improved with a focus on product search. Using additional user assumptions and the same theoretic approach as [17], two new methods for facet sorting were developed. Even though this approach improves upon the original algorithm, it still suffers from the same issues discussed above.

A more recent approach provides another method for facet selection [19], or 'dynamic categorization' as the authors refer to it. The selection process is based on ontological data from a Semantic Web environment. However, due to a limited usage of rich ontological relationships, the algorithms can also be applied to semi-structured data, as also suggested in the paper. The study is an extension of earlier work of the authors, which was based on the idea of selecting more descriptive facets using an entropy-based measure [20]. Similar to [17], [18], this approach does not consider numeric facets and the use of disjunctive semantics for values.

Summarizing, most of the related approaches that have been proposed, with the exception of [18], do not explicitly focus on the e-commerce domain [19], [14], [17]. Furthermore, these solutions often assume that there is a ranking of the results, based on a preceding keyword-based query or external data, which is often not the case for e-commerce. Also, our approach ranks properties and facets, unlike existing algorithms [14], [17], [18], [19], which filter (or select) properties and facets. Last, none of the approaches from the literature that we discussed emphasize the performance aspect of the proposed algorithms. However, in order to be useful in practice, for most Web shops, it is important that the proposed solutions are responsive.

3 Facet Optimization Algorithm

Before discussing the details of our approach, we need to elaborate on the assumptions and the used terminology. From the perspective of user interface design, we distinguish between two main facet types: *qualitative facets* (e.g., `WiFi:true`) and *numeric facets* (e.g., `Lowest price (€):64.00`). We further distinguish between two types of qualitative facets: *nominal facets* and *Boolean facets*. Nominal facets are, for example, those for the property `Display Type`, and can have any nominal value. Boolean facets are for instance `Multitouch`, and have only three options from an interface perspective: `true`, `false`, or `No preference`.

Unlike previous studies, as discussed in Section 2, our approach treats numeric facets differently than qualitative facets. When creating facets from source data (e.g., tabular data), every unique property-value combination is converted into a facet. For numeric facets, the same process is applied. However, numeric values can be widely dispersed, especially in large data sets. For facets, however, that would lead to a list of possibly hundreds of different values. One way to deal with that is to create predefined, fixed ranges of values and use these as facets. However, it is never certain whether the predefined ranges will match the user's preferences. Furthermore, fixed ranges can become useless when a result set has only products that fall into one predefined range. For our approach, we have chosen to let the user define custom ranges of values to select. In a product search engine, such custom ranges can be represented using a slider widget. From a technical point of view, however, these custom ranges are considered as selecting a set of facets in one click, i.e., each numeric value is still represented as a separate facet.

The approach we propose aims to order properties and facets in such a way that any individual product could be found quickly and effectively. We put the leading emphasis on property ordering, as we expect that it has the largest impact on the user effort. A straightforward way to order properties would be by presenting those properties on top that feature equal-sized facet counts for the facets of that property, which is an effect that is for instance visible in the entropy-based approach of [18]. However, this would still require many clicks in total, possibly leading to long search times. Our approach aims to rank more specific properties higher. The reason behind is that we believe that users are to a limited extent, and possibly unconsciously, aware that selecting more unique features of the target product will result in a faster drill-down. Even in situations where this is not true, ranking more specific properties higher will increase the chance that the user will use specific facets for drill-down, resulting in a shorter search session duration. As an example consider a user who is searching for a Nokia smartphone capable of playing his collection of MP3 music, and both features are equally important. We expect the user to start by selecting `Brand:Nokia` instead of `Audio Formats:MP3`. The user may be aware of the fact that most smartphones are capable of playing MP3 audio, thus selecting that facet will not lead to a quick drill-down. Filtering only Nokia phones will presumably have a much larger impact on the result set than filtering phones that support MP3. The effect of ranking the individual facets (i.e., `Nokia` vs. `Samsung`) is assumed to be limited. We expect that popularity is a more suited metric that can be used for this purpose.

When the user selects facets from a more specific

property, the result set will decrease in size quickly. Since the most specific facets only apply to few products, it would be ineffective to present those on top, as the target product is unknown to the system. Given that we assume that ordering properties has more effect than ordering facets, we therefore compute the impurity of properties as a whole, based on the specificity of its facets. Combined with weighting for the number of products on which it applies, this method will give us those properties and facets on top, that will most likely lead to the quickest drill-down for most of the possible target products. At the same time, the weighting that we introduce lowers the rank of properties with many missing values in the data, as those cannot be employed for drill-down.

3.1 Search Sessions

A query in a search session is defined as a collection of previously selected facets. We have decided to apply disjunctive semantics to a selection of facets within a property. For facets across different properties, we use a conjunctive semantics. For example, selecting the facets `Brand:Samsung`, `Brand:Apple`, and `Color:Black` results in `(Brand:Samsung OR Brand:Apple) AND Color:Black`. Several e-commerce stores on the Web (e.g., Amazon.com and BestBuy.com) use the same principle, which, from a user experience point-of-view, is very intuitive.

Our approach assumes that users can undertake two types of actions: drill-down and roll-up. A drill-down is defined as an action of selecting one or more facets, leading to a reduction of the result set size. A roll-up action increases the result set size, which is likely to happen when the user notices that the selected facets are too strict. A roll-up action can be achieved in three ways: (1) selecting a qualitative facet from a property for which a selection already exists (e.g., adding `Brand:Samsung` to a query containing `Brand:Apple`), (2) deselecting the only selected facet of a property, and (3) broadening a numeric range. From this point on, we use the notations described in Table 1, which will be described in further details in the next few sections.

Figure 2 summarize the complete search session flow assumed in our approach. Throughout the search session, we assume that there exists a single target product d_u that the user wants to find, and that the user will eventually be able to find it. Although the user may not know the name of the product, (s)he will be able to identify it by means of the characteristics of the product (F_{d_u}). The process starts with a complete result set containing all products from the catalog D and an empty user query q. Our approach then initiates two processes, i.e., (1) computing the property scores and (2) computing the facet scores, discussed in Section 3.2 and 3.3, respectively. When the system completes, the user view is updated showing the properties and facets in the computed order.

In the next step, the user evaluates the result set size. If the result set size is too large to scan manually ($|D_q| > n$), the user will continue to drill-down. Otherwise, the user will scan the result set and check if the target product is found. If the target product is found, the search session is completed and considered successful. The user will perform a roll-up in the case that the desired product was not found, which will increase the result set size and the same process repeats again.

3.2 Computing Property Scores

We now discuss the details of computing property scores, shown as one of the first two processes in Figure 2. The outcome of the property scores is used to first sort the properties, after which the facet scores, discussed in the next section, are used to sort the values within each property. In Figure 3, we zoom into the main steps of computing the property score. As shown by the diagram, the score for each property is computed separately and can thus be done in parallel.

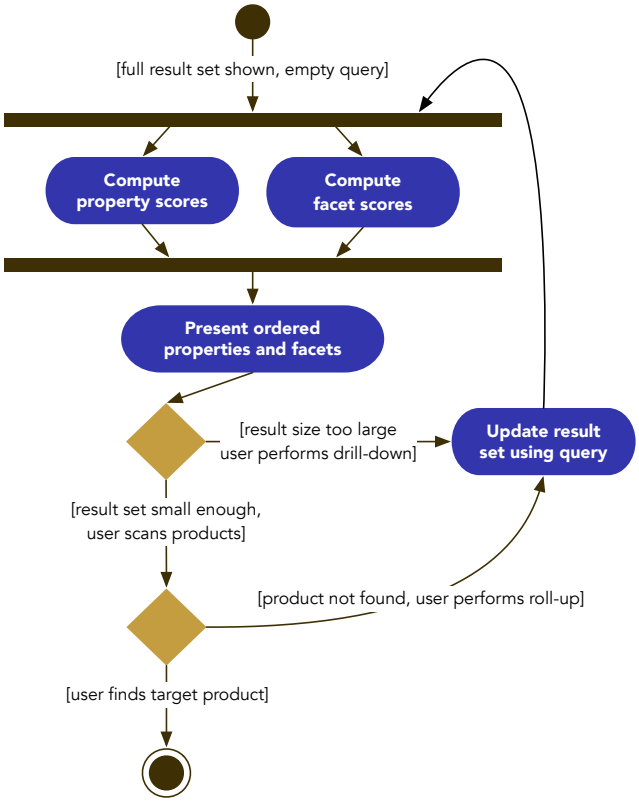


Fig. 2. Activity diagram describing the main flow of a search session.

3.2.1 *Disjoint Facet Counts*

We designed the proposed algorithm in such a way that more specific facets and properties are ranked higher. To support the algorithm in identifying more specific facets, we introduce the *disjoint facet count*. This metric is used to compute the score for qualitative properties. The disjoint facet count is the number of products from the result set

D	Set of products (product catalog)
P	Set of properties
F	Set of facets
$F_p \subseteq F,\ (p \in P)$	Set of facets for property p
$F_d \subseteq F,\ (d \in D)$	Set of facets for product d
$q \subseteq F$	Query
$D_q \subseteq D$	Result set returned for query q
$D_f,\ (\forall d \in D_f : f \in F_d)$	Set of products associated to facet f
$r_q^O(f),\ (f \in F_p)$	Rank of facet f for facet ordering scheme O in the result set (dependent on query q)
$r_q^O(p),\ (p \in P)$	Rank of property p for facet ordering scheme O in the result set (dependent on query q)
$d_u \in D$	Target product for user u
X	Variable indicating user effort
M	Selected drill-down model in user simulation
n	Maximum number of products in the result set the user is willing to scan in the user simulation
t	Iteration indicator (state) of search session

TABLE 1
Summary of notations.

matching each facet f of property p. The classical facet count for a facet f, for a given query q, is defined as:

$$\text{count}(f, q) = |D_q \cap D_f| = \sum_{d \in D_q} \begin{cases} 1 & \text{if } f \in F_d \\ 0 & \text{if } f \notin F_d \end{cases} \quad (1)$$

The disjoint facet count is then defined as:

$$\text{disjointCount}(f, q) = \sum_{d \in D_q} \begin{cases} 1 & \text{if } F_p \cap F_d \equiv \{f\} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where p is the property of facet f, $f \in F_p$, and $\{f\}$ is the singleton set containing f. More general facets such as `Audio Formats:MP3` will thus have a low disjoint count, as most products that have this facet also support other audio formats besides MP3. On the other hand, facets from the property `Brand` are likely to have relatively high counts, as most products are associated to only one brand.

In Table 2 we show the tabular product data of a data sample that was taken from our evaluation dataset from [11]. The table also shows how the tabular data has been transformed into facets and the corresponding final scores.

3.2.2 Scoring Qualitative Properties

Figure 3 shows that qualitative properties are partly treated differently compared to numeric properties. For qualitative properties, we employ the *Gini impurity* [21] to assess their 'uniqueness' or specificity in terms of describing certain products. We could have used Shannon's entropy [22] for the same goal. Various studies have investigated this choice. In [23], the authors find that these two methods produce tree splits that are not significantly different from each other. One of the few differences that tend to be present, is that the Gini impurity tends to produce the most pure nodes [24], which is why we chose to use it.

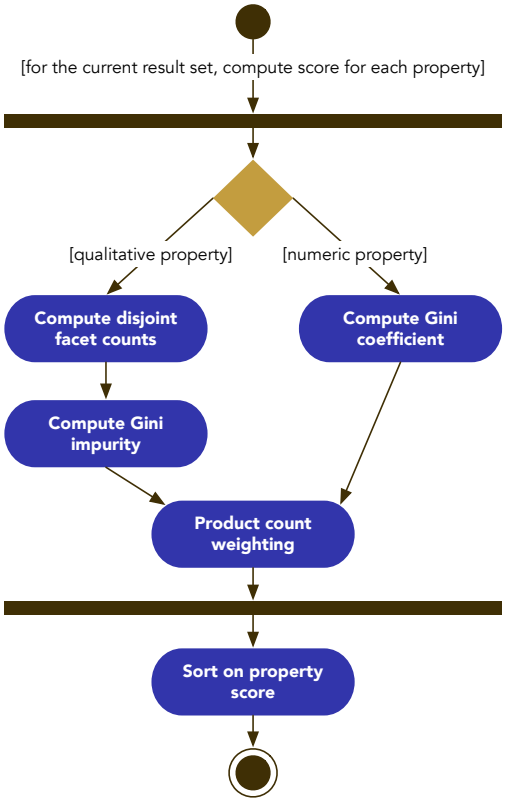


Fig. 3. Activity diagram showing the individual steps in the property score computation process.

In the context of facet properties, we are looking for those properties with the highest impurity. At that point, it becomes desirable to initiate a new 'split', i.e., a facet selection, in order to reduce the impurity. We define the

Product Name	Property			
	Audio Formats	Brand	Diagonal Screen Size (inch)	Lowest Price (€)
Nokia 6230i	mp3	N/A	1.5	80.33
LG KU990 Viewty	aac, midi, mp3, mpeg 4, wav, wma	LG	3	79.00
Sony Ericsson C902	aac, mp3	Sony Ericsson	2	129.95
LG KF510	aac, mp3	LG	2.2	N/A
Apple iPhone 4	aac, aac+, aax, aax+, aiff, mp3, wav	Apple	3.5	459.95
LG Nexus 4 8GB	flac, mp3	LG	4.7	382.90
Samsung Galaxy S4	aac, ac3, amr-nb, eaac+, flac, mp3, ogg, wav, wma	Samsung	N/A	494.99

TABLE 2
This example uses parameter values $|D| = 7$, $|P| = 4$, and $q = \emptyset$. The value 'N/A' stands for 'not applicable' (e.g., Gini coefficient is only computed for numeric properties). Looking at the final property scores (last column of Table 3), we can conclude that `Brand` is more important than `Audio Formats` and that the `Lowest Price (€)` is more important than `Diagonal Screen Size (inch)`.

Property & Facets					Scores		
Property	Facet	Facet Count	Disjoint Facet Count	Prod. Count Weighting	Gini Coeff.	Gini Impurity	Property Score
Audio Formats	aac	5	0	$\frac{1}{7}$	N/A	0.00000	0.00000
	aac+	1	0				
	aax	1	0				
	aax+	1	0				
	ac3	1	0				
	aiff	1	0				
	amr-nb	1	0				
	eaac+	1	0				
	flac	2	0				
	midi	1	0				
	mp3	7	1				
	mpeg4	1	0				
	ogg	1	0				
	wav	3	0				
	wma	2	0				
Brand	Apple	1	1	$\frac{6}{7}$	N/A	0.66667	0.57143
	LG	3	3				
	Samsung	1	1				
	Sony Erricson	1	1				
Diagonal Screen Size (inch)	1.5	1	1	$\frac{6}{7}$	0.21006	N/A	0.18005
	2.0	1	1				
	2.2	1	1				
	3.0	1	1				
	3.5	1	1				
	4.7	1	1				
Lowest Price (€)	79.00	1	1	$\frac{6}{7}$	0.35561	N/A	0.30481
	80.33	1	1				
	129.95	1	1				
	382.90	1	1				
	459.95	1	1				
	494.99	1	1				

TABLE 3
The computed scores for the considered properties.

Gini impurity for facet selection as follows:

$$\text{giniImpurity}(p,q) = 1 - \sum_{f \in F_p} \left(\frac{\text{disjointCount}(f,q)}{\sum_{g \in F_p} \text{disjointCount}(g,q)} \right)^2 \quad (3)$$

where $p \in P_{\text{qualitative}}$ and $q \subset F$, with the fraction denominator being the total number of products from the result set associated to a single facet from property p. It should be noted that since the relative frequency of products is represented by the fraction in Equation (3), the measure is independent of the number of products associated to values by means of property p.

3.2.3 Scoring Numeric Properties

In the previous section, we explained how the Gini impurity can be employed to score qualitative properties. It would be possible to use the same methods for numeric facets as well, similar to related work in which numeric facets are treated as being qualitative [17], [19], [18]. However, this would lead to a loss of information, as each value would be treated as being a nominal. We could for instance imagine a result set of products in a similar price range. Regardless of the fact that the prices are similar, there is a good probability that most products will still have a unique value for price. In the data we used for evaluation, over 90% of the products has a unique price. However, when we disregard the fact that 'unique' prices may actually be quite similar, this would lead to a very high Gini impurity score. With property `Lowest Price (€)` being used in our example for drill-down, however, selecting a certain range of prices would still include most of the products, as their prices are similar. The property is thus not effective for drill-down.

For numeric properties, we have chosen to use the knowledge about the distribution of the numeric values for computing property scores. It is fairly straightforward to imagine that it may be useful to drill-down using a numeric property when the values for the result set are widely dispersed. When the facets are nearly uniformly distributed over the complete range of values, a drill-down using a user-defined range would lead to a large reduction of the result set. On the other hand, when most of the values are similar, such as in the example of having a result set with products of the same price range, drilling down using a numeric property will hardly reduce the result set size and thus be ineffective to use. For assessing the dispersion of numeric facets, we employ the Gini coefficient [25]. We adapt the original Gini index for use in our context:

$$\begin{aligned} &\text{giniCoefficient}(p,q) = \\ &\qquad \frac{1}{m}\left(m + 1 - 2\left(\frac{\sum_{i=1}^{m} (m+1-i) f_i}{\sum_{i=1}^{m} f_i} \right) \right) \quad (4) \\ &\qquad = \frac{2\sum_{i=1}^{m} i f_i}{m \sum_{i=1}^{m} f_i} - \frac{m+1}{m} \\ &\text{given } f_i \in F_p^* \text{ for } i = 1 \text{ to } m \\ &\qquad F_p^* = \{ f_i \mid f_i \in F_p \cap F_d,\ d \in D_q,\ f_i \leq f_{i+1} \} \\ &\qquad m = |F_p^*| \\ &\qquad p \in P_{\text{quantitative}} \end{aligned}$$

where F_p^* represents the values for numeric property p for the products in the result set, indexed in non-decreasing order ($f_i \leq f_{i+1}$), with f_i being the facet ranked at index i.

In Table 3 we give the Gini coefficients for the considered properties. As an example, we will now compute the Gini coefficient for `Diagonal Screen Size (inch)`. We assume that the query is empty and thus all 6 facets can be included in the computation. By ordering these facets in an ascending way, we obtain $F_p^* = \{1.5, 2.0, 2.2, 3.0, 3.5, 4.7\}$ and $m = 6$. The index is then given by:

$$\begin{aligned} \text{G} &= \frac{2\sum_{i=1}^{m} i f_i}{m \sum_{i=1}^{m} f_i} - \frac{m+1}{m} \\ &= \frac{2 \cdot (1 \cdot 1.5 + \ldots + 5 \cdot 3.5 + 6 \cdot 4.7)}{6 \cdot (1.5 + 2.0 + 2.2 + 3.0 + 3.5 + 4.7)} - \frac{6+1}{6} \\ &= \frac{2 \cdot (69.8)}{6 \cdot (16.9)} - \frac{7}{6} \\ &= 0.21006 \end{aligned}$$

which is the index that is also mentioned in Table 3. From the table we can also conclude that the Gini for `Lowest Price (€)` is higher, suggesting that the values for that property are more dispersed than those of `Diagonal Screen Size (inch)`. Similar to the Gini impurity for qualitative facets, the Gini coefficient for properties is independent of the number of products that have this property.

3.2.4 Product Count Weighting

With the Gini impurity and the Gini coefficient, we now have metrics to score both qualitative and numeric properties. As mentioned in the previous sections, this score is independent from the number of products on which it is based. This could possibly lead to problems, as properties that occur within few products will obtain a relatively high score. To compensate for this, we introduce the *product count weighting*. The product count weighting is used to normalize the Gini indices, resulting in the final *property score*. Additionally, it provides a way to cope with missing values, as properties with many missing

associations will be ranked lower. We define the final property score as:

$$\text{propertyScore}(p,q) = gini(p,q) \cdot \sum_{f \in F_p} \frac{\text{disjointCount}(f,q)}{|D_q|} \quad (5)$$

where $gini$ is either the Gini impurity or the Gini coefficient (depending on the property type). The term with which $gini$ is multiplied is the product count weighting term. Table 3 shows the product count weighting for each property. If we take for instance property `Lowest Price (€)`, we can compute the property score using Equation 5 and the Gini from the table as follows:

$$\begin{aligned} \text{score} &= 0.35561 \cdot \frac{1+1+1+1+1+1+1}{7} \\ &= 0.35561 \cdot \frac{6}{7} \\ &= 0.30481 \end{aligned}$$

As we can see, the second term, the product count weighting, is $\frac{6}{7}$, corresponding to the value in Table 3 for `Lowest Price (€)`. Multiplying it by the Gini score obtained earlier this gives us the property score, by which we can rank properties using $r_q^O(p)$, with O referring to our approach in this case.

One should note that, strictly speaking, the Gini impurity and the Gini coefficient are not directly comparable to one another. For our use case, however, this does not lead to problems, as both measure the specificity of a property, one for qualitative and one for quantitative. Another approach to handling qualitative and quantitative properties would be to try to find unified similarity measure. However, we believe that it is difficult to compare qualitative and quantitative properties in the first place and having two separate lists of facets (one for qualitative properties and one for quantitative properties) would make the browsing of products more difficult for the end user. The empirically obtained results suggest that this approach is working adequately in practice.

3.3 Computing Facet Scores

In the previous sections, we have explained how we compute scores for properties. We now discuss the details of computing facet scores, shown as one of the first two processes in Figure 2. However, our approach also sorts the values within each property in order to reduce the value scanning effort. This is in contrast to for instance the approach in [19], which considers property ranking but disregards facets ranking. For numeric properties, value ordering is neglected, as these are often represented with a slider widget in user interfaces. The slider widgets, of which an example is shown in Figure 5, give an indication of the minimum and maximum values for a property, and allow the user to freely define a range of facets within these boundaries. For qualitative properties our approach employs the facet count from Equation (1), ranking facets descending on count, per property. As the target product is unknown to the system, this will increase the chance that a facet matching the target product is placed on top.

In the evaluation, we compare our approach to the one proposed in [19]. To have a fair comparison, we have implemented a version of their method that includes the same facet sorting as our algorithm, as the authors themselves have neglected this aspect. The difference in results can thus be completely accounted to property sorting.

4 Evaluation

In this section, we discuss the evaluation of our proposed approach. The evaluation is based on (1) simulated user sessions, where the simulation framework is derived from previous literature and solid theoretical foundations, and (2) a study involving real users.

4.1 Experimental Framework

Figure 4 gives an overview of the concepts that underlie the evaluation framework. In our experimental setup, one simulation process represents an individual search session, which we will refer to as an experiment. Each experiment contains the selection of one drill-down model, one ordering scheme, and one target product. Furthermore, some of the drill-down models and ordering schemes contain stochastic aspects. Therefore each experiment is repeated 50 times, in order to reduce the variability of results. For each experiment we record six different metrics. For the target products, we have decided to use every product in our data set as a target product d_u, in order to get the most reliable results from the data that we have available.

4.1.1 Drill-Down Models

There are three drill-down models that we consider, based on the ones proposed in [14], [17]. These drill-down models rely on five key assumptions, i.e., (1) rationality: the user will end the session once target product is found, (2) practicality: the user will use no more than a fixed number of clicks when looking for the target product, (3) feasibility: the user will perform a roll-up when the target product disappears from the result set, (4) omniscience: once presented with the facets, the user knows which ones belong to the target product, and (5) linearity: the user scans the properties from top to bottom. Because some of these assumptions are very restrictive, all drill-down models relax one or more of these assumptions. It is, however, useful to identify the theoretical boundaries that may apply to user behavior in order to make a simulation that is more realistic. In the *Least Scanning Drill-Down Model*, M_S, the user u scans the list of facets F starting from the top. When u encounters a facet $f \in F_{d_u}$ (a facet associated with the target product), (s)he will select that facet without further scanning.

The *Best Facet Drill-Down Model*, M_B, assumes that when u is searching for d_u and is scanning F, u identifies the single facet that will reduce the result set size most,

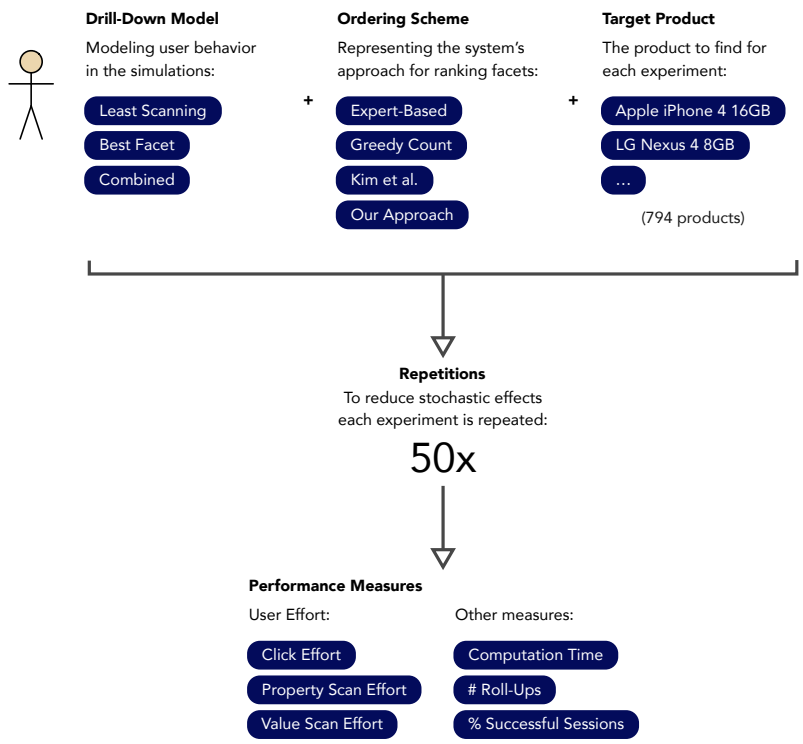


Fig. 4. Overview of the various concepts and phases underlying the evaluation framework. The 50 repetitions are applied to all combinations that include the Combined Drill-Down Model, as this is the only stochastic drill-down model. All considered performance measures are averaged over these 50 repetitions and the t-tests were performed using the metrics for each target product as samples.

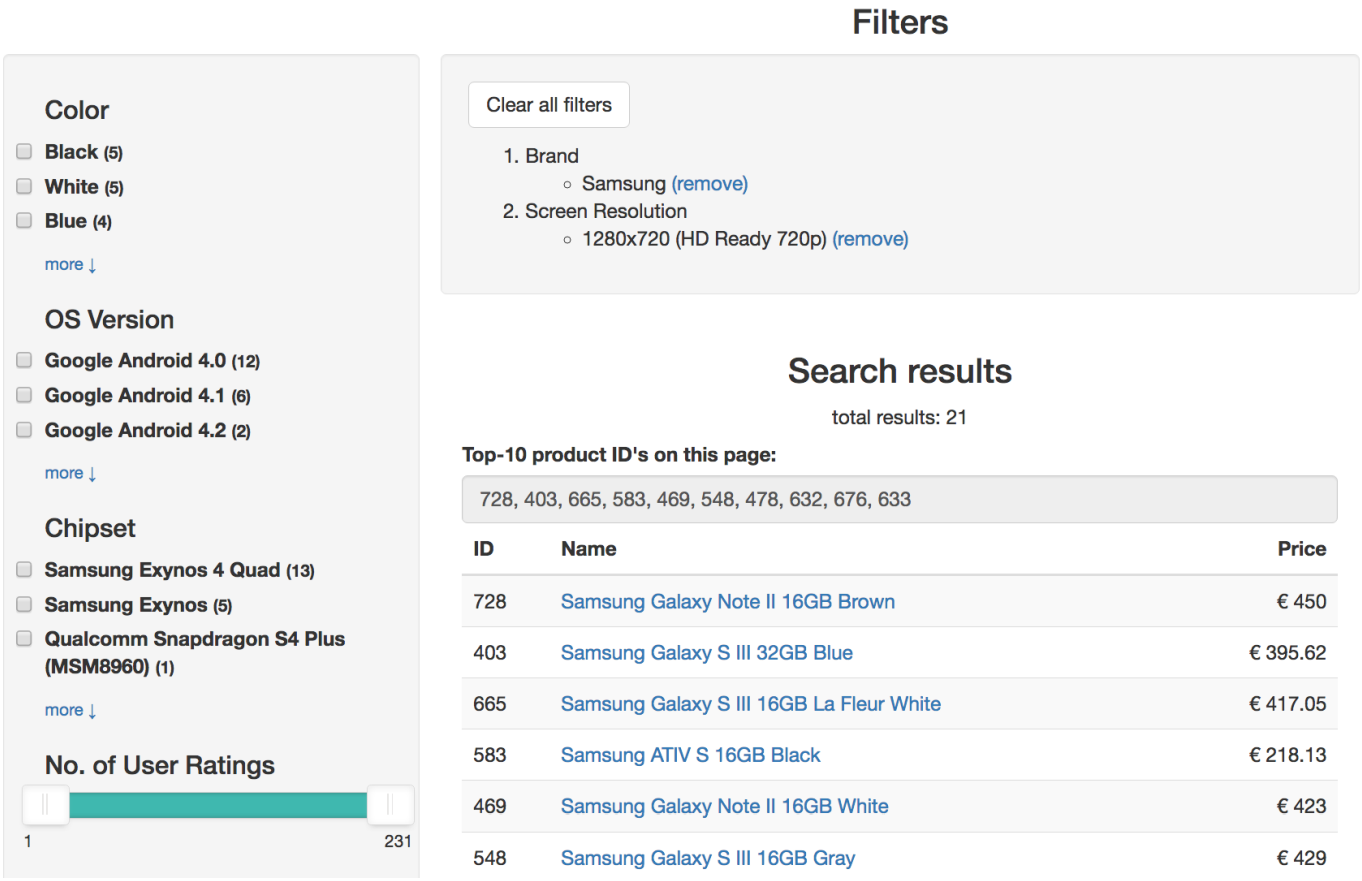


Fig. 5. Screenshot of the Web application that implements our approach. This application can be accessed at http://facet-sorting.eur.dvic.io.

while d_u is still included in the result set. In other words, the user will choose the 'best' drill-down option, regardless of the property or facet rank. The Best Facet Drill-Down Model minimizes the number of clicks at the expense of possibly scanning more facets. This is very useful for comparison with the results from the Least Scanning Drill-Down Model.

Last, the *Combined Drill-Down Model* M_C provides a more realistic simulation of user behavior by allowing faulty selections (i.e., clicks that will exclude the target product from the result set). This model assumes that the user u scans the list of facets F starting from the top. When u encounters a facet f (s)he will consider selecting f with probability α_f when the target product d_u is associated with this facet, and β_f when it is not. For α_f and β_f we use:

$$\alpha_f = \frac{\alpha}{|F_p \cap F_{d_u}|}, \quad \beta_f = \frac{\beta}{|F_p \setminus F_{d_u}|} \tag{6}$$

where $f \in F_p$ and $\alpha + \beta = 1$. Once u has a certain facet in consideration, the decision whether to select it will be made stochastically using the *Facet Importance Factor* γ_f, defined as follows:

$$\gamma_f = \begin{cases} 1 - \frac{r_q^O(f)-1}{|F_{d_u} \setminus q|-1} & \text{if } f \in F_{d_u} \ (\alpha \text{ case}) \\ 1 & \text{if } f \notin F_{d_u} \ (\beta \text{ case}) \end{cases} \tag{7}$$

where $r_q^O(f)$ is a function that returns the rank of f in a list of candidate facets $F_{d_u} \setminus q$ (unselected facets associated with d_u), and the fraction denominator $|F_{d_u} \setminus q| - 1$ is a normalization factor to bring the measure between 0 and 1. When a facet is not selected during a scan, either due to the stochastic effect from α_f or β_f, or due to its Facet Importance Factor γ_f, the user will resume scanning the following facet until a selection has been made.

4.1.2 Ordering Schemes

For effectively evaluating the performance of our approach, we perform a comparison with other ordering schemes. The *Expert-Based* scheme is the fixed-order scheme from [11], which is created manually by a team of dedicated editors. Since manually defined schemes are used in nearly all current applications on the Web, it provides a useful comparison with dynamic ordering methods as the one proposed in this study. The *Kim et al.* approach, proposed in [19], is a state-of-the-art method for sorting properties. Their proposed scheme fits the e-commerce domain well and because it is an entropy-based approach, it is an interesting candidate in the comparison. Although the original paper suggested source data in the form of an ontology, the algorithms can be applied to semi-structured data as well, as the authors also suggest. The last baseline we employ is the *Greedy Count* scheme. Greedy Count appears regularly in related work as a simple baseline for evaluation [14], [17]. It orders properties and facets descending on the number of matching products. In order to fit into our environment, the Greedy Count uses the following definition for the property score:

$$\text{greedyCountPropScore}(p, q) = \frac{\max_{f \in F_p} \text{count}(f, q)}{|D_q|} \tag{8}$$

The properties are thus ordered based on the maximum of the facet counts of their values. The facets themselves are naturally sorted on facet counts as well, as defined in our approach and the one we implemented for the *Kim et al.* approach. This means that all automatic approaches that we evaluate use the same facet ordering technique, which makes the comparison more fair.

4.1.3 Performance Measures

The performance of the ordering schemes given the different drill-down models is measured using various metrics. We consider three user effort metrics. First, the *click effort* X_c measures how often a facet was (de)selected or a range was adapted. Second, the *property scan effort* X_p measures how much effort is put in scanning properties and is defined by:

$$X_p = \sum_{t,\, |D_q^t| > n \,\wedge\, t \leq 100} \frac{r_q^O(p_t^M)}{|P|} \tag{9}$$

where n is the maximum number of products in the result set the user is willing to scan, and p_t^M refers to the property that is selected by the user given drill-down model M at iteration t. Last, the *value scan effort* X_f measures how much effort is put in scanning values and is defined by:

$$X_f = \sum_{t,\, |D_q^t| > n \,\wedge\, t \leq 100} \frac{r_q^O(f_t^M)}{|F_p|} \tag{10}$$

where f_t^M refers to the facet $f \in F_p$ that is selected by the user given drill-down model M at iteration t. As there is no list of facets for quantitative properties, the scanning effort for selecting a range of numerical values is defined as $X_f = 0$.

Besides the user effort metrics, we record three other measures during the experiments:

- **Computation Time** The computation time that is given in the tables measures only the time needed to compute or retrieve the order of facets, thus the selection scheme. Since the computations have been done using machines that are similar in hardware setup, we can use the computation time to compare among the various ordering schemes. The time as given in the tables is the total time in computation of the facet order for one complete experiment, thus depending on the number of states or clicks in the session.
- **# Roll-Ups** The number of roll-up user actions that were needed on average in each search session. This gives an indication of the ability of ordering schemes to cope with errors introduced in the query. Less roll-ups indicate a more efficient search process. The only

drill-down model that allows for faulty selections is the Combined Drill-Down Model, therefore roll-ups will only occur when that model is used.

- **% Successful Sessions** Each experiment is limited to 100 query states ($t \leq 100$) to prevent infinite search sessions. The percentage indicates the amount of experiments in which the target document was found within 100 clicks.

For our experiments, we have gathered data from Tweakers Pricewatch [11]. Tweakers PriceWatch is the largest Dutch price comparison Web site, with a comprehensive collection of product characteristics available in tabular format. The complete catalog contains 794 mobile phones, 53 properties, and 1,816 facets. Of these facets 348 are qualitative, against 1,468 numeric facets. The imported data was cleaned and converted to a more structured format (i.e., we used a custom, predefined schema). With 3 drill-down models, 4 ordering schemes, 794 target products, and 50 repetitions for the Combined Drill-Down Model, we have run over 150,000 experiments, storing over half a terabyte of experimental data. We also implemented our approach in a Web application, shown in Figure 5. Running all these experiments on one computer is unfeasible. Instead, we used a cluster of 100 instances, hosted on Amazon Web Services [26], to run the experiments. In the end, we stored half a gigabyte of performance metrics for the different experiments.

4.2 Results using the simulated experiments

In this section we present and discuss the results obtained from our experiments. We have performed t-tests to assess whether the observed differences for the click, property scan effort, and value scan effort are significant. Based on these tests we can conclude that all the found differences are significant, with the largest p-value being 0.00026.

Tables 4, 5, and 6 show the results for Least Scanning, Best Facet, and Combined Drill-Down models, respectively. We can make several important observations. First, in terms of the number of clicks, our approach seems to outperform the other methods, except in the case of the Best Facet Drill-Down Model, where each approach performs equally well. Furthermore, for the Combined Drill-Down Model, our approach results in the lowest number of roll-ups and the highest percentage of successful sessions.

Second, we observe that our approach, in most cases, performs best in terms of property and facet scan effort, except for the Combined and Least Scanning Drill-Down Model, respectively. However, although the found differences are statistically significant, it can be argued that they are not relevant, as there were no large effect sizes found. Furthermore, we assume that in practice the property and facet scanning efforts are not the key factors that contribute to the true perceived user effort. We assume that the number of clicks and the responsiveness of the approaches play a much more important role here.

Event type	Standard approach	Our approach
List facet select	364	376
Toggle collapsed	182	143
Numeric facet change	198	84
List facet deselect	18	2
Boolean facet change	5	2
Numeric facet remove	2	4
Boolean facet remove	4	0
Clear all filters	2	2
Change page number	2	3

TABLE 7
Event incidence by type for each used system in the user experiment.

Third and last, in terms of computational time, our approach outperforms the other automatic approaches, often needing orders of magnitude less time to return the sorted facets for a query. For example, the total computation time for the *Kim et al.* method, on average, is more than 1 second per click. Our approach needs approximately 100 milliseconds per click, which fits the requirements of Web shops and other e-commerce applications, where latencies in terms of seconds are found to be highly undesired [27]. The reason for why the method of Kim et al. is slower stems from the fact that it relies on computing the conditional entropy for every property pair p_i, p_j ($p_i \neq p_j$), which in turn relies on computing the entropy between the property p_i and all property values $b \in V_j$, where V_j are all the values for property j.

We have also found that ranking specific facets higher does sometimes have a downside. This occurs when a facet is so specific that the user has difficulties to identify it. For instance, the qualitative `Screen Resolution` property is ranked relatively high initially. There are so many different screen resolutions available that the user might be overwhelmed by the decision to choose one. The users might also be indifferent with respect to the different resolutions, which makes the property less attractive. At the same time, the property `Lowest Price (€)`, which is generally considered a more useful property for filtering products, is ranked lower. This shows that achieving faster drill-down does not only involve mathematical optimization but also taking into account user experience and behavior. Our method can be extended by introducing weight parameters for each facet score that positively or negatively influence the final score in order to take into account these aspects.

4.3 Results using the experiment with real users

Besides the extensive experiments performed using simulation, we also performed an experiment with real users. The experiment consisted of 10 small tasks[1], where each task would take the user approximately one minute to

1. https://db.tt/5DRnsIhS

	Ordering Scheme			
	Expert-Based	Greedy Count	Kim et al.	Our approach
user effort:				
# clicks (X_c)	4.0	28.2	19.7	2.3
# clicks std dev	1.24	18.65	14.04	0.68
prop scan effort (X_p)	0.0538	0.1914	0.0630	0.0267
prop scan effort std dev	0.0273	0.0891	0.0351	0.0124
facet scan effort (X_f)	0.1462	0.2438	0.4550	0.2111
facet scan effort std dev	0.0908	0.0952	0.1516	0.1718
other measures:				
computation time (ms)	4	23,386	49,818	187
computation time std dev	3.7	26,832.4	45,129.9	74.9
successful sessions (%)	100.00%	100.00%	100.00%	100.00%

TABLE 4
Results for the Least Scanning Drill-Down Model.

	Ordering Scheme			
	Expert-Based	Greedy Count	Kim et al.	Our approach
user effort:				
# clicks (X_c)	1.5	1.5	1.5	1.5
# clicks std dev	0.52	0.52	0.52	0.52
prop scan effort (X_p)	0.3474	0.7232	0.5804	0.2399
prop scan effort std dev	0.2607	0.2091	0.1939	0.2257
facet scan effort (X_f)	0.4659	0.4796	0.4946	0.4547
facet scan effort std dev	0.2730	0.2736	0.2695	0.2764
other measures:				
computation time (ms)	2	25	1,507	160
computation time std dev	0.9	213.2	638.1	61.9
successful sessions (%)	100.00%	100.00%	100.00%	100.00%

TABLE 5
Results for the Best Facet Drill-Down Model

	Ordering Scheme			
	Expert-Based	Greedy Count	Kim et al.	Our approach
user effort:				
# clicks (X_c)	30.7	62.9	59.8	18.8
# clicks std dev	20.05	27.98	20.01	9.77
prop scan effort (X_p)	0.1220	0.1681	0.1524	0.2268
prop scan effort std dev	0.0232	0.0255	0.0297	0.0261
facet scan effort (X_f)	0.3904	0.4842	0.5443	0.3075
facet scan effort std dev	0.0599	0.1100	0.0325	0.0308
other measures:				
computation time (ms)	16	118,155	113,336	2,843
computation time std dev	12.6	72,772.1	53,871.0	2,094.0
# rollups mean	10.7	10.0	16.6	6.2
successful sessions (%)	90.96%	64.00%	79.53%	99.07%

TABLE 6
Results for the Combined Drill-Down Model

complete. The tasks were generated by a script that randomly selects products and includes all properties of the product in the task description. However, for the sake of brevity, properties with multiple values (e.g., 'Audio Formats') were reduced to one (randomly selected) value. For each task, the user was given a set of product features. The users were instructed to find the product(s) that matched all the given properties in each task. In the experiment, we used two systems, where each user performed the first half of the tasks with one system and the second half of the tasks with the other system. The order of the systems was alternated among users

in order to compensate for the learning effect that may occur. The first system is the Web shop implementation of the algorithm proposed in this paper and has been made available online[2] The second system was the 'standard' Web shop[3], i.e., one that has no special features other than those commonly encountered on the Web. It employs a fixed facet list, which is obtained from the Web shop from which the data set is originating [11].

We had a total of 27 users who participated in the experiment, consisting of 17 males and 10 females. There were 19 users that were between 20 and 30 years old, 6 users that were between 31 and 40 years old, and 2 users that was between 40 and 50 years old. These users were mostly students and colleagues from our university and other universities and there was no financial reimbursement for the participation in the experiment.

Table 7 shows the behavior of the users who participated in the experiment, for each of the systems. We can see that most users chose to filter based on the qualitative facets (such as the brand), as indicated by the event 'List facet select'. We notice that users needed less numeric facet changes with our approach than with the standard approach (event 'Numeric facet change'). The results from our user study also suggest that users do not reformulate the query often. Table 7 shows that the filters were cleared only twice in the whole study (event 'Clear all filters'). We can also see that the users spend more time drilling down or rolling up (events 'List facet select' and 'List facet deselect'). Using a paired t-test (measured per task), we can conclude that the users significantly had less interaction (i.e., less events) with our approach than with the standard approach ($p = 0.001867$). We also considered the user effort in terms of how long it took the users to complete the tasks. On average, the users spent 72.4 seconds per task with our approach and 79.9 seconds with the standard approach. The standard deviation is 33.2 seconds for our approach and 33.0 seconds for the standard approach. A paired t-test shows that the difference is significant although the evidence is not very strong ($p = 0.047170$). This might be due to the fact that there is a large difference among users and 27 users is too little to factor out that effect.

5 Conclusion

In this work, we proposed an approach that automatically orders facets such that the user finds its desired product with the least amount of effort. The main idea of our solution is to sort properties based on their facets and then, additionally, also sort the facets themselves. We use different types of metrics to score qualitative and numerical properties. For property ordering we want to rank properties descending on their impurity, promoting more selective facets that will lead to a quick drill-down of the results. Furthermore, we employ a weighting scheme based on the number of matching products to adequately handle missing values and take into account the property product coverage.

We evaluate our solution using an extensive set of simulation experiments, comparing it to three other approaches. While analyzing the user effort, especially in terms of the number of clicks, we can conclude that our approach gives a better performance than the benchmark methods and in some cases even beats the manually curated 'Expert-Based' approach. In addition, the relatively low computational time makes it suitable for use in real-world Web shops, making our findings also relevant to industry. These results are also confirmed by a user-based evaluation study that we additionally performed.

In future we would like to replicate our study on a different domain than cell phones, thereby addressing one of the limitations of the current evaluation. Also we would like to investigate the use of other metrics, such as facet and product popularity, for determining the order and optimal set of facets.

2. http://facet-sorting.eur.dvic.io
3. http://std-prod-search.eur.dvic.io

Acknowledgement

Damir Vandic is supported by an NWO Mosaic scholarship for project 017.007.142: *Semantic Web Enhanced Product Search (SWEPS)*.

References

[1] H. Zo and K. Ramamurthy, "Consumer Selection of E-Commerce Websites in a B2C Environment: A Discrete Decision Choice Model," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 4, pp. 819–839, 2009.

[2] M. Hearst, "Design Recommendations for Hierarchical Faceted Search Interfaces," in *29th Annual International Conference on Research & Development on Information Retrieval (ACM SIGIR 2006)*. ACM, 2006, pp. 1–5.

[3] D. Tunkelang, "Faceted Search," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 1, no. 1, pp. 1–80, 2009.

[4] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst, "Faceted Metadata for Image Search and Browsing," in *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. ACM, 2003, pp. 401–408.

[5] J. C. Fagan, "Usability Studies of Faceted Browsing: A Literature Review," *Information Technology and Libraries*, vol. 29, no. 2, p. 58, 2010.

[6] M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Yee, "Finding the Flow in Web Site Search," *Communications of the ACM*, vol. 45, no. 9, pp. 42–49, 2002.

[7] B. Kules, R. Capra, M. Banta, and T. Sierra, "What Do Exploratory Searchers Look at in a Faceted Search Interface?" in *9th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2009)*. ACM, 2009, pp. 313–322.

[8] Amazon.com, "Large US-based online retailer," http://www.amazon.com, 2014.

[9] V. Sinha and D. R. Karger, "Magnet: Supporting Navigation in Semi-structured Data Environments," in *24th ACM SIGMOD International Conference on Management of Data (SIGMOD 2005)*. ACM, 2005, pp. 97–106.

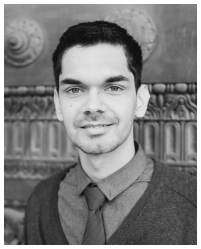
[10] Kieskeurig.nl, "Major Dutch price comparison engine with detailed product descriptions," http://www.kieskeurig.nl, 2014.

[11] Tweakers.net, "Dutch IT-community with a dedicated price comparison department," http://www.tweakers.net, 2014.

[12] Q. Liu, E. Chen, H. Xiong, C. H. Ding, and J. Chen, "Enhancing Collaborative Filtering by User Interest Expansion via Personalized Ranking," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 1, pp. 218–233, 2012.

[13] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering," in *22nd Annual International Conference on Research and Development in Information Retrieval (ACM SIGIR 1999)*. ACM, 1999, pp. 230–237.
[14] J. Koren, Y. Zhang, and X. Liu, "Personalized Interactive Faceted Search," in *17th International Conference on World Wide Web (WWW 2008)*. ACM, 2008, pp. 477–486.
[15] G. M. Sacco and Y. Tzitzikas, *Dynamic Taxonomies and Faceted Search*. Springer, 2009, vol. 25.
[16] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman, "Dynamic Faceted Search for Discovery-Driven Analysis," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008)*. ACM, 2008, pp. 3–12.
[17] S. Liberman and R. Lempel, "Approximately Optimal Facet Value Selection," *Science of Computer Programming*, vol. 94, pp. 18–31, 2014.
[18] D. Vandic, F. Frasincar, and U. Kaymak, "Facet Selection Algorithms for Web Product Search," in *22nd ACM International Conference on Information and Knowledge Management (CIKM 2013)*. ACM, 2013, pp. 2327–2332.
[19] H.-J. Kim, Y. Zhu, W. Kim, and T. Sun, "Dynamic Faceted Navigation in Decision Making using Semantic Web Technology," *Decision Support Systems*, vol. 61, pp. 59–68, 2014.
[20] Y. Zhu, D. Jeon, W. Kim, J. Hong, M. Lee, Z. Wen, and Y. Cai, "The Dynamic Generation of Refining Categories in Ontology-Based Search," in *Semantic Technology*, ser. Lecture Notes in Computer Science, 2013, vol. 7774, pp. 146–158.
[21] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC press, 1984.
[22] C. E. Shannon, "A Mathematical Theory of Communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.
[23] L. E. Raileanu and K. Stoffel, "Theoretical Comparison between the Gini Index and Information Gain Criteria," *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, 2004.
[24] L. Breiman, "Technical Note: Some Properties of Splitting Criteria," *Machine Learning*, vol. 24, no. 1, pp. 41–47, 1996.
[25] L. Ceriani and P. Verme, "The Origins of the Gini Index: Extracts from Variabilità e Mutabilità (1912) by Corrado Gini," *The Journal of Economic Inequality*, vol. 10, no. 3, pp. 421–443, 2012.
[26] AWS, "Amazon Web Services. Large cloud computing provider from Amazon.com," http://aws.amazon.com, 2014.
[27] F. F.-H. Nah, "A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait?" *Behaviour & Information Technology*, vol. 23, no. 3, pp. 153–163, 2004.

Damir Vandic obtained cum laude the master degree in Economics & Informatics from Erasmus University Rotterdam and is currently a PhD candidate at the same university. The focus of his research is on using Semantic Web techniques to improve product search and browsing on the Web and is funded by an NWO Mosaic grant. His research interests cover areas such as machine learning, the Semantic Web foundations and applications, knowledge systems, and Web information systems. He is a member of the editorial board of *Decision Support Systems*.



Steven Aanen has the master degree in Economics & Informatics from Erasmus University Rotterdam with a specialization in Computational Economics and Logistics. His research focuses on improving product search on the Web through the application of Semantic Web technologies. Further research interests include business intelligence, data mining and decision support systems.

Flavius Frasincar obtained the master degree in computer science from "Politehnica" University Bucharest, Romania, in 1998. In 2000, he received the professional doctorate degree in software engineering from Eindhoven University of Technology, the Netherlands. He got the PhD degree in computer science from Eindhoven University of Technology, the Netherlands, in 2005. Since 2005, he is assistant professor in information systems at Erasmus University Rotterdam, the Netherlands. He has published in numerous conferences and journals in the areas of databases, Web information systems, personalization, and the Semantic Web. He is a member of the editorial board of *Decision Support Systems* and the *International Journal of Web Engineering and Technology*.

Uzay Kaymak received the M.Sc. degree in electrical engineering, the degree of chartered designer in information technology, and the Ph.D. degree in control engineering from the Delft University of Technology, Delft, the Netherlands, in 1992, 1995, and 1998, respectively. From 1997 to 2000, he was a reservoir engineer with Shell International Exploration and Production. Currently, he is full professor of information systems in health care at the department of Industrial Engineering & Innovation Sciences of the Eindhoven University of Technology, the Netherlands. Prof. Kaymak is an associate editor of IEEE Transactions on Fuzzy Systems and is a member of the editorial board of several journals.