

Evaluating Faceted Search

Anne Schuth and Maarten Marx

ISLA, University of Amsterdam, The Netherlands
{anneschuth,maartenmarx}@uva.nl

Abstract. We introduce and experimentally compare two evaluation metrics aimed at evaluating systems that select facetvalues for a faceted search interface. Facetvalues are the values of metadata fields in semistructured data and are commonly used to refine queries. It is often the case that there are more facetvalues than can be displayed to a user and thus a selection has to be made. Our metrics evaluate these selections based on binary relevant assessments for the documents in a collection. Both our metrics are based on Normalized Discounted Cumulated Gain, an often used Information Retrieval metric. We show that —as desired— both metrics score a more elaborate system higher than a rather naive one.

1 Introduction

Search interfaces to semistructured data often provide ways of refining a full-text search query by selecting values of metadata fields. These fields and their values are then used to filter the results. The fields are called *facets*, the values are commonly referred to as *facetvalues*.

We can usually only present a limited number of such facetvalues to a user; both because we have limited amount of space (on a screen) but also because we do not want to put a burden of sifting through a large amount of facetvalues on a user. So, out of all facetvalues we need to make a selection. This report investigates ways in which we can do so by defining two evaluation metrics.

In broad terms, we aim at finding a metric that prefers facetvalues that would minimize navigation for a user; that prefers the shortest path through the collection of documents. We want to guide a user in as little as possible steps to all documents that are relevant to his query.

We view the setting in which the selection of facetvalues occurs as follows. We have a collection of documents, some queries and binary relevance judgments (by human assessors) for some documents in the collection for each query, we assume all other documents irrelevant. Besides, we assume that a query defines a strict linear order over the documents.¹ This ordering is given. So, for each query we know which documents are relevant and how all documents should be ordered. Also, all our documents are semistructured, meaning that they contain

¹ Such an ordering can be based on some similarity score between textual data of the document and the query, as is done in our experiments.

textual data (on which the ordering is based) as well as metadata. This metadata determines to which facetvalues a document belongs.

Generally, not all metadata are used as facetvalues since that does not always make sense; only those facetvalues that are shared by more than one document are useful for refining a query. In this work we explicitly select² the metadata fields we want to use as facets. The corresponding facetvalues are taken from the data.

2 Motivation

While there has been work on evaluation faceted search systems from a user interface perspective (Kules et al., 2009; Burke et al., 1996; English et al., 2002; Hearst, 2006, 2008, 2009), no work has focused on evaluation faceted search systems from an Information Retrieval perspective. We view our work as enabling research in that direction and would propose doing so in an evaluation campaign with a setup as described here.

2.1 An Evaluation Campaign

A typical evaluation campaign that focuses on the retrieval of facetvalues could be set up using the following structure:

Task Given are:

- a collection of queries Q ;
- a collection of semistructured documents D ;
- a strict linear order defined on these documents for each query $q \in Q$;
- a set of facets that may be used, the corresponding facetvalues are dictated by the structured part of the document collection.

The task is then to return an ordered list or —depending on the measure— tree of facetvalues that maximizes one of the two metrics defined in this report.

Evaluation Both our metrics, as described in Section 3, can use simple binary relevance judgments on document per query level. And both return a single number that measures how good a list or tree of facetvalues is. This can be averaged over all queries.

To evaluate a participant, the following is needed:

- a collection of semistructured documents D ;
- a strict linear order defined on these documents for each query $q \in Q$;
- binary relevance judgments for each document $d \in D$, for each query $q \in Q$;
- the submission of the participant: a list or tree of facetvalues for each query $q \in Q$.

² With XPath (Berglund et al., 2007) expressions, see Table 3.

Note that these requirements imply that at least the INEX 2010 Data Centric Track (Trotman and Wang, 2010) data and relevance judgments can be used with almost no adjustments, only the order per query over all documents is left to be defined and the topics need slight adjustments. These two issues are discussed in more detail and dealt with in Section 4.1.

3 Two Evaluation Metrics

We introduce two new evaluation metrics. First the Normalized Discounted Cumulated Gain which is an adaption of an existing metric NDCG as described by Järvelin and Kekäläinen (2002). Second, we introduce a new metric called NRDCG which is recursive version of NDCG. Each of our metrics is meant to measure the quality of an *ordered* list or tree of facetvalues.

Notation In Table 1 we first introduce some notation, partly inspired by Dash et al. (2008). This notation is used to describe our two evaluation metrics.

3.1 Normalized Discounted Cumulated Gain

This metric focuses on the following two rather loosely formulated aspects:

1. Prefer facetvalues that would return a lot of relevant documents high in the result list
2. Prefer facetvalues that would return relevant documents we have not seen yet by earlier facetvalues

The first aspect can be measured by counting the number of relevant documents end up in the top p of results if the documentlist D_q were filtered by a facet. The second aspect can then be satisfied by discarding all relevant documents that were already covered by earlier facetvalues, in a ranked list (or three) of facetvalues. Our notion of Gain, as explained below, is designed to capture both aspects.

The fact that we use a discounted measure makes sure we value a facetvalue with a higher Gain more if it is returned earlier in a ranked list of facetvalues. If we allow the (binary) relevance judgments for documents per query to transfer to facetvalues, we get *graded* relevance for facetvalues. We want these relevance judgments —these gains— to reflect the number of relevant documents in the top p result if a facetvalue were selected. Then, to judge the quality of a ranked list of facetvalues, we could use the Normalized Discounted Cumulated Gain (NDCG) measure (Järvelin and Kekäläinen, 2002) that is designed to evaluate a ranking using graded relevance.

In order to be able to compare DCG for several queries, a normalization step is needed. We can use the regular version of Normalized Discounted Cumulated

Table 1. Notation used for the definition of our metrics, inspired by Dash et al. (2008).

d	a document, consisting of triple $\langle t, FV, R \rangle$. With free-text t , set of facetvalues FV and a set of relevance judgments R consisting of $r_q \in \{0, 1\}$ for each query $q \in Q$, where $r_q = 1$ if the document is judged relevant to query q by human judges.
D	list of documents in arbitrary order
D_q	list of documents D ordered by query q
D_f	list of documents D filtered by facetvalue f (in arbitrary order). Or $D_f = \{d : d \in D \wedge f \in FV(d)\}$
f	a facet value pair <i>facet:value</i> . A facetvalue is a property of a document, in filtering operations it preserves only those documents that have this property.
F	list of facetvalues.
FT	tree of facetvalues.
q	a full-text query that can define an ordering on D
Q	a set of full-text queries
$D_q[i \dots j]$	list of documents i up to j in the ordered list of documents D_q . <i>Note that the result of $D[\cdot]$ and $D_f[\cdot]$ is arbitrary since the order of those lists is arbitrary.</i>
$t(d)$	the free-text t of document d
$FV(d)$	the set of facetvalues FV of document d
$r(d, q)$	the binary relevance judgment r of document d with respect to query q
$R(D, q)$	list of the relevant documents given a query q that occur in list of documents D . Or $R(D, q) = \{d : d \in D \wedge r(d, q) = 1\}$
n	maximum number of facetvalues per navigation step
p	maximum number of resulting documents in which to look for relevant documents
λ	used in NRDCG to balance direct Gain with Gain in drill-down, $\lambda = 1$ causes NDRCG to reduce to NDCG. $0 \leq \lambda < 1$.

Gain³

$$NDCG(D, F, q) = \frac{DCG(D, F, q)}{IDCG(D, q)} \quad (1)$$

With a regular definition of Discounted Cumulated Gain.

$$DCG(D, F, q) = \sum_{i=1}^{\min(n, |F|)} \frac{G(D, F, i, q)}{\log_2(i+1)} \quad (2)$$

Only our definition of *gain* is adjusted to reflect the transfer of relevant judgments of documents to facetvalues. We define our version of Gain, $G(D, F, i, q)$

³ We could call DCG and $IDCG$ with $D \setminus D_q[1 \dots p]$ instead of D to reflect the fact that we are interested in *new* relevant documents, and not those that have already shown up in the initial search results.

as follows:

$$G(D, F, i, q) = \left| R(D_{q \& f_i}[1 \dots p], q) \setminus \bigcup_{j=1}^{i-1} R(D_{q \& f_j}[1 \dots p], q) \right| \quad (3)$$

Note how this version of gain does not take relevant documents covered by earlier facetvalues into account. It forces the overall measure to prefer facetvalues that cover *new* relevant documents.

Evidently, for the normalization step, we need to calculate how well an ideal ranked list of facetvalues for this query would do; we calculate the Ideal Discounted Cumulated Gain as follows:

$$IDCG(D, q) = \sum_{i=1}^n \frac{IG(D, i, q)}{\log_2(i+1)} \quad (4)$$

Where our version of the Ideal Gain, $IG(D, i, q)$ is defined as follows:

$$IG(D, i, q) = \max(0, \min(p, |R(D, q)| - (i-1) \cdot p)) \quad (5)$$

It simply states that each i th facetvalue could cover p new relevant document, or less in case all relevant documents were covered by earlier facetvalues.

Averaging over all queries is done by summing up the $NDCG$ for each query and dividing by the number of queries.

$$\overline{NDCG}(D, FT, Q) = \frac{1}{|Q|} \cdot \sum_{q \in Q} NDCG(D, FT, q) \quad (6)$$

Note that this averaging is only possible because we normalized the measure for each query.

3.2 Normalized Recursive Discounted Cumulated Gain

We could also look at the problem of finding the right facetvalues differently. In a real (and possibly even optimal) system a user might have to navigate through multiple facetvalues before he arrives at the desired (relevant) result. In other words, that means that it might take a couple steps before all *relevant* documents end up in the top p results. If we are trying to optimize this navigation we might want to take into account the consecutive facetvalues —and their quality— that a user encounters in a navigation session.

So, instead of looking for the optimal ranked *sequence* of facetvalues, we are looking for an optimal ranked *tree* of facetvalues. We change the setting in the introduction; we now look for a facetvaluetree FT that optimizes our recursive metric.

Such a facetvaluetree FT looks like this:

$$FT = root(f_1(f_1, \dots, f_n), \dots, f_n(f_1, \dots, f_n)) \quad (7)$$

The tree consists essentially of nested lists of facetvalues.

For a tree FT we denote the children of the root with f_i , and we use the notation FT_i to denote the subtree rooted at f_i . The only (natural) restriction on this tree is that paths may not contain a facetvalue more than once.⁴

We define a metric to evaluate this tree in a fashion similar to NDCG, but defined recursively and thus called: Normalized Recursive Discounted Cumulated Gain⁵

$$NRDCG(D, FT, q) = \frac{RDCG(D, FT, q)}{IRDCG(D, q)} \quad (8)$$

We use a recursive version of DCG called Recursive Discounted Cumulated Gain, that is not different except for that it sums up a Recursive Gain.

$$RDCG(D, FT, q) = \sum_{i=1}^{\min(n, |FT|)} \frac{RG(D, FT, i, q)}{\log_2(i+1)} \quad (9)$$

The Recursive Gain, RG , is a mixture model that is composed of a direct gain borrowed from the normal $NDCG$ and a recursive step. Note that for the recursive call we shrink the document set with those that were displayed already; this causes the measure to focus on unseen (relevant) documents in the remaining steps of a drill-down session.

$$RG(D, FT, i, q) = (1 - \lambda) \cdot G(D, FT, i, q) + \lambda \cdot RDCG(D \setminus D_{q, f_i}[1 \dots p], FT_i, q) \quad (10)$$

Setting $\lambda = 0$ reduces the measure to NDCG. Setting $\lambda = 1$ would lead to a zero score (and even division by 0) thus this is not allowed. If $\lambda > 0.5$, the recursive part would get more weight, thereby preferring relevant documents to appear later in a drill-down session. Given that we are after a quick navigation session, we suggest setting $0 < \lambda < 0.5$.

Note that no explicit stopping criteria is needed as $G(\cdot)$ returns 0 for empty document lists and $RDCG(\cdot)$ returns 0 for empty facetvalue lists.

Also, verify that we can indeed simply use $G(D, FT, i, q)$ —even though that function is defined on a list—, as the Gain function is only looking at the children f_i and f_j of the root of FT . Documents (relevant or not) covered by ancestor facetvalues are filtered out by the recursive call to $RDCG(\cdot)$, that is done using $D \setminus D_{q \& f_i}[1 \dots p]$ instead of simply D .

To normalize the $RDCG$, we will need an ideal version, Ideal Recursive Discounted Cumulated Gain ($IRDCG$), which naturally is defined recursively as well.

$$IRDCG(D, q) = \sum_{i=1}^{\min(n, |R(D, q)|)} \frac{IRG(D, i, q)}{\log_2(i+1)} \quad (11)$$

⁴ This restriction is needed because we only filter out the top p results in the recursive call to $RDCG$, and not *all* documents that are covered by a facetvalue

⁵ As with $NDCG$, we could call DCG and $IDCG$ with $D \setminus D_q[1 \dots p]$ instead of D , see Footnote 3 on page 4

The Ideal Recursive Gain, IRG , is defined on the normal IG with a recursive component, balanced with λ .

$$IRG(D, i, q) = (1 - \lambda) \cdot IG(D, i, q) + \lambda \cdot IRDCG(D \setminus R(D, q)[1 \dots p], q) \quad (12)$$

As with $NDCG$, averaging over all queries is simply done by summing up the $NRDCG$ for each query and dividing by the number of queries.

$$\overline{NRDCG}(D, FT, Q) = \frac{1}{|Q|} \cdot \sum_{q \in Q} NRDCG(D, FT, q) \quad (13)$$

4 Experiments

In order to verify that our two metrics indeed measure something meaningful we have set up two systems that are capable of choosing facetvalues. One, called *count* is rather naive. Another called *sumscore* uses a slightly more elaborate technique to select facetvalues. Both systems are evaluated using both $NDCG$ and $RNDCG$ as introduced in the previous section.

We designed these two systems for selecting facetvalues in such a way that one system can be expected to outperform the other; we claim that system *sumscore* is better at selecting the right facetvalues than system *count*. If either or both of our metrics score system *sumscore* higher than system *count* that would be an indication of the correctness of our metrics.

4.1 Experimental Setup

Data We use the data from the INEX 2010 Data Centric Track (Trotman and Wang, 2010). That track used a cleaned version of the Internet Movie Database (IMDb), consisting of 4.418.102 XML files describing movies or persons. The unpacked data is about 23GB in size. The INEX track developed 28 topics (queries) of which 26 topics were assessed. The assessment consisted of a total of 2019 relevance judgments. That comes down to about 78 assessments per topic, topic 2010003 and 2010013 have no judgments. Each topic is defined by a *title*, a *CAStitle* (with structure in the form of an XPath expression), a *description* and a *narrative*. The narrative and description were (supposed to be) used for assessments, the title and CAStitle can be used for querying.

Tools As our XPath/XQuery processor we used eXist version 1.5, a native XML database (Meier, 2009). Running NEXI queries (Trotman and Sigurbjörnsson, 2005) —all CAStitles are NEXI expressions— is easy in eXist because it supports full integration of XQuery with full-text search. Full-text search is implemented using Lucene 2.9. We defined Lucene indexes on all elements E , for which a topic exists which checks whether E is `about()` some full text expression T .

Table 2. Rewrite rules for CASTitles, the resulting NEXI queries can be used by our Baseline system. Note how the last rule simply causes the condition to be deleted; we found that our XQuery processor could not cope with that rule in the context of the dataset.

before	after
OR	or
AND	and
about()	ft:query()
three word phrase	"three word phrase"
./rating < 7	

Baseline Run To provide the strict linear order over the document collection, as mentioned in Section 2, we set up a simple baseline run that uses *only* the CASTitles of each topic as query. We slightly transformed the CASTitles so that they were valid expressions and used our full-text indexes. We used the five rules listed in Table 2 to rewrite the CASTitles.

Besides these rewrites, we have changed the CASTitles of topics 2010010, 2010018, 2010019, 2010022 and 2010028 so that they select either a `//movie` or `//person` node. We did so because it will make our definition of facets, in Table 3, less complex. To illustrate our modifications, the CASTitle of topic 2010018 was originally in the following form:

```
//person[about(.,stanley kubrick)]//direct
```

Using our rules, this was transformed into:

```
//person[ft:query(., "stanley kubrick")]
```

After firing these queries we order the results per topic using the following criteria consecutively to arrive at the strict linear order:

1. `ft:score()` —the eXist interface to the Lucene full-text score— ordered descendingly;
2. the number of descendent nodes of a result ordered descendingly⁶;
3. the document name.

These ranking resulted in a Mean Average Precision (MAP) of 0.4398.⁷

In the remainder of our experiments, we limit the size of these documentlists to a maximum of 200 documents per topic to keep computations tractable. In 17 out of 28 topics this made no difference as the number of returned documents was already below this cutoff of 200. We stored these rankings as documentlist for each topic.

⁶ We noticed that at least for some topics many nearly empty documents were brought up by Lucene.

⁷ With a cutoff of 1000 documents and without topic 2010003 and 20100013 as was done in the evaluation of INEX 2010 Data Centric Track. Our system would have ranked third in that competition and first if the manual runs are excluded.

Parameters The parameters, as described in Table 1, for our two evaluation metrics are set to $n = 5$, $p = 5$ and $\lambda = 0.5$ unless stated otherwise. Besides, we have put a hard limit of 3 on the depth of the facetvalue tree to keep computations within reasonable limits.

Topic Selection Not all 28 topics are fit for our evaluations. If we look at our Baseline run we see that some topics have very few—and some even none—documents in their documentlist. Clearly, facets are of no use and have no meaning if the number of documents that is returned for a topic is very small. Also, documentlists for some topics do not contain any relevant documents. Such topics would be useless for evaluation. We decided to leave out all topics that have either less than p documents in their list or have no relevant documents in their list. This left us with a total of 17 topics⁸ of which 14 ask for movies and 3 ask for persons. Our Baseline Run could potentially return 3400 documents (200 per topic), and returned in fact 3369 documents. From these documents, 349 in total were judged as relevant by the INEX assessors. The median number of relevant documents per topic is 16.

Facet Selection We have to define a limited number of facets on our documents. We selected facets in such a way that each can cover a broad range of values while these facetvalues are not unique for a single document. Movies and persons have different facets. For each facet, we define its name and an XPath expression by which the facetvalues can be retrieved. The possible values are dictated by the data. We list our selection in Table 3.

Two Systems for Facet Selection We defined two strategies for the selection and ordering of facetvalues. The first system, which we will call *count*, ranks facetvalues based on number of—not necessarily relevant—hits that would be the result if the facetvalue were selected. This is captured by the following calculation:

```
count($hits[facet-path eq $value])
```

A slightly more elaborate system, which we call *sumscore*, orders facetvalues based on the sum of the Lucene full-text score of all hits that would be the result if the facetvalue were selected. In XQuery:

```
sum(for $h in $hits[facet-path eq $value] return ft:score($h))
```

4.2 Results

NDCG We have run and evaluated both systems using our NDCG measure. The simple system *count* obtained a \overline{NDCG} score of 0.2351 while the more

⁸ We use topic 2010005, 2010006, 2010007, 2010011, 2010012, 2010014, 2010015, 2010016, 2010017, 2010019, 2010020, 2010021, 2010022, 2010024, 2010025, 2010026 and 2010027.

Table 3. Names and paths, defined as XPath expressions, of our selections of facets. Some apply to topics that ask for movies, others to topics that ask for persons, there is no overlap.

name	facet-path	applies to
director	./directors/director	movie
writer	./writers/writer	movie
genre	./genres/genre	movie
keyword	./keywords/keyword	movie
actor	./cast/actors/actor/name	movie
producer	./cast/producers/producer	movie
certification	./certifications/certification	movie
language	./additional_details/languages/language	movie
country	./additional_details/countries/country	movie
name	./person/name	person
height	./height	person
other-movie	./filmography/act/movie/title	person
other-movie-year	./filmography/act/movie/year	person
nickname	./nicknames/name	person

elaborate system *sumscore* scored \overline{NDCG} 0.3191. If we look in a bit more detail at the results, in Figure 1, we see that according to NDCG the system *sumscore* scored worse than *count* only 3 out of 17 times.

NRDCG Evaluating with NRDCG, gives and \overline{NRDCG} of 0.1785 for system *count* and an \overline{NRDCG} of 0.257 for system *sumscore*. So, again, *sumscore* is ranked higher than *count*, but if we look at Figure 2 we see that *count* is now ranked higher 4 out of 17 times. Also, when this figure is compared to 1 it is clear that scores are much lower in general. This is due to the fact that the ideal scores —IRDCG— are higher in the recursive metric, causing the measure to drop if relatively few relevant documents.

5 Conclusions

We have shown that both our NDCG and NRDCG metric correlate with an intuitive notion of how to select facetvalues. Selecting facetvalues simply based on the *number* of documents that belong to those facetvalues seems not to be the best criteria. Basing the selection on a similarity score between the query and documents seems to be a better criteria. Both measures confirm this.

One might prefer NDCG over NRDCG for its simplicity while the recursive variant might be preferred because it is much more fine grained. Meaning that NRDCG is better at judging a selection of facetvalues were the number of relevant documents is small and harder to retrieve. We acknowledge that our result do not completely support this statement but believe that in further experiments where the limitations that were put on our experiments are lifted, this will show.

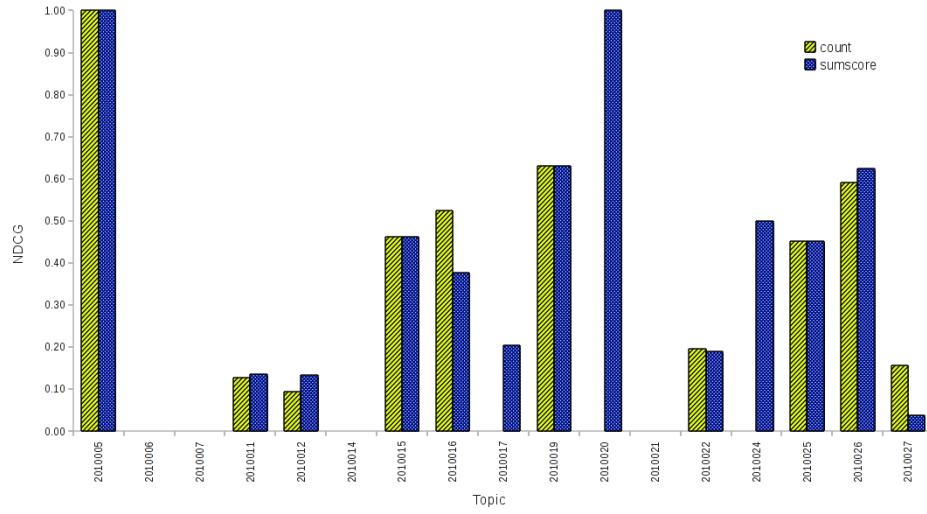


Fig. 1. NDCG measure for our two systems per topic.

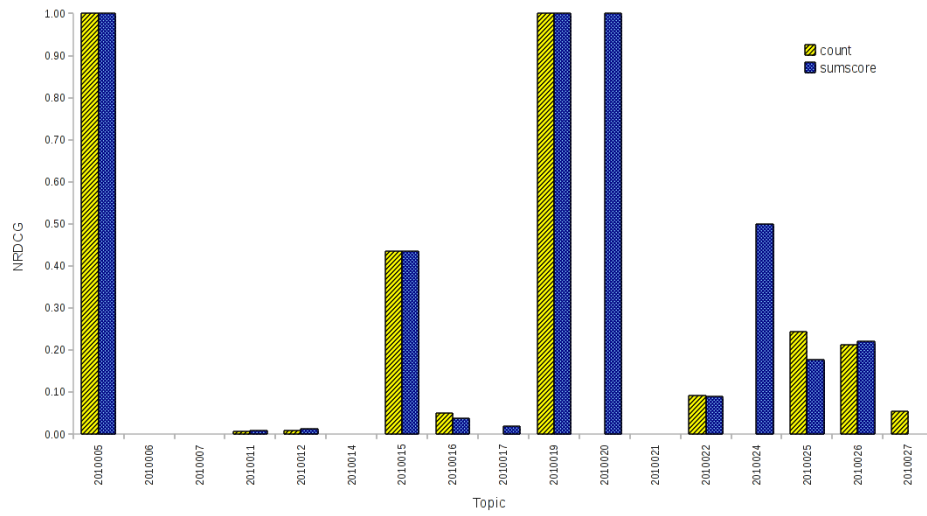


Fig. 2. NRDCG measure for our two systems per topic.

We expect that either of both of our evaluation metrics will foster the development of systems that focus on strategies of selecting the right facet values.

Acknowledgments

Maarten Marx acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599. This research was supported by the Netherlands organization for Scientific Research (NWO) under project number 380-52-005 (PoliticalMashup).

Bibliography

- Berglund, A., Boag, S., Chamberlin, D., Fernandez, M. F., Kay, M., Robie, J., and Simon, J. (2007). XML path language (XPath) 2.0, W3C recommendation 23 January 2007.
- Burke, R. D., Hammond, K. J., and Young, B. C. (1996). Knowledge-based navigation of complex information spaces. In *Proceedings of The National Conference On Artificial Intelligence*, volume 462, page 468.
- Dash, D., Rao, J., Megiddo, N., Ailamaki, A., and Lohman, G. (2008). Dynamic faceted search for discovery-driven analysis. In *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, page 3, Napa Valley, California, USA.
- English, J., Hearst, M., Sinha, R., Swearingen, K., and Yee, K. P. (2002). Hierarchical faceted metadata in site search interfaces. In *CHI'02 extended abstracts on Human factors in computing systems*, page 628639.
- Hearst, M. (2006). Design recommendations for hierarchical faceted search interfaces. In *ACM SIGIR Workshop on Faceted Search*, page 15.
- Hearst, M. (2008). Uis for faceted navigation: Recent advances and remaining open problems. In *Proc. 2008 Workshop on Human-Computer Interaction and Information Retrieval*.
- Hearst, M. (2009). *Search user interfaces*. Cambridge Univ Pr.
- Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20:422446. ACM ID: 582418.
- Kules, B., Capra, R., Banta, M., and Sierra, T. (2009). What do exploratory searchers look at in a faceted search interface? In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, JCDL '09*, page 313–322, New York, NY, USA. ACM. ACM ID: 1555452.
- Meier, W. (2009). eXist: An open source native XML database. *Web, Web-Services, and Database Systems*, pages 169–183.
- Trotman, A. and Sigurbjörnsson, B. (2005). Narrowed extended xpath i (NEXI). *Advances in XML Information Retrieval*, pages 16–40.
- Trotman, A. and Wang, Q. (2010). Overview of the inx 2010 data centric track.