Dynamic Factual Summaries for Entity Cards

Faegheh Hasibi
Norwegian University of Science and Technology
faegheh.hasibi@ntnu.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

Svein Erik Bratsberg
Norwegian University of Science and Technology
sveinbra@ntnu.no

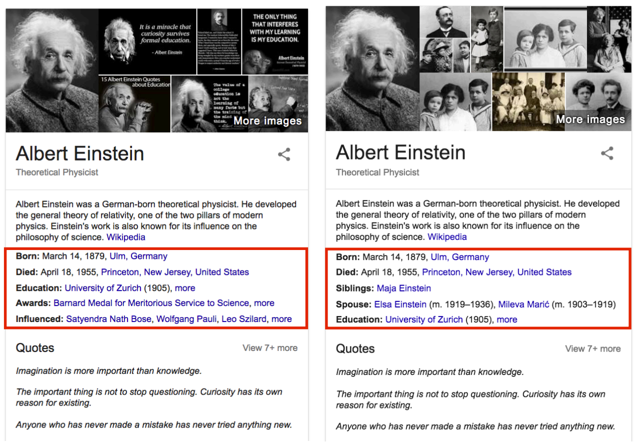
ABSTRACT

Entity cards are being used frequently in modern web search engines to offer a concise overview of an entity directly on the results page. These cards are composed of various elements, one of them being the entity summary: a selection of facts describing the entity from an underlying knowledge base. These summaries, while presenting a synopsis of the entity, can also directly address users' information needs. In this paper, we make the first effort towards generating and evaluating such factual summaries. We introduce and address the novel problem of dynamic entity summarization for entity cards, and break it down to two specific subtasks: fact ranking and summary generation. We perform an extensive evaluation of our method using crowdsourcing. Our results show the effectiveness of our fact ranking approach and validate that users prefer dynamic summaries over static ones.

KEYWORDS

Entity cards; entity summarization; user interfaces

ACM Reference format:
Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Dynamic Factual Summaries for Entity Cards. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080810



Albert Einstein
Theoretical Physicist

Albert Einstein was a German-born theoretical physicist. He developed the general theory of relativity, one of the two pillars of modern physics. Einstein's work is also known for its influence on the philosophy of science. Wikipedia

Born: March 14, 1879, Ulm, Germany
Died: April 18, 1955, Princeton, New Jersey, United States
Education: University of Zurich (1905), more
Awards: Barnard Medal for Meritorious Service to Science, more
Influenced: Satyendra Nath Bose, Wolfgang Pauli, Leo Szilard, more

Quotes — View 7+ more

Imagination is more important than knowledge.

The important thing is not to stop questioning. Curiosity has its own reason for existing.

Anyone who has never made a mistake has never tried anything new.

(a) "einstein education"

Albert Einstein
Theoretical Physicist

Albert Einstein was a German-born theoretical physicist. He developed the general theory of relativity, one of the two pillars of modern physics. Einstein's work is also known for its influence on the philosophy of science. Wikipedia

Born: March 14, 1879, Ulm, Germany
Died: April 18, 1955, Princeton, New Jersey, United States
Siblings: Maja Einstein
Spouse: Elsa Einstein (m. 1919–1936), Mileva Marić (m. 1903–1919)
Education: University of Zurich (1905), more

Quotes — View 7+ more

Imagination is more important than knowledge.

The important thing is not to stop questioning. Curiosity has its own reason for existing.

Anyone who has never made a mistake has never tried anything new.

(b) "einstein family"

Figure 1: Example of entity cards displayed on the Google SERP for different queries. The content of the factual entity summary (marked area) varies depending on the query.

1 INTRODUCTION

Over the recent years, entity cards have become an integral element of search engine result pages (SERPs) on both desktop and mobile devices [6, 25]. Triggered by an entity-bearing search query, a card offers a summary of the entity directly on the results page, helping users to find the information they need without clicking on several documents [33]; see Fig. 1 for examples. Studies have shown that entity cards can enhance the search experience by assisting users to accomplish their tasks [25, 30] and increase their engagement with organic search results [6].

Entity cards are complex information objects, consisting of several components such as images, entity name, short description, summary of facts, related entities, etc. The *factual summary* (or *summary*, for short), which is the focus of this paper, is a truncated view of the top-ranked facts (i.e., predicate-value pairs) about the entity, coming from an underlying knowledge base. Summaries serve a dual purpose on the SERP: they offer a synopsis of the entity and, at the same time, can directly address users' information needs. Consider the examples in Fig. 1, from a commercial search engine, and notice how the summary changes, depending on the query. Even though entity cards are now a commodity in contemporary search engines, to the best of our knowledge, there is no published work on how these (dynamic) summaries are generated and evaluated. In this paper, we make the first effort towards filling this important gap. In other words, the question that we address in this paper is this: *How to generate and evaluate factual summaries for entity cards?*

Looking at the literature, the closest work related to this problem area is the task of ranking or selecting the most important facts about an entity, which has been addressed by different research communities over the recent years [9, 18, 19, 35, 36, 42]. All these works focus on the notion of *fact importance*, as the basis of ranking; a common approach is to compute PageRank-like graph centrality measures on the knowledge graph [9, 35, 36]. When considering factual summaries for entity cards, there are three important aspects that need to be addressed:

(i) Importance vs. Relevance. What is deemed important in general about an entity may be irrelevant in a given query context and vice versa. Take for example the predicate *nationality*, which is generally deemed important for a person; it, however, bears little relevance for the query "einstein awards." This calls for *query-aware* entity summarization, where summaries are created by considering not only *fact importance*, but also *fact relevance* to the query.

(ii) Summary generation. Generating an entity summary that will be shown on an entity card entails more than simply listing the

top-k ranked facts. It needs to deal with, among others, issues such as semantically identical predicates (e.g., *homepage* and *website*), multivalued predicates (e.g., *children*), and presentation constraints (e.g., max height and width, which depend on the device).
~~**(iii) Evaluation.**~~ Given the size of entity cards, it can reasonably be assumed that users consume all facts displayed in the summary. Therefore, in addition to evaluating the ranking of facts, the quality of the summary, as a whole, should also be assessed, with respect to the query. A fair comparison requires side-by-side evaluation of factual summaries by human judges.

In this paper, we aim to address the above challenges head-on. It is important to note that this problem area is not limited to web search, where entity cards are typically displayed on the right hand side of the SERP; it also applies to any information access system that involves entities. Consider, for example, serving entity-annotated documents in response to a search query; when hovering over an entity, a context-dependent entity card is displayed to the user. Deciding when an entity card should be presented is a pivotal question, which should be addressed on its own account. This, however, is beyond the scope of this paper. We shall assume that this decision has already been made by a separate component. ~~Our sole focus is the generation of factual summaries for entity cards.~~

To address (i), we present a method for ~~*fact ranking* that takes both importance and relevance into consideration.~~ We design several novel features for capturing and distinguishing between importance and relevance, and combine these features in a supervised learning framework. For (ii), we introduce ~~*summary generation* as a task on its own account, and develop an algorithm for producing a summary that meets the presentation requirements of an entity card~~. Concerning (iii), we build a ~~benchmark for the fact ranking task, obtaining a large number of crowdsourced judgments with respect to both fact importance and relevance~~. In addition, we evaluate the generated summaries, with regard to search queries, by performing user preference experiments via crowdsourcing. The results show that our proposed fact ranking approach significantly outperforms existing baseline systems. We also find that the summaries uniting both fact importance and relevance are preferred over those that are based on a single aspect. Overall, our results confirm the hypothesis that dynamic (query-aware) summaries are preferred over static (query-agnostic) ones; this is especially true for complex relational queries.

In summary, this paper makes the following novel contributions:

- We present the first study on generating and evaluating dynamic factual summaries for entity cards. We formalize two specific subtasks: fact ranking and summary generation (Section 2).
- We introduce DynES, an approach for generating dynamic entity summaries, composed of *fact ranking* and *summary generation* steps. We introduce a set of novel features, for supervised learning, for the fact ranking task and present an algorithm for summary generation (Section 3).
- We design and make available a benchmark for the fact ranking task, with judgments for around 4K entity facts obtained via crowdsourcing. This test collection may be used in both query-aware and query-agnostic settings, which renders it useful not only for the context of web search, but also for entity summarization in general, which has been addressed in previous work [9, 18, 19, 35, 36] (Section 4).
- We perform an extensive evaluation of the proposed methods by (i) measuring fact ranking using the benchmark we developed (Section 5), and (ii) measuring the overall quality of summaries via a user preference study (Section 6).

The resources developed in the course of this study are made available at http://tiny.cc/sigir2017-dynes.

2 PROBLEM STATEMENT

In this section, we describe and formally define the problem of dynamic entity summarization for entity cards. We assume that entities are represented in a knowledge base (KB) as a set of subject-predicate-object ($\langle \mathrm{s}, \mathrm{p}, \mathrm{o} \rangle$) triples.

Definition 1 (**Entity fact**). *An entity fact (or fact, for short) f is a statement about the entity where the entity stands as subject, i.e., $f = \langle \mathrm{p}, \mathrm{o} \rangle$ is a predicate-object pair. We write $\mathcal{F}_e$ to denote the set of facts about the entity e: $\mathcal{F}_e = \{\langle \mathrm{p}, \mathrm{o} \rangle \mid \langle \mathrm{s}, \mathrm{p}, \mathrm{o} \rangle, s = e\}$.*

This definition implies that multi-valued predicates (i.e., predicates with multiple objects) constitute multiple facts. For example, in Figure 1(b), there are two facts (predicate-object pairs) for the *Spouse* predicate: ⟨*Spouse*, *Elsa Einstein*⟩ and ⟨*Spouse*, *Mileva Marić*⟩. Formulating our problem based on the concept of *fact* (instead of predicate [13, 37]) allows us to handle multi-valued predicates properly. We note that the object of a fact can either be a literal or a URI. A literal object is often presented in the entity cards as it is stored in the KB (e.g., *March 14, 1879*), i.e., as a string. A URI object, on the other hand, links to another entity in the knowledge base and should be converted to link with a human readable anchor, when shown in the card (see, e.g., *Elsa Einstein* in Figure 1(b)).

We now define the "goodness" of a fact for an entity summary from various aspects:

Definition 2 (**Importance**). *The importance of fact f for an entity is denoted by i_f and reflects the general importance of that fact in describing the entity, irrespective of any particular information need.*

Definition 3 (**Relevance**). *The relevance of fact f to query q, indicated by $\mathrm{r}_{f,q}$, reflects how well the fact supports the information need underlying the query. E.g., a fact may hold the answer to the query or help explain why the entity is a good result for that query.*

Definition 4 (**Utility**). *The utility of a fact, $\mathrm{u}_{f,q}$, combines the general importance and the relevance of a fact into a single number, using a weighted combination of the two (where it is assumed that the two are on the same scale):*

$$\mathrm{u}_{f,q} = \alpha \mathrm{i}_f + \beta \mathrm{r}_{f,q}. \quad (1)$$

For the sake of simplicity, ~~we consider both importance and relevance with equal weight in our experiments, i.e.,~~ $\alpha = \beta = 1$. We note that this choice may be suboptimal, and different query types may require different parameter values. This exploration however is left for future work. The central point that we will demonstrate in our experiments is that incorporating fact relevance (as opposed to considering merely importance) leads to better entity summaries.

Definition 5 (**Fact ranking**): *Fact ranking is the task of taking a set of entity facts (and a search query) as input, and returning facts ordered with respect to some criterion (importance, relevance, or utility). We write $\phi(\mathcal{F}_e, q)$ to denote the ranking function ($\phi : \mathcal{F} \times Q \rightarrow \overline{\mathcal{F}}$) which returns a ranked list of entity facts, $\overline{\mathcal{F}_e}$.*

Once facts are ranked, they should be rendered in the from of an entity summary and presented on the entity card. Entity cards have a strong effect on users' search experience [6, 25, 30], and quality of an entity summary can directly impact users' satisfaction. Therefore, simply presenting users with the top-k ranked facts is insufficient for generating an adequate summary. Additional processing steps are required, which may include, but not limited to: (i) resolving semantically identical predicates (e.g., *homepage* and *website*), (ii) grouping related predicates (e.g., *birth place* and *birth date* as single predicate *born*), (iii) dealing with multi-valued predicates (e.g., *children*), (iv) meeting the presentation constraints imposed by the SERP (e.g., max. height and width), and (v) following certain templates or editorial guidelines (e.g., always displaying birth information in the first summary line). Considering these challenges, we formulate *summary generation* as a separate task.

Definition 6 (**Summary generation**): *Given a ranked list of entity facts $\overline{\mathcal{F}_e}$ as input, summary generation is the task of constructing an entity summary with a given maximum size (height and width), such that it maximizes user satisfaction.*

Thus, in this study, we formulate and address two tasks, as defined above: *fact ranking* and *summary generation*. Both of these tasks are novel and challenging on their own; combining the two, the overall goal of this paper is *dynamic entity summarization*, where "dynamic" refers to the query-dependent nature of the generated summaries (as opposed to static ones).

3 APPROACH

In this section we present our proposed approach, referred to as *DynES* (for **Dyn**amic **E**ntity **S**ummarization). It consists of two steps that are performed sequentially. First, we take a set of entity facts and a query as input, and rank these facts based on utility (i.e., a combination of importance and relevance). Second, using a ranked list of facts as input, we generate an entity summary of a given size (ready to be included in the entity card).

3.1 Fact ranking

We approach the entity fact ranking task as a learning to rank problem, where we optimize the ranking of facts w.r.t. a target label. Formally, we define each fact-query pair (f, q) as a learning instance and represent it with a feature vector $\mathbf{x}_i$. Then, a pointwise ranking function $h(\mathbf{x}_i)$ generates a score $\mathbf{y}_i$. We choose *fact utility* to be our target label, where importance and relevance are taken into account with equal weights. We note that the learning framework allows us to optimize for any other target (e.g., more bias towards importance or relevance). The features we introduce here are designed to be able to handle different types of queries, ranging from named entity queries to verbose natural language queries.

We acknowledge that fact ranking could benefit a lot from a query log; however, since we do not have access to that, our feature design is limited to publicly available data sources. Also note that for long tail (unpopular) entities the search log would not be of much help.

Before we proceed, a word on notation and terminology; see Table 1 for a summary. The underlying knowledge base (KB) consists of $\langle$s, p, o$\rangle$ triples, where the subject s is an entity identifier. To help explain the intuition behind the concepts we introduce, we draw an analogy to document retrieval. The concepts fact frequency ($FF(f)$) and entity frequency ($EF(f)$) are loosely analogous to collection frequency and document frequency. The former counts the total number of triples matching a fact, while the latter considers the number of entities that have that fact. Entities have types assigned to them in the KB (typically several, but at least one per entity). Each entity type may be viewed as a document, with predicates of the entities with that type being terms of the document. Using this analogy, the two type-related concepts, entity frequency of predicate for a type ($EF_{\mathrm{p}}(p, t)$) and type frequency of predicate ($TF_{\mathrm{p}}(p)$), are similar to term frequency and document frequency. The former counts the number of times a given predicate appears in the virtual document of the type (i.e., number of entities with that predicate and type), the latter counts the number of documents (types) which contain that predicate.

Next, we describe the features we designed for capturing fact importance and fact relevance. Unless indicated by a reference, the feature is introduced in this paper, and, to the best of our knowledge, represents a novel contribution.

3.1.1 *Importance features.*

The first set of features reflects the general importance of a fact for a given entity and are computed based on various statistics from the knowledge base.

Normalized fact frequency: The feature counts the overall frequency of the fact in the knowledge base, normalized by the total number of $\langle$s, p, o$\rangle$ predicates in the knowledge base ($|\mathcal{F}|$):

$$NFF(f) = \frac{FF(f)}{|\mathcal{F}|}. \tag{2}$$

We compute two other variants of this feature, $NFF_{\mathrm{p}}(p)$ and $NFF_{\mathrm{o}}(o)$, where the numerator is replaced with fact frequency of predicate $FF_{\mathrm{o}}(o)$ and entity frequency of object $FF_{\mathrm{o}}(o)$, respectively.

Normalized entity frequency: This feature captures the entity-wise frequency of a fact, normalized by the cardinality of entities in the knowledge base ($|\mathcal{E}|$):

$$NEF(f) = \frac{EF(f)}{|\mathcal{E}|}. \tag{3}$$

Similarly to the previous feature, we compute predicate and object variations of the feature ($NEF_{\mathrm{p}}(f)$ and $NEF_{\mathrm{o}}(f)$) by substituting $EF_{\mathrm{p}}(p)$ and $EF_{\mathrm{o}}(o)$ in the numerator.

Type-based importance: The importance of a fact for an entity may not always be captured by the overall knowledge base statistics; the specific entity types should be taken into considerations. This is of particular importance for predicates, as their frequencies are biased towards the most frequent types: predicates of less frequent types have low frequency, although they might be important for that specific type. As introduced in [37], the type-based importance

Table 1: Glossary of the notations.

Name	Notation	Definition
Fact	f	$\langle \mathrm{p,o} \rangle : f_p = p, f_o = o$
Entity facts	$\mathcal{F}_e$	$\{\langle \mathrm{p,o} \rangle \mid \langle \mathrm{s,p,o} \rangle \in KB, s = e\}$
Ranked entity facts	$\overline{\mathcal{F}}_e$	$(f_1, f_2, ..., f_n); n = \lvert \mathcal{F}_e \rvert$
Fact frequency	$FF(f)$	$\lvert\{\langle \mathrm{s,p,o} \rangle \mid \langle \mathrm{s,p,o} \rangle \in KB, p = f_p, o = f_o\}\rvert$
Fact frequency of predicate	$FF_{\mathrm{p}}(p)$	$\lvert\{\langle s',p',o' \rangle \mid \langle s',p',o' \rangle \in KB, p = f_p\}\rvert$
Fact frequency of object	$FF_{\mathrm{o}}(o)$	$\lvert\{\langle s',p',o' \rangle \mid \langle s',p',o' \rangle \in KB, o = f_o\}\rvert$
Entity Frequency of fact	$EF(f)$	$\lvert\{e \mid e \in \mathcal{E}, f \in \mathcal{F}_e\}\rvert$
Entity frequency of predicate	$EF_{\mathrm{p}}(p)$	$\lvert\{e \mid e \in \mathcal{E}, \exists f \in \mathcal{F}_e : f_p = p\}\rvert$
Entity frequency of object	$EF_{\mathrm{o}}(o)$	$\lvert\{e \mid e \in \mathcal{E}, \exists f \in \mathcal{F}_e : f_o = o\}\rvert$
Entity frequency of predicate for a given type	$EF_{\mathrm{p}}(p,t)$	$\lvert\{e \mid e \in \mathcal{E}, t \in type(e), \exists f \in \mathcal{F}_e : f_p = p\}\rvert$
Type frequency of predicate	$TF_{\mathrm{p}}(p)$	$\lvert\{t \mid \langle s',p',o' \rangle \in KB : p' = p, t \in type(s')\}\rvert$

is computed as:

$$TypeImp(p,e) = \sum_{t \in type(e)} EF\mathrm{p}(p,t) \cdot \log \frac{|\mathcal{T}|}{TF_{\mathrm{p}}(f)}, \tag{4}$$

where $|\mathcal{T}|$ is the total number of types in the knowledge base.

Predicate specificity: This feature identifies predicate-specific facts; i.e., facts with a common object, but rare predicate. Take for example the fact ⟨*capital, Ottawa*⟩ for the entity Canada, where the predicate is relatively rare (only for capital cities) and the object is frequent. Predicate specificity, hence, combines the fact frequency of the object with the inverse entity frequency of the predicate:

$$PredSpec(f) = FF_{\mathrm{o}}(o) \cdot \log \frac{|\mathcal{E}|}{EF_{\mathrm{p}}(p)}. \tag{5}$$

Object specificity: In contrast to the previous feature, object specificity captures facts with rare objects, but popular predicates; e.g., the object of value of ⟨*Birth date, 1953-10-01*⟩ is relatively unique, while the predicate is frequent. Formally:

$$ObjSpec(f) = EF_{\mathrm{p}}(p) \cdot \log \frac{|\mathcal{F}|}{FF_{\mathrm{o}}(o)}. \tag{6}$$

It is worth noting that both *PredSpec* and *ObjSpec* represent specificity of a fact and highlight important features from the opposite ends of the spectrum; that is, a fact should have either high *PredSpec* or high *ObjSpec* to be considered important.

Other features: Two other binary features are employed: *IsNum* identifies whether the object is a number or not, and *IsEntity* returns true if the object is an entity URI.

3.1.2 *Relevance features.*

The idea behind the second group of features is to determine the relevance of a fact with respect to the information need, specified by the search query (q). Various sources of information are used to extract these features: the query itself, linked entities in the query, retrieved entities in response to the query, and an external corpus to identify the semantic similarity between terms.

Context length: This feature identifies the number of terms in the query that are not linked to any entity. Formally, it is defined as:

$$ConLen(q) = |\{t | t \in q, t \notin Link(q)\}|, \tag{7}$$

where t denotes a term and $Link(q)$ is the set of query terms that are linked to an entity. To obtain entity annotations for queries, we utilize the TAGME entity linking system [15] through its API, as recommended by Hasibi et al. [23]. This feature helps to distinguish keyword queries from other complex queries, such as list or natural language queries. The underlying motivation is that in case of keyword queries targeting a specific entity (e.g., *"eiffel tower"*), users are more concerned about the most important facts of that entity, while for longer and more complex queries (e.g. *"points of interest in paris"*), entity facts that address the underlying information need (i.e., are relevant to the query) would be deemed more useful from the user's perspective.

Semantic similarity: In order to address the vocabulary mismatch between queries and facts, we compute their semantic similarities based on word embeddings, following recent common practice [7, 34]. Specifically, we use Word2Vec [29] with the 300 dimension vectors trained on the Google news dataset, and employ two methods to compute string similarities: aggregated and centroid similarity. The former aggregates the word-wise cosine similarity between each two words of the strings:

$$SemSimAgg(s,s') = \underset{w \in s, w' \in s'}{\mathrm{agg\,func}} \cos(\vec{w}, \vec{w}'), \tag{8}$$

where the w represents a word of string s, and average and maximum are used as the aggregation functions. The second approach performs the aggregation at the vector level and computes the similarity of centroid vectors $\vec{C}, \vec{C}'$: $SemSimCent(s,s') = \cos(\vec{C}, \vec{C}')$.

In our settings, we substitute s with the query and s' with a predicate or object, thereby computing the semantic similarity for query-predicate and query-object pairs.

Lexical similarity: In addition to semantic similarities, we also compute lexical similarity to capture spelling mismatches. Following [34], we employ the Jaro edit distance and apply it to query-predicate and query-object pairs (i.e., $LexSim_p, LexSim_o$).

Inverse rank: This feature promotes facts with an object value that is considered highly relevant to the query [24]. We rank entities from the KB w.r.t. the query and return the inverse rank of the entity that matches the object value (of the fact). Formally:

$$iRank(q,f) = \frac{1}{rank(f_o, Ret(q))}, \tag{9}$$

Algorithm 1 Summary generation algorithm

```
Input: Ranked facts $\overline{\mathcal{F}_e}$, max height $\tau_h$, max width $\tau_w$
Output: Entity summary $lines$
 1: $\mathcal{M} \leftarrow$ Predicate-Name Mapping($\overline{\mathcal{F}_e}$)
 2: $headings \leftarrow []$                                  ▷ Determine line headings
 3: for $f$ in $\overline{\mathcal{F}_e}$ do
 4:     $p_{name} \leftarrow \mathcal{M}[f_p]$
 5:     if ($p_{name} \notin headings$) AND (size($headings$) $\leq \tau_h$) then
 6:         $headings$.add(($f_p$,$p_{name}$))
 7:     end if
 8: end for
 9: $values \leftarrow []$                                    ▷ Determine line values
10: for $f$ in $\overline{\mathcal{F}_e}$ do
11:     if $f_p \in headings$ then
12:         $values[f_p]$.add($f_o$)
13:     end if
14: end for
15: $lines \leftarrow []$                                     ▷ Construct lines
16: for ($f_p$,$p_{name}$) in $headings$ do
17:     $line \leftarrow p_{name}$ + ':'
18:     for $v$ in $values[f_p]$ do
19:         if len(line) + len(v) $\leq \tau_w$ then
20:             $line \leftarrow line + v$                    ▷ Add comma if needed
21:         end if
22:     end for
23:     $lines$.add($line$)
24: end for
```

where $Ret(q)$ is the list of retrieved entities for the query q. In our experiments, we take a state-of-the-art entity retrieval approach [22] to compute this feature.

Other features: Two additional features we use are (i) the *IsLinked* function that looks up the fact object among the linked entities of the query, and (ii) the Jaccard similarity (*JaccSim*) between the terms of the query and predicate and object of the fact (separately).

3.2 Summary generation

We now turn to the task of generating a summary from the ranked list of entity facts. Our proposed approach, presented in Algorithm 1, has three main features that are believed to result in high quality summaries for entity cards: (i) it creates a summary of a given size, (ii) identifies identical facts and filters out unnecessary ones, and (iii) handles multi-valued predicates. We note that summary generation may involve additional processing steps (cf. §2). Our focus of attention here is to emphasize the essence of entity summary generation as a separate task and to address the minimum requirements for summaries that will be used on entity cards.

The algorithm takes as input a ranked list of entity facts $\overline{\mathcal{F}_e}$, and maximum height and width thresholds τ_h, τ_w. The output is maximum τ_h summary lines, each with a heading and one or multiple corresponding values with a maximum width of τ_w characters. Figure 2 illustrates these concepts.

The first step in generating the summary is to map knowledge base predicates to a human readable name. This is of particular

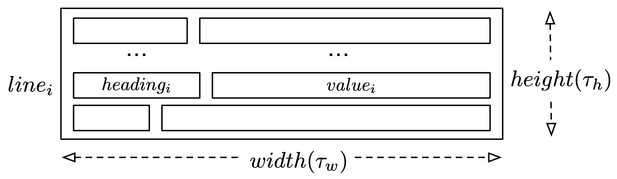


Figure 2: Structure of an entity summary in entity cards.

importance as the same fact may be described with different predicates; e.g., both `dbo:BirthDate` and `dbp:DateOfBirth` have the same meaning. Recognizing these semantically identical predicates and mapping them to a canonical name is encapsulated in the function *Predicate-Name Mapping* (line 1 of Algorithm 1). Depending on the underlying knowledge base, this task can be highly non-trivial. In our experiments using DBpedia, we take two predicates semantically identical if one of the followings holds: (i) one predicate name (irrespective the prefix) is a plural form of the other, (ii) all object values of two predicates are identical, while the predicate names partially match each other.

The summary is built in three stages. First (from line 2 of Algorithm 1), the headings for each summary line are selected; the algorithm keeps the unique predicates corresponding to facts, such that the number of predicates does not exceed the threshold τ_h. Next (from line 9 of Algorithm 1), the values for each line are selected; this is the part where values for the multi-valued predicates are collected. Finally (from line 16 of Algorithm 1), the heading (human-readable predicate) and object values are concatenated together such that they meet the width constraint τ_w.

4 ESTABLISHING A BENCHMARK

There is no existing test set for fact ranking that considers queries. Therefore, we develop and make publicly available a fact ranking benchmark via crowdsourcing, as we shall explain in this section.

4.1 Data sources

To build the collection, we need a set of entity-bearing queries with their corresponding entities that should be summarized. Below, we describe the data sources used for this purpose.

Knowledge base. We use DBpedia (version 2015-10) as our knowledge base, and restrict entities to those with a title and short abstract (`rdfs:label` and `rdfs:comment`); entities without these attributes may not be of sufficient importance to be presented as a card. Since we are concerned with generating entity summaries, we blacklisted predicates that are related to the other parts of the entity card, such as image, entity name and type, abstracts, and related entities. Furthermore, we filtered out noisy predicates that consist of numbers or a single character. To ensure that entity summarization is a meaningful exercise (i.e., entities have enough number of facts to select from), our collection is restricted to entities with at least 5 "valid" predicates after filtering.

Queries. The queries are taken from the DBpedia-entity dataset [2], which is a standard test collection for entity retrieval [2, 8, 27, 31, 44]. It contains 485 queries from 4 different categories: **SemSearch ES** consisting of named entity queries (e.g., "ashley wagner", "carolina"), **List Search** made up of different entity list queries (e.g., "ratt albums"), **QALD-2** containing natural language queries (e.g., "Who founded Intel?"), and **INEX-LD** consisting of general keyword queries, including type, relation, and attribute queries (e.g. "vietnam war facts", "England football player highest paid").

4.2 Selecting entity-query pairs

Given a set of queries, the next step is to form query-entity pairs that will constitute the input for query-dependent entity summarization. For each query in the DBpedia-entity collection, we select a single entity that is (i) known to be relevant and (ii) generally the most easily "retrievable." We measure retrievability by considering several entity retrieval approaches from the literature and establishing a voting schema among them. Bear in mind that we do not decide whether the entity card should be displayed or not; we assume that our information access system generates a card for a retrievable and presumably relevant entity (cf. §1). We also note that our focus of attention in this paper is on generating a summary for a given (assumed to be relevant) entity and not on the entity retrieval task itself. We therefore treat entity retrieval as a black box and combine several approaches to ensure that the findings are not specific to any particular entity retrieval method.

Formally, for a query q, we define $\widehat{E_q}$ as the set of relevant entities according to the ground truth, and $E_{q,m}$ as the set of entities retrieved by method $m \in M$, where M denotes the collection of retrieval methods. A single entity e is selected for q such that:

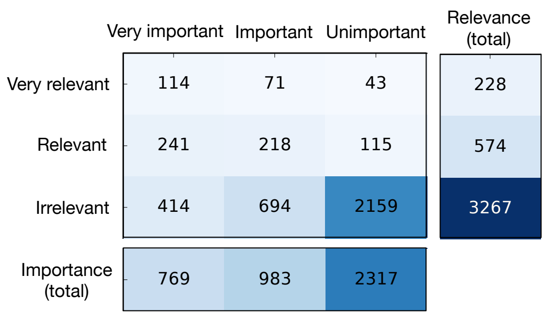
$$\begin{aligned} e &= \arg\max_{e_q \in E_q} \sigma(e_q), \\ \sigma(e_q) &= \frac{1}{|M|} \sum_{m \in M} \frac{1}{rank(e_q, E_{q,m})}, \\ E_q &= \{e | e \in \widehat{E_q}, \exists m \in M : e \in E_{q,m}\}. \end{aligned}$$

Basically, we select the entity that is retrieved at the highest rank by all methods, on average. (If the entity is not retrieved by method m, then the reciprocal rank is set to 0 by definition.) For our experiments, we consider 6 different entity retrieval approaches: BM25 and BM25F-all from [2], SDM and FSDM from [44], and SDM+ELR and FSDM+ELR from [22].

Using our voting mechanism, we were able to extract relevant entities for 421 queries. The average σ score for the selected entities is 0.32, meaning that these entities are retrieved among the top-3 rank positions, on average, by all retrieval methods. For the remaining 64 queries, none of the above methods could retrieve relevant results ($E_q = \emptyset$). These were mostly complex natural language queries. To avoid introducing any bias against these in our collection, we still included them and randomly selected one entity per query from the ground truth entities ($\widehat{E_q}$). Due to pragmatic reasons (i.e., keeping the evaluation costs sensible), we randomly chose 100 entity-query pairs, evenly spread across the 4 different query categories. For each entity in this selection, we extracted the facts from the underlying knowledge base, resulting in a total of approximately 4K facts.

4.3 Fact ranking test set

We build our fact ranking test set by collecting human judgments using the CrowdFlower (CF) platform. We designed two independent tasks to assess the importance and relevance of entity facts. In one task, workers were presented with a single fact for an entity, and were asked to rate the importance of the fact w.r.t. the entity on a 3-point Likert scale: unimportant, important, or very important. In the other task, the search query was also presented in addition, and workers were asked to assess the relevance of the entity fact w.r.t. the query using a 3-point scale: irrelevant, relevant, or very relevant. For both tasks, workers were educated on the concepts of entity cards and entity summaries via examples. They were also supplied with a short description about the entity and a link to the entity's Wikipedia page, to learn more about the entity, if needed.



	Very important	Important	Unimportant	Relevance (total)
Very relevant	114	71	43	228
Relevant	241	218	115	574
Irrelevant	414	694	2159	3267
Importance (total)	769	983	2317	

Figure 3: Distribution of entity facts for different levels of importance and relevance.

Several policies were adopted to obtain high quality results form the crowdsourcing experiments. Only the most trusted workers (level 3 on CF) were allowed to perform the tasks, and they had to retain 80% accuracy throughout the job. The workers who did not meet this threshold or spent very little time on each record, were banned from the rest of task and their judgments were considered untrusted. We also paid with a reasonably high price (¢1 per record) to keep the high quality workers satisfied. Each record was judged by 5 different workers and the Fleiss' Kappa inter-annotator agreement was moderate: 0.52 and 0.41 for importance and relevance, respectively.

Figure 3 shows the distribution of the collected judgments. Considering importance judgments on their own, nearly half of the facts are rated as unimportant, while the rest are (almost evenly) distributed among the two other categories. As for relevance, around 81% of the facts are considered irrelevant, 14% are relevant, and only 5% are judged as very relevant. Taking the combination of these two aspects, the highest correlation is between unimportant and irrelevant facts (53%), while the lowest one is among unimportant facts that are highly relevant to the query (only 1% of all facts). Following our definition of utility (cf. §2), we combine importance and relevance with equal weights. In the end, we have three sets of ground truth for fact ranking, based on importance (3-point scale), relevance (3-point scale), and utility (5-point scale).

5 FACT RANKING RESULTS

In this section we present on our experimental results for the fact ranking task, and address the following research questions: (**RQ1**) How does our fact ranking approach compare against the state-of-the-art? (**RQ2**) How does fact ranking performance compare with respect to importance vs. relevance vs. utility?

5.1 Settings

We chose Gradient Boosted Regression Trees [16] as our learning model because it is shown as one of the best performing learning algorithms on a range of tasks [4, 28, 41]. We set the number of trees to 100 and the maximum depth of the trees to approximately 10% of our feature set; that is $d = 3$ when all features are used and $d = 2$ when trained either on importance or relevance features. All the experiments are performed using 5-fold cross validation, ensuring that facts of the same entities are kept together. We report on NDCG at ranks 5 and 10.

We use a two-tailed paired t-test to measure statistical significance. Significant improvements are marked with $^{\triangle}(\alpha = 0.05)$ or $^{\blacktriangle}(\alpha = 0.01)$, and we write $^{\triangledown}$and $^{\blacktriangledown}$for a drop in performance (for $\alpha = 0.05$ and $\alpha = 0.01$, respectively); $^{\circ}$stands for no significance.

5.2 Experiments

We report on various instantiations of our fact ranking approach in order to be able to tell apart the effect of considering fact relevance in addition/as opposed to fact importance: (i) **DynES** uses all features and is trained on utility judgments; (ii) **DynES/imp** considers importance features only and is trained on importance judgments; and (iii) **DynES/rel** employs relevance features only and is trained on relevance judgments.

We identified three approaches from the literature that can be considered as fact ranking baselines (cf. §7):

- **RELIN [9]** employs a variation of the PageRank algorithm to rank RDF triples for each entity. The scores indicate the importance of a fact for an entity and are computed based on the relatedness (or similarity) between two facts as well as their informativeness.
- **SUMMARUM [36]** computes the PageRank score for all entities in the knowledge graph and then takes the sum of the subject and the object scores as the final score for each entity fact.
- **LinkSUM [35]** combines PageRank scores with the Backlink algorithm [39] (a set-based heuristic for discovering related entities).

Both RELIN and our DynES-based approaches generate the scores for both URI and literal facts, while SUMMARUM and LinkSUM can only score URI entity facts and do not consider literal facts (cf. §2 for URI vs. literal facts). Therefore, when comparing SUMMARUM and LinkSUM with other approaches, we report on the results in a tailored setting, where literal facts are filtered our from the results of other approaches. We implemented RELIN based on the source code kindly provided by the authors and set the parameter $\lambda = 1$, as it delivers robust results across various rank positions [9]. We obtained results for SUMMARUM and LinkSUM from their publicly available API[1], offered for DBpedia ver. 2015-10.

5.3 Results

To answer our first research question, we compare the baseline systems with DynES and DynES/imp with respect to importance and utility. (As these baseline systems only address the importance aspect, we do not report on relevance.) As shown in Table 2, our fact ranking approaches perform significantly better than all baselines (with 16% relative improvement of DynES over SUMMARUM w.r.t. NDCG@10). Interestingly, none of the differences between the baseline systems are significant with respect to utility, even though

[1]http://km.aifb.kit.edu/services/link/

Table 2: Comparison of fact ranking against the state-of-the-art of approaches with URI-only objects. Significance for lines $i > 3$ are tested against lines $1, 2, 3$, and for lines $2, 3$ are tested against lines $1, 2$.

Model	Importance		Utility	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RELIN	0.6368	0.7130	0.6300	0.7066
LinkSum	$0.7018^{\triangle}$	0.7031°	0.6504°	0.6648°
SUMMARUM	**0.7181**$^{\blacktriangle\circ}$	**0.7412**$^{\circ\triangle}$	**0.6719**$^{\circ\circ}$	**0.7111**$^{\circ\circ}$
DynES/imp	$0.8354^{\blacktriangle\blacktriangle\blacktriangle}$	$0.8604^{\blacktriangle\blacktriangle\blacktriangle}$	$0.7645^{\blacktriangle\blacktriangle\blacktriangle}$	$0.8117^{\blacktriangle\blacktriangle\blacktriangle}$
DynES	**0.8291**$^{\blacktriangle\blacktriangle\blacktriangle}$	**0.8652**$^{\blacktriangle\blacktriangle\blacktriangle}$	**0.8164**$^{\blacktriangle\blacktriangle\blacktriangle}$	**0.8569**$^{\blacktriangle\blacktriangle\blacktriangle}$

Table 4: Fact ranking performance by removing features; features are sorted by the relative difference they make.

Group	Removed feature	NDCG@10	Δ%	p
	DynES - all features	0.7873	-	-
Imp.	- NEF_p	0.7757	-1.16	0.08
Imp.	- $TypeImp$	0.7760	-1.13	0.14
Rel.	- $LexSim_o/Max.$	0.7793	-0.8	0.20
Rel.	- $iRank$	0.7793	-0.8	0.22
Rel.	- $SemSimAgg_o/Avg.$	0.7801	-0.72	0.25
Imp.	- $IsURI$	0.7802	-0.71	0.22
Imp.	- $PredSep$	0.7812	-0.61	0.25
Rel.	- $ConLen$	0.7819	-0.54	0.35
Imp.	- $ObjSep$	0.7826	-0.47	0.38
Imp.	- NEF	0.7828	-0.45	0.41
Rel.	- $SemSimCent_p$	0.7834	-0.39	0.49
Imp.	- $IsNumber$	0.7851	-0.22	0.72
Rel.	- $JaccSim_o$	0.7851	-0.22	0.70
	- Other features	0.7810	-0.63	0.37

many of the differences are significant for importance. We select RELIN as our baseline for the rest of the experiments, because it performs in par with other systems in terms of utility. More importantly, it is the only system that can rank both URI and literal facts; SUMMARUM and LinkSUM discard all literal facts (even important ones such as birth and death date), which is not desired for entity card use-case.

For the second research question, we compare RELIN against the three variants of our approach. Table 3 presents the results with respect to importance, relevance, and utility. Our first observation is that all DynES variants significantly outperform RELIN in all aspects; the relative improvements of DynES for NDCG@10 are 48%, 50%, and 47% with regards to importance, relevance, and utility, respectively. We also find that all systems perform better in absolute terms, when they are compared against importance or utility as opposed to relevance. Systems that are designed to capture only the importance of facts (i.e., RELIN and DynES/imp) achieve lower NDCG scores for relevance and utility than for importance. DynES/rel and DynES, on the other hand, deliver better results for utility than for importance. These results, while reflecting the expected behavior of the compared approaches, provide evidence that: (i) capturing the relevance of facts needs special treatment and different features from fact importance, and (ii) capturing the relevance aspect is considerably more challenging than importance.

Table 3: Fact ranking results w.r.t importance, relevance, and utility. Significance for line $i > 1$ is tested against lines $1 .. (i-1)$.

Model	Importance		Relevance		Utility	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RELIN	0.4733	0.5261	0.3514	0.4255	0.4680	0.5322
DynES/imp	**0.7851**$^{\blacktriangle}$	**0.7959**$^{\blacktriangle}$	0.4671$^{\blacktriangle}$	0.5305$^{\blacktriangle}$	0.7146$^{\blacktriangle}$	0.7506$^{\blacktriangle}$
DynES/rel	0.5756$^{\blacktriangle\blacktriangledown}$	0.6151$^{\blacktriangle\blacktriangledown}$	0.5269$^{\blacktriangle\circ}$	0.5775$^{\blacktriangle\circ}$	0.6138$^{\blacktriangle\blacktriangledown}$	0.6536$^{\blacktriangle\blacktriangledown}$
DynES	0.7672$^{\blacktriangle\circ\blacktriangle}$	0.7792$^{\blacktriangle\circ\blacktriangle}$	**0.5771**$^{\blacktriangle\blacktriangle\vartriangle}$	**0.6423**$^{\blacktriangle\blacktriangle\blacktriangle}$	**0.7547**$^{\blacktriangle\vartriangle\blacktriangle}$	**0.7873**$^{\blacktriangle\blacktriangle\blacktriangle}$

5.4 Feature analysis

In Table 4 we report on a feature ablation study, where we remove a single feature based on the relative difference it makes in terms of ranking performance (w.r.t. utility). The table shows the top 13 features individually; the rest of the features are grouped together and removed from the feature set all at once. Interestingly, importance and relevance features are evenly distributed among the most influential features. The top-2 features (NEF_p, $TypeImp$) are computed based on fact predicates, while the rest of importance features involve fact objects. As for relevance features, we see four different versions of similarity features, three of them computed based on the object values: $LexSim_o$,$SemSimAgg_o$,$JaccSim_o$. The feature ablation study reveals that some variant of each of the proposed features (except NFF) are among the top features, indicating that each of the designed features captures utility from a specific angle. To further analyze the performance of each individual feature, we compared the features based on their performance as single feature rankers. The results show a large degree of overlap with the top-13 features identified in Table 4.

6 SUMMARY GENERATION RESULTS

In this section we present on our experimental results for the summary generation task and address the following research questions: (**RQ3**) How satisfied are users with the different types of summaries? (**RQ4**) How does our summary generation algorithm affect user preferences?

6.1 Settings

To evaluate the generated summaries, we performed side-by-side evaluation via crowdsourcing. Workers were presented with two summaries of the same entity along with the corresponding query, and were asked to select the preferred summary w.r.t this query, or the tie option when the two summaries are equally good. Providing users with a tie option enables us to clearly discern user preferences and to avoid randomness in the collected judgments. To avoid any bias, the summaries were randomly placed on the left or right side. We collected 10 judgments from level-3 workers for each pair of summaries. The width and height threshold of Algorithm 1 are set to $\tau_w = 70, \tau_h = 5$ in all experiments, inspired by entity cards used in present-day web search engines. The final results are presented as the total number of user agreements on win, loss, and tie options. We also compute the robustness index (RI) [10], defined as $\frac{N^+ - N^-}{|Q|}$ with N^+ and N^- being the number of wins and losses, and $|Q|$ denoting the total number of queries.

6.2 Experiments

We performed the following side-by-side evaluation of summaries to answer RQ3. In all cases, we apply the same Algorithm 1, but feed it with a ranked list of facts from different sources. (i) **DynES vs. DynES/imp** uses DynES vs. DynES/Imp for fact ranking; (ii) **DynES vs. DynES/imp** uses DynES vs. DynES/rel for fact ranking; (iii) **DynES vs. RELIN** compares DynES vs. the top-5 ranked facts from RELIN; and (iv) **Utility vs. Importance** is an oracle comparison, by taking perfect fact ranking results from crowdsourcing.

For RQ4, we compare our summary generation algorithm with three variations of the algorithm, all applied to the utility-based fact ranking. We compare DynES with: (i) **DynES(-GF)(-RF)**, which is Algorithm 1, without grouping of facts with the same predicate (GF), and removing identical facts (RF); (ii) **DynES(-GF)**, which is Algorithm 1, without the GF feature; and (iii) **DynES(-RF)**, which is Algorithm 1, without the RF feature.

6.3 Results

Table 5 shows the results of summary comparison for different fact ranking methods. According to the first row, query-dependent summaries (DynES) are preferred over query-agnostic ones (DynES/imp) for about half of the queries; the opposite is observed for 31% of queries. We performed the same comparison with the oracle setting (last row of Table 5) and witnessed a similar number of wins, but less losses, which is expected due to imperfect fact ranking. This verifies that the preference of dynamic over static summaries is true for both automatic and human generated summaries. When comparing DynES vs. DynES/rel, we observe that DynES wins in 75% of the cases, signifying that a combination of both importance and relevance is required for a profound entity summary. Finally, we measured the accumulated effect of the improved fact ranking and summary generation method by comparing DynES against RELIN. The results show the superiority of DynES, with a robustness index of 0.9. Based on these experiments, the answer to RQ3 is that dynamic utility-based summaries are indeed preferred over static importance- or relevance-based summaries.

Table 6 presents the comparison between different summary generation algorithms. The results clearly show that DynES summaries are preferred over the ones that do not address the individual presentation aspects. It also reveals that the grouping of multivalued predicates (GF features) is perceived as more important by the users than the resolution of identical facts (RF feature). Based on these results, our answer to RQ4 is that the summary generation algorithm has a major effect on user preferences and thus it should be paid attention within the entity summarization task.

6.4 Analysis

We analyze the differences in users preferences on the query level for the first two set of summarization experiments in Figure 4; i.e., we compare DynES with DynES/imp and DynES/rel. Each value in our query preference distribution indicates the number of users who

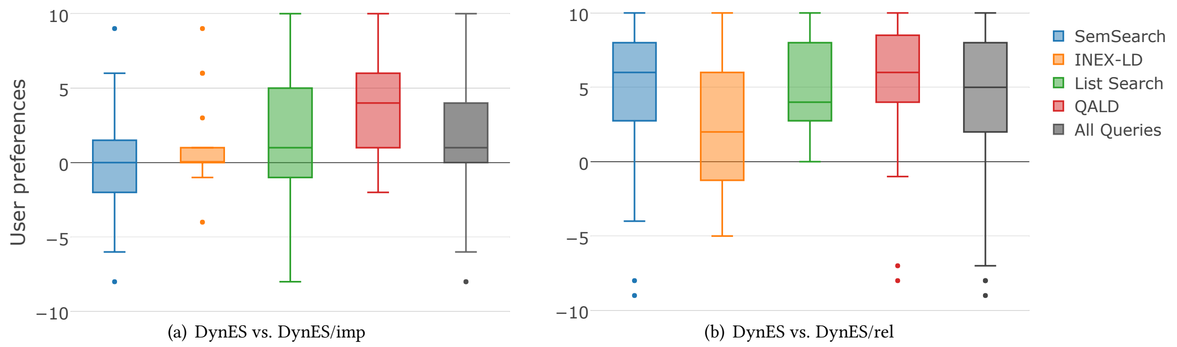


Figure 4: Boxplot for distribution of user preferences for each query subset. Positive values show that DynES is preferred over DynES/imp or DynES/rel.

Table 5: Side-by-Side evaluation of summaries for different fact ranking methods.

Model	Win	Loss	Tie	RI
DynES vs. DynES/imp	46	23	31	0.23
DynES vs. DynES/rel	75	12	13	0.63
DynES vs. RELIN	95	5	0	0.90
Utility vs. Importance	47	16	37	0.31

Table 6: Side-by-side evaluation of summaries for different summary generation algorithms.

Model	Win	Loss	Tie	RI
DynES vs. DynES(-GF)(-RF)	84	1	15	0.83
DynES vs. DynES(-GF)	74	0	26	0.74
DynES vs. DynES(-RF)	46	2	52	0.44

preferred DynES summaries over DynES/imp (or DynES/rel) summaries; ties are ignored. Considering all queries (the black boxes), we observe that the utility-based summaries (DynES) are generally preferred over the other two, and especially over the relevance-based summaries (DynES/rel). These summaries are highly biased towards the query and cannot offer a concise summary; the utility-based summaries, on the other hand, can strike a balance between diversity and bias. Considering the query type breakdowns in Figure 4(a), we observe that the ListSearch and QALD queries, which are identified as complex entity-oriented queries, benefit the most from utility-based summaries. Interestingly, however, we do not observe any clear preferences for SemSearch and INEX-LD queries. This attests that our approach can generate dynamic summaries without hurting named entity and keyword queries.

7 RELATED WORK

The related work pertinent to this paper concerns entity retrieval, entity cards, and entity summarization.

Entity retrieval. Entities play an important role in many information access tasks, including web search [4, 32], enterprise search [1], query understanding [17, 21], document retrieval [11, 14], and table population [43]. Over the past decade, various benchmarking campaigns have focused on entity retrieval, including the INEX 2007-2009 Entity Ranking track [12], the INEX 2012 Linked Data track [40], the TREC 2009-2011 Entity track [3], the Semantic Search Challenge in 2010 and 2011 [5, 20], and the Question Answering over Linked Data (QALD) challenge series [26]. The common goal underlying all these campaign is to address users' information needs by identifying and returning specific entities, as opposed to documents, in response to search queries. In a complementary effort, Balog and Neumayer [2] introduced the DBpedia-Entity test collection, which synthesizes a large number of queries from these benchmarking campaigns and maps the relevant results to DBpedia. Importantly, all these efforts focus exclusively on the ranking of entities and do not deal with the presentation of results to users. In this work, our focus is on the generation of entity cards in a query-dependent manner and not on the actual ranking of entities.

Entity cards. The presentation of entity cards on search engine result pages (SERPs) has recently gained particular attention both in industry [4, 33, 37] and in academia [6, 25, 30, 38]. Most of the research in this area has been geared towards understanding user behavior and interaction with entity cards. Navalpakkam et al. [30] performed eye and mouse tracking on SERPs and showed that relevant entity cards can affect users' attention and, in overall, reduce the amount of time users spend to accomplish their task. In a similar study on mobile search, Lagun et al. [25] interleaved entity cards with organic search results and found that when entity cards are relevant, users can quickly find the answer and complete the task faster. With irrelevant cards, on the other hand, users spend more time on the page looking for the answer and pay attention to the results right below the card. In recent work on card content and structure, Bota et al. [6] showed that entity cards, regardless of their topics, can increase searcher engagement with organic web search results. The focus of attention in these studies is on the search behavior of users, and not on the actual content of the entity cards. Our focus is on the summary part of entity cards.

Entity summarization. Summarizing entities over RDF data has received due attention over the past years [9, 18, 19, 35, 36]. Entity summarization has been also addressed by the more general problem of *attribute ranking* (i.e., ranking entity predicates) [13, 37]. Notable, most of these studies have been performed by different communities in isolation, without knowing about each other.

Cheng et al. [9] introduced entity summarization over RDF data as the task of selecting top-k predicate-object pairs for an entity. Their system, called RELIN, leverages relatedness and informativeness of entity facts using the PageRank algorithm. In similar vein, SUMMARUM [36] and LinkSUM [35] employ the PageRank algorithm to generate ranking scores for predicates involving two entities (i.e., no literal values are considered). These are the closet system to the task we address in this paper and we use them as our baselines. The FACES [19] and FACES-E [18] systems approach entity summarization as a classification task; they partition entity facts into semantically similar groups (facets), and pick the best fact from each facet to form the summaries. The framework presented in [13] ranks RDF attributes for the given entity using learning to rank algorithms. The recent patent by Vadrevu et al. [37] presents an attribute ranking approach for entity summarization. Their proposed approach hinges on a machine-learned ranker (classifier), with the features based on the global and type-specific importance of entity attributes. A major difference between our work and the aforementioned approaches (in addition to being query-dependent) is that we are concerned with the ranking of predicate-object pairs, and not only of predicates.

8 CONCLUSION

In this paper, we have introduced the novel problem of dynamic entity summarization: generating query-dependent entity summaries for entity cards. We have formulated two specific subtasks: fact ranking and summary generation. The first task entails the ranking of facts (predicate-object pairs) with respect to importance and/or relevance. We have addressed it in a learning-to-rank framework, and have demonstrated significant improvements over the most comparable state-of-the-art baselines using a purpose-build test collection. The second task concerns the rendering of ranked facts as a summary to be displayed on the entity card. We have presented a summary generation algorithm and have shown via a series of user preference comparisons that users favor dynamic (query-dependent) summaries over static (query-agnostic) ones. There are several interesting avenues for future work, including the diversification of facts and the personalization of summaries.

Acknowledgments. We thank Andreas Thalhammer for his help and support with the LinkSum and SUMMARUM APIs.

REFERENCES

[1] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. 2009. A Language Modeling Framework for Expert Finding. *Inf. Process. Manage.* 45, 1 (2009), 1–19.

[2] Krisztian Balog and Robert Neumayer. 2013. A Test Collection for Entity Search in DBpedia. In *Proc. of SIGIR '13*. 737–740.

[3] Krisztian Balog, Pavel Serdyukov, Arjen De Vries, Paul Thomas, and Thijs Westerveld. 2010. Overview of the TREC 2009 Entity Track. In *Proc. of TREC '09*.

[4] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. 2013. Entity Recommendations in Web Search. In *Proc. of ISWC '13*. 33–48.

[5] Roi Blanco, Harry Halpin, Daniel M Herzig, Peter Mika, Shady Elbassuoni, Henry S Thompson, and Than Tran Duc. 2011. Entity Search Evaluation over Structured Web Data. In *Proc. of the Intl. Workshop on Entity-Oriented Search*.

[6] Horatiu Bota, Ke Zhou, and Joemon M Jose. 2016. Playing Your Cards Right: The Effect of Entity Cards on Search Behaviour and Workload. In *Proc. of CHIIR '16*. 131–140.

[7] Liora Braunstain, Oren Kurland, David Carmel, Idan Szpektor, and Anna Shtok. 2016. Supporting Human Answers for Advice-Seeking Questions in CQA Sites. In *Proc. of ECIR '16*. 129–141.

[8] Jing Chen, Chenyan Xiong, and Jamie Callan. 2016. An Empirical Study of Learning to Rank for Entity Search. In *Proc. of SIGIR '16*. 737–740.

[9] Gong Cheng, Thanh Tran, and Yuzhong Qu. 2011. RELIN: Relatedness and Informativeness-based Centrality for Entity Summarization. In *Proc. of ISWC '11*. 114–129.

[10] Kevyn Collins-Thompson. 2009. Reducing the Risk of Query Expansion via Robust Constrained Optimization. In *Proc. of CIKM '09*. 837–846.

[11] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity Query Feature Expansion Using Knowledge Base Links. In *Proc. of SIGIR '14*. 365–374.

[12] Gianluca Demartini, Tereza Iofciu, and ArjenP. de Vries. 2010. Overview of the INEX 2009 Entity Ranking Track. In *INEX*. 254–264.

[13] Andrea Dessi and Maurizio Atzori. 2016. A Machine-learning Approach to Ranking RDF Properties. *Future Gener. Comput. Syst.* 54 (2016), 366–377.

[14] Faezeh Ensan and Ebrahim Bagheri. 2017. Document Retrieval Model Through Semantic Linking. In *Proc. of WSDM '17*. 181–190.

[15] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proc. of CIKM '10*. 1625–1628.

[16] Jerome H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Statist.* 29 (2001), 1189–1232.

[17] Dario Garigliotti, Faegheh Hasibi, and Krisztian Balog. 2017. Target Type Identification for Entity-Bearing Queries. In *Proc. of SIGIR '17*.

[18] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Amit Sheth, and Gong Cheng. 2016. Gleaning Types for Literals in RDF Triples with Application to Entity Summarization. In *Proc. of ESWC '16*. 85–100.

[19] Kalpa Gunaratna, Krishnaprasad Thirunarayan, and Amit P Sheth. 2015. FACES: Diversity-Aware Entity Summarization Using Incremental Hierarchical Conceptual Clustering. In *Proc. of AAAI '15*. 116–122.

[20] Harry Halpin, Daniel M Herzig, Peter Mika, Roi Blanco, Jeffrey Pound, Henry S Thompson, and Duc Thanh Tran. 2010. Evaluating Ad-hoc Object Retrieval. In *Proc. of the Intl. Workshop on Evaluation of Semantic Technologies*.

[21] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2015. Entity Linking in Queries: Tasks and Evaluation. In *Proc. of ICTIR '15*. 171–180.

[22] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proc. of ICTIR '16*. 171–180.

[23] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. On the Reproducibility of the TAGME Entity Linking System. In *Proc. of ECIR '16*. 436–449.

[24] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Entity Linking in Queries: Efficiency vs. Effectiveness. In *Proc. of ECIR '17*. 40–53.

[25] Dmitry Lagun, Chih-Hung Hsieh, Dale Webster, and Vidhya Navalpakkam. 2014. Towards Better Measurement of Attention and Satisfaction in Mobile Search. In *Proc. of SIGIR '14*. 113–122.

[26] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. 2013. Evaluating Question Answering over Linked Data. *Web Semantics* 21 (2013), 3–13.

[27] Chunliang Lu, Wai Lam, and Yi Liao. 2015. Entity Retrieval via Entity Factoid Hierarchy. In *Proc. of ACL '15*. 514–523.

[28] Edgar Meij, Wouter Weerkamp, and Maarten De Rijke. 2012. Adding Semantics to Microblog Posts. In *Proc. of WSDM '12*. 563.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proc. of NIPS '13*. 3111–3119.

[30] Vidhya Navalpakkam, LaDawn Jentzsch, Rory Sayres, Sujith Ravi, Amr Ahmed, and Alex Smola. 2013. Measurement and Modeling of Eye-mouse Behavior in the Presence of Nonlinear Page Layouts. In *Proc. of WWW '13*. 953–964.

[31] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. 2016. Parameterized Fielded Term Dependence Models for Ad-hoc Entity Retrieval from Knowledge Graph. In *Proc. of SIGIR '16*. 435–444.

[32] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc Object Retrieval in the Web of Data. In *Proc. of WWW '10*. 771–780.

[33] Filip Radlinski, Nick Craswell, and et. al. 2011. Search Result Driven Query Intent Identification. (2011). US Patent App. 12/813,376.

[34] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2015. Mining, Ranking and Recommending Entity Aspects. In *Proc. of SIGIR '15*. 263–272.

[35] Andreas Thalhammer, Nelia Lasierra, and Achim Rettinger. 2016. LinkSUM: Using Link Analysis to Summarize Entity Data. In *Proc. of ICWE '16*.

[36] Andreas Thalhammer and Achim Rettinger. 2014. Browsing DBpedia Entities with Summaries. In *Proc. of ESWC '14*. 511–515.

[37] Srinivas Vadrevu, Ying Tu, and Franco Salvetti. 2016. Ranking Relevant Attributes of Entity in Structured Knowledge Base. (2016). US Patent 9,229,988.

[38] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Learning to Explain Entity Relationships in Knowledge Graphs. In *Proc. of ACL '15*. 564–574.

[39] Jörg Waitelonis and Harald Sack. 2012. Towards Exploratory Video Search Using Linked Data. *Multimedia Tools Appl.* 59, 2 (2012), 645–672.

[40] Qiuyue Wang, Jaap Kamps, Georgina Ramírez Camps, Maarten Marx, Anne Schuth, Martin Theobald, Sairam Gurajada, and Arunav Mishra. 2012. Overview of the INEX 2012 Linked Data Track. In *CLEF Online Working Notes*.

[41] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. 2016. Beyond Ranking: Optimizing Whole-Page Presentation. In *Proc. of WSDM '16*. 103–112.

[42] Danyun Xu, Gong Cheng, and Yuzhong Qu. 2014. Facilitating Human Intervention in Coreference Resolution With Comparative Entity Summaries. In *The Semantic Web: Trends and Challenges*. 535–549.

[43] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart Assistance for Entity-Focused Tables. In *Proc. of SIGIR '17*.

[44] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *Proc. of SIGIR '15*. 253–262.