A

Composing Structured and Text Databases

Rakesh Agrawal Ariel Fuxman Anitha Kannan Qi Lu John Shafer
Microsoft Corporation {rakesha, arielf, ankannan, qilu, jshafer}@microsoft.com

We postulate a universe of objects in which each object is described by a set of characteristics. Different objects can have the same or different characteristics, but they differ in at least one characteristic. Thus, every object can be conceptually thought to have an implicit unique identity, though the object identity is not manifest. We have two data sources. The first is a collection of structured records, each record containing representative but partial characteristics of some object. The second contains text documents written in a natural language, each discussing some aspects of a small number of objects. There is no marking or structure in the documents that explicitly identifies the objects that the document is about; each text is simply a sequence of words.

Under this general setting, we present a framework for composing structured information about the objects with the textual information about them. The framework is centered around the concept of "trait": a set of characteristics that can serve as the proxy for the identity of an object. Traits might sound similar to database keys, but traits are instance-based rather than schema-based. We present techniques for computing traits, mapping structured records and text documents to traits, and thus joining information about the same object from two repositories. Our extensive experiments using synthetic data demonstrate the effectiveness of our approach under a wide range of operating parameters. Experiments using empirical data validate the results of the synthetic experiments.

1. INTRODUCTION

We live in an era of data abundance. A recent study estimates that the world-wide capacity to store data grew to 295 exabytes in 2007, 94% of which was in the digital form [Hilbert and Lopez 2011]. Increasing amount of this data is becoming accessible on-line [Castle et al. 2009]. For instance, the Word Wide Web now contains tens of billions of web pages [Gulli and Signorini 2005]. A myriad structured repositories are also widely publicly available (see *e.g.,* the Wikipedia entry, 'List_of_online_databases'). It is easy to foresee the usefulness of linking information found in various data sources. In fact, the recent Linked Open Data initiative is precisely aimed at providing specifications for linking data objects from disparate data sources [Bizer et al. 2009].

We consider a particular instance of the problem of automating the task of linking information from various data sources, but in a widely-applicable setting. Many structured records and text documents frequently contain information about some specific object. By object, we simply mean an entity of interest, such as a person, place, product, event, *etc.* We assume we have two data sources. The first is a collection of structured records. Each record contains information about some of the characteristics of some object. A characteristic comprises of a property name and its corresponding value. The record does not explicitly contain the identity of the object that the record is about. The second contains text documents written in a natural language, each discussing some aspects of one or a small number of objects. There is no marking or structure in any of the documents that explicitly identifies the objects that the document is about; each text is simply a sequence of words. Under this general setting, we present techniques for composing structured information about the objects with the textual information about them.

This setting is inspired by the reality of how large collections of structured records and texts are coming into being. Structured database can be arbitrary collection of records obtained from multiple sources. For example, in a product catalog, the product specifications might have been obtained from multiple data providers [Kannan et al. 2011]. By treating records simply as sets of pairs of property names and values, we greatly increase the applicability of our techniques. Similarly, texts can be in any number of formats and by using the

simplest representation for them (sequence of words) and not requiring any categorization of them according to some taxonomy, we greatly increase the reach of our techniques.

The problem we consider is of large practical significance. The applications include scenarios such as:

— A camera manufacturer wants to regularly monitor user attitude towards its products by joining its structured data with web pages and running sentiment analysis over the joined web pages.
— A search engine wants to augment its web index with information about objects mentioned in documents, which can in turn be used at query time. For example, if a user types "sony 55 inch lcd tv", the search engine would be able to return pages about the relevant TVs even if the size of the TVs is not mentioned on the pages. The search engine may also embellish the snippets of its search results with structured information.

The paper proceeds as follows. We first discuss related work in §2. We then formally define the problem of composing objects from text and structured databases in §3 and present our techniques in §4. We present an experimental evaluation of our proposal in §5. We conclude with a summary and directions for future work in §6.

2. RELATED WORK

The research relevant to composing data from multiple sources can be categorized into three streams: i) identifying similar structured records, ii) linking text documents, and iii) matching structured records and text data. The last stream is the one closest to our work. We discuss it in detail and give highlights of the other two. See [Doan and Halevy 2005; Halevy et al. 2006] for an overview of research in the broader field of information integration.

The problem of matching structured records has been studied under various topics including record linkage [Fellegi and Sunter 1969; Newcombe et al. 1959; Winkler 2006], entity resolution [Benjelloun et al. 2009], duplicate detection [Elmagarmid et al. 2007], and merge/purge [Hernández and Stolfo 1995]. The work of Newcombe [Newcombe et al. 1959] (later formalized by Fellegi and Sunter in [Fellegi and Sunter 1969]) pioneered the probabilistic approach to matching structured records. The focus of late has been on designing similarity metrics either at the entire record level [Cohen 1998] or at the attribute level that are subsequently combined to measure record level match [Bilenko et al. 2003].

The thrust of work under the aegis of linking text documents has been on identifying mentions of phrases representing concepts (named entities) in one document and linking them to other documents that have in-depth information about those concepts. Concept phrases are identified and disambiguated using rules, machine learning, or other information extraction techniques [Doan et al. 2008; Paranjpe 2009; Sarawagi 2008]. Examples of work in this direction range from efforts to Wikify Web [Mihalcea and Csomai 2007; Milne and Witten 2008] to enriching educational material by augmenting them with links to authoritative content [Agrawal et al. 2011; Csomai and Mihalcea 2007].

For matching text to structured records, popular approaches involve extracting structured data from text and thus reducing the problem to matching structured records. See [Sarawagi 2008] for a comprehensive survey and taxonomy of various techniques for extracting structure from text. Many of these techniques have been used for algorithmically building structured databases [Carlson et al. 2010; Matuszek et al. 2006; Suchanek et al. 2007; Zhu et al. 2009]. Information extraction methods, however, often have limited accuracy and many require large labeling effort [Dalvi et al. 2009b]. Our proposed approach aims to finesse sophisticated information extraction (in the spirit of [Agrawal and Srikant 2003]).

In [Kannan et al. 2011; Michelson and Knoblock 2008], techniques have been proposed for matching concise text snippets to structured records. Text snippets, for instance, may correspond to brief descriptions of merchant offers that need to be matched to the structured specifications in the product catalog [Kannan et al. 2011]. The essential idea in these

proposals is to identify pieces of a text that are identical to values in structured records, and tag them with the corresponding property names. The text is thus reduced to tuples of pairs, each pair consisting of a value and a set of plausible property names. A match between this representation of a text and a structured record is then scored by choosing the best property name for each value and checking whether the values are the same or different for identical property names of the two. These proposals tacitly assume that the text and structured records have been accurately classified in accordance with some taxonomy. Building good taxonomies and accurate classifiers is hard in practice and we want techniques that do not have this dependency.

Given a set of structured records for objects and a text review, Dalvi *et al.* [Dalvi et al. 2009a] identify the object from the set that is the topic of the review. They hypothesize a language model underlying the creation of reviews: when a review is written about an object, each word in the review is drawn either from a description of the object obtained by concatenating the values in the structured record or from a generic review language that is independent of the object. This mixture model leads to a method for finding the object most likely to be the topic of a review. This work has been generalized in [Dalvi et al. 2009b] so that the values in the structured records are no longer concatenated to allow for attributes to have different weights and admit semantic translations of values. Our hesitation with this proposal again is that its success is dependent upon good pre-categorization of documents and structured records.

The EROCS system views the structured records as a set of predefined entities and identifies those that best match the given document [Chakaravarthy et al. 2006]. The system requires manual input of entity templates, one for each type of entity. An entity template defines how to uniquely determine the best entity match if part or full context of that entity is given. The context for an entity is derived from its corresponding entity template. For example, for a sales transaction, the context could be the customer, store, and product associated with the transaction. It then scores the candidate entities based on matches between the values in the context and the noun phrases in the document. The scoring uses a variant of tf-idf [Manning et al. 2008]. Clearly, the performance of the system depends on how good were the definitions of the entity templates. While our goals are similar, we did not want manual input of the entity templates, which would need to be provided for every category of homogenous entities.

One could hypothetically approach our problem by storing data records in a structured database extended with the keyword querying capability [Yu et al. 2010], treat the document as a query, and apply it to the database. However, these database systems assume that the query consists of a few keywords and does not have many redundant words, which is exactly opposite of our setting. Similarly, one could hypothetically concatenate the values present in the structured record and query the text documents with this string using an IR system [Manning et al. 2008]. This approach will also lead to poor results since IR systems are known to have poor performance when query strings are long [Huston and Croft 2010; Kumaran and Carvalho 2009].

In summary, this paper owes huge intellectual debt to the rich heritage of research in data integration. However, the assumptions, goals, and setting of our work are different, which require new approaches and techniques. As we shall see, we take a novel structured-data-driven approach to solve the problem.

3. CONCEPTUAL FRAMEWORK

3.1. Data Model

Postulate a universe of objects $\mathcal{O}$. Each object $O \in \mathcal{O}$ is described by a set of characteristics. A characteristic X consists of a pair of property name N and its value V. That is, $X \leftarrow \langle N, V \rangle$ (or equivalently $\langle N = V \rangle$), and $O \leftarrow \{X\}$. Property names as well as values can be

arbitrary strings. Denote by $\mathcal{X}$ the set of all characteristics, by $\mathcal{N}$ the set of all property names, and by $\mathcal{V}$ the set of all property values. Different objects can have the same or different characteristics, but they differ at least in one characteristic. Thus, every object can be conceptually thought to have an implicit unique identity, though the object identity is *not* manifest.

We have a structured repository $\mathcal{R}$ as well as a text repository $\mathcal{D}$. The structured repository $\mathcal{R}$ contains the representations of objects from $\mathcal{O}$, called records. Records themselves consist of a set of characteristics (pairs of property names and their values) from objects. There are no duplicates (identical records) in $\mathcal{R}$. An object may be under-represented in $\mathcal{R}$. In other words, property names in a record may be a proper subset of property names in the object. Records do not have primary keys; in fact, an object can have multiple representations in $\mathcal{R}$. That is, there may be more than one record, say R_1 and R_2, corresponding to the same object O, such that they have different set of property names. Since records are in general an under-representation of objects, it is possible that a record may represent more than one object.

The text repository $\mathcal{D}$ comprises of text documents containing information about the objects in $\mathcal{O}$. A text is simply a sequence of words. Specifically, the text does not contain any explicit markings that can be used to identify the objects that the text is about. However, the text may contain some of the property names and values that pertain to the objects described in the text. A text may contain information about objects for which there is no representation in $\mathcal{R}$. Similarly, there may not be any text in $\mathcal{D}$ for an object, although this object has a representation in $\mathcal{R}$.

3.2. Composition

Let $\mathcal{I}$ represent the set of (hypothetical) object identities corresponding to objects in $\mathcal{O}$. Assume a function $RI : \mathcal{R} \rightarrow \mathcal{P}(\mathcal{I})$. Here, $\mathcal{P}(\mathcal{I})$ represents the powerset of identities in $\mathcal{I}$. Given a record $R \in \mathcal{R}$, $RI(R)$ yields the set of object identities $\{I\}$, where every I in this set corresponds to an object $O \in \mathcal{O}$ that R represents. Similarly, the function $DI : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{I})$ is such that $DI(D)$ yields the set of identities of the objects that the text D is about.

Given these mapping functions, we are interested in the result defined by the following composition (written in SQLish syntax):

select R, D
from $\mathcal{R}$ R, $\mathcal{D}$ D
where $\exists I_R \in RI(R) \ \wedge \ \exists I_D \in DI(D) \ \wedge \ I_R = I_D$

We will describe in the next section the algorithms for implementing the above composition. Note that the objects in our framework do not have manifest identities. Our computation methodology therefore does not provide explicit implementation of RI or DI, but aims to obtain the result of the prescribed composition.

4. ALGORITHMS

In our conceptual framework, records and text must be mapped to object identifiers before we can compose $\mathcal{R}$ with $\mathcal{D}$. However, as already mentioned, objects in $\mathcal{O}$ do not have explicit identifiers. Objects are entirely defined and distinguished by their characteristics. We therefore look for distinct characteristics that can proxy for object identifiers. For example, the set of characteristics {brand=Canon, model=350D} may be distinguishing enough to identify a particular camera. Similarly, the single characteristic {model=LN52A750} might be sufficient to identify a particular television. We refer to these distinguishing set of characteristics as *traits*. Thus, given a set of traits for $\mathcal{O}$, if we can identify those traits in $\mathcal{R}$ and $\mathcal{D}$, then we can correctly effect the composition. The mapping functions in our framework are then tantamount to mapping records in $\mathcal{R}$ and documents in $\mathcal{D}$ to a predetermined set of traits. The composition is then performed over these mapped sets of traits.

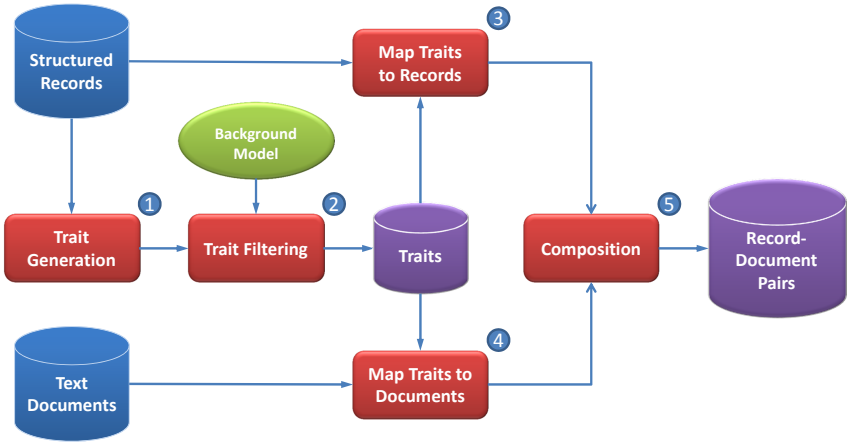


Fig. 1. Overview of Composition

4.1. Overview

Before delving into algorithmic details, we give a quick overview of our approach. Figure 1 shows the major components in our system, which begins with the structured and text databases $\mathcal{R}$ and $\mathcal{D}$ in the left side of the figure and ends with the composition results consisting of scored pairs of records and text documents on the right side.

1. Trait Generation: This is the process by which traits are generated to serve as proxies for object identifiers. This component examines the structured database and computes those combinations of characteristics that can uniquely identify objects. This is arguably the most critical component, as traits are what will ultimately be used to compose $\mathcal{R}$ with $\mathcal{D}$. (See §4.2 and §4.3).

2. Trait Filtering: Initial traits generated in the previous step have strong distinguishing power in the structured database context, but this in not always guaranteed when the same traits are applied in the text corpora context. This component prunes away ineffective traits that make use of distinguishing values that are potentially ambiguous in the text corpus using a background model. For instance, when working with web documents, one could employ a background model built using a web N-gram service (*e.g.,* [Huang et al. 2010]). This step is key to cultivating an effective set of traits and avoiding false-positive results in the final composition. (See §4.4).

3. Mapping Records to Traits: Traits serve as proxy for identifiers, so we need to identify them in both $\mathcal{R}$ and $\mathcal{D}$ in order to effect the composition. This step is responsible for mapping records in $\mathcal{R}$ to the generated set of traits. Since both records and traits are structured sets of characteristics, finding a mapping between the two is fairly straightforward. (See §4.5).

4. Mapping Documents to Traits: The counterpart of the previous step, this step involves mapping documents in $\mathcal{D}$ to the same set of traits. (See §4.6).

5. Composition: Once both $\mathcal{R}$ and $\mathcal{D}$ have been mapped to traits, we compose the two data sets using the mapped traits as the composition key. However, since the presence of a trait is only a binary indicator of a relationship between a record and a document, we use a scoring model to measure final relevance. Relevance scoring not only allows us to eliminate erroneous results, but also allows us to distinguish passing mentions of an object from more in-depth occurrences. (See §4.7).

The remainder of this section describes each of the above components in greater detail.

4.2. Traits

In reality, we do not have access to $\mathcal{O}$ for trait generation. We therefore generate traits from the only concrete representation of $\mathcal{O}$ that we have, namely, $\mathcal{R}$. Traits might sound similar to primary keys in relational databases. But, unlike database keys, traits are instance-based rather than schema-based; different objects may have traits comprised of different characteristics. For example, while a television may be uniquely identified simply by its model characteristic, another television may require several characteristics to disambiguate it. Of course, it is also possible for objects to have more than one trait.

We next formally define what constitutes a trait and then give a trait computation procedure.

4.2.1. Trait Set. We say that a record $R \in \mathcal{R}$ *exhibits* a characteristic $\langle N, V \rangle$ *iff* $\langle N, V \rangle \in R$. For a given integer ρ, a set of characteristics T constitute a ρ-*trait* of $\mathcal{R}$ (or simply *trait*, when ρ is clear from the context) *iff* $\mathcal{R}$ contains at least one and at most ρ records each of which exhibits every characteristic in T and no proper non-empty subset of T constitutes a trait. We will denote the set of all traits of $\mathcal{R}$ as $\mathcal{T}$.

Due to the fact that $\mathcal{R}$ is an imperfect representation of $\mathcal{O}$, the definition of a trait deliberately allows for values of ρ to be greater than one. This flexibility is primarily to accommodate the possibility of objects having multiple representations in $\mathcal{R}$. The amount of relaxation depends on the estimate of multiplicity of representations in $\mathcal{R}$. For example, if $\mathcal{R}$ has been created from n data providers who have overlapping content, setting $\rho \leftarrow n$ is reasonable. The flip side of this relaxation is that a trait may now actually represent multiple objects. We shall see that at the time of composition, we use additional information present in the records and text documents to handle this ambiguity.

4.3. Computing Traits

It turns out that there is a connection between the problem of finding traits and the problem of finding infrequent itemsets. In a database of records consisting of items, a set of items is called a ρ-frequent itemset if all these items are present in at least ρ records [Agrawal et al. 1996]. Correspondingly, an itemset is ρ-infrequent if it is present in fewer than ρ records. Clearly, subsets of frequent itemsets are also frequent and supersets of infrequent itemsets are also infrequent. A minimal ρ-infrequent itemset is one which is ρ-infrequent but each of its proper subsets is ρ-frequent [Boros et al. 2003; Mannila 1997].

CLAIM 4.1. *The problem of computing all ρ-traits is same as the problem of computing all minimal $(\rho + 1)$-infrequent itemsets.*

It has been shown that all maximal ρ-frequent itemsets cannot be found efficiently, unless P=NP. However, all minimal ρ-infrequent itemsets can be found in output quasi-polynomial time [Boros et al. 2003]. In [Haglin and Manning 2007], a depth-first, recursive algorithm has been provided for computing a given number of them. For a fixed budget of total number of minimal ρ-infrequent itemsets, this algorithm is biased toward generating itemsets of larger cardinality. Thus, if used for generating traits, it may generate a trait of cardinality i but exclude a trait of cardinality j, where $j < i$.

As we desire to generate all traits up to a certain maximum cardinality, we present a breadth-first algorithm for trait generation. For composing text and structured databases, traits of large cardinality are not as useful since they are unlikely to be found in arbitrary texts. Additionally, the proposed algorithm is suitable for implementation over big data using database primitives.

4.3.1. TraitGen Algorithm. Recall that a trait $T \leftarrow \{X\}$, where $X \in \mathcal{X}$ and $\mathcal{X}$ denotes the set of all characteristics exhibited in $\mathcal{R}$. Denote by $\mathcal{T}^i$ the set of all traits such that each

Algorithm 1 TraitGen algorithm for mining ρ-traits

Input: Structured Database $\mathcal{R}$, Largest cardinality of desired traits $i_{\max}$, Parameter ρ.
Output: $\bigcup_{j=0}^{i\max} \mathcal{T}^j$.

1: $\mathcal{T}^0 \leftarrow \emptyset$
2: **for** $i \in 1..i_{\max}$ **do**
3: $\quad \mathcal{C}^i \leftarrow \{X^i \in \mathcal{P}(\mathcal{X}) \mid |\{X^i \in R \mid R \in \mathcal{R}\}| \leq \rho\}$
4: $\quad \mathcal{F}^i \leftarrow \{T^i \in \mathcal{C}^i \mid \exists T \in \mathcal{P}(T^i) \text{ s.t. } T \in \bigcup_{j=0}^{i-1} \mathcal{T}^j\}$
5: $\quad \mathcal{T}^i \leftarrow \mathcal{C}^i - \mathcal{F}^i$
6: Output $\bigcup_{j=0}^{i\max} \mathcal{T}^j$

trait has a cardinality of i. For a given ρ, TraitGen (Algorithm 1) produces all ρ-traits of cardinality up to $i_{\max}$.

CLAIM 4.2. *TraitGen correctly computes all ρ-traits of $\mathcal{R}$ for the specified values of ρ and i_{max}.*

PROOF. In iteration i, step 3 finds $\mathcal{C}^i$, the candidate traits of cardinality i, by counting the numbers of records of $\mathcal{R}$ in which all the characteristics of a certain trait X^i are exhibited. However, $\mathcal{C}^i$ may contain false positives. Step 4 checks if a candidate trait T^i contains a confirmed true trait of smaller cardinality and adds it to set of false positives $\mathcal{F}^i$ if true. Step 5 removes such false positives from the candidate set. □

Strategies for efficient implementations of TraitGen are beyond the scope of this paper. However, we give below one pruning strategy to illustrate the non-triviality of the task.

CLAIM 4.3 (VERTICAL PRUNING OF $\mathcal{R}$). *The output of Algorithm 1 does not change if the following statement were added as the last statement of the for loop:* ***if*** $(i = 1)$ $\forall T \in \mathcal{T}^1$ $\forall R \in \mathcal{R}$ $R \leftarrow R - T$.

Observe that the **if** condition is necessary as the following example shows. Consider $\mathcal{R} \leftarrow \{\{a, b, c, d\}, \{a, b, e\}, \{b, c, e\}, \{a, c, e\}, \{d, e\}\}$, $\rho \leftarrow 1$. We will have $\mathcal{T}^1 \leftarrow \emptyset$ in iteration 1 and $\mathcal{T}^2 \leftarrow \{\{a, d\}, \{b, d\}, \{c, d\}, \{d, e\}\}$ in iteration 2. The vertical pruning without the **if** condition would make the first record empty at the end of iteration 2, which would cause the valid trait $\{a, b, c\}$ not to be generated. Thus, we will incorrectly have $\mathcal{T}^3 \leftarrow \{a, b, e\}$, $\{b, c, e\}$, and $\{a, c, e\}$.

4.3.2. Anchored Traits. Because $\mathcal{R}$ is an imperfect representation of $\mathcal{O}$, we must be careful in how we generate traits. We do not want to simply use any unique combination of characteristics in $\mathcal{R}$ as uniqueness in $\mathcal{R}$ is easily fooled by sparsity and duplicity of object representations.

In order to have effective traits, we insist that every trait contain at least one anchor property. Intuitively, anchor properties can be thought of as those that endow pseudo keys to the data. For example, names are a useful, albeit imperfect, method for distinguishing people. Phone numbers do a fairly good job of distinguishing businesses. Anchor properties should have a large number of unique values that are well-distributed amongst records in the database. We next describe how we algorithmically determine anchor properties.

We first compute an anchor score for every property name. A property's anchor score is defined to be the normalized entropy computed over all the values for that property. Intuitively, anchor score indicates the distinctiveness of values for the corresponding property name and we want to have traits comprising of distinctive properties. Let $\mathcal{N}$ represent all the property names present in $\mathcal{R}$. Using $\{\{\cdot\}\}$ to denote a bag, we compute the anchor score

of every property name $N \in \mathcal{N}$ as follows:

$$\mathcal{V}^{'} \leftarrow \{\{V \mid \langle N, V\rangle \in R \ \wedge \ R \in \mathcal{R}\}\}$$
$$\mathcal{V} \leftarrow \{V \mid \langle N, V\rangle \in R \ \wedge \ R \in \mathcal{R}\}$$
$$\text{Anchor Score} \leftarrow \text{Entropy}(\mathcal{V}^{'})/\log|\mathcal{V}|$$

Thus, $\mathcal{V}^{'}$ contains all the property values corresponding to the property N present in $\mathcal{R}$. In contrast, $\mathcal{V}$ will get just the unique values. The anchor score in step 3 is then computed by normalizing the entropy of values in $\mathcal{V}^{'}$ by the cardinality of $\mathcal{V}$.

Now, order the property names in the decreasing order of their anchor score and designate the top k of them as anchor properties. We experimented with several empirical databases and found that generally there is a sudden steep fall off in the anchor scores and determining k is quite easy in practice. Note that the choice of k influences the choice of $i_{\max}$ in Algorithm 1, as a trait comprised entirely of anchors would be a powerful trait. As such, it would make sense to set $i_{\max} \geq k$. The algorithm as stated can also easily be adapted to ensure that every trait is anchored.

4.4. Filtering ineffective traits

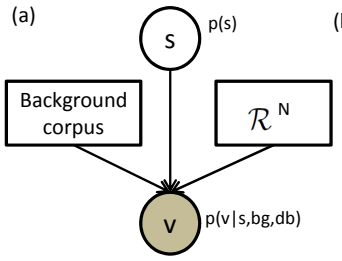
The notion of trait relies on the availability of distinguishing values. However, it is quite possible that very distinguishing values within the structured database might be ambiguous in the text database. For example, both 'LunarFly+' and 'Free' are lines of Nike running shoes that might find their way into the trait set generated from a product database. While the former value continues to be quite distinguishing, the latter has a significant ambiguity issue in many text corpora. We would like to remove traits that contain values that are common in the context of the text repository on which composition is performed. Our approach uses statistics of words (in particular, probability of occurrence) in text documents in comparison to their occurrence in the structured repository $\mathcal{R}$. The comparison is performed for each property name and their corresponding values.

We pose the problem of identifying valid values that can appear in traits as probabilistic inference in a generative model for values. The model explains an observed value as either generated from the values present in $\mathcal{R}$ for the corresponding property name or from a background text corpus. We train the model using an unsupervised learning algorithm that does not require any manually labeled input. Once the model is learned, we perform inference on the model to obtain the probability of a value being explained using $\mathcal{R}$. Those values whose inferred probabilities in $\mathcal{R}$ are below a threshold are identified as ambiguous (and hence ineffective) values, and traits that contain those property name, value combinations are pruned as invalid.

Let $\mathcal{R}^N = \{v_1, \cdots v_K\}$ be the values for the property name N, corresponding to the K records in $\mathcal{R}$. Note that $v_i = v_j$ when the records i and j share the same value for the property name N. Figure 2(a) shows the generative model of a value that is present in the background corpus and as a value for the property name N in $\mathcal{R}$. The binary selection variable s with prior $p(s)$ defines which of the two sources is responsible for generating the value v. Hence, s takes two values, 'db' and 'bg' corresponding to whether the value is generated from $\mathcal{R}^N$ or from the background corpus. Thus the probability of generating a value is defined by the choice of s and is given by $p(v|s)$:

$$p(v|s) = \begin{cases} p(v|bg) & \text{if} \ \ \text{s} = \text{bg} \\ p(v|\mathcal{R}^N) & \text{if} \ \ \text{s} = \text{db} \end{cases}$$

where $p(v|bg)$ and $p(v|\mathcal{R}^N)$ are probabilities of observing the value in the background corpus and in $\mathcal{R}^N$. The joint probability $p(v, s)$ is given by $p(v, s) = p(s)p(v|s)$. When we consider the text documents that correspond to web pages, we use corpus of web pages to build the



(b)

Property Name	Property Values (v)	Frequency of v in db	P(s = db|v)	
Brand	Canon	758	0.961	Canon as product line is invalid while as a brand is not
~~Product Line~~	~~Canon~~	~~22~~	~~5.18E-05~~	
Brand	NextWindow	23	0.951	
~~Brand~~	~~Orion~~	~~1~~	~~1.21E-05~~	
~~Product Line~~	~~Orion~~	~~2~~	~~1.24E-05~~	
Brand	iSonic	26	0.95	While both 'Other' and 'iSonic' have similar frequencies, iSonic is inferred as a valid value for brand and 'Other' as invalid
~~Brand~~	~~Other~~	~~20~~	~~1.15E-05~~	
Brand	Kobian Group	12	0.945	

Fig. 2. (a) Graphical Model for identifying valid values (b) Examples of valid and invalid values (the invalid values are shown by crossing them out).

background model. In particular, we make use of web N-gram service [Huang et al. 2010] to estimate $p(v|bg)$.

Probabilistic inference: Given an observed value $v = v^*$, probabilistic inference involves computing the posterior distribution over the unknown variables in the model. In our setting, this amounts to inferring $p(s = s^*|v = v^*)$, where $s^* \in \{bg, db\}$. We perform probabilistic inference by applying the Bayes rule:

$$p(s = s^*|v = v^*) = \frac{p(v = v^*|s = s^*)p(s = s^*)}{\sum_{s=\{bg,db\}} p(v = v^*|s)p(s)} \tag{1}$$

Learning $p(v|\mathcal{R}^N)$: While $p(v|\mathcal{R}^N)$ can be approximated using relative frequency counts of values in $\mathcal{R}^N$, we instead learn these parameters so that the values that are better explained using the background model are down-weighted in the $p(v|\mathcal{R}^N)$ calculation. For learning $p(v|\mathcal{R}^N)$, we require training data consisting only of those values, $\{v_k\}$, currently present in $\mathcal{R}^N$. Assuming each value in $\mathcal{R}^N$ is independent and identically distributed, we estimate $\theta^* = p(v|\mathcal{R}^N)$ by maximizing $p(\mathcal{R}^N)$:

$$\begin{aligned}
\theta^* &= \arg\max_\theta p(\mathcal{R}^N) = \arg\max_\theta \prod_k p(v_k) \\
&= \arg\max_\theta \log \prod_k p(v_k) = \arg\max_\theta \sum_k \log \sum_{s_k} p(v_k, s_k) \\
&\geq \arg\max_\theta \sum_k \sum_{s_k} p(s_k|v_k) \log \frac{p(v_k|s_k)p(s_k)}{p(s_k|v_k)}
\end{aligned} \tag{2}$$

We solve for θ^* by maximizing the observation probability using the iterative Expectation Maximization algorithm [Dempster et al. 1977]. We initialize the parameters close to uniform distribution. Then, we iterate between the Expectation(E-) step and the Maximization(M-) step. In the E-step, we perform probabilistic inference (Eq. 1) to obtain $p(s_k|v_k)$ for each data point. In the M-step, we use this distribution to fill-in for the unobserved s_k to compute the current estimates for θ:

$$p(v = v'|\mathcal{R}^N) = \frac{\sum_k p(s_k = db|v_k = v')I(v_k = v')}{\sum_k p(s_k = db|v_k = v')}, \tag{3}$$

where $I(v_k = v')$ is the indicator function that evaluates to 1 when $v_k = v'$. This equation can be derived from Eq. 2 by taking derivatives with respect to $p(v = v'|\mathcal{R}^N)$.

Ineffective traits: After learning the parameters of the model, for each value in the database, we perform inference (Eq. 1) to find the probability of the value being explained by the database. We prune traits that contain common values that do not meet a chosen threshold on this probability. Fig. 2(b) shows examples of property name, value pairs that are identified as valid (or invalid) to be present in the traits.

4.5. Mapping Records to Traits

First consider the case when the trait set $\mathcal{T}$ to which records of $\mathcal{R}$ need to be mapped has been generated from $\mathcal{R}$ itself. In this situation, the TraitGen algorithm can simply be modified to concurrently enumerate and persist the $\mathcal{R} \rightarrow \mathcal{T}$ mappings while generating traits.

However, it is conceivable that the trait set has been obtained from an independent source. It could even be a manually curated set. We consider this general case next. Postulate that every record $R \in \mathcal{R}$ has an internal identifier, referred to as R.rid, which uniquely designates R. The mappings from records to their traits are kept in a table RT, whose entries are of the form $\langle R.\text{rid}, R.\text{trait}\rangle$. Given a trait set $\mathcal{T}$, the following will bind all the records in $\mathcal{R}$ to their traits: $RT \leftarrow \{\langle R.\text{rid}, T\rangle \mid T \in \mathcal{T} \ \wedge \ \exists R \in \mathcal{R} \ s.t. \ T \subseteq R\}$.

4.6. Mapping Documents to Traits

The relevant traits in the document are located by employing fast data structures (*e.g.,* [Sarkas et al. 2010]). One may also account for differences in the vocabulary used in the documents and in structured data by incorporating semantic translations of the values present in a trait [Cheng et al. 2010; Dalvi et al. 2009b].

4.7. Composition

Now that we have mappings from structured records as well as text documents to the same set of traits, we can compose information about the same objects from the two repositories by doing a natural join on traits. However, since traits are computed from an imprecise representation of $\mathcal{O}$, we additionally need a function that provides a probabilistic score of agreement between matched documents and structured records.

Our scorer is defined using binary logistic regression based on a set of features that measure the similarity between a text D and a record R. Given matched and mismatched training pairs, let $\{\mathbf{F}, \mathbf{Y}\} = \{(\mathbf{f}_1, y_1), \ldots, (\mathbf{f}_T, y_T)\}$ be the set of feature vectors along with their corresponding binary labels. Logistic regression maximizes the conditional log-likelihood of the training data: $\arg\max_{\mathbf{w}} \log P(\mathbf{Y}|\mathbf{F}, \mathbf{w}) \equiv \arg\max_{\mathbf{w}} \sum_{i=1}^{T} \log P(y_i|\mathbf{f}_i, \mathbf{w})$, where $\mathbf{w}$ is the weight vector wherein each component w_j measures the relative importance of the feature f_j for predicting the label y, and $P(y = 1|\mathbf{f}, \mathbf{w}) \leftarrow 1/(1 + \exp(-(b + \mathbf{f}^T\mathbf{w})))$.

Documents generally contain larger number of property values corresponding to more relevant objects than less relevant ones and also mention them with a much higher repetition frequency. Thus, we define two features: (i) fraction of distinct property names corresponding to property values found in the joining structured record and the text document that overlap (ii) weighted fraction of property values that overlap, where the weight is determined through the relative frequency of the corresponding property name, property value pairs. Because we use a small number of features, we do not need a large amount of training data.

5. EXPERIMENTS

In designing experiments for studying the performance characteristics of the proposed techniques, we primarily employed synthetic data but then validated the results using proprietary empirical data. The use of synthetic data allows us to systematically explore the performance with respect to various operating parameters, which would be difficult, if not

Ψ^O	significance vector for object generation
ϱ	intra-duplication function
β	sparsity function
Ψ^T	importance vector for text generation
M	number of prominent objects in a text
Λ	relative proportions of prominent objects in text
σ	overlap between words in structured data and general vocabulary
γ	overlap between words in prominent objects and remaining objects in structured data

Fig. 3. Key data parameters

impossible, to do with empirical data. It also allows the creation of ground truth against which the accuracy of results could be benchmarked. Finally, it enables independent reproduction of the results by interested researchers. The validation using empirical data ensures that we have not assumed away reality in the synthetic data experiments. See, for example, [Agrawal et al. 1996; DeWitt 1993] for similar reasoning.

5.1. Performance Metric

We focus on the performance of the proposed techniques with respect to the accuracy of results. Regarding execution time, notice that traits are computed from the structured database independently of the corpus (which is usually much larger). This step is also amenable to massive parallelization. The composition with records using traits is done for the documents in the corpus independently of each other, and is therefore embarrassingly parallelizable.

Given text documents about up to M objects, we study the accuracy of the composed records. As our performance measure, we adapt the normalized discounted cumulative gain (NDCG), a metric widely used in IR for measuring relevance [Manning et al. 2008]. We use NDCG rather than precision because it is a stricter metric as it additionally takes into account the ordering in the result. For example, NDCG would penalize the case in which the main object of a document is ranked below (*i.e.,* given a lower match score) a secondary object.

In a typical IR setting, there is a ranked list of result documents for a given a textual query. In our case, we have a ranked list of structured records for a given document along with their match scores. We also have the ideal ordering of the structured records: for synthetic data, this ordering is known a priori; for empirical data, this ordering is obtained by employing human judges.

Given an observed ranked list of records $\mathcal{Q}$ for a given document and an ideal ordering of the same set $\mathcal{H}$, define the discounted cumulative gain at rank threshold M as $\mathrm{DCG}(\mathcal{Q}, M) \leftarrow \sum_{j=1}^{M} (2^{r(j)} - 1)/\log(1+j)$, where $r(j)$ is the match score of the record at rank j in $\mathcal{Q}$. Now, define the normalized discounted cumulative gain as $\mathrm{NDCG}(\mathcal{Q}, M) \leftarrow \mathrm{DCG}(\mathcal{Q}, M)$ / $\mathrm{DCG}(\mathcal{H}, M)$. When $\mathcal{Q}$ is clear from the context, we write $\mathrm{NDCG}(\mathcal{Q}, M)$ equivalently as $\mathrm{NDCG@}M$.

5.2. Synthetic Data Experiments

5.2.1. Synthetic Data Generation. We employ three generators: i) Objects generator (ObjGen), ii) Structured data generator (StructGen), and iii) Text generator (TextGen). We have attempted to design these generators such that the synthetic structured records and text documents exhibit the salient characteristics of those found in empirical data. For ease of exposition, we will describe the generators pretending all objects in $\mathcal{O}$ have the same set of

property names $\mathcal{N}$ (universal schema). The key data generation parameters are summarized in Figure 3.

Algorithm 2 ObjGen for creating $\mathcal{O}$

Inputs: Empty objects $\mathcal{O}$, Property names $\mathcal{N}$, Significance vector Ψ^O, Intra and inter duplication functions: ϱ, ς (which are functions of Ψ^O).
Output: Instantiated $\mathcal{O}$.

1: $\forall O \in \mathcal{O}\ \forall N \in \mathcal{N}$ **do** O.Add($\langle N$, *newValue*()$\rangle$)
2: $\forall N \in \mathcal{N}\ \forall O \in \mathcal{O}$ **do**
3: **if** *perturb*(N, Ψ^O, ϱ, *intra*) **then**
4: Choose $O' \in \mathcal{O}$
5: Replace O's property value for N with O''s property value for N
6: **else if** *perturb*(N, Ψ^O, ς, *inter*) **then**
7: Choose $O' \in \mathcal{O}$
8: Choose N' according to **1**-Ψ^O
9: Replace O's property value for N with O''s property value for N'

Algorithm 3 StructGen for creating $\mathcal{R}$

Inputs: Objects $\mathcal{O}$, $\mathcal{R}$'s cardinality $|\mathcal{R}|$, Significance vector Ψ^O, sparsity function β.
Output: $\mathcal{R}$.

1: $\mathcal{R} \leftarrow$ uniformly sample with replacement $|\mathcal{R}|$ records from $\mathcal{O}$
2: $\forall R \in \mathcal{R}\ \forall N \in R.\mathcal{N}$ **do**
3: **if** *nullify*(N, β) **then**
4: Delete $\langle$R.N, R.V$\rangle$ from R
5: Remove all but one amongst identical records in $\mathcal{R}$

ObjGen (Algorithm 2). ObjGen is designed to incorporate two important features of the real-world objects. First, some property names tend to have the same value for more objects than others (*e.g.,* name of a person vs. gender) and hence some characteristics are more differentiating than others. Second, some property values are shared across different property names (*e.g.,* '5x' is a valid value for both optical and digital zoom). Thus, ObjGen takes as input a significance vector Ψ^O such that property N_j's significance for differentiating objects is given by $\Psi^O[N_j] \in [0, 1]$. The functions that govern the amount of duplication in the objects are defined in terms of $\Psi^O[N_j]$, with less significant properties N attracting more duplicates. These functions are: ϱ, that governs duplication of property values within a property name; and ς, that determines duplication across property names. Generally, duplicate values across property names are rarer than within a property name. Step 1 of the algorithm creates new objects, setting them initially with unique values by invoking newValue(). The rest of the steps change some of the values of the objects thus created by borrowing values from other objects depending on the function perturb(), resulting in duplicate values within and across property names.

StructGen (Algorithm 3). $\mathcal{R}$ contains records corresponding to some of the objects from $\mathcal{O}$. In general, it is possible for an object to have multiple representations, and not all of the characteristics from an object may be present in a particular representation. StructGen therefore chooses $|\mathcal{R}|$ records uniformly with replacement from $\mathcal{O}$ and then invokes nullify() for every property name within a record to determine if a particular property name, value pair should be retained. The probability of a property being retained is independent of how differentiating is the property. Thus, nullify() employs a uniform sparsity function β.

Algorithm 4 TextGen for creating a document $D \in \mathcal{D}$ consisting of L words

Inputs: Number of prominent objects M from $\mathcal{O}$ along with their relative proportions Λ, Overlap σ between words in $\mathcal{R}$ and general vocabulary $\mathcal{W}$, Overlap γ between prominent objects and the remaining objects in $\mathcal{R}$, Fraction ε_n (ε_v) of property names (values) to include from prominent objects, Importance vector Ψ^T indexed by property names.
Output: D consisting of L words.

```
Select a distribution of objects, Λ' = Dirichlet(Λ)
O' ← Uniformly sample without replacement M objects from O
D ← ∅
while |D| ≤ L do
    if FromStruct(σ) & FromDominantObjects(γ) then
        o ← GetObjectIndex(Λ')
        n ← GetPNameIndex(Ψ^T)
        if Include(ε_n) then
            D.Add(N_n)
        if Include(ε_v) then
            D.Add (O'_o's property value for N_n)
    else if FromStruct(σ) then
        O ← Sample an object uniformly from O
        n ← GetPNameIndex(1-Ψ^T)
        D.Add(O's property value for N_n)
    else
        D.Add(word ∈ W)
```

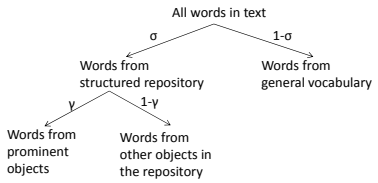


Fig. 4. Parameters σ and γ of text generation

TextGen (Algorithm 4). Our design goal for TextGen was to be able to create a text database such that some documents contain information predominantly about one or more objects from $\mathcal{O}$, and some about none. Henceforth, we will call these the "prominent" objects of the document. Information about an object may consist of some property names and some values from the object, respecting that one usually finds more property values than property names and more important properties get more mention than less important ones. When a document contains information about more than one object, it may be the case that both the objects are equally prominent or one is more prominent than the other. In addition to the prominent objects, the document may also contain passing mentions to other objects.

Many of the terms in a document may not appear at all in the structured repository. We say that those terms come from the "general vocabulary". For example, in a document about reviews, the term "review" may appear on the document but it is unlikely to appear in any property value of the structured repository. The fraction of words appearing in the document from values in the structured repository is specified by the parameter σ; the smaller the value of this parameter, the greater the number of words from general vocabulary $\mathcal{W}$.

Let us now focus on the words in the document that do appear in the structured repository. Some of these might correspond to properties of the prominent objects of the document, while others might be just passing mentions to properties of other objects. For example, a

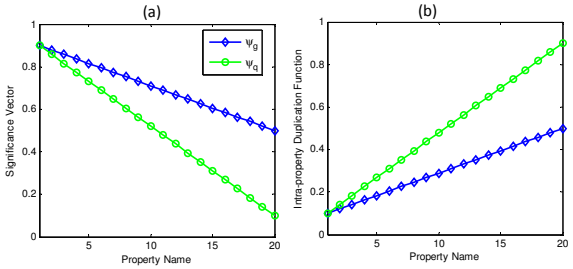


Fig. 5. (a)Significance Vector (b) Corresponding Intra-property duplication with $\varphi = 1$

user reviewing a Nikon 15 megapixel digital camera may mention that she had a 5 megapixel digital camera in the past. We capture this situation with the parameter γ, which specifies the fraction of structured values in the document that come from the prominent objects. See Figure 4 for a pictorial illustration of the σ and γ parameters.

The parameter M determines the number of prominent objects and Λ determines the relative proportion of information from respective prominent objects. The importance vector Ψ^T, which is indexed by property names, specifies the probability of the corresponding property to appear in a document. In general, Ψ^T could be different from Ψ^O. Having determined that a property will be included in the document, probability of including property name (value) is given by parameters ε_n (ε_v).

5.2.2. Parameter Settings. Clearly, one can generate any number of families of synthetic objects by simply varying values of the parameters of ObjGen. An object family is defined by two key parameters: $|\mathcal{N}|$, the number of property names, and Ψ^O, the vector specifying significance of the property names (which in turn determines the intra- and inter- duplication functions ϱ and ς). We number property names from 1 to $|\mathcal{N}|$ in decreasing order of their significance and use linear vectors to specify their significance. Thus, the line segment connecting $(1, \Psi^O[1])$ to $(|\mathcal{N}|, \Psi^O[|\mathcal{N}|])$ defines the significance of different property names.

We study two settings of Ψ^O: (1) $|\mathcal{N}| = 20$ and Ψ^O_q, and (2) $|\mathcal{N}| = 20$ and Ψ^O_g. For both the significance vectors, $\Psi^O_\cdot[1] \leftarrow 0.9$, but $\Psi^O_q[20] \leftarrow 0.1$ whereas $\Psi^O_g[20] \leftarrow 0.5$. Thus, the significance has a quick drop off with Ψ^O_q, but decreases only gradually with Ψ^O_g. Fig. 5(a) pictorially depicts the two significance vectors. Both intra and inter property name duplications depend on Ψ^O, with less significant properties attracting greater number of duplicates. The parameter $\varphi \in [\Psi^O[|\mathcal{N}|], 1]$ determines the amount of duplication of values within a property name. Let $\widehat{\Psi}^O[.]$ be defined by the line connecting $(1, \varphi\Psi^O[1])$ to $(|\mathcal{N}|, \Psi^O[|\mathcal{N}|])$. Then $\varrho_\varphi[.] \leftarrow 1 - \widehat{\Psi}^O[.]$. Fig. 5(b) plots ϱ_φ for the two significance vectors. The duplication of values across property names is set to $\varsigma = 0.005$.

Corresponding to Ψ^O_q and Ψ^O_g, we have two families of synthetic structured databases: $\mathcal{R}^s_q$ and $\mathcal{R}^s_g$. For the sparsity function, we use $\beta = 0.005$.

On the text side, the document length L is fixed to 1000 words. We study two families of documents, assuming every document predominantly discusses at most two objects ($M = 2$), though they may mention many other objects. In one, referred to as $\mathcal{D}^s_u$, both the objects are equally prominent ($\Lambda \in \{(.5, .5)\}$), while in the other, referred to as $\mathcal{D}^s_p$, their prominence is skewed ($\Lambda \in \{(.8, .2)\}$). We say that a document in $\mathcal{D}^s_u$ has two main objects, whereas a document in $\mathcal{D}^s_p$ has one main and one secondary object. Different documents will generally have different objects. As documents are typically dominated by words from the general vocabulary, we consider documents containing a small fraction of words from structured data and vary $\sigma \in \{.1, .15, .2\}$. We vary $\gamma \in \{.2, .35, .5, .65, .8\}$ to study the impact of variations in the amount of words from the prominent objects. Since documents seldom

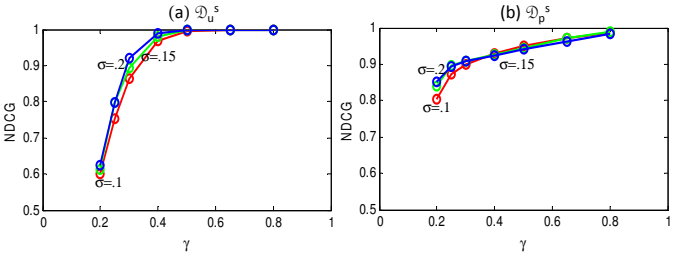


Fig. 6. Experiments with $\mathcal{R}_q^s$

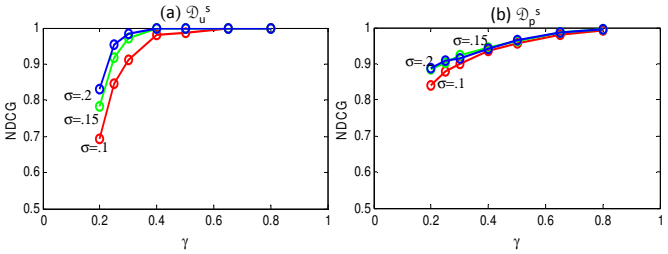


Fig. 7. Experiments with $\mathcal{R}_g^s$

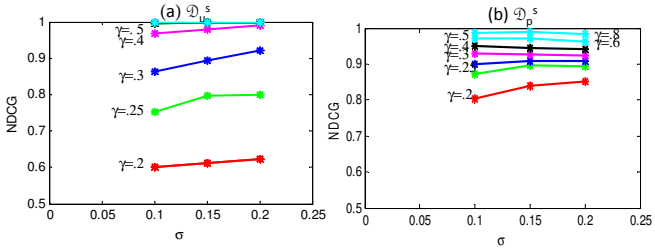


Fig. 8. Pivoted Fig. 6

mention property names, we set $\varepsilon_n \leftarrow 0$ and $\varepsilon_v \leftarrow \gamma$. We start by focusing on $\Psi^T \equiv \Psi^O$. Later, we also report results of choosing different forms of Ψ^T that differs from Ψ^O.

Experiments were run for synthetic database sizes of $10,000$ records and $100,000$ documents, but results were insensitive to these sizes (beyond reasonable minimums). Experiments were repeated sufficient number of times for results to have 95% confidence.

5.2.3. Performance. Using the above parameter settings, we carried out a large number of experiments, but will limit ourselves to discussing only the salient results. The results are presented in Fig. 6 and Fig. 7 for the $\mathcal{R}_q^s$ and $\mathcal{R}_g^s$ family of structured databases respectively, and where $\Psi^T \equiv \Psi^O$. The first (second) panel in both the figures contains results for the $\mathcal{D}_u^s$ ($\mathcal{D}_p^s$) family of documents. NDCG@2 values are shown on Y-axis against different values of γ on X-axis. In each graph, we have shown results for three different values of σ. In these figures, $\varphi = 1$, $\Psi_q^O[1] = 0.9$ and $\Psi_q^O[20] = 0.1$ for $\mathcal{R}_q^s$, and $\Psi_g^O[1] = 0.9$ and $\Psi_g^O[20] = 0.5$ for $\mathcal{R}_g^s$.

We can observe that γ has the most significant impact on the performance. Recall that while σ determines what fraction of a document's total words come from structured data, γ governs what fraction of these structured words come from the two prominent objects. As an example, suppose that we have a document of 1000 words where $\sigma = 0.1$ and $\gamma = 0.2$. Then, the document will have 100 words that appear in the structured repository. Of these,

there will be only 20 words (including repetitions) from the prominent objects and 80 from other objects. Fig. 6(a) corresponds to $\mathcal{D}_u^s$ documents in which the two objects are equally prominent. Thus, there might only be 10 non-unique words about one of them. Clearly, the document has little information about the target objects, but fairly large amount of extraneous words and information about other objects. Thus, it is quite possible that the document contains traits of other objects, but not the target ones. This imbalance of relevant versus irrelevant information negatively affects composition scoring. Naturally, NDCG is low in this setting. Looking at Fig. 6(b) corresponding to $\mathcal{D}_p^s$ documents, we see that NDCG has improved. Continuing our example, there are still only 20 words about the target objects, but now 16 of them could come from the main object. Thus, there is a greater chance that some trait of the main object might appear in the document, leading to better NDCG.

When γ is increased to 0.35, while holding σ at 0.1, we see in Fig. 6(a) that NDCG becomes higher as we now have more structured words coming from the prominent objects. For the documents in which γ can approach 0.4, NDCG starts approaching acceptable values even in this configuration with a low σ. To the right in Fig. 6(b), for low values of γ, we again see higher values for NDCG compared to Fig. 6(a) because the skewed distribution for Λ in $\mathcal{D}_p^s$ versus $\mathcal{D}_u^s$ means a greater chance of finding traits and scoring evidence for the main products.

Let us now focus on NDCG's sensitivity to σ in Fig. 6. Recall σ governs what fraction of words in a document come from structured data (irrespective of whether or not they are from the prominent objects). As expected, NDCG improves with increasing σ. However, a more interesting trend becomes apparent if we pivot the information in Fig. 6 as depicted in Fig. 8. We now have σ varying along X-axis and we have different plots for different values of γ. We see large gains in NDCG when γ increases from 0.2 to 0.3 for a given value of σ. But gains are relatively less pronounced when γ increases from 0.3 to 0.5. Similarly, gains from increasing γ are relatively lower for higher values of σ. This points to a notion of "enoughness" for the amount of structured information from target objects that needs to appear in documents in order to have accurate compositions. Certain minimum information is necessary for enabling traits to start showing up, but it does not help much to have more than enough. Enough information can easily appear in focused documents (higher quality documents!).

We find the next interesting trend by studying Fig. 7 for the $\mathcal{R}_g^s$ family of structured databases in relation to Fig. 6. We observe higher NDCG values in Fig. 7. When compared to $\mathcal{R}_q^s$, the relative significance of property names in $\mathcal{R}_g^s$ decreases more gradually. Consequently, traits tend to have larger cardinality. Longer traits can be harder to find, but when they do appear in a document, they serve as strong identifiers since large combination of values rarely occur by accident (*cf.* [Agrawal and Srikant 2003]). Thus, even for as low a value as 0.2 for γ, we have good NDCG. In addition, even for small values of γ (such as 0.2), NDCG increases with increasing fraction (σ) of structured data; longer traits require larger fraction of structured data. For small values of γ, having larger σ or $\mathcal{D}_p^s$ documents has similar effect.

5.2.4. Effect of varying Ψ^T . In the above experiments, we assumed that the importance vector Ψ^T used in TextGen is same as the significance vector Ψ^O used in *ObjGen*. Here, we are interested in studying the robustness of the techniques when Ψ^T differs from Ψ^O. The motivation is that the properties that are significant for distinguishing between two structured records might not be the ones perceived to be important by the document writer and vice versa. We report results for $\mathcal{R}_q^s$ structured database with $\Psi^O = \Psi_q^O$ and $|\mathcal{N}| = 20$ in the adversarial parameter setting of $\sigma = 0.1$ and $\gamma = 0.2$ for text generation (implying only 10% of the tokens in the document correspond to values in $\mathcal{R}_q^s$ and out of which only 20% correspond to the prominent objects). Note that the proposed techniques perform better with Ψ_g^O when compared to Ψ_q^O significance vector.

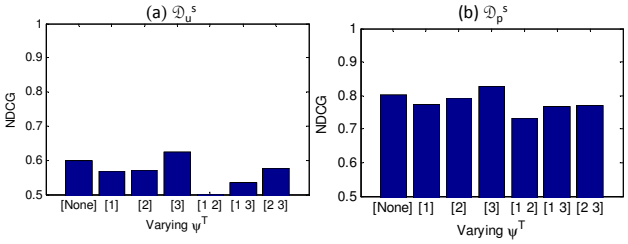


Fig. 9. Effect of variations of Ψ^T from Ψ^O

We initially kept the first three property names, which also serve as anchor property names, to have the same importance values in Ψ^T as the corresponding significance values for Ψ^O, but experimented with three variants on the remaining property names: (1) equal importance (at the mean value of significance vector for those property names), (2) increasing importance (inverse of corresponding values in Ψ^O), and (3) similar trends with corresponding property names in Ψ^O but with more gradual fall-off in importance. We found no noticeable difference in performance across these different variants. The implication is that as long as the document includes the properties that are relevant for identifying the objects, the performance is not affected by over or under inclusion of lesser significant properties.

We next studied more drastic variations in which some of the anchor properties were completely excluded from the text. Fig. 9 plots the results, separately for $\mathcal{D}_u^s$ and $\mathcal{D}_p^s$ family of documents. Each bar corresponds to NDCG performance for a particular instantiation of Ψ^T in which certain anchor property names have importance values of zero. The bar labeled '[None]' corresponds to text generation in which $\Psi^T = \Psi^O$. The bar labeled [1] corresponds to the case in which $\Psi^T[1] = 0$, *i.e.,* the corresponding document will have no mention of the most significant property for uniquely identifying the structured record of the object that the document is about. Similarly, the bar labeled '[1 2]' corresponds to the variant $\Psi^T[1] = 0$ and $\Psi^T[2] = 0$, *i.e.,* both first and second significant properties are absent in the corresponding document.

We observe that the results are robust even when the first property name is missing. The degradation is even less in the $\mathcal{D}_p^s$ family of documents. The implication is that as long as enough of trait appears in the document, the composition can be effected. The surprise was the increase in NDCG when the third property name was missing. The analysis revealed the third property name, due to its larger intra-duplication factor, was not serving as a good anchor property, thereby leading to spurious compositions. The removal of this property leads to higher NDCG.

We also varied σ and γ and observed similar trends. Of course, for lesser adversarial settings of σ and γ, the degradations were progressively even less noticeable.

5.3. Empirical Data Experiments

5.3.1. Data Sets. We present results from experiments using two structured databases from a commercial search engine. The first, referred to as $\mathcal{R}_h$, has data about household appliances (refrigerators, dishwashers, washing machines, cooking ranges, air conditioners, etc.). The second, referred to as $\mathcal{R}_e$, has data about electronic products (televisions, digital cameras, printers, camcorders, DVD players, etc.). $\mathcal{R}_h$ has 163,892 records and 284 distinct property names, whereas $\mathcal{R}_e$ has 111,731 records and 738 property names. For more statistics on these databases, see Figure 10.

The text corpus, henceforth $\mathcal{D}^w$, was created as follows. We first obtained upwards of a billion documents from the web index of the search engine, removing boilerplate content [Kohlschtter et al. 2010]. For a document to compose with one of the records, it must

	$\mathcal{R}_h$	$\mathcal{R}_e$
#records	163,892	111,731
#distinct property names (values)	284 (1,475,729)	738 (930,777)
#initial (effective) traits	130,326 (107,580)	854,694 (250,026)
#initial traits of cardinality 1/2/3	116,391/13,754/181	142,950/242,131/469,613
#effective traits of cardinality 1/2/3	98,245/9,171/164	96,532/56,371/97,123
#records having at least one initial (effective) trait	121,480 (102,090)	107,816 (89,976)
#distinct values in initial (effective) traits	105,786 (87,933)	165,926 (96,313)

Fig. 10. Statistics on $\mathcal{R}_h$, $\mathcal{R}_e$, and the traits computed from them.

necessarily contain at least one anchor value present in some record. To avoid wasted effort, therefore, we discarded those documents that did not contain even a single anchor value from $\mathcal{R}_h$ and $\mathcal{R}_e$. This resulted in upwards of 10 million documents, from which we sampled 200,000 documents. Since the filtering used just anchor values, not entire traits, $\mathcal{D}^w$ contains many documents that are not about any of the products present in the database but happened to have a token corresponding to one of the anchor values. For example, although there is no automobile in our database, $\mathcal{D}^w$ still has documents about Chevrolet S10 pickup truck because 'S10' is an anchor value for the Canon Powershot S10 digital camera that is present in $\mathcal{R}_e$. Clearly, $\mathcal{D}^w$ contains documents both about household appliances and electronic products. We did not cluster or classify them into these categories since our technique does not require such pre-processing.

5.3.2. Trait Computation. We computed traits of cardinality up to three; we did not consider traits of larger cardinality since they are unlikely to be found in arbitrary texts. We also set the maximum trait frequency (parameter ρ) to three, as our data set of products is provided to us by a couple of sources, and there is some overlap between them. In Figure 10, we show a number of statistics. Recall that initial traits refer to the traits before the trait filtering step and effective traits to the traits after removing those with invalid values (§4.4).

We note that although the databases contain a large number of distinct values, only a small fraction of them participate in forming traits. There is also not a large growth in number of traits as the trait cardinality increases. These effects are particularly pronounced in $\mathcal{R}_h$. The end result is that the trait computation is very fast. Yet, a majority of records end up having traits. Some 96% of the records in $\mathcal{R}_e$ get initial traits and 81% of them are assigned an effective trait, while these numbers are 74% and 62% respectively for $\mathcal{R}_h$. In both the databases, most of the records having no initial trait were partial records that had no anchor value. The lower trait coverage for $\mathcal{R}_h$ is due to the providers of that data set being of lower quality than for $\mathcal{R}_e$.

It is instructive to examine the values that were identified as invalid for forming effective traits. In $\mathcal{R}_e$, one of the initial traits computed for the product 'Panasonic c220' dvd player is {model=c220, productline=dvd}. However, by consulting the text corpus, the algorithm could determine that the token 'dvd' is very common and was able to filter out this error-prone trait. Similarly, an initial trait computed for the product 'InSinkErator Evolution Excel' garbage disposal in $\mathcal{R}_h$ was {brand=In-Sink-Erator, model=XL}. While 'XL' is indeed a valid model designation for the product and is distinguishing in the database, the value is far too ambiguous in the web corpus. Many other examples of ineffective traits in both the databases include cases where characteristic values are integer values (*e.g.,* Model=500), which tend to be ambiguous.

5.3.3. Evaluation. We composed $\mathcal{R}_h$ with the entire $\mathcal{D}^w$ and then $\mathcal{R}_e$ with the same $\mathcal{D}^w$. This results in 1,502 documents that are assigned some record in $\mathcal{R}_e$, and 66 documents that are assigned some record in $\mathcal{R}_h$.

Unfortunately, unlike synthetic data, we do not have ground truth needed for computing the accuracy of composition. We therefore had to resort to the rather expensive process

	$\mathcal{R}_h$	$\mathcal{R}_e$
NDCG@1	0.97	0.94
NDCG@2	0.95	0.88

Fig. 11. Composition performance on $\mathcal{R}_h$ and $\mathcal{R}_e$.

of manually labeling the results. The labeling was done as follows. For each document, we asked a human annotator to identify all the products that the document was about. Each such product was labeled as main or secondary. A product is considered to be main if the document is primarily about this product, and secondary if the product has been discussed in connection with the main topic. Products mentioned in passing were considered irrelevant. The annotator was then asked to identify which of these products were actually present in the structured database and their corresponding records.

We thus randomly sampled and labeled 62 documents that appeared in the result of composing $\mathcal{R}_e$ with $\mathcal{D}^w$, and all the results of composing $\mathcal{R}_h$ with $\mathcal{D}^w$. We refer to them as $\mathcal{D}_h^w$ and $\mathcal{D}_e^w$ respectively. These documents cover a broad spectrum of heterogenous content about products including manufacturer pages, merchant pages, reviews, and how-to articles.

5.3.4. Performance. Using the labels on $\mathcal{D}_h^w$ and $\mathcal{D}_e^w$, we computed NDCG for each, defining the perfect ordering to be the main product followed by the secondary products. Figure 11 tabulates the results. Thus, for the composition of $\mathcal{R}_h$ with $\mathcal{D}^w$, we obtained an NDCG@1 of 0.97 and NDCG@2 of 0.95. For the second composition, *i.e.,* composing $\mathcal{R}_e$ with $\mathcal{D}^w$, we obtained an NDCG@1 of 0.94 and NDCG@2 of 0.88. Given the high score for NDCG, we can say that our approach is doing a good job of composition.

Upon in-depth examination of the results, we found that the approach worked well across a wide variety of documents, successfully identifying both main and secondary objects in product reviews[1], merchant pages[2], manufacturer pages[3], articles[4] and forums[5]. In fact, we were pleasantly surprised by many of the results. For example, one particular merchant page[6] describes a guide book titled 'Nikon D60 Guide to Digital SLR Photography', which was matched to a Nikon D60 digital camera. The diversity of traits is also well-leveraged, as we have many instances of composition results that relied entirely on less obvious traits, such as {model=DC5000, memory=2GB, harddisk=80GB} for correctly joining a classified page[7] to a Hewlett-Packard PC, and {model=s300, color-print-speed=7.5ppm} for joining a resources page[8] to a Canon Printer. In the case of former, we were able to successfully identify the object on the page, even though the database and web page had syntactically different representations of the Brand value ('Hewlett-Packard' versus 'HP'). In the case of latter, the database record was missing the Brand value entirely. These examples highlight the flexibility and generality of the approach versus a domain-specific approach that relies on matching fixed keys such as Brand and Model.

[1]http://www.digitalcamerainfo.com/content/Fujifilm-FinePix-F470-Digital-Camera-Review-.htm (matched to a Fuji F470 camera)
[2]http://www.overstock.com/Electronics/Polaroid-i1037-Blue-2.7-inch-10MP-Digital-Camera-Refurbished/4275862/product.html (matched to a Polaroid i1037 camera)
[3]http://www1.lexmark.com/US/en/catalog/product.jsp?prodId=3220 (matched to a Lexmark E120N printer)
[4]http://www.slashgear.com/pentax-a40-shoots-divx-video-at-640-x-480-037753/ (matched to Pentax A40 and Pentax S7 cameras)
[5]http://answers.yahoo.com/question/index?qid=20101122173550AANj4Vq (matched to Canon 500D/T1i, Nikon D3100 and Canon XSi cameras)
[6]http://search.barnesandnoble.com/David-Buschs-Nikon-D60-Guide-to-Digital-SLR-Photography/David-D-Busch/e/9781598635775/
[7]http://cgi.ebay.com/HP-DC5000-DEsktop-P4-3-2Ghz-2GB-Mem-80GB-HDD-XP-PRO-/220730984330
[8]http://www.downloadatoz.com/driver/articles/introduction-of-canon-s300-color-bubble-jet-printer-and-driver-download.html

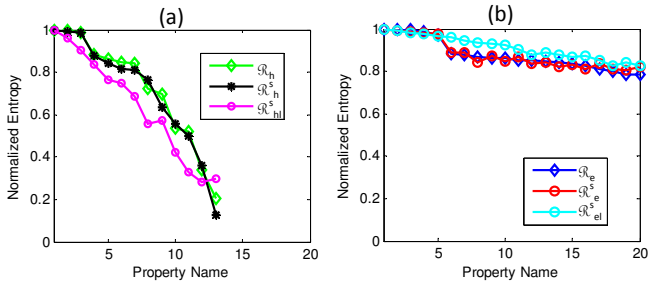


Fig. 12. Synthetic and empirical data agreement

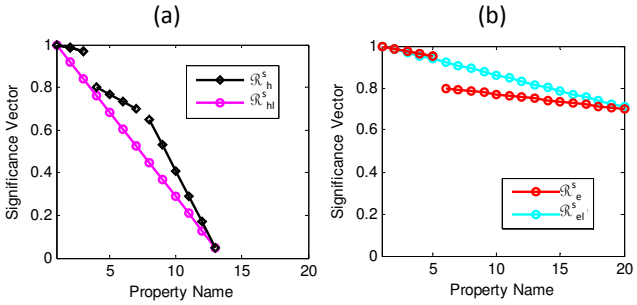


Fig. 13. Significance Vectors (Ψ^O)

We also drilled down on the cases where the results were judged to be erroneous. Mostly, the root cause was the incompleteness of our structured databases. For example, there is a web page about the Samsung S5200 cell phone[9] that is joined with the digital camera 'Fujifilm FinePix S5200'. The cell phone is not in $\mathcal{R}_e$; however, both the cell phone and the digital camera share the same anchor value (S5200), and the page contains terms that appear in the traits of the digital camera (e.g., 'electronic' for the property name 'Viewfinder type'). Some additional NDCG loss was due to correct matches that were nevertheless deemed irrelevant and therefore treated as errors. While these matched products had legitimate mentions in the page, their presence was purely in passing and not contributing to the main topic.

5.4. Validation Experiments

One can readily see that it is quite cumbersome and expensive to conduct experiments using empirical data because of non-availability of ground truth. Yet another difficulty is that a different empirical dataset might not provide useful additional insights if the characteristics that determine the performance of the composition are not significantly different in the two datasets. Synthetic data allowed us to explore a wide range of operating regions. We tried to incorporate many realistic aspects to datasets found in practice in our synthetic data generators, yet one should legitimately question whether they are indeed generating realistic datasets.

This section reports on experiments we performed to validate our synthetic data generators. We also study if we could use the synthetic data generator to estimate the expected performance of the composition, utilizing the measured values of the text generation parameters from the target text repository.

[9]http://www.mobileheart.com/cell-phone-softwares/1771-Samsung-S5200-Software.aspx

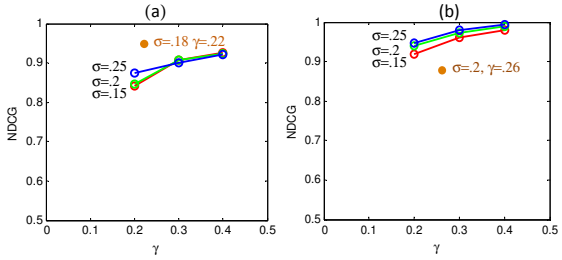


Fig. 14. Composition of (a)$\mathcal{R}_h^s$ with $\mathcal{D}_h^s$, (b)$\mathcal{R}_e^s$ with $\mathcal{D}_e^s$. The solid circle corresponds to the composition of (a)$\mathcal{R}_h$ with $\mathcal{D}_h^w$,(b)$\mathcal{R}_e$ with $\mathcal{D}_e^w$.

5.4.1. StructGen Validation. We investigated if StructGen could be used to generate synthetic databases to mimic the two empirical databases, $\mathcal{R}_h$ and $\mathcal{R}_e$, discussed in §5.3.1. Our analysis of $\mathcal{R}_h$ revealed that its 14 attributes were clustered in two groups. The first group, consisting of three attributes, had the most significant attributes that could differentiate between the records in $\mathcal{R}_h$. There was a sharp drop off in the differentiation capability of other attributes. Fig. 12(a) shows the plot of normalized entropy of different attributes in $\mathcal{R}_h$ that corroborates this observation. We therefore used a piece-wise linear significance vector Ψ_h^O, consisting of two pieces shown in Fig. 13(a), to generate the synthetic database $\mathcal{R}_h^s$ for mimicking $\mathcal{R}_h$. The intra property name duplication was set to: $\varrho[.] \leftarrow 1 - \Psi^O[.]$, and inter property name duplication to: $\varsigma[.] \leftarrow 0$. We have plotted in Fig. 12(a) the normalized entropy of different attributes in $\mathcal{R}_h^s$. We see a close match. In fact, we further approximated Ψ_h^O by the plain linear vector Ψ_{hl}^O, also shown in Fig. 13(a). The normalized entropies of the corresponding synthetic database $\mathcal{R}_{hl}^s$ is also plotted in Fig. 12(a). We continue to see reasonably good match with the entropies for $\mathcal{R}_h$.

We repeated this exercise with $\mathcal{R}_e$. As Fig. 12(b) shows, $\mathcal{R}_e$ consists of three groups of attributes in terms of their significance in differentiating various records. We therefore used a piece-wise linear vector Ψ_e^O, consisting of three pieces shown in Fig. 13(b), for generating synthetic $\mathcal{R}_e^s$. Again, $\varrho[.] \leftarrow 1 - \Psi^O[.]$, and $\varsigma[.] \leftarrow 0$. We see in Fig. 13(b) a close match between the entropies of $\mathcal{R}_e^s$ and $\mathcal{R}_e$. Entropies of synthetic and empirical databases continue to reasonably match even when a plain linear vector Ψ_{el}^O is used to generate synthetic $\mathcal{R}_{el}^s$.

5.4.2. TextGen Validation. The approach we took was to make use of the document set $\mathcal{D}_h^w$ ($\mathcal{D}_e^w$) and compute from it the manifest values of the TextGen parameters, pretending that $\mathcal{D}_h^w$ ($\mathcal{D}_e^w$) was produced by TextGen. We thus estimated $\langle \sigma = .18, \gamma = .22 \rangle$ ($\langle \sigma = .20, \gamma = .26 \rangle$) from $\mathcal{D}_h^w$ ($\mathcal{D}_e^w$). We then instantiated TextGen with parameter values bracketed around those computed from $\mathcal{D}_h^w$ ($\mathcal{D}_e^w$). Every such instantiation yields a set of synthetic text documents $\mathcal{D}_h^s$ ($\mathcal{D}_e^s$). We next compose $\mathcal{D}_h^s$ ($\mathcal{D}_e^s$) with synthetic database $\mathcal{R}_h^s$ ($\mathcal{R}_e^s$) described in §5.4.1, and compute NDCG. We then compare these NDCG values to those obtained from composing empirical data, *i.e.,* composing $\mathcal{D}_h^w$ ($\mathcal{D}_e^w$) with $\mathcal{R}_h$ ($\mathcal{R}_e$), and examine if the results are similar or wide apart. We show the results in Fig. 14(a) (Fig. 14(b)).

We can see that synthetically generated data with similar parameters performs comparably with the empirical data. This study indicates that the text generator can act as the proxy for documents found in practice. It also suggests that our synthetic data generators initialized with parameter values from empirical data can be used to simulate the behavior of the system in empirical setting.

6. SUMMARY

We studied the problem of composing structured and text databases about objects in a general setting, making minimal assumptions. Structured databases contain partial and

possibly multiple representations of objects consisting of property name and value pairs. Texts are simply sequences of words devoid of any markings or structure that explicitly identifies the objects that the document is about. We postulated the notion of a trait, a set of characteristics that can serve as the proxy for the identity of an object, and presented techniques for algorithmically computing traits, mapping structured records and text documents to traits, and using traits for joining information about the same object from two repositories.

To understand the performance characteristics of the proposed techniques, we defined a realistic model of an object and how structured records are created from objects. We also defined a model of text generation that captures many intricacies of text documents found in practice. Using empirical structured databases in active use, we verified that we can indeed generate synthetic databases having salient characteristics of their empirical counterparts. These synthetic generators could be of independent interest to researchers wanting to base their research on non-proprietary data.

With the synthetic data generators on hand, we carried out extensive experiments to delineate the performance regions for the proposed techniques under different operating parameters. The take home lesson from these experiments is that the proposed techniques can yield accurate compositions provided the objects have good traits and the text documents satisfy the enoughness property. The former is true for objects that have at least a few important attributes, whence the traits comprising of combinations of them become quite differentiating. The enoughness property holds for focused documents as they usually contain enough important words about the objects that drag along the traits of the corresponding objects with them. Our experiments with the empirical data validated the findings of the synthetic experiments. As a byproduct of this validation exercise, we also confirmed the sanctity of our synthetic data generators. It also showed that our synthetic data generators initialized with parameter values from empirical data can be used to simulate the behavior of the system in an empirical setting.

This paper assumed that the composition is computed off-line and materialized for use by different applications. Techniques for identifying when the composition should be recomputed and incremental algorithms for updating materialized compositions are interesting topics for future research. Similarly, parallel implementations and fast indexing structures and algorithms for on-line computation of the composition are fruitful directions for future work.

Acknowledgments. Many members of the Search Lab, including Sreenivas Gollapudi, Samuel Ieong, Krishnaram Kenthapadi, Nina Mishra, Alex Ntoulas, Stelios Paparizos, Livia Polanyi, and Panayiotis Tsaparas generously contributed their time and thoughts during various phases of this work. We particularly wish to thank Stelios Paparizos for customizing his annotator code and Sreenivas Gollapudi for providing the indexing code. Partha Talukdar played a key role in the initial phases of the work. He built an early prototype and carried out experiments that had formative influence. Ekaterina Gonina built an early prototype of a parallel implementation and graciously helped with exploratory investigations.

REFERENCES

Agrawal, R., Gollapudi, S., Kannan, A., and Kenthapadi, K. 2011. Data mining for improving textbooks. *SIGKDD Explorations 13,* 2, 7–19.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. 1996. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI/MIT Press, 307–328.

Agrawal, R. and Srikant, R. 2003. Searching with numbers. *IEEE Transactions on Knowledge and Data Engineering 15*, 855–870.

Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., and Widom, J. 2009. Swoosh: a generic approach to entity resolution. *VLDB Journal 18,* 1, 255–276.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. 2003. Adaptive name matching in information integration. *IEEE Intelligent Systems 18,* 5, 16–23.

Bizer, C., Heath, T., and Berners-Lee, T. 2009. Linked data – the story so far. *International Journal on Semantic Web and Information Systems 5,* 3.

Boros, E., Gurvich, V., Khachiyan, L., and Makino, K. 2003. On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence 39*, 211–221.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Castle, J. L., Fawcett, N. W., and Hendry, D. F. 2009. Nowcasting is not just contemporaneous forecasting. *National Institute Economic Review 210,* 1.

Chakaravarthy, V., Gupta, H., Roy, P., and Mohania, M. 2006. Efficiently linking text documents with relevant structured information. In *VLDB*. 667–678.

Cheng, T., Lauw, H., and Paparizos, S. 2010. Fuzzy matching of web queries to structured data. In *ICDE*.

Cohen, W. 1998. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD*. 202–212.

Csomai, A. and Mihalcea, R. 2007. Linking educational materials to encyclopedic knowledge. In *AIED*.

Dalvi, N., Kumar, R., Pang, B., and Tomkins, A. 2009a. Matching reviews to objects using a language model. In *EMNLP*. 609–618.

Dalvi, N., Kumar, R., Pang, B., and Tomkins, A. 2009b. A translation model for matching reviews to objects. In *CIKM*. 167–176.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*.

DeWitt, D. J. 1993. The wisconsin benchmark: Past, present, and future. In *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*, J. Gray, Ed. Morgan Kaufmann.

Doan, A., Gravano, L., Ramakrishnan, R., and Vaithyanathan, S. 2008. Special issue on information extraction. *SIGMOD Record 37,* 4.

Doan, A. and Halevy, A. Y. 2005. Semantic integration research in the database community: A brief survey. *AI Magazine 6*.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. 2007. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng. 19,* 1, 1–16.

Fellegi, I. P. and Sunter, A. B. 1969. A theory for record linkage. *Journal of the American Statistical Association 64,* 328, 1183–1210.

Gulli, A. and Signorini, A. 2005. The indexable web is more than 11.5 billion pages. In *WWW*.

Haglin, D. J. and Manning, A. M. 2007. On minimal infrequent itemset mining. In *DMIN*. 141–147.

Halevy, A., Rajaraman, A., and Ordille, J. 2006. Data integration: the teenage years. In *VLDB*. 9–16.

Hernández, M. A. and Stolfo, S. J. 1995. The merge/purge problem for large databases. In *SIGMOD*.

Hilbert, M. and Lopez, P. 2011. The world's technological capacity to store, communicate, and compute information. *Science 10*.

Huang, J., Gao, J., Miao, J., Li, X., Wang, K., Behr, F., and Giles, C. L. 2010. Exploring web scale language models for search query processing. In *WWW*.

Huston, S. and Croft, W. B. 2010. Evaluating verbose query processing techniques. In *SIGIR*.

Kannan, A., Givoni, I., Agrawal, R., and Fuxman, A. 2011. Matching unstructured offers to structured product descriptions. In *KDD*.

Kohlschtter, C., Fankhauser, P., and Nejdl, W. 2010. Boilerplate detection using shallow text features. *WSDM*.

Kumaran, G. and Carvalho, V. R. 2009. Reducing long queries using query quality predictors. In *SIGIR*.

Mannila, H. 1997. Methods and problems in data mining. In *ICDT*.

Manning, C. D., Raghavan, P., and Schütze, H. 2008. *Introduction to Information Retrieval*. Cambridge Univ. Press.

Matuszek, C., Cabral, J., Witbrock, M., and DeOliveira, J. 2006. An introduction to the syntax and content of cyc. In *AAAI Spring Symposium*.

Michelson, M. and Knoblock, C. 2008. Creating relational data from unstructured and ungrammatical data sources. *Journal of Artificial Intelligence Research 31*, 543–590.

Mihalcea, R. and Csomai, A. 2007. Wikify! linking documents to encyclopedic knowledge. In *CIKM*.

Milne, D. N. and Witten, I. H. 2008. Learning to link with wikipedia. In *CIKM*.

Newcombe, H. B., Kennedy, M. J., Axford, S. J., and James, A. P. 1959. Automatic linkage of vital records. *Science 130*, 954–959.

Paranjpe, D. 2009. Learning document aboutness from implicit user feedback and document structure. In *CIKM*.

Sarawagi, S. 2008. Information extraction. *Foundations and Trends in Databases 1,* 3, 261–377.

Sarkas, N., Paparizos, S., and Tsaparas, P. 2010. Structured annotations of web queries. In *SIGMOD*.

Suchanek, F., Kasneci, G., and Weikum, G. 2007. Yago : A core of semantic knowledge. In *WWW*. 697–706.

Winkler, W. E. 2006. Overview of record linkage and current research directions. Tech. rep., Bureau of the Census.

Yu, J. X., Qin, L., and Chang, L. 2010. *Keyword Search in Databases*. Morgan and Claypool Publishers.

Zhu, J., Nie, Z., Liu, X., Zhang, B., and Wen, J.-R. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*.