Cost-Effective Conceptual Design for Information Extraction

ARASH TERMEHCHY, Oregon State University
ALI VAKILIAN, MIT
YODSAWALAI CHODPATHUMWAN and MARIANNE WINSLETT, University of Illinois at Urbana-Champaign

It is well established that extracting and annotating occurrences of entities in a collection of unstructured text documents with their concepts improves the effectiveness of answering queries over the collection. However, it is very resource intensive to create and maintain large annotated collections. Since the available resources of an enterprise are limited and/or its users may have urgent information needs, it may have to select only a subset of relevant concepts for extraction and annotation. We call this subset a *conceptual design* for the annotated collection. In this article, we introduce and formally define the problem of *cost-effective conceptual design* where, given a collection, a set of relevant concepts, and a fixed budget, one likes to find a conceptual design that most improves the effectiveness of answering queries over the collection. We provide efficient algorithms for special cases of the problem and prove it is generally NP-hard in the number of relevant concepts. We propose three efficient approximations to solve the problem: a greedy algorithm, an *approximate popularity maximization* (APM for short), and *approximate annotation-benefit maximization* (AAM for short). We show that, if there are no constraints regrading the overlap of concepts, APM is a fully polynomial time approximation scheme. We also prove that if the relevant concepts are mutually exclusive, the greedy algorithm delivers a constant approximation ratio if the concepts are equally costly, APM has a constant approximation ratio, and AAM is a fully polynomial-time approximation scheme. Our empirical results using a Wikipedia collection and a search engine query log validate the proposed formalization of the problem and show that APM and AAM efficiently compute conceptual designs. They also indicate that, in general, APM delivers the optimal conceptual designs if the relevant concepts are not mutually exclusive. Also, if the relevant concepts are mutually exclusive, the conceptual designs delivered by AAM improve the effectiveness of answering queries over the collection more than the solutions provided by APM.

Categories and Subject Descriptors: H.2.1 [**Logical Design**]: Schema and subschema

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Conceptual design, information extraction, effective query answering

ACM Reference Format:
Arash Termehchy, Ali Vakilian, Yodsawalai Chodpathumwan, and Marianne Winslett. 2015. Cost-effective conceptual design for information extraction. ACM Trans. Datab. Syst. 40, 2, Article 12 (June 2015), 39 pages.
DOI: http://dx.doi.org/10.1145/2716321

1. INTRODUCTION

Discovering structured data from large unstructured or semi-structured document collections is an active research area in data management [Chiticariu et al. 2010;

The authors are supported by NFS grants CCF-0938071, CCF-0938064, CNS-0716532, IIS-1423238, and IIS-1421247. A. Termehchy is also supported by a Yahoo! Key Scientific Challenges award.
Authors' addresses: A. Termehchy (corresponding author), Oregon State University, OR; email: termehca@oregonstate.edu; A. Vakilian, MIT, Cambridge, MA; Y. Chodpathumwan, M. Winslett, University of Illinois at Urbana-Champaign, IL.

DOI: http://dx.doi.org/10.1145/2716321

```
<article>
 ... Michael Jeffrey Jordan is a former American professional
 basketball player ...
</article>
<article>
 ... Michael Jordan is a full professor at the
 University of California, Berkeley ...
</article>
<article>
  ... All six championship teams of Chicago Bulls were led by
  Michael Jordan and Scottie Pippen ...
</article>
```

Fig. 1. Wikipedia article excerpts.

```
<article>
  ... <athlete> Michael Jeffrey Jordan</athlete> is a former
 <nationality> American </nationality> professional
 basketball player ...
</article>
<article>
  ... <scientist> Michael I. Jordan </scientist> is a full
 professor at the
 <organization>
  University of California, Berkeley
 </organization> ...
</article>
<article>
  ... All six championship teams of
 <organization> Chicago Bulls </organization> were led by
 <athlete> Michael Jordan </athlete> and
 <athlete> Scottie Pippen </athlete> ...
</article>
```

Fig. 2. Semantically annotated Wikipedia article excerpts.

Doan et al. 2009; Sarawagi 2008; Dalvi et al. 2009]. A popular method of discovering structured data from unstructured or semi-structured documents is *semantic annotation*: extracting the mentions of named entities in a collection of unstructured or semi-structured text documents and annotating these mentions by their concepts [Chakrabarti et al. 2007, 2010; Chu-Carroll et al. 2006; Dill et al. 2003; Schenkel et al. 2007; Finin et al. 2010; van Zwol and Loosbroek 2007; Egozi et al. 2011]. It is well established that semantically annotating a collection improves the effectiveness of answering queries over the collection [Chu-Carroll et al. 2006; Dalvi et al. 2009; Chakrabarti et al. 2007]. Figure 2 depicts excerpts of semantically annotated Wikipedia articles (*www.wikipedia.org*) whose original and unannotated versions are shown in Figure 1. Since the mentions to the entities named *Michael Jordan* are disambiguated by their concepts, a query interface can deliver more effective results for the queries about *Michael Jordan*, the *scientist*, over the semantically annotated collection than the unannotated one.

We call the set of all concepts that have at least one entity in the collection a *conceptual domain* (*domain* for short). A possible domain for the articles shown in Figure 1 is the set of concepts {*athlete*, *scientist*, *position*, *organization*, *sport*, *nationality*}.

Intuitively, annotating all concepts in a domain will provide more effective results for the input queries. Recent studies, however, indicate that accurately annotating the entities of a concept in a large collection requires developing, deploying, and maintaining complex pieces of software, manual labor, and/or collecting training data, which may take a long time and substantial amount of computational and financial resources [Chiticariu et al. 2010; Anderson et al. 2013; Elhelw et al. 2012; Doan et al. 2009; Finin et al. 2010]. Given a concept, developers have to design and write a program called an *annotator* or *extractor* that finds and annotates all instances of the concept in the collection. One may write hand-tuned programming rules, such as regular expressions, that leverage formatting or language patterns in the text to identify and extract instances of some concepts, such as *Email* [Chiticariu et al. 2010; McCallum 2005]. However, formatting and language patterns for most concepts, such as *person* or *protein*, are subtle and involve many exceptions. In these cases, it is not unusual to have thousands of rules to extract concepts [McCallum 2005]. Hence, developing and maintaining annotating programs becomes extremely time consuming [Chiticariu et al. 2010].

One may also use machine learning methods for concept extraction [McCallum 2005; Chiticariu et al. 2010; Anderson et al. 2013]. Nevertheless, studies of several recent concept extraction systems, such as DeepDive (*deepdive.stanford.edu*), show that using and deploying machine learning methods for concept extraction are also very time consuming and labor intensive [Anderson et al. 2013; Chiticariu et al. 2010]. In order to extract the instances of a concept, developers have to first inspect the dataset and identify a set of clues, called *features*, which indicate whether some terms in a document refer to an instance of the concept. For instance, the occurrence of word *said* in a sentence may suggest that the subject of the sentence refers to an instance of concept *person*. Then, developers have to write programs to extract these features from the documents. Each concept extractor may use hundreds of features [Anderson et al. 2013; McCallum 2005]. These efforts are more costly for concepts that are defined in specific domains, such as geology and medicine, as they require extensive collaborations between developers and scarce domain experts. As communication between developers and domain experts is often ineffective, developers may have to spend a great deal of time and sift through the data to find the relevant features [Anderson et al. 2013].

After finding candidate features, developers have to perform *feature selection*: they have to analyze available features, remove some of them (e.g., those that are highly correlated), and select a subset of those features that predict the instances of the concept accurately [Anderson et al. 2013; McCallum 2005]. Developers iterate the steps of finding, extracting, and revising features and testing the annotation program several times in order to create an annotation program with a reasonable accuracy [Anderson et al. 2013; Doan et al. 2009].

Finally, since annotation modules need to perform complex text analysis, it may take days or weeks, plus a great deal of computational resources, to execute them over a large collection [Jain et al. 2008a; Shen et al. 2008; Agichtein and Gravano 2003]. Thus, users have to wait a long time for the execution of extraction programs before they have a fully annotated collection. The long delays to execute extraction programs and to create and/or update fully annotated collections are well recognized as an issue in concept extraction [Shen et al. 2008; Gulhane et al. 2011; Elhelw et al. 2012; Doan et al. 2009]. They are particularly problematic in domains with *urgent* information needs [Jain et al. 2008a; Shen et al. 2008; Elhelw et al. 2012]. For example, an FBI agent who needs to query the evolving content of Web sites and social media pages to find and respond to new activities in human trafficking, a stock analyst who has to respond to changes in the stock market in a timely fashion, and an epidemiologist who

must act quickly to control the spread of an infectious disease cannot afford to wait for all annotation programs to be (re-)executed [Shen et al. 2008].

Since the structure and content of documents in many domains evolve over time, the annotation programs should be regularly rewritten and rerun to create an updated annotated collection [Gulhane et al. 2011; Elhelw et al. 2012; Kowalkiewicz et al. 2006; Chen et al. 2012]. A recent study from Yahoo! Research indicates that the average lifetime of most extractors is about two months [Gulhane et al. 2011]. Hence, users have to wait a long time for annotation programs to be rewritten and rerun in order to pose their queries over an updated and fully annotated collection [Shen et al. 2008; Elhelw et al. 2012; Chen et al. 2012].

Therefore, an enterprise may decide to select only a subset of the concepts in the domain for annotation or re-annotation to provide a partially annotated collection relatively quickly. Users can pose their queries over the partially annotated collection and get reasonably effective answers. Moreover, since the available financial and computational resources of most enterprises are limited, they may not be able to hire a sufficient number of machine learning experts and acquire computational resources to (re-)write and (re-)execute the annotation programs for all concepts in their domains and select subsets of these domains for annotation. We call this subset of concepts a *conceptual design* (*design* for short) for the annotated collection.

Clearly, an enterprise wants to find a design whose required time (or resources) for annotation does not exceed its limit on turnaround time (or budget) and that most improves the effectiveness of answering queries over the annotated collection. Each concept may require different amounts of time and resources for annotating its entities in a collection. For instance, an enterprise may use a freely available and relatively simple annotation program from OpenNLP (*opennlp.apache.org*) to discover the entities of concept *Email*, to purchase and deploy a more sophisticated annotation program from companies (such as *ontotext.com*) to annotated instances of concept *position*, or to develop and deploy in-house annotators to identify entities of more domain-specific concepts, such as *athlete*. The latter annotators may require more financial resources and/or time to develop and execute than the former. This scenario suggests a novel conceptual design problem: given a domain and a document collection, we want to find a design for the collection that most improves the overall effectiveness of answers to input queries, while its annotation costs do not exceed a given budget.

Although building and maintaining annotation modules are among the most expensive stages of managing an annotated collection, to the best of our knowledge, the choice of a cost-effective design for a collection is generally guided only by intuition and has not been studied before. One cannot address this problem by conceptual or logical design guidelines in classic database literature, as they neither consider the cost of creating or maintaining a concept nor the impact of having a concept in the design on the degree of effectiveness of answering queries [GarciaMolina et al. 2008]. In this article, we introduce and formalize the problem of *cost-effective conceptual design* for semantic annotation. Our formal treatment paves the way for systematic analysis of the problem and shows that intuitively appealing heuristics, such as choosing relatively less costly concepts and/or those that appear most often in queries, are not generally optimal, even for cases where all annotators have equal cost. We prove the problem to be generally NP-hard in the number of concepts in the domain and provide efficient algorithms with provably bounded or sufficiently small worst-case approximation ratios to solve the problem. Our extensive experiments using a large-scale real-world document collection, queries from a search engine query log, and real-world conceptual domains show that our algorithms efficiently select designs that provide effectiveness close to that of the optimal design for queries. In summary, we make the following contributions.

—We formally analyze the impact of possibly inaccurate annotation of a concept in a collection on the effectiveness of answering queries over the collection. We quantify this impact using a function called *annotation benefit* for two categories of real-world domains: those with mutually exclusive concepts and those that do not have any constraints regrading the overlap of concepts.
—We introduce and formally define the problem of cost-effective conceptual design for semantic annotation as maximizing the value of the annotation-benefit function over a set of concepts in a domain, given a limited time or budget. We propose efficient exact algorithms for some interesting special cases of the problem. We prove that the problem over both categories of domains is generally NP-hard in the number of concepts in the domain.
—We propose three efficient approximation algorithms for the problem: a greedy algorithm, *an approximate popularity maximization* (APM for short) algorithm, and an *approximate annotation-benefit maximization* (AAM for short) algorithm. We prove that the designs returned by the greedy algorithm improve the effectiveness of answering queries almost as much as the optimal design if the concepts are mutually exclusive and have equal cost. We also prove that the designs returned by APM improve the effectiveness of answering queries by at most a constant factor less than the optimal design and that the AAM algorithm is a fully polynomial-time approximation scheme in mutually exclusive domains: the effectiveness improvement achieved by its designs will get sufficiently close to the improvement achieved by optimal designs, given sufficient running time. We also show that APM is a fully polynomial-time approximation scheme for those domains without any constraint regarding the overlap of concepts.
—Our extensive experiments over the collection of Wikipedia articles, concepts from the YAGO ontology [Schenkel et al. 2007], and queries from the MSN query log [Demidova et al. 2010] show that the annotation-benefit formula accurately quantifies the impact of a design on the amount of improvement in the effectiveness of answering queries over the annotated collection for both categories of domains.
—Our empirical results indicate that APM finds optimal designs for most cases where the domain does not have any constraint regrading the overlap of its concepts. They also show that the designs delivered by the AAM algorithm improve the effectiveness of answering queries more than the APM algorithm across domains with mutually exclusive concepts. We evaluate the scalability of APM and AAM and show that both algorithms can find designs that improve the effectiveness of answering queries in a reasonable amount of time and with modest memory overheads for a preprocessing task.
—Because the complete information about the values of input parameters for AAM may not be available at design time, we explore the sensitivity of this algorithm to errors in estimating its input parameters and show that, when using the input parameters computed over a small sample of the collection, AAM still returns designs that are generally more effective than those returned by APM over domains with mutually exclusive concepts.

This article is organized as follows. Section 2 reviews the related work. Section 3 describes the basic definitions. Section 4 quantifies the impact of a design on the improvement in effectiveness of answering queries over a collection annotated by the design. Section 5 introduces the problem of cost-effective conceptual design and explores its hardness. Section 6 proposes efficient approximation algorithms for the problem and explores their worst-case approximation ratios. Section 7 contains the empirical results

about the accuracy of the annotation-benefit function and the average approximation ratios of the algorithms, and Section 8 concludes the article.

2. RELATED WORK

Conceptual design is a topic of research in data management [GarciaMolina et al. 2008]. Our work extends this line of research by introducing and exploring the ability of a conceptual design in effectively answering input queries and its cost effectiveness.

There is a large body of work on building programs that extract entities that belong to a given concept, and systems that manage the extracted data [Chu-Carroll et al. 2006; Dalvi et al. 2009; Chiticariu et al. 2010; Chakrabarti et al. 2007, 2010; Dill et al. 2003; Sarawagi 2008; Doan et al. 2009; Dalvi et al. 2009; Anderson et al. 2013; Chen et al. 2012; Graupmann et al. 2004]. We build on this work by offering a new preprocessing design phase that can be followed by and coupled with any of these previously proposed approaches.

As developing concept extraction programs is very time consuming, researchers have proposed frameworks for iterative development of these programs so that developers can check the accuracy of a concept extraction program in each iteration of its development and stop if the program delivers the desired level of accuracy [Shen et al. 2008; Anderson et al. 2013]. Our work is orthogonal to these efforts and focuses on selecting those concepts for which one will (re-)write an extraction program.

In order to address the long delays in concept and relation extractions, researchers have proposed several techniques to optimize the execution time of SQL queries over existing databases whose information comes from concept and relation extraction programs [Ipeirotis et al. 2006; Jain et al. 2008a, 2008b; Huang and Yu 2010; Agichtein and Gravano 2003; Doan et al. 2009; Shen et al. 2007; Elhelw et al. 2012]. Similar systems optimize the use of information extraction programs to add missing data values to an existing database [Kanani and McCallum 2012]. These techniques generally improve execution time or storage capacity by processing only those "promising" documents in the collection that contain information about the database relations, instead of the whole collection. Our work differs in addressing the issues raised at design time rather than query time. We also consider ranking queries as opposed to SQL queries. Our model covers other types of cost in annotation in addition to runtime and storage space. Moreover, we explore using both structured data (i.e., annotated documents) and unstructured data (i.e., unannotated documents) in effectively answering queries.

Researchers have proposed methods to discover the schema of the noisy output of information extraction modules [Cafarella et al. 2007]. Our work, however, finds the set of concepts that are worth extracting and annotating, given their costs of annotation.

Researchers have empirically shown that annotating all concepts in a domain using sufficiently precise annotators improves the effectiveness of answering queries [Chu-Carroll and Prager 2007; Sanderson 2008; Krovetz and Croft 1992; Stokoe et al. 2003]. We extend this line of work by considering the cost of building concept annotators, providing a formal analysis of the impact of concept annotation on the improvement in effectiveness of answering queries, and proposing systematic methods to select concepts for annotation given a fixed budget.

We have described and studied the problem of cost-effective conceptual design in Termehchy et al. [2014]. The current article improves this work in seven directions. First, it provides the formal proofs and further analysis of the results in Termehchy et al. [2014]. Second, it studies the hardness of the cost-effective conceptual design problem in more detail and proves its connection to the well-known SUBSET SUM problem. Third, it proposes exact and efficient (polynomial-time) algorithms for the problem in some interesting special cases. Fourth, it proposes a greedy approximation algorithm for some special cases of the problem that are more efficient than the algorithms

proposed in Termehchy et al. [2014]. Fifth, we evaluate and analyze the effectiveness of our proposed models and algorithms using an additional effectiveness metric (i.e., mean reciprocal rank) and provide more insight on the effectiveness of the investigated models and algorithms. Sixth, we provide a detailed empirical study on the trade-off between the running times and the output qualities of the proposed algorithms. Finally, we analyze the parameter estimation techniques for cost-effective conceptual design algorithms over a large collection.

3. BASIC DEFINITIONS

Similar to previous work, we refrain from rigorously defining the notions of named entities [Dalvi et al. 2009] and define a named entity (entity for short) as a unique name in some (possibly infinite) domain. A concept is a set of entities, that is, its instances. For example, *person* and *country* are concepts corresponding to entities *Albert Einstein* and *Jordan*, respectively. A collection is a set of text documents. There may be several mentions, that is, occurrences, of an entity in a collection. For example, *Michael Jeffrey Jordan* and *Michael Jordan* refer to the famous athlete in the collection shown in Figure 2. We call these mentions *instances* of the entity and, for brevity, also the instances of its concept (*athlete*).

A domain may have some constraints on the relationship between its concepts [Abiteboul et al. 2011]. Concepts C_1 and C_2 are *mutually exclusive* in a domain if and only if no entity belongs to both C_1 and C_2. For instance, concepts *person* and *location* are mutually exclusive, as no entity is both a person and a location. When all concepts in domain D are mutually exclusive, then each occurrence of an entity in a collection over D maps to exactly one concept. Our study of real-world ontologies, such as DBPedia (*wiki.dbpedia.org/Ontology*), Schema.org (*schema.org*), and YAGO, indicates that mutually exclusive concepts appear frequently in these ontologies. For example, in Schema.org, each entity should belong to only one of the concepts of *Action*, *BroadcastService*, *Class*, *CreativeWork*, *Event*, *Intangible*, *MedicalEntity*, *Organization*, *Person*, *Place*, *Product*, and *Property*, which are mutually exclusive. As another example, in DBPedia, different types of organizations, places, events, devices, and creative works are described by mutually exclusive concepts. Mutually exclusive concepts are also easier to annotate via learning-based methods, as one can use the positive training examples of one concept as negative training examples for the others [Riloff and Jones 1999]. When this constraint is available in the domain, we exploit it to find cost-effective conceptual designs. Concepts in a domain may have other types of relationships such as a subclass/superclass relationship (e.g., person and scientist). Analyzing and solving the problem of cost-effective conceptual design for concepts with other types of relations is a larger undertaking and provides interesting subjects for future work.

Extracted entities can be represented and stored in a variety of data formats, such as XML files [Chu-Carroll et al. 2006] or annotation stores [Kandogan et al. 2006; Fagin et al. 2010]. Our framework and algorithms are oblivious to the ways in which the annotated documents are stored.

In this article, we consider queries that seek information about named entities [Chu-Carroll et al. 2006; Chakrabarti et al. 2007; Chu-Carroll and Prager 2007]. A query takes the form (C, T), where C is a concept and T a finite set of keywords. Example queries include (*person,* {*Jordan*}) and ((*location,* {*Jordan attractions*}). This type of query has been widely used to search annotated collections [Chu-Carroll et al. 2006; Graupmann et al. 2005; Chakrabarti et al. 2007; Pound et al. 2010]. Query (C, T) over collection *CO* is answered by a function that maps T to a ranked list of documents in *CO* such that each document in the list contains an occurrence of an entity in C. One may use a variety of functions to rank the documents [Manning et al. 2008].

Empirical studies on real-world query logs indicate that the majority of entity-centric queries refer to a single entity [Sanderson 2008]. Since this article is the starting effort to address the problem of cost-effective conceptual design, it is reasonable to start with the aforementioned class of queries. We do not consider complex queries such as aggregations (e.g., the average number of actors in a movie) or those that seek information about relationships between several entities. Such queries require a different model and algorithms, and thus are beyond the scope of this work.

4. THE BENEFIT OF A CONCEPTUAL DESIGN

4.1. Objective Function

Let $\mathcal{S}$ be the design of annotated collection CO and $\mathcal{Q}$ be a set of queries over CO. We would like to quantify the degree by which $\mathcal{S}$ improves the effectiveness of answering queries in $\mathcal{Q}$ over CO. The value of this function should be larger for those designs that help the query interface to answer a larger number of queries in $\mathcal{Q}$ more effectively. It has been shown that most information needs over annotated collections are precision oriented [Dill et al. 2003; Chu-Carroll et al. 2006]. In some settings, users may be more interested in improving query answer recall, rather than precision at k. For instance, a biologist may want to view all possible genomic causes of cancer listed in the scientific literature. The problem of cost-effective design to maximize other objective functions, such as recall, is an interesting subject for future work.

We choose the standard metric of *precision at k* ($p@k$ for short) to measure the effectiveness of answering queries over an annotated collection [Manning et al. 2008]. The value of $p@k$ for a query is the ratio of the number of relevant answers in the top-k returned answers for the query, divided by k. Precision at k has also a simpler form than other precision-oriented metrics such as *mean reciprocal rank* (MRR) or *normalized discounted cumulative gain* (NDCG), thus, is easier to optimize [Manning et al. 2008]. We average the values of $p@k$ over queries in $\mathcal{Q}$ to measure the amount of effectiveness in answering queries in $\mathcal{Q}$.

4.2. Effectiveness Improvement for Queries of Annotated Concepts

Let $Q : (C, T)$ be a query in $\mathcal{Q}$. If C is annotated, that is, $C \in \mathcal{S}$, the query interface will find and return only those documents that contain information about entities in C. It will then rank them according to its ranking function, such as the traditional TF-IDF scoring methods or learning to rank techniques [Manning et al. 2008]. Our model is orthogonal to the method used to rank the candidate answers. Annotating C in CO will help the query interface to avoid nonrelevant results that otherwise may have been placed in the top-k answers for Q. We call the fraction of queries in $\mathcal{Q}$ whose concept is C the *popularity* of C in $\mathcal{Q}$. Let $u_{\mathcal{Q}}$ be the function that maps concept C to its popularity in $\mathcal{Q}$. When $\mathcal{Q}$ is clear from the context, we simply use u instead of $u_{\mathcal{Q}}$. The portion of queries for which the query interface returns only those documents about entities in their desired concepts is $\sum_{C \in \mathcal{S}} u(C)$. Given all other conditions are the same, the larger the value of $\sum_{C \in \mathcal{S}} u(C)$, the more likely that the query interface will achieve a larger $p@k$ value over queries in $\mathcal{Q}$. Hence, we may use $\sum_{C \in \mathcal{S}} u(C)$ to compare the degrees of improvement in the value of $p@k$ over queries in $\mathcal{Q}$ achieved by various designs.

Annotators, however, may make mistakes in identifying the correct concepts for occurrences of entities in a collection [Chu-Carroll et al. 2006]. An annotator may annotate some appearances of entities from concepts other than C as the occurrences of entities in C. For instance, the annotator of concept *person* may annotate *Lincoln Building* as a person. The *accuracy* of annotating concept C over CO is the number of correct annotations of C divided by the number of all annotations of C in CO. We

denote the accuracy of annotating C over CO as $\text{pr}_{CO}(C)$. When CO is clear from the context, we show $\text{pr}_{CO}(C)$ as $\text{pr}(C)$. Given query $Q : (C, T)$ and $C \in \mathcal{S}$, it is reasonable to assume that $1 - \text{pr}(C)$ of the top-k results may contain information about entities that do not belong to C. Hence we should refine our estimate to

$$\sum_{C \in \mathcal{S}} u(C)\text{pr}(C) \tag{1}$$

in order to reward those designs whose concepts are annotated more accurately.

4.3. Effectiveness Improvement for Queries of Unannotated Concepts

Given query $Q : (C, T) \in \mathcal{Q}$, if $C \in \mathcal{C} - \mathcal{S}$ (C is not annotated by $\mathcal{S}$), there is insufficient metadata information in the collection for the query interface to identify the occurrences of the entities in C. Therefore, it may view the concept name C and the keywords in T as a bag of words and use some document ranking function to return the top-k answers for Q. We like to estimate the fraction of the results for Q that contain a matching entity in concept C. Given all other conditions are the same, the larger this fraction, the more likely that the query interface delivers more relevant answers, and therefore a larger $p@k$ value for Q. Based on the available constraints on the relations between concepts in the domain, we provide two different estimations of the fraction of the results for Q that contain a matching entity in concept C.

Domains with mutually exclusive concepts. If concepts in the domain are mutually exclusive, the annotated concepts may help the query interface to eliminate some nonrelevant answers from its results for Q. For example, assume the instances of concept *location* are annotated and the instances of concept *person* are not annotated in the collection. As these concepts are mutually exclusive, given query *(person,* {*Jordan*}*)*, the query interface can ignore matching instances like *Jordan River* for this query. Because text documents are coherent, they do not usually contain information about entities with similar or the same name but from mutually exclusive concepts. For instance, it is unlikely to find a document that contains information about both *Jaguar*, the vehicle, and *Jaguar*, the animal. Hence, the query interface can eliminate those candidate answers for Q whose matched terms are annotated by concepts other than the concept of Q. By removing these nonrelevant answers from its ranked list, the query interface may improve the value of $p@k$ for Q.

In order to compute the fraction of candidate answers for Q whose matching instances belong to C, we have to first calculate the fraction of candidate answers that survive the elimination. This ratio, however, may vary across different queries in $\mathcal{Q}$ as some queries may have more candidate answers with matched annotated instances from concepts in $\mathcal{S}$ than others. Estimating this ratio per query is generally hard, as it may require estimating and computing model parameters per query. In particular, detailed information about queries in a query workload such as their candidate answers may not be always available. Hence, in order to have an estimation which can be efficiently and effectively computed over a large number of queries, we assume all queries in $\mathcal{Q}$ have equal ratios of candidate answers that contain matched instances of a certain concept in the domain. We further estimate this ratio for the concept by the fraction of documents in the collection that contain instances of the concept. Our empirical results using queries from a real-world search engine query log and collection (reported in Section 7) show that, in spite of these simplifying assumptions, our model effectively estimates the degrees of improvement achieved by various designs for a collection.

Let $d_{CO}(E)$ denote the fraction of documents that contain instances of concept E in collection CO. These instances may or may not be annotated, depending on whether $E \in \mathcal{S}$. We call $d_{CO}(E)$ the *frequency* of E over CO. When CO is clear from the context, we

denote the frequency of E as $d(E)$. Given design $\mathcal{S}$ for collection CO, we want to compute the fraction of those candidate answers for query $Q : (C, T)$ that contain a matching instance of concepts $E \in \mathcal{C} - \mathcal{S}$. In order to simplify our model, we estimate this fraction as $\sum_{E \in \mathcal{C}-\mathcal{S}} d(E)$. Our experimental results in Section 7.2 indicate that, despite of this simplification, our objective function effectively captures the degree of improvement delivered by a conceptual design over a collection. This portion of answers will stay in the list of results for Q after the query interface eliminates all candidate answers with matching instances from concepts in $\mathcal{S}$. Hence, the fraction of candidate answers that contain a matching instance of concept C in the list of answers for a query in $\mathcal{Q}$ is $d(C)/\sum_{E \in \mathcal{C}-\mathcal{S}} d(E)$. We assume that the documents in the collection are not so long such that finding information about a matching entity from a single document takes a great deal of time and effort.

Using this estimation and Eq. (1), we formally define the function that estimates the likelihood of improvement for the value of *p@k* for queries that both belong and those that do not belong to the conceptual design in a query workload over a collection that is annotated by concepts in design $\mathcal{S}$.

Definition 4.1. Given domain $\mathcal{C}$ with mutually exclusive concepts, query workload $\mathcal{Q}$, and conceptual design $\mathcal{S} \subseteq \mathcal{C}$, the *annotation benefit* of $\mathcal{S}$ is

$$AB(\mathcal{S}) = \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) + \sum_{C \in \mathcal{C}-\mathcal{S}} u(C)\frac{d(C)}{\sum_{E \in \mathcal{C}-\mathcal{S}} d(E)}. \quad (2)$$

Overall, the annotation benefit estimates the likelihood in improving users' satisfaction by answering queries more precisely. The larger the value of the annotation benefit for design $\mathcal{S}$ over collection CO, the more likely it is that $\mathcal{Q}$ will have a larger average *p@k* over the version of CO annotated by concepts in $\mathcal{S}$.

The first term of the annotation benefit in Eq. (2) reflects the portion of queries for which the query interface returns only those candidate answers with instances matching the concept of the query. It is larger for those concepts that are more frequently used in queries. For example, let a domain contain concepts C_1, C_2, and C_3, where the instances of C_1 appear in 90% of queries and 1% of documents, the instances of C_2 occur in 1% of queries and 90% of documents, and the instances of C_3 appear in 9% of queries and 9% of documents. If all annotators have perfect accuracies (i.e., $\text{pr}(C) = 1$, $C \in \{C_1, C_2, C_3\}$), we have $\sum_{C \in \{C_1\}} u(C) > \sum_{C \in \{C_2\}} u(C)$. Although C_1 appears in only 1% of documents, it is used in 90% of queries. Hence, it is more likely that the query interface will answer the input queries more effectively if we annotate the instances of C_1 rather than C_2 in the collection.

The second term represents the impact of annotating the concepts in $\mathcal{S}$ on the likelihood of improving the precision of answering queries whose concepts are not in $\mathcal{S}$. Given that the concept of a query does not belong to $\mathcal{S}$, the more frequent the concepts in $\mathcal{S}$ in the collection, the more nonrelevant answers the query interface can eliminate.

Domains without constraints regarding the overlap of concepts. If there is no constraint on the relations between concepts in the domain, such as whether they are mutually exclusive or superclass/subclass, the query interface has to examine all documents in the collection to answer Q. For example, assume that a domain contains concepts *actress* and *director* and the entities of *actress* are annotated in the collection. Given query *(director, {Rossellini})*, the query interface cannot filter out its matching instances from concept *actress* like *Isabella Rossellini* because concepts *actress* and *director* are not mutually exclusive. Thus, if the instances of concept C are not annotated, $C \notin \mathcal{S}$, the fraction of candidate answers of $Q : (C, T)$ that contain a matching instance

of concepts C is $d(C)$. Using Eq. (1), we formally define the function that estimates the likelihood of improvement for the value of $p@k$ for all queries in a query workload over a collection that is annotated by concepts in design $\mathcal{S}$ over domains without any constraint.

Definition 4.2. Given domain $\mathcal{C}$ without any constraint, query workload $\mathcal{Q}$, and conceptual design $\mathcal{S} \subseteq \mathcal{C}$, the *annotation benefit* of $\mathcal{S}$ is

$$AB(\mathcal{S}) = \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) + \sum_{C \in \mathcal{C}-\mathcal{S}} u(C)d(C). \tag{3}$$

Similar to the formula for annotation benefit in Eq. (2), the first term of the annotation benefit in Eq. (3) reflects the group of queries for which the query interface returns only those candidate answers with instances matching their concepts. The second term of the annotation benefit in Eq. (3), however, is different from the second term in Eq. (2) and represents the impact of the frequency of a concept that is not in $\mathcal{S}$ on the likelihood of the precision of its queries.

Some domains may contain a mix of mutually exclusive and overlapping concepts. Analyzing and solving the problem of cost-effective conceptual design for such domains is a larger undertaking which requires more space than one article and provides an interesting subject for future work.

4.4. Estimating the Input Parameters

According to Definitions 4.1 and 3, we need popularities, frequencies, and accuracies of annotation for each concept in a domain in order to compute the annotation benefit of a design over the domain. These values, however, are not usually available before annotating the instances of concepts in the collection. Similar issues arise in database query optimization, where complete information about running times of operators in a query is not available before running the query [GarciaMolina et al. 2008]. Our empirical results indicate that one can effectively estimate the values of popularities and frequencies of concepts over a small sample of a collection (e.g., 384 out of about 1 million documents). The enterprise may use methods such as crowd sourcing to compute the popularities and frequencies of concepts over such small samples. These annotated documents may be also used as training data for the annotation program of the concept if it is selected for annotation. In some settings, a sample workload of queries with their concepts is not available, that is, we may have access only to pure keyword queries. The enterprise can use the click-through information of sample queries to effectively find their associated concepts [Bennett et al. 2007]. Nevertheless, rigorous study of parameter estimation methods for concept extraction requires a deeper theoretical investigation and empirical analysis over more than one collection. An interesting future work is to find practical parameter estimation methods that can effectively estimate such parameters over many different collections.

An enterprise may use the annotation-benefit function to choose those concepts for which it should develop annotation programs, therefore it may not know the accuracies of the annotation programs in design time. Because one has to spend more time and resources to develop a more accurate annotator, the accuracy of annotating a concept somewhat represents the cost of developing its annotation program. Hence, the enterprise may set the accuracy of annotating a concept to a reasonable value that can be achieved using its associated cost. It may also compute and compare the values of annotation benefit for designs across multiple assignments of costs and accuracies of annotating concepts in the domain.

5. COST-EFFECTIVE CONCEPTUAL DESIGN

5.1. Costs of Concept Extractions

Researchers have noticed the overheads and costs of curating and organizing large datasets [Dong et al. 2013; Kanani and McCallum 2012; Jain et al. 2008a]. For example, some researchers have recently considered the problem of selecting datasets for fusion such that the marginal cost of acquiring and processing a new dataset does not exceed its marginal gain, where cost and gain are measured using the same metric, such as U.S. dollars [Dong et al. 2013]. Other researchers have considered the problem of filling missing values in a relational database by extracting information from the Web, given that the resources available for information extraction are limited [Kanani and McCallum 2012; Jain et al. 2008a]. They identify the costs associated with information extraction as the computational power and network bandwidth needed to issue queries to find and download relevant documents and the time to execute information extraction programs over the downloaded documents. In addition to these recurring overheads, concept extraction may incur one-time costs such as the amount of time, money, or manual labor spent on developing, training, and/or maintaining concept extractors.

Our goal is to propose a rather general framework that can capture various types of costs involved in concept extraction. Given collection CO, we define function w_{CO} that maps each concept to a real number, which reflects the amount of resources used to annotate instances of this concept over collection CO. When the collection is clear from the context, we simply denote the cost function as w. The cost function captures both one-time and recurring overheads. In some settings, concept annotation incurs multiple types of costs which may be measured in different units. For instance, an enterprise may spend financial resources to develop concept extractors, and it will take some time to execute these annotators. The enterprise may not be able to represent these multiple cost elements using a unified measure. We will explain how our proposed framework handles these cases in Section 6.1.

We assume that annotating certain concepts does not impact the cost and accuracy of other concepts in the collection. The costs of (re-)writing, (re-)running, and maintaining an extractor for a concept are still considerable in most cases after coupling its extraction with other related concepts. For example, programmers have to find, extract, and select a great deal of relevant features for each concept separately, run its extraction programs, and rewrite and/or rerun it as the underlying collection evolves. The problem also becomes rather complex to express and solve without this assumption.

Researchers have developed effective methods to estimate the required time and computational power for concept extraction [Jain et al. 2008a]. An enterprise may also use the amount of money needed to purchase extraction programs from other companies (e.g., *ontotext.com*) to estimate the cost of acquiring concept extraction programs. It may predict the cost of annotator programs that are developed in-house using current techniques for predicting costs of software development and maintenance [Boehm et al. 2000]. In the absence of any evidence, one may assume that all concepts require an equal amount of resources for annotation. As we show in Section 5, it is still challenging to find cost-effective designs in this setting.

5.2. The Cost-Effective Conceptual Design Problem

Since the resources available to develop, maintain, and execute concept annotators are limited, our goal is to find a conceptual design $\mathcal{S}$ such that annotating the concepts in $\mathcal{S}$ in the queries and collection maximizes the annotation benefit. Let B denote the amount of resources available to perform the annotation. Annotating a set of concepts $\mathcal{S}$ is feasible if $\sum_{C \in \mathcal{S}} w(C) \leq B$. We formally define the annotation-benefit problem as follows.

Problem 5.1. Given a domain $\mathcal{C}$, the goal of the COST-EFFECTIVE CONCEPTUAL DESIGN problem is to construct a conceptual design $\mathcal{S}$ that maximizes the annotation benefit (AB) while satisfying the constraint $w(\mathcal{S}) \leq B$.

In the case of domains with no constraints, we can rewrite the annotation-benefit function as follows.

$$\begin{aligned} AB(\mathcal{S}) &= \sum_{C \in \mathcal{S}} u(C)\mathrm{pr}(C) + \sum_{C \in \mathcal{C}-\mathcal{S}} u(C)d(C) \\ &= \sum_{C \in \mathcal{S}} u(C)(\mathrm{pr}(C) - d(C)) \\ &\quad + \sum_{C \in \mathcal{C}} u(C)d(C), \end{aligned}$$

where the term $\sum_{C \in \mathcal{C}} u(C)d(C)$ is independent of $\mathcal{S}$. Concept extraction is intended to be an informative process, that is, it should perform better than random selection [Downey et al. 2006]. A random annotator for concept C with frequency $d(C)$ will achieve $\mathrm{pr} = d(C)$. Thus it is reasonable to assume that, for an annotation of C that takes resources, we have $\mathrm{pr}(C) \geq d(C)$. Moreover, if for a concept C, $\mathrm{pr}(C) < d(C)$, it is better to leave the concept unannotated. If we do not annotate C, the benefit we get from C is at least $u(C)d(C)$, while annotating C gives a benefit of $\mathrm{pr}(C)u(C)$ for concept C, which is less than $u(C)d(C)$.

Thus, the optimization problem in this setting is the following:

$$\max \quad \sum_{C \in \mathcal{S}} u(C)(\mathrm{pr}(C) - d(C)), \ \text{s.t.} \sum_{C \in \mathcal{S}} w(C) \leq B.$$

If the domain has n concepts, the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains with no constraints is equivalent to the 0-1 KNAPSACK problem with n objects, where the value of each object O_C is $u(C)(\mathrm{pr}(C) - d(C))$ and its weight is $w(C)$. Since the 0-1 KNAPSACK problem is NP-hard, the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains with no constraints is also NP-hard.

Next, we prove that the COST-EFFECTIVE CONCEPTUAL DESIGN problem is NP-hard for domains with mutually exclusive concepts by a reduction from the following NP-hard variant of the PARTITION problem [Korte and Schrader 1981].

Problem 5.2. Let $\mathcal{A} = \{a_1, \ldots, a_{2m}\}$ be a set of $2m$ positive integers that sum up to $2A$, such that for each $a \in \mathcal{A}$, $\frac{A}{m+1} < a < \frac{A}{m-1}$. The goal is to decide whether there exists a set $\mathcal{I} \subset \mathcal{A}$ such that $\sum_{a \in \mathcal{I}} a = A$.

THEOREM 5.3. *Problem 5.2 polynomially reduces to the* COST-EFFECTIVE CONCEPTUAL DESIGN *problem over a domain with mutually exclusive concepts.*

PROOF. Given an instance of Problem 5.2, we construct an instance of the COST-EFFECTIVE CONCEPTUAL DESIGN problem with $2m$ concepts as follows. For each $1 \leq i \leq 2m$, let $w(C_i) = u(C_i) = a_i$, $\mathrm{pr}(C_i) = 1$, and $d(C_i) = 1$ and let $B = A$.

A conceptual design $\mathcal{S}$ is a maximal design for the COST-EFFECTIVE CONCEPTUAL DESIGN problem if there exists no $C' \in \mathcal{C} - \mathcal{S}$ such that $w(C') + \sum_{C \in \mathcal{S}} w(C) \leq B$; there is no concept C such that its annotation cost is less than the leftover annotation budget.

Moreover, since for each i we have $w(C_i) < \frac{A}{m-1}$ and $B = A$, the size of each maximal feasible solution is either $m-1$ or m. Next we show that the optimal conceptual design of the constructed instance of the COST-EFFECTIVE CONCEPTUAL DESIGN problem is at most $A + \frac{A}{m}$ and this value is obtained iff there exists a set $\mathcal{S}$ such that $w(\mathcal{S}) = A$. By a

straightforward analysis, we have

$$\begin{aligned} AB(\mathcal{S}) &\leq \sum_{C \in \mathcal{S}} u(C) + \frac{2A - \sum_{C \in \mathcal{S}} u(C)}{m} \\ &= \frac{2A}{m} + \sum_{C \in \mathcal{S}} u(C) \left(1 - \frac{1}{m}\right) \\ &= \frac{2A}{m} + \sum_{C \in \mathcal{S}} w(C) \left(1 - \frac{1}{m}\right) \\ &\leq \frac{2A}{m} + A\left(1 - \frac{1}{m}\right) = A + \frac{A}{m}, \end{aligned}$$

where $\mathcal{S}$ is a feasible conceptual design. Moreover, $AB(\mathcal{S}) = A + \frac{A}{m}$ iff $w(\mathcal{S}) = A$. Thus there exists a set $\mathcal{I} \subseteq \mathcal{A}$ such that $\sum_{a \in \mathcal{I}} a = A$ iff its corresponding Cost-Effective Conceptual Design instance (which is constructed in polynomial time) has a solution of value $A + \frac{A}{m}$. □

Problem 5.4. (*Subset Sum Problem*). In the Subset Sum problem we are given a set $\mathcal{S}$ of n positive integer numbers and the goal is to determine whether there exists a subset $\mathcal{A} \subseteq \mathcal{S}$ whose total sum of its element is W, where W is a positive integer.

Theorem 5.5. *The* Subset Sum *problem polynomially reduces to the* Cost-Effective Conceptual Design *problem over a domain with mutually exclusive concepts.*

Proof. Given an instance of the Subset Sum problem with n integers $a_1, \ldots, a_n$ and query value W, we construct the following instance of Cost-Effective Conceptual Design. For each $i \leq n$, create a concept C_i such that $u(C_i) = w(C_i) = a_i$ and $d(C_i) = 0$. Moreover, we add two extra concepts C_{n+1} and C_{n+2} with $w(C_{n+1}) = w(C_{n+2}) = W$, $u(C_{n+1}) = u(C_{n+2}) = 0$ and $d(C_{n+1}) = d(C_{n+2}) = 1$. Since for each i, $u(C_i)d(C_i) = 0$, the benefit of the mixed part is always zero. Thus for the constructed instance, W is an upper bound for Cost-Effective Conceptual Design and it obtains W iff there exists a subset $\mathcal{A}$ such that $\sum_{a_i \in \mathcal{A}} a_i = W$. □

By Theorem 5.3 and Theorem 5.5 and the fact that both the Subset Sum problem and Problem 5.2 are NP-hard, we have the following corollary.

Corollary 5.6. *The* Cost-Effective Conceptual Design *problem is NP-hard.*

If a domain contains a manageable number of concepts, developers can manually estimate the popularities of concepts among users and their associated costs, and manually select the most cost-effective concepts for extraction. However, we observe a new trend in concept extraction where the number of possible concepts is too large for a manual approach. For instance, search engine companies such as Google and Microsoft have proposed a set of hundreds of concepts, called *schema.org*. Enterprises can annotate their Web pages using the concepts from *schema.org*, so that users of Google and Bing can satisfy their information needs from the enterprises' Web sites more easily. It is very difficult to manually navigate through hundreds of concepts, pick the ones that are most likely to be queried by users, and select a cost-effective subset of these concepts. This is particularly true for enterprises with diverse sets of Web pages, such as newspapers, like *The New York Times*, and large companies such as IBM. Further, as shown in Section 6, a cost-effective design does not necessarily include the concepts most popular among users, as a concept may be sufficiently frequent in the

collection for an effective document ranking method to provide answers with reasonable ranking quality for queries involving this concept.

The obvious first step toward a solution for the COST-EFFECTIVE CONCEPTUAL DESIGN problem is to adopt a greedy solution, that is, pick those concepts with the largest value of $u(C)\text{pr}(C)$ or the largest values for both $u(C)\text{pr}(C)$ and $d(C)$. However, the following examples show the problems of using such ideas.

Example 5.7. Consider a domain with three concepts $\{event, personality, shape\}$ from the YAGO ontology over the collection of English Wikipedia articles [Schenkel et al. 2007]. Some instances of concept *event*, *personality*, and *shape* in this collection are *Jesus Christ Crucifixion*, *Julia Child*, and *sphere*. These concepts have the following normalized u and d values over the collection.

	event	*personality*	*shape*
u	0.57	0.38	0.05
d	0.91	0.07	0.02

Assume they can be annotated with perfect accuracy. Given that developing an annotator costs the same for all three concepts, that is, $w(C) = 1$, $C \in \{event, personality, shape\}$, and $B = 1$, we would like to select a concept for annotation that delivers the maximum annotation benefit. Although *event* has larger u and d values than the other two concepts, annotating *place* maximizes the annotation-benefit function.

$$\begin{aligned} AB(event) &= 0.57 + \frac{1}{1-0.91}(0.38 \times 0.07 + 0.05 \times 0.02) \\ &= 0.877 \end{aligned}$$

By annotating concept *personality*, the value of the annotation benefit becomes

$$\begin{aligned} AB(Personality) &= 0.38 + \frac{1}{1-0.07}(0.57 \times 0.91 + 0.05 \times 0.02) \\ &= 0.939. \end{aligned}$$

This is because $d(personality) \approx d(shape)$. However, *event* has considerably higher d value than the other two concepts. Thus, after *event* has been annotated, *personality* and *shape* are still indistinguishable, but annotating *personality* helps *event* to be recognizable as well.

An interesting special case of the COST-EFFECTIVE CONCEPTUAL DESIGN problem over mutually exclusive domains is its *unweighted* variant, the setting in which all costs are equal. It will provide useful insights to the general problem and accepts exact algorithms in some cases. It may also occur in practice. For instance, if the enterprise does not have sufficient information about the costs of the concepts in a domain, it may assume their costs to be equal. In this setup, the goal is to find a set of B concepts, $\mathcal{S}$, whose annotations maximize the value of the annotation benefit over the collection where B is the available budget for annotation.

5.3. Equally Popular or Frequent Concepts

An interesting variant of the COST-EFFECTIVE CONCEPTUAL DESIGN problem is the case in which all concepts are equally popular; for each $C \in \mathcal{C}$, $u(C) = u$, where u is a fixed constant.

PROPOSITION 5.8. *Suppose that for each $C \in \mathcal{C}$, $u(C) = u$ where u is a fixed constant. The greedy approach that picks concepts in descending order of* pr() *values achieves an optimal solution.*

PROOF. By annotating a set $\mathcal{S}$ of concepts, the value of the annotation benefit is equal to

$$\begin{aligned} AB(S) &= \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) + \frac{\sum_{C \in \mathcal{C}-\mathcal{S}} u(C)d(C)}{\sum_{C \in \mathcal{C}-\mathcal{S}} d(C)} \\ &= u \sum_{C \in \mathcal{S}} \text{pr}(C) + \frac{u}{\sum_{C \in \mathcal{C}-\mathcal{S}} d(C)} \sum_{C \in \mathcal{C}-\mathcal{S}} d(C) \\ &= u \left(\sum_{C \in \mathcal{S}} \text{pr}(C) + 1 \right). \quad \square \end{aligned}$$

Another case of interest is when all concepts have almost the same number of instances in the collection; for each $C \in \mathcal{C}$, $d(C) = d$, where d is a fixed constant.

PROPOSITION 5.9. *Given that $d(C) = d$ for each $C \in \mathcal{C}$, choosing concepts in descending order of $u()\text{pr}()$ achieves the maximum annotation benefit.*

PROOF. Let $d(C) = d$ for each $C \in \mathcal{C}$, where d is a fixed constant, and let B be the given budget for annotation. After annotating an arbitrary set $\mathcal{S}$ of concepts, the value of the annotation benefit will be equal to

$$\begin{aligned} AB(S) &= \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) + \frac{\sum_{C \in \mathcal{C}-\mathcal{S}} u(C)d(C)}{\sum_{C \in \mathcal{C}-\mathcal{S}} d(C)} \\ &= \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) + \frac{d}{1 - B \cdot d} \sum_{C \in \mathcal{C}-\mathcal{S}} u(C) \\ &= \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) + \frac{d}{1 - B \cdot d} \left(\sum_{C \in \mathcal{C}} u(C)\text{pr}(C) - \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) \right) \\ &= \sum_{C \in \mathcal{S}} u(C)\text{pr}(C) \left(1 - \frac{d}{1 - B \cdot d} \right) + \frac{d}{1 - B \cdot d} \sum_{C \in \mathcal{C}} u(C)\text{pr}(C). \end{aligned}$$

Since the values of both $1 - \frac{d}{1-B \cdot d}$ and $\sum_{C \in \mathcal{C}}$ are independent of choice of $\mathcal{S}$, if we want to annotate a set of concepts $\mathcal{S}$ of size B from $\mathcal{C}$ to maximize the annotation benefit, we should choose those concepts with the m highest $u()\text{pr}()$ values. $\square$

5.4. Greedy Approximation Algorithm

We have shown in Section 5.3 that the greedy algorithm which picks those B concepts with the largest values of $u()\text{pr}()$ delivers the optimal answer for some cases of the COST-EFFECTIVE CONCEPTUAL DESIGN problem for mutually exclusive domains. We can show that the greedy algorithm has a reasonable approximation ratio for the unweighted variant of the problem over mutually exclusive domains. Consider a maximization problem $\mathcal{M}$. A polynomial-time algorithm $\mathcal{A}$ is an α-approximation to $\mathcal{M}$ if $\text{SOL}_{\mathcal{A}} \geq \frac{1}{\alpha}\text{OPT}_{\mathcal{M}}$, where $\text{SOL}_{\mathcal{A}}$ is the value of the solution returned by $\mathcal{A}$ and $\text{OPT}_{\mathcal{M}}$ is the value of the optimal solution to $\mathcal{M}$.

THEOREM 5.10. *The greedy algorithm is a* $(\text{pr}_{\min} \frac{B}{B+1})$*-approximation where* $\text{pr}_{\min} = \min_{C \in \mathcal{C}} \text{pr}(C)$.

PROOF. We use the method of Gal and Klots [1995] to analyze the algorithm. It is clear that the greedy algorithm at least picks those B concepts with largest $u()\text{pr}()$ value. On the other hand, the optimal solution cannot pick more than B concepts of $\mathcal{C}$. If the optimal solution picks ℓ concepts, we have $\ell + 1$ terms in the annotation-benefit formula (ℓ for the extracted concepts and one for the mixed one). The total benefit of these terms is less than the sum of the $\ell + 1$ largest values of $u()\text{pr}()$. Since the annotation-benefit of the output of the greedy algorithm is more than the sum of the $u()\text{pr}()$ of the B concepts with the largest $u()\text{pr}()$,

$$AB(S_{\text{greedy}}) > \text{pr}_{\min}\frac{B}{B+1}\text{OPT}. \quad \square$$

Furthermore, we can show that the ratio obtained for the greedy algorithm in Theorem 5.10 is tight. Consider the following example in Gal and Klots [1995]. A set $\mathcal{C}$ of $n+1$ concepts with the following u and d is given:

—$u(C_i) = 1 - (i-1)\epsilon, 1 \le i \le n$; $u(C_{n+1}) = 0$,
—$d(C_i) = (1 - \epsilon - \epsilon^2)/(n-1), 1 \le i \le n-1$; $d(C_n) = \epsilon^2$; $d(C_{n+1}) = \epsilon$,
—$\text{pr}(C_i) = 1, 1 \le i \le n+1$.

Here, ϵ is a small positive number. Set $w(C) = 1$ for each concept C in $\mathcal{C}$. For $B = n-1$, the value of the solution returned by the greedy algorithm is close to $n-1$, while the value of the optimal solution, which selects $\{C_2, \ldots, C_n\}$ is about n.

Using a similar idea to the proof of Theorem 5.10, we prove the approximation ratio of the greedy algorithm for the case where the concepts are not equally costly.

THEOREM 5.11. *The greedy algorithm is a* $(\text{pr}_{\min} \cdot \frac{w_{\min}}{w_{\max}} \cdot \frac{B}{B+1})$*-approximation, where* $\text{pr}_{\min} = \min_{C\in\mathcal{C}} \text{pr}(C)$, $w_{\min} = \min_{C\in\mathcal{C}} w(C)$, *and* $w_{\max} = \max_{C\in\mathcal{C}} w(C)$.

Hence, the greedy algorithm may not deliver a reasonably effective design if the concepts are not equally costly. In Section 6, we propose efficient algorithms with constant or sufficiently small approximation ratios for the general case of the problem.

6. APPROXIMATION APPROACHES FOR THE GENERAL CASE

A brute-force algorithm for the COST-EFFECTIVE CONCEPTUAL DESIGN problem may take several days or weeks of computation, even if the domain contains only 50 concepts. In this section, we design some efficient approximation algorithms for the general case of the COST-EFFECTIVE CONCEPTUAL DESIGN problem.

6.1. Approximate Popularity Maximization Algorithm (APM)

We can use available efficient algorithms with bounded approximation ratios for the 0-1 KNAPSACK problem to solve the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains without any constraint with the same approximation ratios. An algorithm $\mathcal{A}$ for an optimization problem P is *fully polynomial-time approximation scheme (FPTAS)* if, given $\varepsilon > 0$, $\mathcal{A}$ achieves approximation guarantee $(1+\varepsilon)$, and $\mathcal{A}$ finds the solution in time polynomial in the size of the input of P and $(1/\varepsilon)$. Since the 0-1 KNAPSACK problem is NP-hard, FPTAS is the best possible approximation for the problem, unless $P = NP$. In our experiments, we consider an FPTAS algorithm of the 0-1 KNAPSACK problem described in Ibarra and Kim [1975] that uses a dynamic programming approach. Note that, since the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains with no constraint is simply a 0-1 KNAPSACK problem, there is an FPTAS algorithm for the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains with no constraint.

Moreover, we use the idea behind FPTAS algorithms of the 0-1 KNAPSACK problem to devise an approximation algorithm for the COST-EFFECTIVE CONCEPTUAL DESIGN problem

over domains with mutually exclusive concepts. This algorithm ignores the improvement in effectiveness of answering those queries whose concepts are not in the design of a collection. As discussed in Section 4, this improvement is achieved by eliminating those nonrelevant answers whose concepts are in the design of the collection from the list of candidate answers for these queries. This degree of improvement is represented by the second term of the annotation-benefit function. Hence this algorithm picks a conceptual design $\mathcal{S}$ with maximum value of $\sum_{C\in\mathcal{S}} u(C)\text{pr}(C)$. We call this modified problem the POPULARITY MAXIMIZATION problem. More formally, given a domain $\mathcal{C}$, the POPULARITY MAXIMIZATION problem maximizes $\sum_{C\in\mathcal{S}} u(C)\text{pr}(C)$ subject to $\sum_{C\in\mathcal{S}} w(C) \leq B$.

The following lemma shows that we can design a constant factor approximation algorithm for the COST-EFFECTIVE CONCEPTUAL DESIGN problem by applying an algorithm with bounded approximation ratio for the POPULARITY MAXIMIZATION problem.

LEMMA 6.1. *A ρ-approximation algorithm for the* POPULARITY MAXIMIZATION *problem is a $(\rho + 1/\text{pr}_{\min})$-approximation for the* COST-EFFECTIVE CONCEPTUAL DESIGN *problem over domains with mutually exclusive concepts, where* $\text{pr}_{\min} = \min_{C\in\mathcal{C}} \text{pr}(C)$.

PROOF. Let $P = \langle \mathcal{C}, B, d, u, \text{pr}, w\rangle$ be an instance of the COST-EFFECTIVE CONCEPTUAL DESIGN problem. Let SOL be the solution returned by the ρ-approximation of the POPULARITY MAXIMIZATION problem on P and let OPT be the optimal solution of the COST-EFFECTIVE CONCEPTUAL DESIGN problem on P. Note that, for any set $\mathcal{S}$, the value of the second term in the annotation-benefit function

$$\sum_{C\in\mathcal{C}-\mathcal{S}} u(C)\frac{d(C)}{\sum_{E\in\mathcal{C}-\mathcal{S}} d(E)}$$

is less than $u(C_{\max})$, where $C_{\max} = \arg\max_{C\in\mathcal{C}} u(C)$. Let M_{OPT} be the value of the second term in the annotation-benefit function in the optimal solution, $M_{\text{OPT}} = \frac{\sum_{C\in\mathcal{C}\setminus\text{OPT}} u(C)d(C)}{\sum_{C\in\mathcal{C}\setminus\text{OPT}} d(C)}$. Since for each $C \in \mathcal{C}$, $w(C) \leq B$,

$$\sum_{C\in\text{SOL}} \text{pr}(C)u(C) \geq \text{pr}(C_{\max})u(C_{\max}) \geq \text{pr}(C_{\max})M_{\text{OPT}}.$$

Moreover, since SOL is a ρ-approximate solution of the POPULARITY MAXIMIZATION problem on P, we have

$$\rho \sum_{C\in\text{SOL}} \text{pr}(C)u(C) \geq \sum_{C\in\text{OPT}} \text{pr}(C)u(C).$$

These two together imply that $(\rho + 1/\text{pr}(C_{\max}))AB(\text{SOL}) \geq AB(\text{OPT})$; thus SOL is a $(\rho + 1/\text{pr}(C_{\max}))$-approximate solution. Let $\text{pr}_{\min} = \min_{C\in\mathcal{C}} \text{pr}(C)$. Since for any $C \in \mathcal{C}$, $\text{pr}(C) \geq \text{pr}_{\min}$, SOL is a $(\rho + 1/\text{pr}_{\min})$-approximate solution of the COST-EFFECTIVE CONCEPTUAL DESIGN problem. □

In particular, if $\text{pr}(C) = 1$ for all $C \in \mathcal{C}$, a ρ-approximation of the POPULARITY MAXIMIZATION problem is a $(\rho + 1)$-approximation for the COST-EFFECTIVE CONCEPTUAL DESIGN problem.

The POPULARITY MAXIMIZATION problem is also a version of the 0-1 KNAPSACK problem with n objects, if we choose the value of each object O_C to be $u(C)\text{pr}(C)$ and its weight to be $w(C)$. Thus, in our experiments we use the FPTAS algorithm for the KNAPSACK problem [Ibarra and Kim 1975] to solve this problem.

COROLLARY 6.2. *An FPTAS algorithm that returns a $(1 + \varepsilon)$-approximate solution to the* POPULARITY MAXIMIZATION *problem is a $(1 + \varepsilon + 1/\text{pr}_{min})$-approximation algorithm for the* COST-EFFECTIVE CONCEPTUAL DESIGN *problem over domains with mutually exclusive concepts whose running time is polynomial in $\frac{1}{\varepsilon}$ and the number of concepts.*

COROLLARY 6.3. *A greedy algorithm that returns a 2-approximate solution to the* POPULARITY MAXIMIZATION *problem is a* $(2 + 1/\text{pr}_{min})$*-approximation algorithm for the* COST-EFFECTIVE CONCEPTUAL DESIGN *problem whose running time is* $O(|\mathcal{C}| \log |\mathcal{C}|)$.

We call this algorithm the *approximate popularity maximization* (APM) algorithm.

Concept annotation may incur multiple types of costs which may be measured in different units therefore cannot be represented using a single measure. APM can be used to solve the COST-EFFECTIVE DESIGN PROBLEM in this case. The enterprise can categorize its cost elements into groups with compatible cost measures and add a separate budget constraint for each cost group to the COST-EFFECTIVE CONCEPTUAL DESIGN problem. We call this problem MULTIPLY-CONSTRAINED COST-EFFECTIVE CONCEPTUAL DESIGN. When there is no constraint regarding the overlap of concepts in the domain, one may use existing PTAS algorithms for the MULTIPLY-CONSTRAINED KNAPSACK problem to obtain a PTAS algorithm for the MULTIPLY-CONSTRAINED COST-EFFECTIVE CONCEPTUAL DESIGN problem [Freville 2004]. Unfortunately, there is no FPTAS algorithm for the MULTIPLY-CONSTRAINED KNAPSACK problem unless $P = NP$ [Korte and Schrader 1981].

Consider the case of the MULTIPLY-CONSTRAINED COST-EFFECTIVE CONCEPTUAL DESIGN problem where the concepts are mutually exclusive. We define the problem of MULTIPLY-CONSTRAINED POPULARITY MAXIMIZATION similar to the problem of POPULARITY MAXIMIZATION but with multiple budget constraints. Clearly, a PTAS algorithm for the MULTIPLY CONSTRAINED KNAPSACK problem is also a PTAS algorithm for the MULTIPLY-CONSTRAINED POPULARITY MAXIMIZATION problem. Since the objective function of the new problem is identical to that of the POPULARITY MAXIMIZATION problem with a single budget constraint, according to Lemma 6.1, this algorithm will be a $(1 + \varepsilon + 1/\text{pr}_{min})$-approximation for the MULTIPLY-CONSTRAINED COST-EFFECTIVE CONCEPTUAL DESIGN problem and its running time is polynomial in the number of concepts and exponential in $(\frac{1}{\varepsilon})$. In this article, however, we focus on the case where cost elements can be expressed through a single metric, and we leave empirical analysis of the MULTIPLY-CONSTRAINED COST-EFFECTIVE CONCEPTUAL DESIGN problem for future work.

6.2. Approximate Annotation-Benefit Maximization Algorithm (AAM)

In this section we present an FPTAS algorithm for the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains with mutually exclusive concepts. Since in Theorem 5.3 we proved that the problem is NP-hard, FPTAS is the optimal approximation guarantee for the problem unless $P = NP$. The algorithm is based on the dynamic programming method of the 0-1 KNAPSACK problem in addition to some scaling techniques. For simplicity in exposition of the algorithm, we assume that $\text{pr}(C) = 1$ for each $C \in \mathcal{C}$. However, in Remark 6.8 we state that our approach works for an arbitrary pr function, given an additional property that usually holds in practice. Without loss of generality, we can also assume that $u(C)$ and $d(C)$ are positive integers for all concepts.

Given a fixed constant N, we define the BOUNDED COST-EFFECTIVE$(\mathcal{C}, B, N)$ problem as follows.

$$\max_{\mathcal{S}} \quad f(N, \mathcal{S}) = \frac{1}{N}\left(N \sum_{C \in \mathcal{S}} u(C) + \sum_{C \in \mathcal{C}-\mathcal{S}} u(C)d(C)\right) \tag{4}$$

$$\text{s.t.} \sum_{C \in \mathcal{C}-\mathcal{S}} d(C) \leq N$$

$$\sum_{C \in \mathcal{C}-\mathcal{S}} w(C) \leq B$$

In addition to the cost constraint that we had previously, BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) has a constraint over frequency of documents. Let $\langle \mathcal{C}, B, d, u, w \rangle$ be an instance of COST-EFFECTIVE CONCEPTUAL DESIGN ($\mathcal{C}$, B). For any value of N, the value of the optimal solution of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) on $\langle \mathcal{C}, B, d, u, w \rangle$ is not more than the optimal solution of the annotation benefit of COST-EFFECTIVE CONCEPTUAL DESIGN on $\langle \mathcal{C}, B, d, u, w \rangle$. Moreover, for a fixed N, the objective function of the BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) is a separable function. Thus it is easier to find the maximum value of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) for a fixed N rather than finding the optimal conceptual design of the COST-EFFECTIVE CONCEPTUAL DESIGN problem.

LEMMA 6.4. *Let* OPT *be the value of the optimal solution of the* COST-EFFECTIVE CONCEPTUAL DESIGN ($\mathcal{C}$, B) *problem and let* OPT_{bnd} *be the maximum value of an optimal solution of* BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) *over different values of* N. *Then* $\text{OPT} = \text{OPT}_{\text{bnd}}$. *Moreover, the same set of concepts (conceptual design) obtains the optimal value in both functions.*

PROOF. Consider the COST-EFFECTIVE CONCEPTUAL DESIGN problem and assume that AB obtains its maximum value for set $\mathcal{S}_{\text{OPT}}$. Let $N_{\text{OPT}} = \sum_{C \notin \mathcal{C} - \mathcal{S}_{\text{OPT}}} d(C)$. Then, we have

$$\text{OPT}(\text{BOUNDED COST-EFFECTIVE}(\mathcal{C}, B, N_{\text{OPT}})) = AB(\mathcal{S}_{\text{OPT}}) = \text{OPT}.$$

Thus $\text{OPT}_{\text{bnd}} \geq \text{OPT}$. For the other direction, we know that the value of $f(N, \mathcal{S})$ is at most $AB(\mathcal{S})$ for all feasible solutions $\mathcal{S}$ of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N). This implies that $\text{OPT}_{\text{bnd}} \leq \text{OPT}$.

Hence $\text{OPT} = \text{OPT}_{\text{bnd}}$. It also implies that the set $\mathcal{S}$ that achieves the maximum value of COST-EFFECTIVE CONCEPTUAL DESIGN ($\mathcal{C}$, B) obtains the maximum value of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) over different values of N as well. □

Lemma 6.4 implies that, in order to find a set with the maximum annotation benefit, we can instead solve BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) for all different values of N and return the set that obtains the maximum value. In other words, first we give an FPTAS for BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N), where N is a given fixed value. The first step is to check whether for the given N there exists a feasible solution to BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N). For the given N, a feasible solution has to contain all concepts C that $d(C) > N$. Let $\mathcal{S}_{\text{rem}} = \{C | d(C) > N\}$ and $\mathcal{C}_{\text{rem}} = \mathcal{C} - \mathcal{S}_{\text{rem}}$. If $w(\mathcal{S}_{\text{rem}}) > B$, there is no feasible solution for BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N). Otherwise, we select all concepts in $\mathcal{S}_{\text{rem}}$ and we set $B_{\text{rem}} = B - w(\mathcal{S}_{\text{rem}})$ to be the leftover budget. The problem is equivalent to optimizing the bounded problem on $\mathcal{C}_{\text{rem}}$, B_{rem}, and N. Now, for each $C \in \mathcal{C}_{\text{rem}}$ we have $d(C) \leq N$. To solve BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) optimally for the given N, we can apply dynamic programming. Let $V_{\text{init}}(N) = \sum_{C \in \mathcal{C}_{\text{rem}}} u(C)d(C)/N$. We can rewrite the objective function of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) as follows:

$$\sum_{C \in \mathcal{S}} v(C) + V_{\text{init}}(N),$$

where $v(C) = u(C)(1 - d(C)/N)$ for each $C \in \mathcal{C}_{\text{rem}}$.

Let $\mathcal{C}_{\text{rem}} = \{C_1, \ldots, C_n\}$. We define $Q[i, P, X]$ to be the minimum required cost that we must pay to obtain a solution of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) of value at least $P - V_{\text{init}}$ if we are only allowed to annotate concepts from the first i concepts of $\mathcal{C}_{\text{rem}}$. We can state the recursive relation of $Q[i, P, X]$ as follows:

(1) $Q[0, 0, X] = 0$ for all $0 \leq X \leq N$;
(2) $Q[0, P, X] = \infty$ for all $P > 0$ and $0 \leq X \leq N$;
(3) $Q[i, P, X] = \min(Q[i - 1, P, X - d(C_i)],$
$Q[i - 1, \min\{P - v(C_i), 0\}, X] + w(C_i))$.

To find the optimal solution of BOUNDED COST-EFFECTIVE($\mathcal{C}, B, N$), we need to find the maximum value of V such that $Q[n, V, N] \leq B_{\text{rem}}$. The running time of the described dynamic programming is $O(nVN)$, where V is the value of the optimal solution of the bounded problem for the given N. The described dynamic programming is pseudo-polynomial and we can convert it to an FPTAS via scaling techniques.

LEMMA 6.5. *There exists a* $(1+\varepsilon)$*-approximation algorithm to* BOUNDED COST-EFFECTIVE($\mathcal{C}, B, N$) *which runs in* $O(Nn^3/\varepsilon)$.

PROOF. To eliminate the dependency of the running time on V, we scale $v(C)$ for each $C \in \mathcal{C}_{\text{rem}}$. Let $M = \max_{C \in \mathcal{C}_{\text{rem}}} v(C)$ and $\lambda = \frac{\varepsilon M}{n}$, where $n = |\mathcal{C}_{\text{rem}}|$. We define $\hat{v}(C) = \lfloor v(C)/\lambda \rfloor$. This implies that

$$\lambda \hat{v}(C) \leq v(C) \leq \lambda(\hat{v}(C) + 1).$$

The maximal value of BOUNDED COST-EFFECTIVE($\mathcal{C}, B, N$) over the scaled values is at most $\sum_{C \in \mathcal{C}_{\text{rem}}} \hat{v}(C) < \sum_{C \in \mathcal{C}_{\text{rem}}} (n/\varepsilon)(v(C)/M) = O(n^2/\varepsilon)$.

Now, let $\mathcal{C}_{\text{scl}}$ be the set returned by dynamic programming after scaling and let $\mathcal{C}_{\text{org}}$ be the optimal solution of the problem before scaling and $\text{OPT} = \sum_{C \in \mathcal{C}_{\text{org}}} v(C)$. Thus

$$\begin{aligned}
\sum_{C \in \mathcal{C}_{\text{scl}}} v(C) &\geq \lambda \sum_{C \in \mathcal{C}_{\text{scl}}} \hat{v}(C) \\
&\geq \lambda \sum_{C \in \mathcal{C}_{\text{org}}} \hat{v}(C) \\
&\geq \sum_{C \in \mathcal{C}_{\text{org}}} v(C) - \lambda |C_{\text{org}}| \\
&\geq \text{OPT} - \lambda n \\
&\geq \text{OPT} - \varepsilon M \geq (1-\varepsilon)\text{OPT} \geq \frac{1}{(1+\varepsilon)}\text{OPT}.
\end{aligned}$$

The first inequality comes from $\lambda \hat{v}(C) \leq v(C)$. The second one is because of the optimality of $\mathcal{C}_{\text{scl}}$ over the scaled values. The third inequality is derived from $v(C) \leq \lambda(\hat{v}(C)+1)$.

Thus, the proposed dynamic programing algorithm with the scaled profits is a $(1+\varepsilon)$-approximation whose running time is $O(Nn^3/\varepsilon)$. □

Although we need to satisfy the document frequency constraint of BOUNDED COST-EFFECTIVE($\mathcal{C}, B, N$), $\sum_{C \in \mathcal{C}_{\text{rem}} - \mathcal{S}} d(C) \leq N$, for a given N we can allow $\mathcal{S}$ to violate the document frequency constraint by ε; our ultimate goal is to maximize the annotation benefit of COST-EFFECTIVE CONCEPTUAL DESIGN ($\mathcal{C}, B$). Later we show that the value of AB for a $(1+\varepsilon)$-approximate solution of BOUNDED COST-EFFECTIVE($\mathcal{C}, B, N$), $\mathcal{S}$, that violates the document frequency constraint by at most ε is comparable to the optimal solution of BOUNDED COST-EFFECTIVE for the given N.

LEMMA 6.6. *There is a* $(1+\varepsilon)$*-approximation algorithm for* BOUNDED COST-EFFECTIVE($\mathcal{C}, B, N$) *that violates* $\sum_{C \in \mathcal{C}_{\text{rem}} - \mathcal{S}} d(C) \leq (1+\varepsilon)N$*, and its running time is* $O(n^4/\varepsilon^2)$.

PROOF. First we apply the scaling introduced in Lemma 6.5 and then scale d for concepts as follows. Define $\hat{d}(C) = \lfloor d(C)/\gamma \rfloor$ where $\gamma = \varepsilon N/n$. Then we work with $\hat{d}(C)$, $\hat{v}(C)$ and $\hat{N} = \lfloor N/\gamma \rfloor = \lfloor n/\varepsilon \rfloor$. Thus the running time of the algorithm on the scaled value is $O(\hat{N}n^3/\varepsilon) = O(n^4/\varepsilon^2)$. For the optimal solution $\mathcal{C}_s$ of the scaled instance,

we have $\sum_{C \in \mathcal{C}_{\text{rem}} - \mathcal{C}_{\text{scl}}} \hat{d}(C) \leq \hat{N}$. Thus

$$\begin{aligned}\sum_{C \in \mathcal{C}_{\text{rem}} - \mathcal{C}_{\text{scl}}} d(C) &\leq \gamma \left(\sum_{C \in \mathcal{C}_{\text{rem}} - \mathcal{C}_{\text{scl}}} \hat{d}(C) + (n - |\mathcal{C}_{\text{scl}}|) \right) \\ &\leq \gamma \hat{N} + \gamma n \\ &\leq N + \gamma n = (1+\varepsilon)N.\end{aligned}$$

This implies that the returned solution $\mathcal{C}_{\text{scl}}$ may violate N by a factor of ε. Thus, we have a $(1+\varepsilon)$-approximation algorithm for BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) that may violate constraint $\sum_{C \in \mathcal{C}_{\text{rem}} - \mathcal{C}_{\text{scl}}} d(C) \leq N$ by ε and its running time is $O(n^4/\varepsilon^2)$. □

By Lemma 6.6, we have an algorithm that finds a solution $\mathcal{S}_s$ with value at least $(1+\varepsilon)$ times the optimal solution of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N). However, $\mathcal{S}_s$ may violate the document frequency constraint by ε. Suppose that $\mathcal{S}_s$ is the set returned by the algorithm after performing the described scaling. Let $N_r = \sum_{C \in \mathcal{S}_s} d(C)$ and let N_s be the value of N for which $\mathcal{S}_s$ is returned in our algorithm (since we allow the algorithm to violate the constraint by ε, $N_r \leq (1+\varepsilon)N_s$). Suppose that $\mathcal{S}_{\text{OPT}}$ is the optimal solution of COST-EFFECTIVE CONCEPTUAL DESIGN($\mathcal{C}$, B). Lemmas 6.4 and 6.6 imply that $AB(\mathcal{S}_{\text{OPT}}) \leq (1+\varepsilon) f(N_s, \mathcal{S}_s)$. Thus $AB(\mathcal{S}_s) = f(N_r, \mathcal{S}_s) \geq f(N_s, \mathcal{S}_s)/(1+\varepsilon) \geq (1/(1+\varepsilon)^2) AB(\mathcal{S}_{\text{OPT}}) \geq (1-2\varepsilon) AB(\mathcal{S}_{\text{OPT}}) \geq \frac{1}{(1+2\varepsilon)} AB(\mathcal{S}_{\text{OPT}})$, where f is the objective function of the BOUNDED COST-EFFECTIVE problem. By maximizing BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) over all possible values of N ($0 < N \leq D_{total} = \sum_{C \in \mathcal{C}_{\text{rem}}} d(C)$), we can find a $(1+\varepsilon)$-approximation[1] of COST-EFFECTIVE CONCEPTUAL DESIGN($\mathcal{C}$, B) in $O(D_{\text{total}} \frac{n^4}{\varepsilon^2})$.

THEOREM 6.7. *The* COST-EFFECTIVE CONCEPTUAL DESIGN *problem admits an FPTAS algorithm.*

Instead of checking all possible values of N which lead to a pseudo-polynomial algorithm, we solve BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N) for some specific values of N (which is polynomial in the size of input) and still guarantees a $(1-\varepsilon)$-approximation (in the relaxed version we allow the solution to violate the document frequency constraint by a factor of ε). Consider the set $\mathcal{N} = \{N_1, \ldots, N_p\}$ such that $N_i = D_{\min}(1/(1-\varepsilon))^i$ where $D_{min} = \min_{C \in \mathcal{C}_{\text{rem}}} d(C)$ and $N_{p-1} \leq D_{\text{total}} \leq N_p$. This implies that $p < \log_{(1-\varepsilon)^{-1}}(D_{\text{total}}/D_{min}) + 1 = (\log D_{\text{total}} - \log D_{\min})/(-\log(1-\varepsilon)) + 1 < O((\log D_{\text{total}})/\varepsilon)$, where the last inequality comes from $-\log(1-\varepsilon) = -\ln(1-\varepsilon)/\ln 2 > \varepsilon/\ln 2$. Thus the number of different values of N we need to examine is polynomial in $\log D_{\text{total}}$ and $1/\varepsilon$. Suppose that $(N_{\text{OPT}}, \mathcal{S}_{\text{OPT}})$ is a pair that maximizes f, that is, $\text{OPT} = f(N_{\text{OPT}}, \mathcal{S}_{\text{OPT}})$ where OPT is the value of an optimal solution of COST-EFFECTIVE CONCEPTUAL DESIGN($\mathcal{C}$, B). Let N_g be the smallest member of $\mathcal{N}$ that is greater than N_{OPT}. Note that $(N_g, \mathcal{S}_{\text{OPT}})$ is a feasible solution to BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N_g) and, since $N_g > N_{\text{OPT}}$, $N_g f(N_g, \mathcal{S}_{\text{OPT}}) > N_{\text{OPT}} f(N_{\text{OPT}}, \mathcal{S}_{\text{OPT}})$. Thus $f(N_g, \mathcal{S}_{\text{OPT}}) > (N_{\text{OPT}}/N_g)\text{OPT}$. Since our algorithm examines $N = N_g$, the solution returned by our algorithm is at least $(1-\varepsilon)$ times the optimal solution of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N_g). Since $N_{\text{OPT}} > N_{g-1}$, $N_{\text{OPT}}/N_g \geq N_{g-1}/N_g$ and thus $f(N_g, \mathcal{S}_{\text{OPT}}) \geq (N_{g-1}/N_g)\text{OPT} \geq (1-\varepsilon)\text{OPT}$. This implies that the value of an optimal solution of BOUNDED COST-EFFECTIVE($\mathcal{C}$, B, N_g) is at least $(1-\varepsilon)\text{OPT} \geq \frac{1}{(1+\varepsilon)}\text{OPT}$. This fact, along with Lemma 6.6, leads us to obtain an FPTAS algorithm for the COST-EFFECTIVE CONCEPTUAL DESIGN problem with runtime $O((n^4 \log D_{\text{total}})/\varepsilon^3)$. Note that

[1]We let $\varepsilon' = 2\varepsilon$.

Table I. Summary of Approximation Ratios and Time Complexities of Approximation Algorithms for the COST-EFFECTIVE CONCEPTUAL DESIGN Problem

Algorithm	Approximation ratio	Running time
APM	$2+\varepsilon$	$O(n^3/\varepsilon)$
AAM	$1+\varepsilon$	$O(n^5/\varepsilon^3)$

$\log D_{\text{total}} \leq \log(nD_{\max}) \leq \log n + \log D_{\max}$, where $D_{\max} = \max_{C \in \mathcal{C}_p} d(C)$ and is polynomial in the size of input. Hence the running time is bounded by $O(n^5/\varepsilon^3)$.

Remark 6.8. We assumed that $\text{pr}(C) = 1$ for all $C \in \mathcal{C}$. However, our approach also works for a realistic function pr. For a given N, we define $v(C) = u(C)(\text{pr}(C) - \frac{d(C)}{N})$ and the proof holds as long as $v(C)$ is a positive value for all concepts.

Figure 3 depicts the steps of AAM algorithms. Table I summarizes the approximation algorithms we presented for the COST-EFFECTIVE CONCEPTUAL DESIGN problem over domains with mutually exclusive concepts.

7. EXPERIMENTS

7.1. Experiment Setting

Domains. To validate the accuracy of the annotation-benefit function and the effectiveness of our conceptual design algorithms, we use concepts from YAGO ontology version 2008-w40-2 [Schenkel et al. 2007]. YAGO organizes its concepts using IS-A (i.e., parent-child) relationships in a DAG with a single root. We define a *level* as a set of concepts that have the same distance (in terms of the number of edges) from the root of the ontology. Most levels in the DAG generally contain a set of mutually exclusive concepts. We select three domains from this ontology for our experiments. All concepts in each domain are mutually exclusive and have at least one instance in our dataset. *Domain M1* consists of seven concepts from the third level of the ontology. Examples of concepts in M1 are *object*, *causal agent*, and *psychological feature*. We use domain M1 to validate how accurately the annotation-benefit function estimates the effectiveness of answering queries over annotated collections. Since some validation experiments require running brute-force algorithms, we have to use a domain with a small number of concepts while containing as many documents in the dataset as possible. So we need to select concepts in the top level in the ontology tree. Thus the concepts in M1 are vague for users.

The popularities (u) and frequencies (d) of concepts in domain M1 are shown in Figure 4. We further select two larger domains from the YAGO ontology to evaluate the average-case performance ratios and efficiency of our approximation algorithms. *Domain M2* consists of 76 mutually exclusive concepts from the fourth level of the ontology, such as *location* and *event*. Since we would like to evaluate our algorithms over domains with concepts that have more specific meanings, that is, concepts in the lower level of the ontology tree, we expand some relatively abstract concepts such as *whole* to their descendants on the sixth level of the ontology and create a third domain, called *domain M3*. This domain consists of 87 concepts such as *person* and *animal*. We also select an additional domain, called *N1*, from the fifth level of YAGO with 10 concepts whose concepts are not guaranteed mutually exclusive. We use this domain to validate the annotation-benefit formula for those domains with no constraint and we measure the empirical approximation ratio of APM over these domains.

Dataset. We use a semantically annotated version of the Wikipedia collection that is created from the October 8, 2008 dump of English Wikipedia articles [Schenkel et al. 2007]. This collection uses concepts from the YAGO ontology. It contains 2,666,190

AAM-algorithm $\langle\langle$*Input:* $\langle \mathcal{C}, B, w, u, d, \varepsilon \rangle\rangle$

$\langle\langle \mathcal{C}$*: set of concepts,* B*: available budget to annotator*$\rangle\rangle$
$\langle\langle w$*,* u *and* d *represent cost, popularity and frequency of concepts respectively.*$\rangle\rangle$
$\langle\langle \varepsilon$*: approximation guarantee is* $1+\varepsilon\rangle\rangle$
Let $D_{\min} = \min_{C \in \mathcal{C}} d(C)$, $D_{\text{total}} = \sum_{C \in \mathcal{C}} d(C)$ and $\text{SOL} = \emptyset$.
$\langle\langle$*Solving several instances of* BOUNDED COST EFFECTIVE$(N)\rangle\rangle$
For $(N = D_{\min}; N \leq D_{\text{total}}; N = \frac{N}{1-\varepsilon})$
  $\mathcal{C}_{\text{rem}} \leftarrow \mathcal{C}$ and $V_{\text{init}} = 0$.
  For each $C \in \mathcal{C}$
    If $d(C) > N$
      $\mathcal{C}_{\text{rem}} \leftarrow \mathcal{C}_{\text{rem}} \setminus C$ $\langle\langle \mathcal{C}_{\text{rem}} = \{C \in \mathcal{C} \mid d(C) \leq N\}\rangle\rangle$
      $B \leftarrow B - w(C)$ $\langle\langle$*We have to pick* $\mathcal{C} \setminus \mathcal{C}_{\text{rem}}$ *in* $\text{SOL}_{\text{bnd}}\rangle\rangle$
  If $B < 0$ $\langle\langle$*Check whether* BOUNDED COST EFFECTIVE(N) *has feasible solution*$\rangle\rangle$
    break;
  Give an arbitrary ordering to $\mathcal{C}_{\text{rem}}$,$\{C_1, \cdots, C_p\}$.
  For each $C \in \mathcal{C}_{\text{rem}}$
    $v(C) \leftarrow u(C)(1 - \frac{d(C)}{N})$ $\langle\langle v(C)$*: profit of* $C\rangle\rangle$
  Let $M_{\text{rem}} = \max_{C \in \mathcal{C}_{\text{rem}}} v(C)$ and $n_{\text{rem}} = |\mathcal{C}_{\text{rem}}|$.
  $\lambda \leftarrow \frac{\varepsilon M_{\text{rem}}}{n_{\text{rem}}}$, $\gamma \leftarrow \frac{\varepsilon N}{n_{\text{rem}}}$, $\hat{N} \leftarrow \lfloor \frac{N}{\gamma} \rfloor$
  For each $C \in \mathcal{C}_{\text{rem}}$
    $\hat{v}(C) \leftarrow \lfloor \frac{v(C)}{\lambda} \rfloor$ $\langle\langle$*Scale profit function*$\rangle\rangle$
    $\hat{d}(C) \leftarrow \lfloor \frac{d(C)}{\gamma} \rfloor$ $\langle\langle$*Scale frequency function*$\rangle\rangle$
  For $(X = 0; X \leq \hat{N}; X = X + 1)$
    $Q[0, 0, X] \leftarrow 0$
    $\langle\langle \text{SOL}_{\text{bnd}}[i, P, X]$ *contains a set of concepts achieving* $Q[i, P, X]\rangle\rangle$
    $\text{SOL}_{\text{bnd}}[0, 0, X] \leftarrow \emptyset$
  $\langle\langle$*Dynamic programming (DP) on scaled values;* Q *represents the table of DP*$\rangle\rangle$
  For $(X = 0; X \leq \hat{N}; X = X + 1)$
    For $(P = 1; P \leq P_{\max}; P = P + 1)$
      $Q[0, P, X] \leftarrow \infty$
  For $(i = 1; i \leq n_{\text{rem}}; i = i + 1)$
    For $(X = 0; X \leq \hat{N}; X = X + 1)$
      For $(P = 0; P \leq P_{\max}; P = P + 1)$
        Let $\hat{P}_i \leftarrow \min\{P - \hat{v}(C_i), 0\}$
        If $Q[i-1, P, X - d(C_i)] < Q[i-1, \hat{P}_i, X] + w(C_i)$
          $Q[i, P, X] \leftarrow Q[i - 1, P, X - d(C_i)]$
          $\text{SOL}_{\text{bnd}}[i, P, X] \leftarrow \text{SOL}[i - 1, P, X - d(C_i)]$
        Else
          $Q[i, P, X] \leftarrow Q[i - 1, \hat{P}_i, X] + w(C_i)$
          $\text{SOL}_{\text{bnd}}[i, P, X] \leftarrow \text{SOL}[i - 1, \hat{P}_i, X] \cup C_i$
  $P_N \leftarrow \max\left\{A \mid Q[n_{\text{rem}}, A, \hat{N}] \leq B\right\}$ $\langle\langle$*Max profit with budget* $B\rangle\rangle$
  If $AB(\text{SOL}_{\text{bnd}}[n_{\text{rem}}, P_N, \hat{N}] > AB[\text{SOL}])$ $\langle\langle$*Compare* AB *values*$\rangle\rangle$
    $\text{SOL} \leftarrow \text{SOL}_{\text{bnd}}[n_{\text{rem}}, P_N, \hat{N}]$

Return SOL

Fig. 3. Description of AAM algorithm.

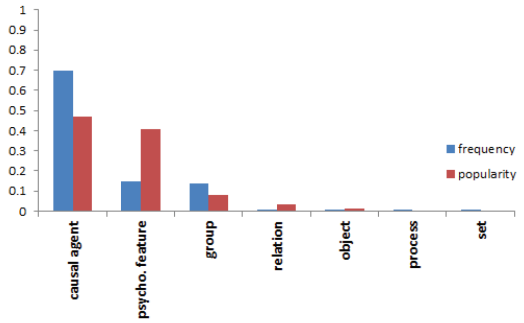

Fig. 4. Popularities (u) and frequencies (d) of concepts in domain M1.

Table II. Original Number of Queries and Queries Remaining After Filtering Out those whose Ranking Quality is Not Improved by Annotating All Concepts in the Domain

	Domain	M1	M2	M3	N1
Number of Queries	Before filtering	1281	888	5820	714
	After filtering	98	187	1737	199
	% Remaining	7.65%	21.06%	29.85%	27.87%
Number of Distinct Queries	Before filtering	928	595	3650	537
	After filtering	98	118	972	138
	% Remaining	10.56%	19.83%	26.63%	25.70%

Wikipedia articles, of which 1,470,661 are annotated. For each domain, we select all documents that contain an annotation of a concept in the domain and, create a dataset for this domain. The datasets for domain M1, M2, M3, and N1 contain 525,703, and 399,792, and 927,848, and 186,952 documents, respectively. Each annotation contains a confidence value that indicates the accuracy of the annotation. We have used the average confidence values over all annotations of a concept to compute its annotation accuracy. The accuracies of annotations are between 0.75-0.95 in domain M1, between 0.8-0.95 in domain M2 and M3, and between 0.8-0.94 in domain N1.

Query Workload. We use a subset of the Bing query log whose target URLs are Wikipedia articles [Demidova et al. 2010]. Each query contains up to six keywords and has one to two relevant answers that are the Wikipedia URL. Because the query log does not list the concept behind each query, we adopt an automatic approach to find those concepts associated with the query. Given a domain, for each query we find that concept from the domain whose instance(s) match the query terms in its relevant answers. We ignore queries that match instances from multiple concepts in their relevant answers, as these queries do not comply with our query model.

The effectiveness of answering some queries may not be improved from semantically annotating the collection [Chu-Carroll et al. 2006; Sanderson 2008]. For instance, all candidate answers for a query may contain matched instances of the same concept. In order to reasonably evaluate our algorithms, we have not considered those queries whose rankings are the same over the unannotated and the fully annotated version (i.e., annotating all concepts in the domain) of the collection. Table II shows the original number of queries and the queries remaining after filtering out the ones whose ranking quality is not improved by annotating all concepts in the domain. Overall, the more concepts in a domain, the greater the fraction of queries whose ranking quality is improved by annotation. Since most concepts in a domain with few concepts, such as M1, contain many entities, all matches to most queries refer to entities from a single

Table III. Examples of Annotated Queries for Domains M1, M2, M3, and N1

Domains	Queries
M1	object:“Rosetta Stone” causal_agent:“Israeli Prime Ministers” causal_agent:“Noah” causal_agent:“Scottie Pippen” group:“Wizard of Oz and Emerald City” group:“Niall Noigiallach” psychological_feature:“Final Fantasy 7 info on the remake” psychological_feature:“The book of Revelation” relation:“NATO” relation:“What is NATO”
M2	agent:“Carson Kit” arrangement:“Tarzan” event:“Final Fantasy 7 info on the remake” event:“Jesus Christ crucifixion” message:“Book of Matthew” message:“Shadow Wikipedia” personality:“Larry King” shape:“Liberty Statue in Paris” written_communication:“The Adventures of Tom Sawyer” written_communication:“The story of Robin Hood”
M3	artifact:“Book of Esther” artifact:“Hebrews” event:“Final Fantasy 7 info on the remake” event:“When was Muhammad born” person:“Hilary Clinton biography” person:“Michael Jordan” person:“Noah” social_group:“way of the master radio” social_relation:Prime Minister of Pakistan” written_communication:“Planet of Apes”
N1	area:“largest city in the USA” area:“New York City” dramatic_conposition:“essays Macbeth blood” dramatic_conposition:“Degrassi” literary_composition:“Gone With The Wind” literary_composition:“Dracula” literary_composition:“Phantom of the opera the book” series:“American Idol” series:“Desperate Housewives” series:“Ultraman”

concept. Therefore, annotating concepts in these domains does not disambiguate many queries. On the other hand, annotating concepts in domains with a relatively large number of concepts, such as M3 and N1, disambiguates many queries and improves their ranking quality. Real-world domains usually contain many concepts. This method leads to collecting 98 (98 unique), 187 (118 unique), 1,737 (972 unique), and 199 (138 unique) queries for domain M1, M2, M3, and N1, respectively. Examples of queries for each domain are shown in Table III.

We use twofold cross-validation to train the u values for concepts in each domain. Because some concepts may not appear in the query workload, we smooth the u values

using the Bayesian m-estimate method with smoothing parameter 1 and uniform priors [Hastie et al. 2009] as

$$\hat{u}(C) = \frac{\hat{P}(C|QW) + mp}{m + \sum_C \hat{P}(C|QW)},$$

where $\hat{P}(C|QW)$ is the probability that C occurs in the query workload and p denotes the prior probability. We set the value of the smoothing parameter, m, to 1 and use uniform prior.

Retrieval System. We index the datasets using Lucene (*lucene.apache.org*) and use BM25 as the underlying retrieval algorithm [Manning et al. 2008]. Given a query, we first rank its candidate answers using BM25. Then, we apply information about the concepts in the query and documents to return those documents whose matching instances have the same concept as the concept of the query, or to filter out the nonrelevant candidate answers for the query if using domain M1, M2, and M3, as explained in Section 4. We performed our experiments on a Linux server with 250GB main memory and two quad core processors. We implemented our retrieval system and optimization algorithms using JAVA 1.7.0_51.

Effectiveness Metrics. Most queries in our query workloads have one relevant answer, and the maximum of number of relevant answers per query in the workload is two. Hence we measure the effectiveness of answering queries over the dataset using precision at 3 (p@3). Since many of our queries have a single relevant answer, we also use *mean reciprocal rank* (MRR), which is $\frac{1}{r}$ where r is the rank of the first relevant answer to the query in the ranked list [Manning et al. 2008]. The value of MRR is larger for ranked lists where the first relevant answer appears at higher positions in the list. We have not defined the annotation-benefit function based on MRR. However, it will provide additional insights on how accurately the annotation-benefit function measures the effectiveness and user satisfaction from conceptual designs and the ranking qualities delivered by our cost-effective conceptual design algorithms. We measure the statistical significance of our results using the paired-t test at significance level of 0.05. The statistically significant improvements are marked in bold in the reported results.

Cost Metrics. We use two types of costs for concept annotation in our experiments. We hypothesize that the cost of running an annotator for a concept may be proportional to its frequency in a collection. It is true that some concepts, such as phone numbers, are both frequent and quick to extract. However, if all other conditions, such as the complexity of the concept extraction program, are the same, then the more frequent concepts are likely to take more computational resources and time for extraction. For instance, one may use a fast and easy-to-build classifier to separate documents about sports from those about science in a large collection, and run extractors for concepts *scientist* and *athlete* only on their relevant sub-collections [Huang and Yu 2010]. Given that the extractors for *scientist* and *athlete* are almost equally complex and that the sub-collection on sports contains more documents than the one on science, it will take more time and computational resources to extract instances of *athlete* than *scientist*. We call this type of cost assignment *frequency-based cost*. We also evaluate our algorithms by assigning randomly generated costs to the concepts in a domain. We call this type of cost assignment *random cost*. We report the average p@3 over 40 sets of random costs for each budget. We use a range of budgets between 0 and 1 with step size of 0.1, where 1 means a sufficient budget to annotate all concepts in a domain.

Table IV. Average p@3 for Oracle, PM, and AM over Domain M1

	Frequency-based Cost			Random Cost		
Budget	Oracle	PM	AM	Oracle	PM	AM
0.1	0.146	0.146	0.146	0.190	0.188	**0.190**
0.2	0.207	0.207	0.207	0.208	0.205	**0.208**
0.3	0.218	0.218	0.218	0.216	0.216	0.216
0.4	0.218	0.218	0.218	0.218	0.218	0.218

Table V. Average MRR for Oracle, PM, and AM over Domain M1

	Frequency-based Cost			Random Cost		
Budget	Oracle	PM	AM	Oracle	PM	AM
0.1	0.271	0.271	0.271	0.442	0.440	**0.442**
0.2	0.491	0.491	0.491	0.513	0.509	**0.513**
0.3	0.551	0.551	0.551	0.543	0.542	0.542
0.4	0.551	0.551	0.551	0.549	0.549	0.549
0.5	0.551	0.551	0.551	0.551	**0.551**	0.550
0.6	0.551	0.551	0.551	0.551	0.551	0.551

7.2. Model Validation

In this section, we investigate whether the annotation-benefit function accurately estimates the likelihood of improvement in effectiveness of answering queries over annotated collections.

7.2.1. Domain with Mutually Exclusive Concepts. We use three algorithms in this set of experiments. Given complete information about the relevant answers of queries, *Oracle* checks all possible designs in a domain whose costs do not exceed a fixed budget, and delivers that design with maximum p@3 or MRR over all queries. Clearly, Oracle cannot be used in a real-world setting as a query interface does not know the relevant answers for the queries at query time. Because the designs returned by Oracle deliver the maximum possible effectiveness for answering queries, we use its results to measure how accurately practical methods predict the amount of improvement in effectiveness of answering queries achieved by a design. *AM* is a brute-force algorithm that picks the design with maximum annotation benefit over a domain given a fixed budget. An intuitively appealing heuristic for finding a design is to select those concepts that are most queried by users. The *PM* algorithm implements this heuristic. PM is a brute-force algorithm that finds the design with the maximum value of $\sum_{C \in \mathcal{S}} u(C)\text{pr}(C)$ over a domain given a fixed budget. Since all these algorithms use exhaustive search methods, running them over a domain with a large number of concepts is not practical. Thus we evaluate these algorithms only over domain M1. In order to precisely evaluate the estimation accuracy of the annotation-benefit function, we assume that AM has the exact values of concept frequencies. We will explain how to estimate the frequencies of concepts without fully annotating them in Section 7.3.

Tables IV and V show the values of p@3 and MRR over domain M1 delivered by AM and PM using frequency-based and random costs. We have omitted the results of Oracle, AM, and PM for budgets from 0.5-0.9 in Table IV and from 0.7-0.9 in Table V, because their results are the same as those for budgets 0.4 and 0.6 in these tables. Since the number of concepts in domain 1 is rather small, there are few feasible solutions that can exhaust the budget, given a modest or large budget. Hence all algorithms find the same or very similar designs for these budgets over domain M1.

The designs returned by Oracle, AM, and PM deliver the same values of p@3 for answering queries over all budgets for frequency-based cost. In this setting, AM and

PM pick the same designs for all budgets between 0.1 and 0.8. The designs selected by AM and PM are different for budget 0.9. Nonetheless, both designs contain 6 out of 7 available concepts in the domain. Answering queries over an annotated collection that contains annotation for all but one of the concepts in the domain will be as effective as will answering queries over the fully annotated collection. Therefore they both achieve the same values of $p@3$ and *MRR*. Further, the cost distribution in the frequency-based cost setting is very skewed in domain M1. Since the number of concepts is rather small in domain M1 and the cost distribution is skewed, there are very few feasible solutions that can maximize the objective functions of either AM or PM given a small budget. For example, with budget equal to 0.2, there are only two feasible designs that exhaust the budget and one of them maximizes the objective functions of both AM and PM.

Tables IV and V show that the designs produced by AM deliver more effective results for queries than the ones generated by PM for budgets 0.1–0.2 using a random cost metric. Since the cost distribution of random costs is not as skewed as that for frequency-based costs, there are more feasible solutions for both objective functions than in the frequency-based cost setting for small budgets. For example, PM and AM include *causal agent* and *psychological feature*, respectively, in their designs for budget equal to 0.1. These designs are not feasible in the frequency-based cost setting. Since *causal agent* is quite frequent in the collection, the matching instances of this concept appear in most of the top candidate answers for queries with this concept. Hence, AM does only slightly worse than PM in returning answers whose matching instances belong to *causal agent* for queries with this concept. Because AM picks *psychological feature* in its design, it is able to effectively answer the queries from this concept. PM, however, does not pick this concept in its design. Because this concept is not very frequent in the collection, the matching instances of most candidate answers for queries with this concept belong to other concepts. Hence the PM design returns considerably less effective results for these queries than the AM design. AM returns the same designs as Oracle for budgets 0.1 and 0.2 in the random cost setting.

As the budget becomes larger, both algorithms pick almost all useful (relatively popular and/or frequent) concepts. Thus, overall, the ranking qualities provided by the designs from these methods are almost the same for larger budgets. The designs generated by Oracle, AM, and PM deliver equal values of $p@3$. Since *MRR* is more sensitive to the position of the top relevant answer than $p@3$, the values of *MRR* for the Oracle designs are different from those for the designs of AM and PM in budget 0.3. In this budget, all methods pick the same designs and deliver the same ranking in most runs. There are some runs where the designs selected by Oracle are different from those of AM and PM. For instance, in one of the runs, Oracle picks *object*, *causal agent*, and *group*, but AM and PM choose *relation causal agent* and *group* as their designs. It seem reasonable to select *relation* because it has a higher popularity and frequency than *object*. It turns out that the concept of the matching entities in many nonrelevant answers, for queries whose entities belong to *psychological feature*, is *object*. Hence, extracting entities of concept *object* helps the query interface to return more effective results for queries of both concepts *object* and *psychological feature*. Documents that contain instances of concept *relation* do not appear as nonrelevant answers of most concepts. And extracting them does not improve the effectiveness of answering queries of other concepts as much. We have made the simplifying assumption that the frequency of a concept in the collection is proportional to the number of its instances in the top nonrelevant answers for queries of other concepts. This observation shows that such assumption might not always hold. We observe a similar situation in budget 0.5. Although *MRR* is not the objective function of the AM algorithm, AM delivers higher values of *MRR* than PM, and close to Oracle overall.

Table VI. Average $p@3$ for Oracle, AM-N, and APM ($\epsilon = 0.001$) over Domain N1

	Frequency-based Cost			Random Cost		
Budget	Oracle	AM-N	APM	Oracle	AM-N	APM
0.1	0.191	0.191	0.191	0.173	**0.173**	0.160
0.2	0.229	0.229	0.229	0.205	**0.205**	0.196
0.3	0.238	0.238	0.238	0.230	0.230	0.230
0.4	0.238	0.238	0.238	0.235	0.235	0.235
0.5	0.238	0.238	0.238	0.237	0.237	0.237
0.6	0.238	0.238	0.238	0.238	0.238	0.238

Table VII. Average MRR for Oracle, AM-N, and APM ($\epsilon = 0.001$) over Domain N1

	Frequency-based Cost			Random Cost		
Budget	Oracle	AM-N	APM	Oracle	AM-N	APM
0.1	0.380	0.380	0.380	0.327	**0.327**	0.298
0.2	0.457	0.457	0.457	0.401	**0.400**	0.377
0.3	0.475	0.475	0.474	0.452	0.451	0.450
0.4	0.475	0.475	0.474	0.465	0.464	0.464
0.5	0.475	0.475	0.474	0.473	0.472	0.472
0.6	0.475	0.475	0.474	0.475	0.475	0.475

7.2.2. Domains without Constraints Regarding the Overlap of Concepts. We use two algorithms in this set of experiments. *Oracle* is the same algorithm used in the validation experiments for domains with mutually exclusive concepts. *AM-N* is a brute-force algorithm that picks that design with maximum annotation benefit over a domain with no constraint given a fixed budget. Since the *PM* heuristic is very similar to *AM-N*, we do not report its results in this section.

Tables VI and VII show the values of $p@3$ and *MRR*, respectively, over domain N1 delivered by Oracle and AM-N using frequency and random cost metrics. All results for budget 0.7–0.9 are omitted as they are the same as the results over budget 0.6. Note that, since the number of concepts in domain N1 is larger than in domain M1, this increases the running time of Oracle. Due to the limited amount of time, we perform the experiment over domain N1 using 20 sets of random costs.

Overall, the ranking quality delivered by AM-N is the same as Oracle. AM-N and Oracle pick the same design for all budgets for frequency-based cost. Intuitively, more popular and accurately annotated concepts should deliver a better ranking quality for domains without any constraint, because query interface can use only the annotations for the concept of each query to improve the ranking quality of its answers. As opposed to the domain with mutually exclusive concepts, the query interface cannot use annotations of concepts other than the concept in the query to filter nonrelevant answers. AM-N generally picks the same designs as Oracle for runs of random cost. AM-N chooses different designs from Oracle in a few runs. For instance, in one of the runs, PM picks *area*, which is slightly more popular than *dramatic composition*, where Oracle picks the latter instead. However, both designs lead to almost the same amount of improvement in ranking qualities for queries. These results confirm our assumption that the annotation-benefit formula is suited as an objective function for domains with no constraints.

7.3. Effectiveness of Approximation Algorithms

7.3.1. Parameters Estimation. In addition to the popularities (u) of concepts in the query workload, AAM requires the value of the frequency (d) for each concept in the collection.

Table VIII. Average p@3 for AAM (with $\epsilon = 0.1$) and APM (with $\epsilon = 0.001$) Using Frequency-Based Costs

	Domain M1		Domain M2		Domain M3	
Domain	APM	AAM	APM	AAM	APM	AAM
0.1	0.146	0.146	0.196	**0.230**	0.145	0.145
0.2	0.177	**0.207**	0.203	**0.237**	0.164	0.165
0.3	0.218	0.218	0.205	**0.239**	0.175	0.176
0.4	0.218	0.218	0.203	**0.241**	0.183	**0.196**
0.5	0.218	0.218	0.237	0.241	0.175	**0.198**
0.6	0.218	0.218	0.239	0.241	0.202	0.202
0.7	0.218	0.218	0.241	0.241	0.202	0.202
0.8	0.211	0.218	0.235	0.241	0.202	0.202
0.9	0.218	0.218	0.241	0.241	0.202	0.202

Table IX. Average MRR for AAM (with $\epsilon = 0.1$) and APM (with $\epsilon = 0.001$) Using Frequency-Based Costs

	Domain M1		Domain M2		Domain M3	
Domain	APM	AAM	APM	AAM	APM	AAM
0.1	0.271	0.271	0.461	**0.585**	0.311	0.310
0.2	0.383	**0.491**	0.521	**0.616**	0.377	0.369
0.3	0.551	0.551	0.549	**0.642**	**0.419**	0.406
0.4	0.551	0.551	0.535	**0.646**	0.445	**0.476**
0.5	0.551	0.551	0.637	0.646	0.430	**0.484**
0.6	0.551	0.551	0.641	0.646	0.503	0.503
0.7	0.551	0.551	0.646	0.646	0.503	0.503
0.8	0.510	**0.551**	0.638	**0.644**	0.503	0.503
0.9	0.551	0.551	0.646	0.646	0.503	0.503

The exact frequency of a concept, however, cannot be determined before annotating all its instances. One may estimate the values of frequencies for the concepts from previous rounds of annotating the collection [Gulhane et al. 2011], or using fast and easy-to-develop classification algorithms [Huang and Yu 2010]. We estimate the frequencies of concepts using a small sample of randomly selected documents from the collection. For each domain, we calculate the frequency of each concept over a random sample of 384 documents from the collection, which corresponds to an estimation error rate of 5% under the 95% confidence level. Similar to computing concepts' popularities, we smoothed the value of d using a Bayesian m-estimate with smoothing parameter of 1 and uniform priors to smooth the estimations, particularly for those concepts with estimated frequencies of 0 [Hastie et al. 2009].

7.3.2. Domains with Mutually Exclusive Concepts. Tables VIII, IX, X, and XI show the values of p@3 and MRR for APM and AAM algorithms for all mutually exclusive domains using frequency-based and random costs, respectively. Generally, the designs generated by AAM improve the values of p@3 and MRR significantly more than the designs produced by APM, over all domains and both types of cost metric. As discussed in Section 4, the optimal design should balance three types of impacts. First, it should contain the most popular concepts so that query interface can return potentially relevant answers to as many queries as possible. Second, if a concept is very frequent, most candidate answers for queries with this concept contain matching instances of this concept. Hence, if the concept is relatively costly, the optimal design should not include these concepts as they may not be worth annotating. Third, it should contain relatively frequent and inexpensive concepts so that the query interface can eliminate

Table X. Average p@3 of AAM (with $\epsilon = 0.3$) and APM (with $\epsilon = 0.001$) Using Random Costs

	Domain M1		Domain M2		Domain M3	
Domain	APM	AAM	APM	AAM	APM	AAM
0.1	0.179	**0.189**	0.221	**0.240**	0.192	**0.202**
0.2	0.201	**0.207**	0.223	**0.240**	0.193	**0.202**
0.3	0.215	0.214	0.226	**0.240**	0.194	**0.202**
0.4	0.218	0.217	0.227	**0.240**	0.195	**0.202**
0.5	0.218	0.218	0.229	**0.241**	0.197	**0.202**
0.6	0.218	0.218	0.231	**0.241**	0.197	**0.202**
0.7	0.218	0.218	0.232	**0.241**	0.198	**0.202**
0.8	0.218	0.218	0.234	**0.241**	0.199	**0.202**
0.9	0.218	0.218	0.237	**0.241**	0.202	0.202

Table XI. Average MRR of AAM (with $\epsilon = 0.3$) and APM (with $\epsilon = 0.001$) Using Random Costs

	Domain M1		Domain M2		Domain M3	
Domain	APM	AAM	APM	AAM	APM	AAM
0.1	0.388	**0.438**	0.517	**0.641**	0.479	**0.502**
0.2	0.471	**0.512**	0.532	**0.643**	0.482	**0.503**
0.3	0.534	**0.541**	0.554	**0.644**	0.486	**0.503**
0.4	0.550	0.549	0.565	**0.645**	0.488	**0.503**
0.5	0.551	0.551	0.568	**0.645**	0.491	**0.503**
0.6	0.551	0.551	0.585	**0.645**	0.493	**0.503**
0.7	0.551	0.551	0.594	**0.645**	0.494	**0.503**
0.8	0.551	0.551	0.609	**0.646**	0.495	**0.503**
0.9	0.551	0.551	0.622	**0.646**	0.503	0.503

many nonxrelevant answers from the list of results for those queries whose concepts are not in the design.

In our experiments, the designs produced by APM have larger overall popularities (u values) than the designs selected by AAM, across all domains and cost metrics. We have observed that, in general, the most popular concepts in users' queries may not be the most frequent ones in the collection. The designs picked by AAM do not normally include the most popular concepts. Instead, they contain a larger number of relatively popular concepts than the designs selected by APM over all domains and cost metrics. Generally, relatively popular concepts are also rather frequent. Since the overall frequencies of the designs produced by AAM are generally larger than those selected by APM, they help the query interface to eliminate more nonrelevant answers from the results of those queries whose concepts are not included in these designs. Because these designs include relatively popular concepts, they also help the query interface to return those relevant answers to a relatively large number of queries.

In a small number of cases, the designs generated by APM deliver a larger p@3 than those produced by AAM. Although the differences between AAM and APM in these cases are not statistically significant, it is interesting to explore the reasons behind these improvements. The designs generated by APM for budgets 0.3 and 0.4 over domain M1, using random costs, deliver larger values of p@3 than the designs of AAM. In both budgets, the relative effectiveness improvement of APM over AAM in each budget is due to a single run where APM selects a design with larger value of annotation benefit than the design picked by AAM. This illustrates the fact that both methods are approximation algorithms and sometimes may return quite different answers from their optimal solutions.

Generally, the differences between ranking quality achieved by the designs of AAM and APM are smaller for larger budgets across all domains and cost metrics. This is mainly due to the fact that AAM and APM can afford to include most of the popular and frequent concepts in their designs for medium or large budgets. Adding the concepts that are rare in the collection or query workload does not considerably improve the effectiveness of answering queries. Because domain M1 has a relatively small number of concepts, both algorithms pick similar designs given a smaller amount of budget for this domain than other domains.

In some cases, the designs generated by APM deliver smaller values of $p@3$ and MRR for larger budgets. For instance, the design for budget 0.8 delivers a smaller value of $p@3$ than the one for budget 0.7 over domain M1 when frequency-based cost is used. Given sufficient budget, APM may replace reasonably popular and frequent concepts with more popular and less frequent concepts. As discussed in Section 4 and the beginning of this section, this may have a negative impact on $p@3$ for answering queries over the annotated collection.

The values of $p@3$ and MRR for the designs generated by AAM over domain M2 and domain M3 when random costs are used are almost the same over all budgets greater than 0.1. The distributions of frequencies and popularities of concepts are very skewed in these domains, where a relatively small number of concepts (e.g., 10 concepts in domain M2) have a large portion of the total frequency and popularity in the domain. Since the costs are assigned randomly, in most runs AAM is able to pick these concepts using a relatively small budget. AAM adds new concepts to this set given larger budgets. The new concepts, however, do not improve the effectiveness of answering queries over the annotated collection.

APM cannot find the set of more popular concepts given a small or moderate budget. The algorithm used in APM has two main steps [Korte and Schrader 1981]. It separates concepts into two sets: popular and unpopular. It then uses a dynamic programming method to find the optimal solution from the set of popular concepts. If there is still some budget left, it greedily picks concepts from the set of unpopular concepts until the budget is exhausted. The decision of how to partition concepts into these two sets is based on an approximation which is not accurate in many cases. Hence, in some cases those concepts that belong to the optimal design may be placed in the set of unpopular concepts. The greedy algorithm used to pick the concepts in the unpopular set sorts them based on the ratio of $\frac{u(C)\times pr(C)}{w(C)}$, where u is the popularity, pr the accuracy, and w the cost of concept C, and then selects the top concepts. This method leaves out some desired concepts that are relatively popular, but expensive.

7.3.3. Domains without Constraint. We investigate the effectiveness of the version of APM algorithm introduced in Section 6.1 for domains without any constraint. Tables VI and VII show the values of $p@3$ and MRR over domain N1 delivered by Oracle, AM-N, and APM using frequency-based and random costs. Overall, the results delivered by APM are the same as those of Oracle and AM-N, except at budgets 0.1 and 0.2 of random cost. APM generally selects the same designs as those of PM. Hence it delivers the same ranking qualities as do the optimal solutions. Since APM is an approximation algorithm, it cannot find solutions that maximize concept popularity in some runs of random cost. For example, APM picks the design consisting of *literary composition*, where as AM-N picks the design with *series* and *dramatic composition* in one of the runs. The overall popularity of the latter design is larger than the former. Thus APM cannot answer queries as effectively as AM-N does. As the costs are quite skewed in frequency-based runs, APM almost always picks the same designs as AM-N in these runs.

Table XII. Average Running Times of AAM and APM (in minutes)

	ϵ	0.5	0.3	0.1	0.01	0.001
AAM	Domain M1	1	2	2	-	-
	Domain M2	1	5	102	-	-
	Domain M3	4	15	128	-	-
APM	Domain M1	1	2	2	2	5
	Domain M2	1	2	2	3	12
	Domain M3	4	14	15	15	23

Table XIII. Average Memory Usage of AAM and APM (in MB)

	ϵ	0.5	0.3	0.1	0.01	0.001
AAM	Domain M1	348	492	635	-	-
	Domain M2	1667	6498	84139	-	-
	Domain M3	1326	5608	63466	-	-
APM	Domain M1	184	184	184	215	1976
	Domain M2	184	184	184	215	4933
	Domain M3	184	184	184	471	7732

7.4. Efficiency and Scalability of Approximation Algorithms

This section studies the efficiency and scalability of our approximation algorithms. We show the efficiency and scalability of AAM and APM using frequency-based costs over domains M1, M2, and M3. Our experiments on the random-based costs show similar results for the scalability of the algorithms. Since the running time and memory consumption of APM are similar over the domains with mutually exclusive concepts and those without any constraints, we report the scalability results for APM only over those domains with mutually exclusive concepts.

7.4.1. Efficiency. Table XII shows the average running time of APM and AAM algorithms over domains M1, M2, and M3 with budgets 0.1 to 0.9 using values between 0.5 - 0.001 for ϵ. As we expect, the smaller the value of ϵ, the longer the running times of both algorithms. APM is generally more efficient than AAM, particularly for smaller values of ϵ. This observation confirms our comparative analysis of the time complexities of these algorithms in Section 6. We set the value of ϵ to 0.1 for AAM and to 0.001 for APM in our experiments to evaluate the improvement in effectiveness of answering queries achieved by the designs produced by AAM and APM for frequency-based cost as reported in Section 7.3, and we set the value of ϵ to 0.1 and 0.001, respectively. According to Table XII, the running times of the algorithms for these values of ϵ are reasonable for a design-time task. As we have to run AAM 40 times per budget in the experiments using random costs, and reported in Section 7.3, we set the value of ϵ to 0.3 in AAM for these experiments. Table XII indicates that the running time of AAM with this value of ϵ is reasonable for a design-time task.

Both APM and AAM use a dynamic programming approach and keep a table in the main memory to maintain the solutions of their subproblems. Table XIII shows the average memory usage of APM and AAM algorithms over domains M1, M2, and M3 using values from 0.5-0.001 for ϵ. Similar to running time, the smaller the value of ϵ, the larger the memory AAM and APM need. Interestingly, AAM uses smaller amount of memory over domains M3 than M2, even though the size of M2 is smaller than M3. We have found that the distributions of costs and frequencies of concepts in domain M3 are more skewed than those of domain M2. Thus the size of $\mathcal{C}_{rem}$ for domain M3 tends to be smaller than the one for domain M2. Hence the amount of memory space required

Table XIV. Average $p@3$ over All Budgets for AAM and APM Using Different Values of ϵ

	ϵ	0.5	0.3	0.1	0.01	0.001
AAM	Domain M1	0.209	0.209	0.209	-	-
	Domain M2	0.229	0.238	0.239	-	-
	Domain M3	0.188	0.188	0.188	-	-
APM	Domain M1	0.204	0.205	0.204	0.204	0.204
	Domain M2	0.233	0.233	0.229	0.222	0.222
	Domain M3	0.184	0.183	0.182	0.184	0.184

Table XV. Average MRR over All Budgets for AAM and APM Using Different Values of ϵ

	ϵ	0.5	0.3	0.1	0.01	0.001
AAM	Domain M1	0.325	0.501	0.513	-	-
	Domain M2	0.636	0.635	0.635	-	-
	Domain M3	0.451	0.449	0.451	-	-
APM	Domain M1	0.497	0.501	0.497	0.497	0.497
	Domain M2	0.595	0.595	0.584	0.586	0.586
	Domain M3	0.445	0.444	0.439	0.444	0.444

to construct the dynamic programming table for AAM in domain M3 is smaller than the one for M2. The size of the main-memory table becomes very large (e.g., for some budgets it exceeds the available main memory) for $\epsilon \leq 0.01$ in AAM and for $\epsilon \leq 0.001$ in APM. Our results in Section 7.3 indicate that one does not need such small values for ϵ, particularly for AAM, in order to find effective designs. Hence in this article we have not used such values for *epsilon* for APM and AAM.

7.4.2. Scalability. One may have to set ϵ to values larger than 0.3 or 0.1 for AAM and 0.001 for APM in order to find the desired designs for large domains in a reasonable amount of time and using modest memory overheads. Hence, we empirically examine the effect of changes on the values of ϵ for the effectiveness of the algorithms. Tables XIV and XV show the average values of $p@3$ and MRR for APM and AAM algorithms over domains M1, M2, and M3 using values between 0.5 and 0.001 for ϵ. The average values of $p@3$ and MRR delivered by the designs of AAM are relatively stable across different values of ϵ in domains M1, M2, and M3. Except for some cases, such as $\epsilon = 0.5$ in domain M3, generally, the ranking qualities delivered by the designs of AAM and APM tend to improve when using smaller value of ϵ.

Furthermore, Table XVI indicates that AAM with relatively small values of ϵ, that is, 0.5 and 0.3, generally provides better ranking qualities than APM with considerably smaller values of ϵ, and a comparable ranking quality to AAM using $\epsilon = 0.1$. With this choice of ϵ, AAM requires significantly less amount of resources than the ideal value of ϵ, or that of APM with $\epsilon = 0.001$, while sustaining its effectiveness.

8. CONCLUSIONS AND FUTURE WORK

Extracting and annotating the occurrences of entities in an unstructured or semi-structured text collection by their concepts improves the effectiveness of answering queries over the collection. Nonetheless, annotating the occurrences of a concept and maintaining the annotated collection are resource intensive. Thus, an enterprise may have to select a subset of the concepts for annotation, called a *conceptual design*, whose cost of extraction does not exceed its budget and improves the effectiveness of answering queries most. To surpass the intuition-based approaches to conceptual design, we introduced and formalized this problem and proved it NP-hard in the number of

Table XVI. Average Ranking for AAM with $\epsilon = 0.5, 0.3, 0.1$ and APM with $\epsilon = 0.001$ Using Frequency-Based Cost

		p@3				MRR			
		AAM				AAM			
Domain	Budget	0.5	0.3	0.1	APM	0.5	0.3	0.1	APM
M1	0.1	0.146	0.146	0.146	0.146	0.271	0.271	0.271	0.271
	0.2	**0.207**	**0.207**	**0.207**	0.177	**0.491**	**0.491**	**0.491**	0.383
	0.3	0.218	0.218	0.218	0.218	0.551	0.551	0.551	0.551
	0.4	0.218	0.218	0.218	0.218	0.551	0.551	0.551	0.551
	0.5	0.218	0.218	0.218	0.218	0.551	0.551	0.551	0.551
	0.6	0.218	0.218	0.218	0.218	0.551	0.551	0.551	0.551
	0.7	0.218	0.218	0.218	0.218	0.551	0.551	0.551	0.551
	0.8	0.218	0.218	0.218	0.211	**0.551**	**0.551**	**0.551**	0.510
	0.9	0.218	0.218	0.218	0.218	0.551	0.551	0.551	0.551
M2	0.1	**0.232**	**0.230**	**0.230**	0.196	**0.590**	**0.585**	**0.585**	0.461
	0.2	**0.233**	**0.233**	**0.237**	0.203	**0.616**	**0.614**	**0.616**	0.521
	0.3	**0.239**	**0.239**	**0.239**	0.205	**0.642**	**0.642**	**0.642**	0.549
	0.4	**0.241**	**0.241**	**0.241**	0.203	**0.646**	**0.646**	**0.646**	0.535
	0.5	0.241	0.241	0.241	0.237	0.646	0.646	0.646	0.637
	0.6	0.241	0.241	0.241	0.239	0.646	0.646	0.646	0.641
	0.7	0.241	0.241	0.241	0.241	0.646	0.646	0.646	0.646
	0.8	0.241	0.241	0.241	0.235	**0.646**	**0.646**	**0.644**	0.638
	0.9	0.241	0.241	0.241	0.241	0.646	0.646	0.646	0.646
M3	0.1	0.147	0.143	0.145	0.145	0.302	0.292	0.310	0.311
	0.2	0.165	0.165	0.165	0.164	0.369	0.369	0.369	0.377
	0.3	**0.179**	**0.179**	0.176	0.175	0.409	0.409	0.406	**0.419**
	0.4	**0.196**	**0.196**	**0.196**	0.183	**0.476**	**0.476**	**0.476**	0.445
	0.5	**0.198**	**0.197**	**0.198**	0.175	**0.493**	**0.484**	**0.484**	0.430
	0.6	0.202	0.202	0.202	0.202	0.500	0.503	0.503	0.503
	0.7	0.202	0.202	0.202	0.202	0.502	0.503	0.503	0.503
	0.8	0.202	0.202	0.202	0.202	0.503	0.503	0.503	0.503
	0.9	0.202	0.202	0.202	0.202	0.503	0.503	0.503	0.503

relevant concepts in the general case. We proposed two efficient approximation algorithms for it: *approximate popularity maximization* (APM) and *approximate annotation-benefit maximization* (AAM). We proved that if concepts are mutually exclusive, APM has a constant factor approximation ratio and AAM is a fully polynomial-time approximation scheme. If there is not any constraint regrading the overlap of concepts, APM is a fully polynomial-time approximation scheme. Our empirical studies over real-world datasets, concepts, and query workloads showed that APM and AAM efficiently compute conceptual designs and return effective ones over real-world concepts, with AAM delivering more effective results over domains with mutually exclusive concepts.

We plan to extend this work in multiple directions. First, the information needs behind some queries may be more complex than finding information about a single entity. For example, a user may ask about the relationships between multiple entities. We plan to extend our model to consider the dependencies between occurrences of multiple concepts in the input queries and the collection. For instance, assume that the references to concept *job* appear mostly in queries that also refer to instances of concept *person*. Given limited resources, it may be worth extracting only *person*, instead of both *person*, and *job*, as the extracted instances of *person* may also improve the effectiveness of answering queries about instances of *job*. Similarly, because some concepts

often occur together in documents, extracting one of them may make annotating the rest less costly. For instance, if concepts *person* and *job* appear together in a collection quite frequently, it will be less time consuming to develop and/or run the extractor of *job*, given that the instances of *person* are already annotated in the collection. We plan to represent this problem using a collection of precedence-constrained knapsack problems [Johnson and Niemi 1983] and to leverage the approximation algorithms for the precedence-constrained knapsack problem to solve this extension of the cost-effective design problem. We also plan to improve estimating the frequencies of concepts using relatively inexpensive concept extraction programs, for example, simple classifiers that do not use deep parsing, that can deliver a reasonable estimation of the ratio of those documents that may contain instances of certain concepts, in a rather short amount of time [Huang and Yu 2010].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments and Wolfgang Nejdl and Elena Demidova for providing the query workload.

REFERENCES

Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. 2011. *Web Data Management*. Cambridge University Press.

Eugene Agichtein and Luis Gravano. 2003. Querying text databases for efficient information extraction. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'03)*.

Michael Anderson, Dolan Antenucci, Victor Bittorf, Matthew Burgess, Michael Cafarella, Arun Kumar, Feng Niu, Yongjoo Park, Christopher Re, and Ce Zhang. 2013. Brainwash: A data system for feature engineering. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR'13)*.

Paul N. Bennett, Krysta Svore, and Susan T. Dumais. 2007. Classification-enhanced ranking. In *Proceedings of the International Conference on World Wide Web (WWW'07)*.

Barry Boehm, Chris Abts, and Sunita Chulan. 2000. Software development cost estimation approaches, a survey. *Ann. Softw. Engin.* 10, 177–205.

Michael Cafarella, Dan Suciu, and Oren Etzioni. 2007. Navigating extracted data with schema discovery. In *Proceedings of the International Conference on World Wide Web (WWW'07)*.

Soumen Chakrabarti, Kriti Puniyani, and Sujatha Das. 2007. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *Proceedings of the International Conference on World Wide Web (WWW'07)*.

Soumen Chakrabarti, Sunita Sarawagi, and S. Sudarshan. 2010. Enhancing search with structure. *IEEE Data Engin. Bull.* 33, 1.

Fei Chen, Xixuan Feng, Chris Re, and Min Wang. 2012. Optimizing statistical information extraction programs over evolving text. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'12)*.

Laura Chiticariu, Yunyao Li, Sriram Raghavan, and Frederick Reiss. 2010. Enterprise information extraction: Recent developments and open challenges. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'10)*.

Jennifer Chu-Carroll and John Prager. 2007. An experimental study of the impact of information extraction accuracy on semantic search performance. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'07)*.

Jennifer Chu-Carroll, John Prager, Krzysztof Czuba, David Ferrucci, and Pablo Duboue. 2006. Semantic search via XML fragments: A high-precision approach to IR. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*.

Nilesh Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, Sathiya Keerthi, and Srujana Merugu. 2009. A Web of concepts. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'09)*.

Elena Demidova, Xuan Zhou, Irina Oelze, and Wolfgang Nejdl. 2010. Evaluating evidences for keyword query disambiguation in entity centric database search. In *Proceedings of the International Workshop on Database and Expert Systems Application (DEXA'10)*.

Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John Tomlin, and Jason Zien. 2003. SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation. In *Proceedings of the International Conference on World Wide Web (WWW'03)*.

Anhai Doan, Jeff Naughton, Akanksha Baid, Xiaoyang Chai, Fei Chen, et al. 2009. The case for a structured approach to managing unstructured data. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR'09)*.

Xin Luna Dong, Barna Saha, and Divesh Srivastava. 2013. Less is more: Selecting sources wisely for integration. *Proc. VLDB Endow.* 6, 2.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2006. A probabilistic model of redundancy in information extraction. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI'06)*.

Ofer Egozi, Shaul Markovitch, and Evgeniy Gabrilovich. 2011. Concept-based information retrieval using explicit semantic analysis. *ACM Trans. Inf. Syst.* 29, 2, 1–34.

Amr Elhelw, Mina Farid, and Ihab Ilyas. 2012. Just-in-time information extraction using extraction views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*.

Ronald Fagin, Benny Kimelfeld, Yunyao Li, Sriram Raghavan, and Shivakumar Vaithyanathan. 2010. Understanding queries in a search database system. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'10)*.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In *Proceedings of the CSLDAMT Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (CSLDAMT-NAACL HLT'10)*.

Arnaud Freville. 2004. The multidimensional 0-1 knapsack problem: An overview. *Euro. J. Oper. Res.* 155, 1–21.

Shmuel Gal and Boris Klots. 1995. Optimal partitioning which maximizes the sum of weighted averages. *Oper. Res.* 43, 3.

Hector Garciamolina, Jeff Ullman, and Jennifer Widom. 2008. *Database Systems: The Complete Book*. Prentice Hall.

Jens Graupmann, Michael Biwer, Christian Zimmer, Patrick Zimmer, Matthias Bender, Martin Theobald, and Gerhard Weikum. 2004. COMPASS: A concept-based Web search engine for HTML, XML, and Deep Web data. In *Proceedings of the Conference on Very Large Databases (VLDB'04)*.

Jens Graupmann, Ralf Schenkel, and Gerhard Weikum. 2005. The SphereSearch engine for unified ranked retrieval of heterogeneous XML and Web documents. In *Proceedings of the Conference on Very Large Databases (VLDB'05)*.

Pankaj Gulhane, Amit Madaan, Rupesh Mehta, Jeyashankher Ramamirtham, Rajeev Rastogi, Sandeep Satpal, Srinivasan H. Sengamedu, Ashwin Tengli, and Charu Tiwari. 2011. Web-scale information extraction with vertex. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'11)*.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer.

Jian Huang and Cong Yu. 2010. Prioritization of domain-specific Web information extraction. *Oper. Res.* 43, 500–508.

Oscar Ibarra and Chul Kim. 1975. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM* 22, 4, 463–468.

Panagiotis Ipeirotis, Eugene Agichtein, Pranay Jain, and Luis Gravano. 2006. To search or to crawl? Towards a query optimizer for textcentric tasks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'06)*.

Alpa Jain, Anhai Doan, and Luis Gravano. 2008a. Optimizing SQL queries over text databases. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'08)*.

Alpa Jain, Panagiotis Ipeirotis, and Luis Gravano. 2008b. Building query optimizers for information extraction: The SQoUT project. http://www.cs.columbia.edu/~gravano/Papers/2008/sigmod-record08.pdf.

David S. Johnson and K. Niemi. 1983. On knapsacks, partitions, and a new dynamic programming technique for trees. *Math. Oper. Res.* 8, 1–14.

Pallika Kanani and Andrew Mccallum. 2012. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'12)*.

Eser Kandogan, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2006. Avatar semantic search: A database approach to information retrieval. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'06)*.

Bernhard Korte and Rainer Schrader. 1981. On the existence of fast approximation schemes. In *Nonlinear Programming*, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds. Academic Press, New York, 41–437.

Marek Kowalkiewicz, Tomasz Kaczmarek, and Witold Abramowicz. 2006. Myportal: Robust extraction and aggregation of Web content. In *Proceedings of the Conference on Very Large Databases (VLDB'06)*.

Robert Krovetz and W. Bruce Croft. 1992. Lexical ambiguity and information retrieval. *ACM Trans. Inf. Syst.* 10, 115–141.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *An Introduction to Information Retrieval.* Cambridge University Press.

Andrew Mccallum. 2005. Information extraction: Distilling structured data from unstructured text. *ACM Queue* 3, 9, 48–57.

Jeffrey Pound, Ihab Ilyas, and Grant Weddell. 2010. Expressive and flexible access to Web-extracted data: A keyword-based structured query language. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'10)*.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'99)*.

Mark Sanderson. 2008. Ambiguous queries: Test collections need more sense. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*.

Sunita Sarawagi. 2008. Information extraction. *Foundat. Trends Databases* 1, 3, 261–377.

Ralf Schenkel, Fabian Suchanek, and Gjergji Kasneci. 2007. YAWN: A semantically annotated Wikipedia XML corpus. In *Proceedings of the Symposium on Database Systems for Business, Technology and Web (BTW'07)*.

Warren Shen, Pedro Derose, Robert Mccann, Anhai Doan, and Raghu Ramakrishnan. 2008. Toward best-effort information extraction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*.

Warren Shen, Anhai Doan, Jeff Naughton, and Raghu Ramakrishnan. 2007. Declarative information extraction using Datalog with embedded extraction predicates. In *Proceedings of the Conference on Very Large Databases (VLDB'07)*.

Christopher Stokoe, Michael P. Oakes, and John Tait. 2003. Word sense disambiguation in information retrieval revisited. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*.

Arash Termehchy, Ali Vakilian, Yodsawalai Chodpathumwan, and Marianne Winslett. 2014. Which concepts are worth extracting? In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'14)*.

Roelof Van Zwol and Tim Van Loosbroek. 2007. Effective use of semantic structure in XML retrieval. In *Proceedings of the European Conference on IR Research (ECIR'07)*.

Received June 2014; revised November 2014; accepted December 2014