Entity-Relationship Queries over Wikipedia

XIAONAN LI and CHENGKAI LI, University of Texas at Arlington
CONG YU, Google Research

Wikipedia is the largest user-generated knowledge base. We propose a structured query mechanism, *entity-relationship query*, for searching entities in the Wikipedia corpus by their properties and interrelationships. An entity-relationship query consists of multiple predicates on desired entities. The semantics of each predicate is specified with keywords. Entity-relationship query searches entities directly over text instead of preextracted structured data stores. This characteristic brings two benefits: (1) Query semantics can be intuitively expressed by keywords; (2) It only requires rudimentary entity annotation, which is simpler than explicitly extracting and reasoning about complex semantic information before query-time. We present a ranking framework for general entity-relationship queries and a position-based Bounded Cumulative Model (BCM) for accurate ranking of query answers. We also explore various weighting schemes for further improving the accuracy of BCM. We test our ideas on a 2008 version of Wikipedia using a collection of 45 queries pooled from INEX entity ranking track and our own crafted queries. Experiments show that the ranking and weighting schemes are both effective, particularly on multipredicate queries.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*

General Terms: Design, Languages, Performance

Additional Key Words and Phrases: Entity search and ranking, structured entity query, Wikipedia

ACM Reference Format:
Li, X., Li, C., and Yu, C. 2012. Entity-relationship queries over Wikipedia. ACM Trans. Intell. Syst. Technol. 3, 4, Article 70 (September 2012), 20 pages.
DOI = 10.1145/2337542.2337555 http://doi.acm.org/10.1145/2337542.2337555

1. INTRODUCTION

Since its inception in January 2001, Wikipedia has become the largest encyclopedia ever created, containing more than 3 million articles in English alone as of 2010. In the meantime, Wikipedia articles have amazingly evolved, from mostly plain texts at an earlier stage to current ones with substantial structural annotations. It is now the primary knowledge source for many users on a wide variety of *entities*, including people, institutions, geographical locations, events, etc. For discovering and exploring the entities that fascinate them, users are in need of structured querying facilities, coupled with text retrieval capabilities, that explicitly deal with the entities, their properties, and relationships.

This material is based on work partially supported by NSF Grant IIS-1018865. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of NSF.

Authors' addresses: X. Li, Department of Computer Science and Engineering, University of Texas at Arlington; email: xiaonan.li@mavs.uta.edu; C. Li, Department of Computer Science and Engineering, University of Texas at Arlington; C. Yu, Google Research.

DOI 10.1145/2337542.2337555 http://doi.acm.org/10.1145/2337542.2337555

The prevalent manner in which users access Wikipedia is still keyword-based document search. Although keyword search has been quite effective in finding specific pages matching the keywords, there clearly exists a mismatch between its document-centric view and the aforementioned entity-centric user information needs. Users' tasks often cannot be clearly expressed with simple keyword queries and processing the query results may require substantial user efforts.

Example 1 (*Motivating Example*). Consider a business analyst investigating the development of Silicon Valley. Particularly, she is interested in this task: Find the list of *companies* and their *founders*, where the companies are in Silicon Valley and the founders are Stanford graduates.

There are two major mismatches that make keyword search unsuitable for resolving this task. First, the task focuses on *typed* entities, PERSON and COMPANY, and, in database terminology, their "join" relationships. Second, the task involves synthesizing information scattered across different places, therefore a simple list of pages is not sufficient. For instance, one page may tell the analyst that JERRY YANG is a founder of YAHOO!, but whether YAHOO! is a Silicon Valley company and whether JERRY YANG is a Stanford graduate may have to be found in other pages.

While conceptually simple, with only keyword search, tasks like the given one require substantial user efforts to perform multiple searches and assemble information from a potentially large number of articles. Our analyst may start with a search on "Silicon Valley company" and scan through the potentially long list of result articles to, hopefully, fetch a list of companies that are likely to be in Silicon Valley. She then similarly issues another search on "Stanford graduate" to find a list of people graduated from Stanford University. She then manually combine entities in these two lists and, by multiple additional searches, check if a company was founded by a person, for each pair of person and company. Alternatively, she can also go through the list of companies and, for each company, find its founders and check if Stanford is their alma mater by multiple search queries. Both are painful options and require the user to break down the task into a time-consuming, error-prone iterative procedure of searching, reading, and re-searching.

Wikipedia (and the Web) contains various tables and lists, which can be extracted into databases for powerful queries [Cafarella et al. 2008]. For example, a page listing all Silicon Valley companies may exist. However, it is unrealistic to expect such pages exist for arbitrary user tasks. Moreover, it is less common to find such tables/lists for relationships between entities, for instance, who founded which company.

We propose *entity-relationship query*,[1] a declarative query mechanism for the aforementioned task. The results of such queries are tuples of entities that are likely to meet the semantic requirements of the query, instead of articles containing such entities. For Example 1, our analyst can write the following SQL-like query.

Query 1 (*Entity-Relationship Query for Example 1*).

```
SELECT x, y
FROM   PERSON x, COMPANY y
WHERE  x:["Stanford", "graduate"]  // Predicate p1
  AND  y:["Silicon Valley"]        // Predicate p2
  AND  x,y:["found"]               // Predicate p3
```

We take a DB-IR integration approach in proposing this direction. On the one hand, entity-relationship queries have explicit structured components: *typed entity variables*

[1] It should not be confused with the well-known ER model.

(e.g., x, bound to entities of type PERSON, and y, for entities of type COMPANY), *selection predicates* for selecting entities by their properties (e.g., predicate p_1), and *relation predicates* for specifying relations between entities (e.g., predicate p_3 for "x founded y"). On the other hand, the individual predicates are specified by keyword-based constraints. The query semantics dictates that entities satisfy a predicate by a simple and intuitive requirement: the entities cooccur with the keywords in some contexts. Such contexts can be sentences, windows of texts, etc. For simplicity, we use sentence as context in this article. For example, predicate p_1 requires every PERSON in the query answer to cooccur with "Stanford" and "graduate" in at least one sentence. In short, we aim to capture entity properties and relationships through shallow syntax requirements implied by users at query-time,[2] while previous works [Agichtein and Gravano 2000; Brin 1998; Cafarella et al. 2007; Chu et al. 2007; DeRose et al. 2007; Etzioni et al. 2008; Kandogan et al. 2006] explicitly extract and reason about semantic information before query-time. Although such syntax clue is by no means rigorous or error-proof, it becomes robust when we take into account the redundancy in a corpus: true facts are more likely repetitively stated in multiple places. This intuition has been widely used in Web search and mining, for instance, information extraction [Agichtein and Gravano 2000; Brin 1998] and entity search and ranking [Cheng et al. 2007].

Entity-relationship queries can yield many false answers due to abundant accidental cooccurrences (e.g., false evidence such as "X's partner is a Stanford graduate" for predicate p_1). Therefore, how to rank query answers presents a critical challenge. First, the presence of multiple predicates in a query requires us to aggregate the rankings of entities for multiple predicates. Second, many true answers have small numbers of cooccurrence contexts (i.e., low redundancy) in Wikipedia. Using redundancy solely is not sufficient to tell such entities apart from false answers.

We present a ranking framework for general entity-relationship queries. It first evaluates how well an answer satisfies individual predicates and then aggregates multiple predicate scores into an answer score. A Bounded Cumulative Model (BCM) is proposed for scoring predicates. BCM relies on redundant cooccurrence contexts for robust evaluation. To improve ranking accuracy for answers with small numbers of supporting contexts, BCM performs refined assessment on each cooccurrence context based on three positional features: *proximity*, *ordering pattern*, and *mutual exclusion*. Contexts that are likely true evidence are given higher importance. Existing systems only exploit proximity feature [Chakrabarti et al. 2006; Cheng et al. 2007] and may not leverage redundancy [Chakrabarti et al. 2006]. In certain sensitive cases, BCM tends to yield sketically high scores. To address this issue, we further study two weighting schemes for BCM. The basic idea is to detect and penalize susceptible answers based on the number of supporting contexts.

In summary, we make the following contributions in this article.

- We propose the concept of entity-relationship queries for structured querying of entities directly over Wikipedia text with multiple predicates.
- We design a ranking framework and a position-based Bounded Cumulative Model for ranking the answers to entity-relationship queries.
- We develop two weighting schemes, maximal-support weighting and document-frequency weighting, for improving the accuracy of BCM.
- We conduct comprehensive experiments, and demonstrate the effectiveness of the ranking and weighting methods.

[2] The effectiveness of such entity-relationship queries partially relies on the user's capability in forming proper keyword constraints, like in IR queries.

The article is organized as follows. Section 2 reviews related work. Section 3.1 defines entity-relationship query and its ranking problem. In Section 3.2, we discuss the three position-based features in our Cumulative Model (Section 3.3) and Bounded Cumulative Model (Section 3.4). Section 4 explores the weighting schemes for improving BCM's performance. Section 5 reports empirical results. Section 6 discusses limitations and future work. Section 7 concludes the article.

2. RELATED WORK

Previous studies on structured querying of the Web focus on DB-based approach that explicitly extracts structured information into databases [Agichtein and Gravano 2000; Brin 1998; Cafarella et al. 2007; Chu et al. 2007; DeRose et al. 2007; Etzioni et al. 2008; Kandogan et al. 2006]. This approach lends itself to the rich techniques of database querying. It is constrained by the capability of the information extraction (IE) and natural language processing (NLP) techniques. Particularly, it requires explicit identification of the "names" of entity relationships. For example, if a "found" relation between JERRY YANG and YAHOO! was not detected during the extraction phase, such information is lost and could not be queried.

Some systems [Auer et al. 2007; Dill et al. 2003; Kasneci et al. 2008; Suchanek et al. 2007] explicitly encode entities and their relations in RDF, the W3C recommendation of data model for Semantic Web. They can thus leverage the expressiveness of query languages like SPARQL.[3] Some of them [Auer et al. 2007; Kasneci et al. 2008; Suchanek et al. 2007] only capture structured and semistructured information, for instance, infoboxes in Wikipedia, leaving out implicit information in unstructured text. For example, YAGO only supports around 100 relations unified from WordNet and Wikipedia [Suchanek 2009]. Other systems apply IE techniques over Web pages to bootstrap RDF extraction [Dill et al. 2003], thus bearing the same limitation as the aforementioned DB-based approach.

Question answering has been an important task in TREC conferences.[4] The basic factoid QA task expects systems to provide a short answer (typically less than 250 characters) to a factoid question such as Who invented the paper clip? [Voorhees 2003]. The list QA task instead asks for a list of answers to a question. Evolved from the QA tasks, entity ranking tasks of both INEX [5] and TREC have gained significant interest [Demartini et al. 2008; Petkova and Croft 2007; Vercoustre et al. 2008; Zaragoza et al. 2007]. These tasks are highly relevant to our single-predicate queries. However, they do not have the concept of "predicates" and do not deal with multiple predicates. Answering complex and multipredicate queries is what distinguishes our work from previous studies. Moreover, an important focus in question answering and entity ranking is to accurately understand the narrative question descriptions, which we do not study.

The studies most related to ours are [Chakrabarti et al. 2006; Cheng et al. 2007; Zhou et al. 2010]. Chakrabarti et al. [2006] learns an optimal scoring function on proximity feature. It only scores entities by single context, without integrating information found in multiple documents. EntityRank [Cheng et al. 2007] aggregates scores of locally evaluated cooccurrence contexts into global scores to improve ranking. [Zhou et al. 2010] extends EntityRank with more context matching patterns. All three systems only focus on queries comparable to our single-predicate queries and thus do not study multipredicate queries.

The concept of entity-relationship query and its query result ranking problem were initially studied in Li et al. [2010a]. In this extension, we propose two weighting

[3] http://www.w3.org/TR/rdf-sparql-query

[4] http://ilps.science.uva.nl/trec-entity/guidelines/

[5] http://www.inex.otago.ac.nz/tracks/entity-ranking/entity-ranking.asp

Table I. Notations

$q=\langle V, P\rangle$	an entity-relationship query
V	a set of entity variables
P	a set of predicates
$p=\langle V_p, C_p\rangle$	a predicate
V_p	a subset of V (the entity variables relevant to p)
C_p	a set of phrases
$t=\langle e_1, ..., e_{\|V\|}\rangle$	a tuple, where each e_i is an entity instantiated from entity variable V_i
t_p	the sub-tuple of t with regard to p
s	a context (sentence)
$\langle doc, sent, \hat{V}_p, \hat{C}_p\rangle$	a cooccurrence context of some answer tuple t for predicate p
$\hat{V}_p$	the positions of entities in t_p, in the context.
$\hat{C}_p$	the positions of phrases in C_p, in the context.
$\phi_p(t)$	the set of all contexts of t_p
ϕ_p	the set of all contexts of answer tuples to predicate p
ϕ	the set of all contexts of all predicates
$prox_p(t, s)$	the proximity of t_p in s
o	an ordering pattern
O_p	all possible ordering patterns for predicate p
$\phi_p(o)$	the set of contexts following pattern o for predicate p
$f_p(o)$	the weight of pattern o with regard to predicate p
$O_p(s)$	colliding patterns in s
$credit_p(o, s)$	the credit of o in s with regard to predicate p
$F_p(t)$	single-predicate score of an answer tuple t
$F^A(t)$	overall score of t
$\phi_p(t, o)$	all contexts of t for predicate p that follow pattern o

schemes for improving the BCM-based ranking model (Section 4). We also experimentally evaluate the effectiveness of the weighting schemes in Section 5.5. In addition, we have reprocessed our Wikipedia dataset with an updated version of preprocessing module, which fixed some parsing errors in previous version. As a result, Section 5 now reports our findings on the updated data.

3. POSITION-BASED RANKING

3.1. Preliminaries

To facilitate our discussion, Table I summarizes the major notations that we use. We formalize an *entity-relationship query* as $q=\langle V, P\rangle$. V is a set of entity variables. Each $v \in V$ is bound to entities of certain type, for instance, PERSON. P is a set of predicates. Each $p \in P$ is a pair $\langle V_p, C_p\rangle$, where $V_p \subseteq V$ and C_p is a set of phrases[6]. Query 1 is thus $q_1=\langle V, P\rangle$, $V=\langle x$:PERSON, y:COMPANY$\rangle$, and $P=\{p_1, p_2, p_3\}$. The predicates $p_1=\langle V_{p_1}=\{x\}, C_{p_1}=\{$"Stanford","graduate"$\}\rangle$ and $p_2=\langle V_{p_2}=\{y\}, C_{p_2}=\{$"Silicon Valley"$\}\rangle$ are selection predicates, and $p_3=\langle V_{p_3}=\{x, y\}, C_{p_3}=\{$"found"$\}\rangle$ is a relation predicate. Note that although Query 1 only involves binary relationship between two entities, a query in general can have multientity relationships. For instance, a query `SELECT p,c,l FROM PERSON p, COMPANY c, PROGRAMMING_LANGUAGE l WHERE p,c,l:["design"]` can be used to find programming languages designed by scientists at some companies.

[6] A single keyword is treated as a phrase of length 1.

An *answer* to a query q is a tuple of entities, denoted by t. For each $v \in V$, there is a corresponding entity $e \in t$ instantiated from v, for instance, t=⟨JERRY YANG, YAHOO!⟩ for Query 1. Given a predicate $p=\langle V_p, C_p \rangle$, we use t_p to represent the subtuple of t such that each entity $e \in t_p$ is instantiated from a corresponding $v \in V_p$. Take p_1 in Query 1, for example. t_{p_1}=⟨JERRY YANG⟩ because V_{p_1} has only one variable x and JERRY YANG is instantiated from x. Similarly, $t_{p_3}=t$.

Given a predicate p, if a sentence contains all the phrases in C_p and one entity for each variable in V_p, it is a (cooccurrence) context for p. These entities in whole are said to satisfy p. Suppose these three sentences are found in the corpus.

s$_1$: Stanford University graduates JERRY YANG and ...
s$_2$: ...a senior manager at YAHOO! in Silicon Valley.
s$_3$: JERRY YANG cofounded YAHOO!.

JERRY YANG satisfies p_1 by sentence s_1; YAHOO! satisfies p_2 by sentence s_2; and they together satisfy p_3 by s_3. Assembling the information together, the entity tuple ⟨JERRY YANG, YAHOO!⟩ is an answer to the query since it satisfies all the query predicates. Note that in s_1, "Stanford University" is treated as plain text since it is neither a PERSON nor a COMPANY.

A cooccurrence context of answer t for predicate $p=\langle V_p,\ C_p\rangle$ is a quadruple $\langle doc, sent, \hat{V}_p, \hat{C}_p \rangle$. *doc* and *sent* refer to the document ID and the sentence number that together identify a unique sentence in the corpus. $\hat{V}_p$ are the positions of entities in the aforementioned subtuple t_p and $\hat{C}_p$ are the positions of phrases in C_p. Suppose the aforementioned s_1 is the 8th sentence of document 9. In this context, JERRY YANG spans from position 3 to 4. "Stanford" and "graduate" are at positions 0 and 2, respectively. Hence, the context is represented as $\langle 9, 8, \{\langle 3, 4\rangle\}, \{0, 2\}\rangle$.

Note that there can be multiple contexts of t_{p_1}, each being a sentence containing JERRY YANG, "Stanford," and "graduate." We denote all contexts of t_p by $\phi_p(t)$. Without loss of generality, we use sentence and context interchangeably. Note that in reality contexts with coarser granularity, such as paragraphs and sections, can be applied.

Problem Statement. Denote all answers to query $q=\langle V, P\rangle$ by A. Our goal is to rank the answers in A according to $\phi=\{\phi_p | p \in P\}$, where $\phi_p=\bigcup_{t \in A} \phi_p(t)$.

Since the information that is used for ranking, ϕ, is primarily position information (i.e., documents IDs, sentence numbers, entity spans and phrase positions), the problem is called the *position-based* ranking problem.

Given a query $q=\langle V, P\rangle$, our *ranking framework* consists of three scoring functions F^S, F^R and F^A, such that for each answer t: (1) its score on a selection predicate $p \in P$ is given by $F^S_p(t)$; (2) its score on a relation predicate $p \in P$ is given by $F^R_p(t)$; and (3) its final score $F^A(t)$ aggregates all predicate scores obtained via F^S and F^R. In this framework, the scores of different predicates are computed independently from each other. The intuition can be explained as follows. In Query 1, whether a PERSON is a Stanford graduate (p_1) is assumed independent from whether she founded any COMPANY (p_3) and irrelevant to whether a COMPANY is in Silicon Valley (p_2).

3.2. Position-Based Features

This section studies three position-based features that are derivable from cooccurrence contexts. These features are the key components in our ranking models.

3.2.1. Proximity. Intuitively, if the entities in t_p and the keywords in C_p are close to each other in a context $s \in \phi_p(t)$, they likely belong to the same grammatical unit of the corresponding sentence (e.g., a phrase like “Stanford University graduate JERRY YANG”) and thus form a piece of true evidence. Given predicate p, we define the proximity of t_p in s as

$$prox_p(t,s) = prox_p(t_p,s) = \frac{\sum_{e\in t_p} |token(e,s)| + \sum_{c\in C_p} |c|}{|scope_p(t_p,s)|},$$

where $|token(e,s)|$ is the number of tokens in s representing entity e; $|c|$ is the number of tokens in phrase c; $scope_p(t_p,s)$ is the shortest token sequence in s that covers all the entities in t_p and all the phrases in C_p; and $|scope_p(t_p,s)|$ is the number of tokens in the sequence. Note that $prox_p(t,s)$ is in [0,1] by this definition.

Different representations may be used in various places to refer to the same entity and may have different numbers of tokens. For example, the entity IBM may be represented by “IBM,” “Big Blue,” or “International Business Machine.” Hence, $|token(\text{IBM},s)|$ may be 1, 2, or 3 in different context s.

Example 2. The following two sentences are both contexts of the highlighted (all capitalized) entities for predicate p_1 in Query 1. Context s_1 is true, supporting a true positive, while s_4 is false, supporting a false positive.

s$_1$: Stanford University graduates JERRY YANG and ...
s$_4$: A professor at Stanford University, COLIN MARLOW had a relationship with Cristina Yang before she graduated ...

Predicate p_1 has two phrases, “Stanford” and “graduate,” each of which has one token. Hence $\sum_{c\in C_{p_1}} |c|$=2. In s_1, the PERSON JERRY YANG is represented by two tokens, “Jerry” and “Yang”, hence $\sum_{e\in t_{p_1}} |token(e,s_1)|$=2. The scope covering the entity and the two phrases spans 5 tokens, from “Stanford” to “Yang”, thus $|scope_{p_1}(t_{p_1},s_1)|$=5. Therefore, the proximity of JERRY YANG in sentence s_1 is $prox_{p_1}(t_{p_1},s_1)=\frac{2+2}{5}$=0.8. Similarly, the proximity of COLIN MARLOW in s_4 is $\frac{2+2}{13}$=0.31. Based on proximity alone, we say that s_1 is more likely a piece of true evidence. Therefore JERRY YANG is more likely to satisfy p_1 than COLIN MARLOW, given no other context.

3.2.2. Ordering Pattern. An ordering pattern refers to the order of entities and phrases in a cooccurrence context. Consider again predicate p_1=⟨{x}, {“Stanford”, “graduate.”}⟩ Let c_1=“Stanford” and c_2=“graduate.” This predicate has six different ordering patterns (xc_1c_2, xc_2c_1, c_1xc_2, c_2xc_1, c_1c_2x and c_2c_1x). Generally, if we denote all possible patterns of a predicate p by O_p, we have $|O_p|=(|V_p| + |C_p|)!$. Note that, extra tokens and punctuations between entities and phrases are irrelevant to the patterns. Hence, “Stanford University graduate, JERRY YANG” and “Stanford graduate JERRY YANG” follow the same pattern, c_1c_2x.

We observe that some ordering patterns are better indicators of true evidence than others. For example, to express that somebody is a graduate of Stanford University, true evidence usually follows the pattern c_1c_2x (e.g., s_1). Contexts following another pattern, c_1xc_2, are likely to be false evidence (e.g., s_4). To distinguish good patterns (those that tend to indicate true evidence) from others, we may assign a different weight to each pattern, so that entities supported by contexts following good patterns are scored higher. However, it is impossible to predetermine the weights since the goodness of ordering patterns are predicate-dependent. To illustrate, c_1c_2x is a good pattern for predicate p_1 in Query 1, but may not be equally good for another predicate p=⟨{x:NOVEL}, {“by”, “Jane Austen”}⟩, because it is less common to see true evidence such as

... written by Jane Austen, *PRIDE AND PREJUDICE*...

In our approach, the weights of ordering patterns for a predicate p are dynamically derived from ϕ_p, the set of all cooccurrence contexts for p. Denoting $\phi_p(o)$ as the subset of contexts following pattern o, we define the weight of o for predicate p as its frequency in ϕ_p,

$$f_p(o) = |\phi_p(o)|/|\phi_p|.$$

This definition captures the intuition that good patterns appear more often than bad ones. Although in theory there might be a pattern frequently appearing in false evidence, making a bad pattern more common, we do not observe such case in our experiments.

3.2.3. Mutual Exclusion. Given a predicate p, multiple contexts in ϕ_p may have the same $\langle doc, sent\rangle$ value, that is, coming from the same sentence. They are contexts of different entities and may follow different ordering patterns in that sentence. The coexistence of different patterns in one sentence is called a *collision* and the patterns are *colliding patterns*. The mutual exclusion rule dictates that, when collision happens, at most one colliding pattern is effective and the sentence is only considered evidence following that pattern.

Example 3. The following sentence appears as three contexts for p_1, one for each highlighted (all capitalized) entity. Ric Weiland follows the pattern o_1=xc_2c_1. PAUL ALLAN and BILL GATES follow o_2=c_2c_1x. Semantically, the former pattern is the effective pattern and the sentence is only a piece of evidence for RIC WEILAND.

s_5: After RIC WEILAND graduated from Stanford University, PAUL ALLEN and BILL GATES hired him in 1975 ...

Without understanding the semantics, it is difficult to decide which colliding pattern is absolutely effective. Therefore, we relax the rule with a *credit* mechanism, where every colliding pattern is considered partially effective, and patterns with higher credits are more likely to be effective than those with lower credits. We assume each sentence s (that is a context of at least one subtuple t_p for predicate p) has a total credit of 1, meaning that there is only one effective pattern. Given a predicate p, we denote the colliding patterns in s by $O_p(s) \subseteq O_p$. Each $o \in O_p(s)$ gets a credit $credit_p(o, s)$, and $\sum_{o\in O_p(s)} credit_p(o, s)$=1.

To allocate credits to the colliding patterns $O_p(s)$, we adopt the intuition that patterns followed by more prominent entities are more likely to be effective. Specifically, let $T_p(o, s)$ be all subtuples on p following pattern o in s. For each $o \in O_p(s)$, we choose a representative from $T_p(o, s)$, denoted by $T^*_p(o, s)$, which is the one with the highest proximity value, that is, $T^*_p(o, s)$=$\arg\max_{t_p\in T_p(o,s)} prox_p(t_p, s)$. We compare the representatives (and thus the patterns that they follow) by how prominent they are, that is, by their overall numbers of contexts in ϕ_p. The credit of o in s is

$$credit_p(o, s) = \frac{|\phi_p(T^*_p(o, s))|}{\sum_{o'\in O_p(s)} |\phi_p(T^*_p(o', s))|},$$

where $\phi_p(T^*_p(o, s))$ is the set of contexts of $T^*_p(o, s)$ for predicate p. Note that we choose the most proximate sub-tuple as the representative of a colliding pattern and allocate credits based on representatives only. The intuition is that the most proximate subtuple is most likely to form a grammatical unit with phrases in C_p, and hence the most reliable one for allocating credits.

In Example 3, t^1=$T^*_{p_1}(o_1, s)$=RIC WEILAND (i.e., the representative of pattern o_1 is RIC WEILAND) since he is the only PERSON in s following pattern o_1; and

t^2=$T^*_{p_1}(o_2, s)$=PAUL ALLEN because he has higher proximity (0.67) than BILL GATES (0.44), though both follow o_2. Suppose RIC WEILAND is found in 4 contexts ($|\phi_{p_1}(t^1)|$=4) and PAUL ALLEN in 2 ($|\phi_{p_1}(t^2)|$=2). Then, the credits of their corresponding patterns in s would be $\frac{4}{4+2}$=0.67 (for o_1) and 0.33 (for o_2).

Note that the pattern credit here is different from the pattern weight in Section 3.2.2. The weight of pattern o is a global measure (aggregated over ϕ_p) of how frequent o is. The credit of o, on the contrary, is a local measure particular to each sentence s, indicating how likely o is the effective pattern in s.

3.3. Single-Predicate Scoring

So far, we have introduced all the position-based features for assessing individual contexts. Integrating these features together, this section presents the Cumulative Model (CM) for scoring answers for a single predicate. We assume that F^S is the same as F^R (i.e., the same function is used for scoring all predicates), hence for brevity, we use $F_p(t)$ instead of $F^S_p(t)$ and $F^R_p(t)$.

Let $\phi_p(t, o) \subseteq \phi_p(t)$ be all contexts of t for predicate p that follow pattern $o \in O_p$. Our Cumulative Model (CM) is

$$F_p(t) = \sum_{o \in O_p} \left(f_p(o) \sum_{s \in \phi_p(t,o)} prox_p(t, s) credit_p(o, s) \right),$$

where $f_p(o)$ is the weight of pattern o, $prox_p(t, s)$ is t_p's proximity in context s, and $credit_p(o, s)$ is the credit of o in s.

The model divides $\phi_p(t)$, t's contexts for p, into $|O_p|$ groups, $\{\phi_p(t, o)|o \in O_p\}$, so that contexts in each group follow the same pattern. For each group $\phi_p(t, o)$, the model computes a *group score* (the inner summation). The group scores are linearly combined using weights $f_p(o)$ (the outer summation), such that the group scores of better patterns account more in $F_p(t)$. The kernel of the function, $prox_p(t, s)$ $credit_p(o, s)$, assesses how likely s is a piece of true evidence of t for predicate p. It is monotonic to both the proximity of t_p and the credit of t_p's pattern o. Answers supported by contexts having higher proximities and pattern credits will accumulate higher scores and thus be ranked higher.

It is interesting to note that CM can be customized easily by switching its component features on and off, so that we can evaluate the effectiveness of individual features. While detailed evaluations are presented in Section 5, below we list three important customizations.

$$\texttt{COUNT:}\ F_p(t) = \sum_{o \in O_p} (1 \sum_{s \in \phi_p(t,o)} 1) = \sum_{o \in O_p} |\phi_p(t, o)| = |\phi_p(t)|$$

$$\texttt{PROX:}\ F_p(t) = \sum_{o \in O_p} \sum_{s \in \phi_p(t,o)} prox_p(t, s) = \sum_{s \in \phi_p(t)} prox_p(t, s)$$

$$\texttt{MEX:}\ F_p(t) = \sum_{o \in O_p} \sum_{s \in \phi_p(t,o)} credit_p(o, s) = \sum_{s \in \phi_p(t)} credit_p(o, s).$$

COUNT is a straightforward method that scores t by the count of its supporting evidence. It can be reduced from CM by turning off all the features, that is, setting $prox_p(t, s) \equiv 1$, $credit_p(o, s) \equiv 1$ and $f_p(o) \equiv 1$. PROX only applies the proximity feature and is reduced from CM by setting $credit_p(o, s) \equiv 1$ and $f_p(o) \equiv 1$. MEX only applies the mutual exclusion feature. It is derived from CM by setting $prox_p(t, s) \equiv 1$ and $f_p(o) \equiv 1$.

Table II. Example Answers

	x	y	p_1	p_2	p_3	Π	Σ
t_1	Jerry Yang	Yahoo!	0.8	0.7	0.8	0.448	2.3
t_2	Larry Page	Google	0.6	0.5	0.6	0.18	1.7
t_3	Scott McNealy	Cisco	0.9	0.8	0.2	0.144	1.9
t_4	Bill Gates	IKEA	0.3	0.1	0.2	0.006	0.6

3.4. Multipredicate Scoring

We extend our single-predicate scoring model to handle multipredicate queries. Given a query answer, CM computes a score on each predicate. However, it remains a task to derive the final score, $F^A(t)$, from multiple predicate scores.

With CM, predicate scores are unbounded, that is, the more contexts the higher scores. When multiple predicate scores are aggregated, some could be so high that they dominate the aggregate score. To alleviate this predicate dominance problem, we propose the *Bounded Cumulative Model* (BCM) for better scoring of individual predicates:

$$F_p(t) = \sum_{o \in O_p} \left(f_p(o) \left[1 - \prod_{s \in \phi_p(t,o)} \big(1 - prox_p(t,s) credit_p(o,s)\big) \right] \right). \tag{1}$$

BCM uses the same three features as CM does, but differs from CM in computing group scores, each of which is computed from a set of contexts $\phi_p(t, o)$. Basically, BCM bounds all group scores in the range [0,1], and consequently it bounds the predicate scores within [0,1], since $\sum_{o \in O_p} f_p(o)$=1 according to Section 3.2.2.

Given an answer t to query q=$\langle V, P \rangle$, t's final score, $F^A(t)$, is computed as the product of its scores on all predicates,

$$F^A(t) = \prod_{p \in P} F_p(t), \tag{2}$$

where $F_p(t)$ can use either BCM or CM. For our problem, product is a more reasonable aggregate function than summation, another common aggregate function, because it favors answers with balanced predicate scores over those with polarized ones. To illustrate why balanced scores should be favored, consider Table II. The table shows four answers to Query 1. For each answer, it lists all three predicate scores (by BCM), as well as the final scores using product and summation, respectively. The two aggregates agree on the ranking of t_1 and t_4, which get unanimously (i.e., balanced) high and low predicate scores, but disagree on t_2 and t_3. The true positive, t_2, gets modest and balanced scores on all the predicates. It is correctly ranked higher than t_3, a false positive, by using product as the aggregate function, but loses the competition when summation is used. Answer t_3 gains high scores on p_1 and p_2 (both indeed satisfied by t_3), but low score on p_3 (It does not satisfy p_3 in the real world.) When summation is the aggregate function, the final score of t_3 is dominated by the high scoring predicates and t_3 is mistakenly ranked above t_2.

4. WEIGHTING

We have introduced our multipredicate ranking method based on Bounded Cumulative Model. In this section, we will look further into BCM and explore several weighting schemes to improve its accuracy.

As discussed earlier, BCM is proposed to alleviate the problem of predicate dominance. But its multiplicative nature (the product in Formula 1) also makes predicate scoring sensitive to a few good contexts. As an extreme example, if only one ordering pattern is involved (thus, $f_p(o)$=1), the predicate score $F_p(t)$ will become 1 (the maximum) as long as any context is scored 1 ($prox_p(t,s)credit_p(o,s)$=1), regardless of other supporting contexts. In multipredicate queries, such sensitivity of predicate scores will be further magnified by Formula 2. For instance, a small change in one predicate score from 0.1 to 0.2 will double the answer score. Apparently, ranking judgment made in such sensitive situations is susceptible and should be downplayed.

Our idea for solving the sensitivity problem is to penalize susceptible answers by weighting. The new weighted scoring model for general entity-relationship queries is Formula 3. Compared to Formula 2, an exponential weight $W_p(t)$ is applied over each $F_p(t)$, the BCM score of answer t for predicate p. In the ideal case, the weight should adjust itself according to the degree of susceptibility of each answer. Intuitively, if $F_p(t)$ is computed according to a small number of contexts, it is susceptible (even though the score may be high) and should receive a penalty for that. Briefly speaking, the weight $W_p(t)$ needs to be larger for more susceptible answers (i.e., answers supported by less contexts). Note that $F_p(t)$ is bounded between [0, 1] by BCM, hence larger weight will lower the predicate score.

$$F^A(t) = \prod_{p \in P} F_p(t)^{W_p(t)}. \tag{3}$$

The rest of this section discusses two weighting schemes that match our intention: higher weights $W_p(t)$ for answers with less contexts. With regard to a predicate, let us define the *support* of a candidate answer t to be the number of supporting contexts for t_p, which is denoted by $|\phi_p(t)|$. We measure the significance of t_p by comparing its support with some "best-possible" support. In the first weighting scheme, maximal-support weighting, we compare the support of t_p with the largest support among all the answers. The larger the difference is, the higher the weight will be. In the second scheme, corpus-frequency weighting, the support of t_p is compared with its "corpus frequency," which refers to the number of contexts in the corpus that contain t_p.

4.1. Maximal-Support Weighting

The maximal-support weight is defined as

$$W_p(t) = \alpha_p(t) = \frac{\max_{t'} \log(|\phi_p(t')| + 1)}{\log(|\phi_p(t)| + 1)}.$$

The denominator is the logarithm of answer t's support (plus 1) for predicate p and the numerator is the maximum of such logarithm values among all the answers. The smoothing constant 1 is used to avoid zero-denominator when $|\phi_p(t)| = 1$. By this weighting scheme, answers with less support (i.e., smaller denominators) will receive higher penalties. Suppose there are two answers for predicate p, where t_1 has 3 contexts ($\log(3+1) = 2$) and t_2 has much more contexts, say 127 ($\log(127+1) = 7$). Their maximal-support weights are $\alpha_p(t_1) = \frac{\max\{2,7\}}{2} = 3.5$ and $\alpha_p(t_2) = \frac{7}{7} = 1$, respectively. Thus, t_1 is penalized more with weight 3.5. Note that, although we use base 2 logarithm in the calculation for illustration purpose, the weight is actually independent of the choice of base.

For an individual predicate, if two answers have the same support (hence, the same denominator), their maximal-support weights will be the same, and their relative ranking is preserved in spite of the weighting, according to the monotonicity that $x < y \Leftrightarrow x^w < y^w$ for $w > 0$.

The numerator of $\alpha_p(t)$ corresponds to the maximum of observed support for a predicate. It is a hint on how much support a true answer could get from the corpus. Here, we implicitly assume that the maximally supported answer is true, which is usually the case. We utilize this hint to evaluate the significance of the supports of candidate answers and weight them accordingly. To illustrate, let us consider an answer t with 3 contexts. We observed two situations in our corpus: (1) the maximal support is much higher than 3 (e.g., 127), and (2) the maximal support is also low (e.g., 7). The former indicates that true answers tend to have many contexts, while the latter indicates that it is unlikely to find many contexts for this predicate. Intuitively, answer t should be penalized more in the former situation since it does not look like a true answer in terms of support. In this regard, the definition of $\alpha_p(t)$ tends to penalize poorly-supported answers more severely.

It is important to note that the maximal support (i.e., the numerator), has no effect on ranking single-predicate queries because it is the same for all answers. For multi-predicate queries, each predicate may have a different maximal support. Predicates with higher maximal support tend to yield higher weights, and penalize the answer more severely given the same support value, and thus impact the ranking of candidate answers. In sum, maximal-support weighting applies different treatment of predicates according to their maximal supports, relying on predicates with high maximal support to penalize their poorly-supported answers.

The maximal-support weighting has its own concern, as the maximal support itself is sensitive to outliers. In an undesirable scenario, it is possible that most true answers of a predicate have less than 10 contexts, except for one outlier with 100 contexts. In this case, the maximum support is 100, resulting in high penalty to other true answers. A more robust solution could use the average of the largest k supports as the numerator in $\alpha_p(t)$. We leave this issue for future study.

4.2. Corpus-Frequency Weighting

The corpus-frequency weighting takes into consideration the corpus background of an answer for evaluating the significance of its support. Given the same support, the more an answer appears in the corpus, the less significant the support is. This intuition is reflected in our corpus-frequency weighting as

$$W_p(t) = \beta_p(t) = \frac{\log(N(t_p) + 1)}{\log(|\phi_p(t)| + 1)},$$

where t_p is the projection of t onto the variables in p. $N(t_p)$ is t_p's corpus frequency, that is, the number of sentences in the whole corpus that contain t_p. Note that $|\phi_p(t)| \leq N(t_p)$ because $\phi_p(t)$ is the set of supporting contexts of t with regard to p, that is, those sentences that not only contain t_p but also C_p, the phrases in p.

For example, for predicate p_2=$\langle\{y\}$, "Silicon Valley"$\rangle$ on COMPANY, NASDAQ is supported by 3 contexts out of its 845 occurrences in the corpus while MAYFIELD FUND has 3 out of 11. Given its high corpus frequency, NASDAQ is quite susceptible because its cooccurrence with "Silicon Valley" likely happens just by coincidence. In this sense, we consider the support of MAYFIELD FUND more significant than that of NASDAQ. By applying corpus-frequency weighting, the weight for NASDAQ is $\frac{\log(846)}{\log(4)}$, while the weight for MAYFIELD FUND is $\frac{\log(12)}{\log(4)}$. This meets our goal of giving higher weights (penalties), thus lower scores, to more susceptible answers. In contrast, given this example, maximal-support weighting will weight the two answers equally since they have the same support. Hence by looking at the corpus frequency, $\beta_p(t)$ is able to capture the difference in case of tied support.

Corpus-frequency weighting shares the same intuition as IDF (inverse document frequency) weighting in document retrieval. Both weighting schemes favor specificity over generality. The difference is that IDF weight depends on query terms only and is the same for all hit answers, while our corpus-frequency weight is dependent on both query predicates and the returned answers.

A concern is how to weight between two answers of the same specificity, for instance, (1) $|\phi_p(t_1)| = 1$ with $N(t_{1p}) = 5$ versus (2) $|\phi_p(t_2)| = 20$ and $N(t_{2p}) = 100$. Here, t_1 and t_2 have equal specificity if we measure specificity by the ratio between support and corpus frequency, $N(t_p)/|\phi_p(t)|$. While the conclusion is unclear in general, for our problem and corpus, the latter case should be favored. The intuition is that, since t_2 already has large support, its BCM score is not as susceptible as t_1 and hence the corpus frequency becomes less important in determining its significance. Our definition of corpus-frequency weight is capable of capturing this intuition by using logarithm in both numerator and denominator. As a real example from the predicate on Silicon Valley companies, APPLE (a true answer) has 19 contexts out of its 2639 occurrences with specificity $\frac{19}{2639} = 0.72\%$, while SAP (a false answer) has 2 out of 211 with specificity 0.94%. Even though Apple has lower specificity, its weight is $\frac{\log(2639+1)}{\log(19+1)} = 2.6$, compared to SAP's 4.9.

5. EXPERIMENTS

This section provides empirical evaluation of our prototype system implemented in Apache Lucene.[7] A demo of the system is maintained at `http://idir.uta.edu/erq` [Li et al. 2010b]. The system consists of several components. An *Indexer* constructs an entity-centric index over our corpus. It associates each term w with a list of entities (ordered by entity IDs) that cooccur with w somewhere in the corpus. For each entity e in the list, it further records where w and e cooccur. Leveraging this design, the *Retriever* efficiently retrieves entities and cooccurrence contexts. The results are fed to *Ranker* for ranking, which is based on the ideas in previous sections.

5.1. Data and Query Sets

We used the 2008-07-24 snapshot of Wikipedia.[8] After we removed all the irrelevant pages (such as category and administrative pages), there were about 1.8 million articles. This article set is used as the entity catalog. Each article is the description of an entity, by Wikipedia's nature of being an encyclopedia, and the article title corresponds to the entity name. We predefined 10 entity types (Table III) and assigned about 0.63 million entities to these types based on simple handcrafted rules, mainly using their categories in Wikipedia. For example, if an article belongs to a category whose name ends with "novels" (e.g., British novels) it is treated as an entity of type NOVEL. This simple method turns out sufficiently accurate for our experiments. These 10 types represent some of the major types of entities available in Wikipedia, for instance, PERSON. They also include the main types (people, places, and organizations) used in the study of named entity recognition (NER). Although a more comprehensive, fine-grained, and accurate categorization of entities can improve the system's query capability, it is not the focus of this study.

The same article set is used as the query corpus. For each article, we removed its section titles, tables, infoboxes, references, etc., retaining only the main content. The main text is segmented into sentences. We removed punctuation marks and stemmed all words using the Porter stemmer.[9] We consider the hyperlinks between Wikipedia

[7] http://lucene.apache.org/

[8] http://download.wikimedia.org

[9] http://tartarus.org/ martin/PorterStemmer/

Table III. Ten Types from Wikipedia

Type	(E)ntities	(O)ccurrences	O/E
AWARD	979	83,001	85
CITY	54,296	2,269,771	42
CLUB	13,365	701,042	52
COMPANY	20,966	842,849	40
FILM	37,455	595,774	16
NOVEL	15,385	181,156	12
PERSON	368,833	5,985,231	16
PLAYER	74,171	750,708	10
SONG	27,805	251,186	9
UNIVERSITY	17,491	677,140	39
TOTAL	630,746	11,742,084	19

articles as occurrences of the link targets (entities). In this way, we collected nearly 12 million occurrences of the 0.63 million typed entities.

Named entity recognition (NER) [Nadeau et al. 2007] and entity disambiguation [Dill et al. 2003] are intensively studied problems. Our hyperlink-based annotation can be viewed as a rudimentary entity disambiguation method. Recently we have seen advanced entity recognition and disambiguation methods using Wikipedia as entity catalog [Kulkarni et al. 2009; Mihalcea and Csomai 2007; Milne and Witten 2008] to automatically link entities mentioned in plain text to their corresponding Wikipedia articles. One of our ongoing efforts is to use Wikify [Milne and Witten 2008] to annotate entities occurrences that are not hyperlink anchortexts. This method will give us more comprehensive entity occurrences. Furthermore, it can be applied on generic Web pages, enabling entity-relationship queries on Web corpus.

We have previously collected two query sets, INEX17 and OWN28. INEX17 is adapted from the topics in the Entity Ranking track of INEX 2009. From the 60 available topics, we adapted the ones that are on our predefined 10 entity types. We obtained 11 single-predicate queries and 6 multipredicate queries. OWN28 contains our own crafted 28 queries, including 16 single-predicate queries and 12 multipredicate queries. Since both query sets are relatively small, we merged them into one set for more robust evaluation. It contains 27 single-predicates (Single-27) and 18 multi-predicate queries (Multi-18).

Ground Truth. We manually checked query answers returned by our system to collect the ground truth. However, some of the queries return hundreds of answers. Exhaustive checking is prohibitively time consuming. Therefore, for such queries, we adopted the depth-N pooling approach used by INEX. Basically for each query, we only check the top N answers returned by each ranking method. Since different methods overlap greatly in their top N answers, a lot of time is saved. This pooling approach allows us to evaluate ranking accuracy up to rank N. As a typical choice, N is set to 100 in our experiments.

5.2. Answers to Sample Queries

In this section we use several sample queries to demonstrate the accuracy of our query evaluation system, based on BCM scoring model. The top-10 answers to each query are displayed. The true answers are marked by bullets.

Case 1. Find the list of big ten universities.
Case 2. Our Query 1– Silicon Valley companies founded by Stanford graduates.

Case 3. Find Academy Award winning films starring Australian actors.
Case 4. A basketball fan looking for team leaders of NBA champions. Particularly, she is interested in those NBA Finals MVPs.

Case 1	**Case 2**

`SELECT x FROM UNIVERSITY x`
`WHERE x:["Big Ten"]`
	`SELECT x, y FROM PERSON x, COMPANY y`
	`WHERE x:["Stanford", "graduate"]`
	`AND y:["Silicon Valley"] AND x,y:["found"]`

• ⟨Indiana University (Bloomington)⟩ • ⟨Ohio State University⟩ • ⟨University of Michigan⟩ • ⟨Pennsylvania State University⟩ • ⟨University of Iowa⟩ • ⟨Purdue University⟩ • ⟨Michigan State University⟩ • ⟨University of Wisconsin-Madison⟩ • ⟨University of Illinois at Urbana-Champaign⟩ • ⟨Northwestern University⟩	• ⟨Jerry Yang, Yahoo!⟩ • ⟨David Packard, Hewlett-Packard Company⟩ • ⟨Scott McNealy, Sun Microsystems⟩ • ⟨Vinod Khosla, Sun Microsystems⟩ ⟨Bill Gates, Microsoft⟩ • ⟨William Hewlett, Hewlett-Packard Company⟩ ⟨Vinod Khosla, Kleiner Perkins Caufield & Byers⟩ • ⟨Larry Page, Google⟩ • ⟨Andy Bechtolsheim, Sun Microsystems⟩ • ⟨Sergey Brin, Google⟩

Case 3	**Case 4**

`SELECT x, y FROM FILM x, PERSON y`
`WHERE x:[win "Academy Award"]`
`AND y:["Australian" "actor"]`
`AND x,y:["star"]`
	`SELECT x, y FROM CLUB x, PERSON y`
	`WHERE x:["NBA" "champion"]`
	`AND y:["Finals MVP"]`
	`AND x,y:["led"]`

• ⟨Braveheart, Mel Gibson⟩ • ⟨Brokeback Mountain, Heath Ledger⟩ • ⟨The Adventures of Priscilla, Guy Pearce⟩ • ⟨L.A. Confidential, Guy Pearce⟩ ⟨Mad Max, Mel Gibson⟩ • ⟨The Year of Living Dangerously, Mel Gibson⟩ • ⟨Elizabeth, Geoffrey Rush⟩ • ⟨A Beautiful Mind, Russell Crowe⟩ • ⟨The Adventures of Priscilla, Hugo Weaving⟩ • ⟨Gladiator, Russell Crowe⟩	• ⟨Chicago Bulls, Michael Jordan⟩ • ⟨Los Angeles Lakers, Magic Johnson⟩ • ⟨San Antonio Spurs, Tim Duncan⟩ • ⟨Los Angeles Lakers, Kareem Abdul-Jabbar⟩ • ⟨Boston Celtics, Larry Bird⟩ • ⟨Los Angeles Lakers, Shaquille O'Neal⟩ • ⟨Houston Rockets, Moses Malone⟩ • ⟨Los Angeles Lakers, James Worthy⟩ ⟨Philadelphia 76ers, Kareem Abdul-Jabbar⟩ • ⟨Detroit Pistons, Isiah Thomas⟩

5.3. Comparing Ranking Methods

In this section, we compare and analyze the multiple ranking methods discussed in Section 3, namely COUNT, MEX, PROX, CM, and BCM. All the methods differ in how they compute predicate scores, that is, $F_p(t)$. For multipredicate queries, the same Formula (2) is used to aggregate predicate scores into answer score. We compare these ranking methods using three popular measures: *MAP*, *nDCG*, and *Precision-at-k* (P@k for short). Since it is extremely time-consuming to obtain the complete ground truth in

Table IV. Comparing Different Ranking Methods

Query	COUNT	MEX	PROX	CM	BCM	ER
MAP						
Single-27	0.826	0.858	0.901	0.909	0.911	0.829
Multi-18	0.573	0.603	0.682	0.708	0.763	0.703
nDCG						
Single-27	0.888	0.908	0.932	0.937	0.940	0.901
Multi-18	0.729	0.753	0.806	0.826	0.855	0.812

Wikipedia (which exactly motivates this study), we do not evaluate the absolute recall or F-measure.

The MAP *(Mean Average Precision)* is the arithmetic mean of average precisions for a set of queries. It is a good indicator of both precision and recall. The top part of Table IV shows MAP for Single-27 and Multi-18. Both MEX and PROX improve over COUNT, by 0.032 and 0.075 on Single-27, respectively, and by 0.03 and 0.109 on Multi-18. PROX is shown to be more effective than MEX. CM and BCM have similar performance on Single-27, but BCM excels over CM by 0.055 on Multi-18. This demonstrates that BCM is a better model for ranking multipredicate queries.

The nDCG *(Normalized Discounted Cumulative Gain)* discounts the gain of a true answer by its rank in the result list and normalizes the cumulated gain with that of a perfect ranking. The bottom part of Table IV shows the average nDCG on Single-27 and Multi-18. The observation is similar to the case of MAP.

Overall, our conclusion is that, CM and BCM are comparable for ranking single-predicates and BCM has clear advantage on multipredicate queries. We refer interested readers to Li et al. [2010a] for separate evaluation of INEX17 and OWN28 query sets in earlier experiments.

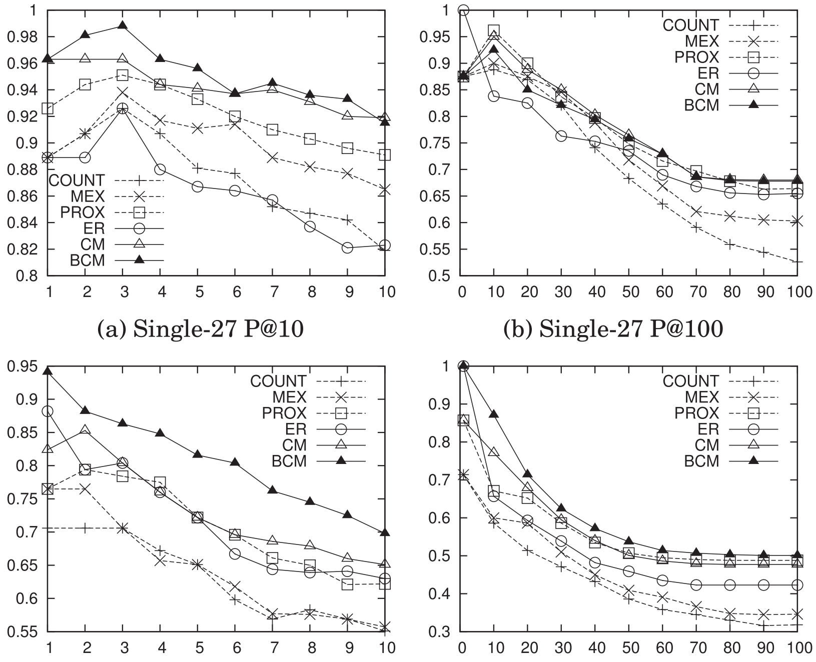
To further analyze how various methods perform at different ranks, we plot *precision-at-k* curves. Figure 1(a) shows the results for Single-27 with *k* up to 10 (denoted as P@10). COUNT has the worst performance. PROX is consistently better than MEX across all ranks, but worse than CM and BCM, agreeing with the conclusion drawn from MAP and nDCG analysis. BCM is the best among all, especially for top 5. We further collected a subset of queries in Single-27, which returned more than 100 answers, and plot the P@100 curve in Figure 1(b). While COUNT and MEX are still the worst, the other methods are fairly close to each other. Our observation on Single-27 indicates that, for single-predicate queries, BCM is good at ranking top answers, but does not have advantage over the long range. Figure 1(c) and (d) reports the same experiment on Multi-18. It can be easily observed that BCM has clear advantage in ranking multipredicate queries for top answers and beyond. This reinforces our conclusion drawn from Table IV.

Note that, if all true answers are ranked before position K, the precision after K will not change. This is reflected by the flat tails in Figure 1(b) and (d).

In summary, the individual features are effective for ranking entity-relationship queries and they work best in concert when integrated in BCM. BCM rivals CM on single-predicate queries, and excels on multipredicate queries. This is because BCM alleviates the predicate dominance problem as discussed in Section 3.4.

5.4. BCM vs. Other Entity Ranking Methods

As a first study on multipredicate entity-relationship query, we did not find directly comparable systems. By our best effort, we have chosen three state-of-the-art systems proposed for related problems: EntityRank, INEX and INRIA. All the three systems work on the entity ranking problem, though under different task settings. Besides,



(a) Single-27 P@10

(b) Single-27 P@100

(c) Multi-18 P@10

(d) Multi-18 P@100

Fig. 1. Precision-at-k for Single-27 and Multi-18.

they are all evaluated using Wikipedia as corpus and entity catalog, though INEX and INRIA worked on different snapshots from ours.

EntityRank (ER) [Cheng et al. 2007] outperforms another closely related system [Chakrabarti et al. 2006] by a large margin in terms of MRR (Mean Reciprocal Rank), since Chakrabarti et al. [2006] do not focus on aggregating supporting evidence from multiple articles. We reimplemented ER in our system for scoring individual predicates. As ER focuses on single-predicate queries, performance on such queries can be fairly compared. For multipredicate queries, we can also use ER to compute predicate scores and aggregate them by the same Formula 2.

We tested ER with our data set. In Table IV, ER is worse than BCM on Single-27, by 0.082 in MAP and 0.039 in nDCG. It is only marginally better than the baseline COUNT. For multipredicate queries (Multi-18), ER shows relatively better performance than PROX, however, it is still worse than BCM by a large margin (0.06 in MAP and 0.043 in nDCG).

From Figure 1, it can be seen that ER is good at ranking top 2 answers (with the exception of (a)), rivaling CM and BCM. However, it deteriorates very fast when $k > 2$. In (c), ER drops below 0.7 around $k = 6$, while BCM remains above 0.7 at $k = 10$. It indicates that BCM is more robust for queries with multiple true answers. This is because BCM exploits more features than ER and is thus able to promote the ranking of some *hard* true answers indistinguishable by ER.

The INEX Entity Ranking track targets on a different problem setting. INEX queries are specified as narrative descriptions on the desired entities. Participating systems can use any techniques to answer the queries, but need to understand the query descriptions, which itself is challenging. Hence their MAPs tend to be low. The MAP achieved by the best system participating in the 2009 track is 0.517. To avoid

Table V. Compare Effectiveness of Different Weighting Schemes

Query	BCM	BCM-α	BCM-β	BCM-γ
MAP				
Single-27	0.911	0.910	0.913	0.915
Multi-18	0.763	0.771	0.785	0.788
nDCG				
Single-27	0.940	0.938	0.942	0.944
Multi-18	0.855	0.863	0.866	0.871

the overhead of assessing participating systems, INEX used a sampling strategy to estimate their MAPs.

INRIA [Vercoustre et al. 2008] works on the same problem as INEX. Unlike INEX participants, it is not based on cooccurrence of entities and query inputs. Rather, it finds the Wikipedia articles that best match query descriptions, through link analysis and TF-IDF weighting. It achieves MAP of 0.390 on 18 topics adapted from INEX 2006 ad hoc track.

In comparison with INEX and INRIA, the MAP achieved by BCM is 0.852 (averaged over all 45 queries). We acknowledge that this comparison is not strictly fair. First, the results are based on different query sets and snapshots of Wikipedia. Second, they focus on different query styles (structured query vs. narrative description). However, our argument is that the high MAP of BCM at least indicates that the structured entity-relationship queries can be highly effective in reality.

5.5. Effectiveness of Weighting

In this section we evaluate different weighting schemes introduced in Section 4. We compare plain BCM with BCM-α (applying maximal-support weight only) and BCM-β (applying corpus-frequency weight only). We also consider the case of combining the two weights as $\gamma_p(t) = \alpha_p(t) + \beta_p(t)$. Table V compares the results using both MAP and nDCG.

The benefits of the two weighting schemes are not clearly observed on Single-27. One reason is that the accuracy of BCM on single-predicate queries is already quite high, therefore any significant improvement would be challenging. Recall that both weighting schemes are designed to penalize susceptible answers instead of promoting promising ones. Their effects tend to diminish when there are few susceptible answers ranked at top positions, which is the case of Single-27. Nevertheless, the combined weighting, BCM-γ, does show marginal improvement over BCM.

The result on Multi-18 is different. Both BCM-α and BCM-β achieve better MAP than BCM, by 0.008 and 0.022, respectively. The corpus-frequency weighting (BCM-β) appears to be more effective than maximal-support weighting (BCM-α). BCM-γ benefits from both weighting components, achieving the highest MAP (0.788) and nDCG (0.871). There are two main reasons for the improvement. First, the sensitivity of Formula 2 bring more susceptible answers into top ranks in case of multipredicate queries. Second, as discussed in Section 4.1, the maximal support (the numerator) in BCM-α becomes effective for multipredicate queries.

6. FUTURE WORK

The distinguishing characteristic of our approach is to combine IR-style keyword constraints with SQL-style structured query constructs. As the initial exploratory study along this line, our work has mainly focused on designing an accurate ranking framework and building a prototype system, to demonstrates the effectiveness and promises of the approach. We have thus only supported basic keyword constraints in the queries. To deploy a production query system with good usability in practice, we need to support more advanced query features. Here we briefly enumerate several such features.

Our current definition of entity-relationship query does not support querying entities by name (e.g., find entities whose names contain "Michael") or querying relationships between entities (e.g., find the relationship between BILL GATES and MICROSOFT). It does not distinguish the subject and the object in a binary relationship between two entities of the same type. It does not allow disjunctive conditions such as `x:["Stanford", "graduate"] OR x:["Berkeley", "graduate"]`.

Although the predicates in entity-relationship query are keyword-based, it can be a burden for users to choose the right keywords. We are exploring two directions to address this concern. First, query suggestion can directly help users find the proper keywords. For example, after the user types "Stanford" for p_1 of Query 1, the system could suggest a list of keywords that commonly cooccur with "Stanford" in the corpus or in the query log, such as "graduate," "professor," etc. Second, the strict keyword matching can be relaxed by query expansion, for instance, allowing a context for p_1 to contain "alumni," if not "graduate." Synonym thesaurus and paraphrases mined from the corpus may be used for this purpose. New challenges on ranking shall arise with query expansion.

In addition to position-based features, the assessment of cooccurrence contexts can be improved with more features such as syntactic information (e.g., part-of-speech tags) and lexicographic information. The BCM model may be extended with new factors for these features.

7. CONCLUSION

Entity-relationship query is a structured facility to query entities over Wikipedia. It distinguishes itself by (1) allowing multiple keyword-based predicates in a query and (2) searching directly in corpus instead of preextracted data stores. We presented a ranking framework for entity-relationship queries and a Bounded Cumulative Model under this framework. Our ranking method exploits three intuitive positional features, which are shown to be effective in the experiments. To address the sensitivity problem in our ranking model, we further studied how to detect and weight susceptible answers. We explored two weighting schemes, both leveraging the intuition that predicate scores supported with less contexts are more susceptible and should be penalized accordingly.

REFERENCES

AGICHTEIN, E. AND GRAVANO, L. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*. 85–94.

AUER, S., BIZER, C., KOBILAROV, G., LEHMANN, J., CYGANIAK, R., AND IVES, Z. 2007. DBpedia: A nucleus for a Web of open data. In *Proceedings of the 6th International Semantic Web Conference*. 11–15.

BRIN, S. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of the International Workshop on the World Wide Web and Databases*. 172–183.

CAFARELLA, M. J., RÉ, C., SUCIU, D., ETZIONI, O., AND BANKO, M. 2007. Structured querying of Web text: A technical challenge. In *Proceedings of the Conference on Innovative Data Systems Research*. 68–74.

CAFARELLA, M. J., HALEVY, A., WANG, D. Z., WU, E., AND ZHANG, Y. 2008. WebTables: Exploring the power of tables on the web. *Proc. VLDB Endowment 1*, 1, 538–549.

CHAKRABARTI, S., PUNIYANI, K., AND DAS, S. 2006. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *Proceedings of the 15th International Conference on World Wide Web*. 717–726.

CHENG, T., YAN, X., AND CHANG, K. C.-C. 2007. EntityRank: Searching entities directly and holistically. In *Proceedings of the 33rd International Conference on Very Large Data Bases*. 387–398.

CHU, E., BAID, A., CHEN, T., DOAN, A., AND NAUGHTON, J. 2007. A relational approach to incrementally extracting and querying structure in unstructured data. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*. 1045–1056.

DEMARTINI, G., FIRAN, C. S., IOFCIU, T., KRESTEL, R., AND NEJDL, W. 2008. A model for ranking entities and its application to Wikipedia. In *Proceedings of the Latin American Web Conference*. 29–38.

DEROSE, P., SHEN, W., CHEN, F., DOAN, A., AND RAMAKRISHNAN, R. 2007. Building structured Web community portals: A top-down, compositional, and incremental approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases*. 399–410.

DILL, S., EIRON, N., GIBSON, D., GRUHL, D., GUHA, R., JHINGRAN, A., KANUNGO, T., RAJAGOPALAN, S., TOMKINS, A., TOMLIN, J. A., AND ZIEN, J. Y. 2003. SemTag and seeker: Bootstrapping the semantic Web via automated semantic annotation. In *Proceedings of the 12th International Conference on World Wide Web*. 178–186.

ETZIONI, O., BANKO, M., SODERLAND, S., AND WELD, D. S. 2008. Open information extraction from the Web. *Comm. ACM 51*, 12, 68–74.

KANDOGAN, E., KRISHNAMURTHY, R., RAGHAVAN, S., VAITHYANATHAN, S., AND ZHU, H. 2006. Avatar semantic search: A database approach to information retrieval. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 790–792.

KASNECI, G., SUCHANEK, F., IFRIM, G., RAMANATH, M., AND WEIKUM, G. 2008. NAGA: Searching and ranking knowledge. In *Proceedings of the IEEE 24th International Conference on Data Engineering*. 953–962.

KULKARNI, S., SINGH, A., RAMAKRISHNAN, G., AND CHAKRABARTI, S. 2009. Collective annotation of Wikipedia entities in Web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 457–466.

LI, X., LI, C., AND YU, C. 2010a. Entity-relationship queries over Wikipedia. In *Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents*. 21–28.

LI, X., LI, C., AND YU, C. 2010b. Entityengine: Answering entity-relationship queries using shallow semantics. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (Demonstration Description)*. 1925–1926.

MIHALCEA, R. AND CSOMAI, A. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the 16th ACM International Conference on Information and Knowledge Management*. 233–242.

MILNE, D. AND WITTEN, I. H. 2008. Learning to link with Wikipedia. In *Proceedings of the 17th ACM International Conference on Information and Knowledge Management*. 509–518.

NADEAU, DAVID, SEKINE, AND SATOSHI. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes 30*, 1, 3–26.

PETKOVA, D. AND CROFT, W. B. 2007. Proximity-based document representation for named entity retrieval. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*. 731–740.

SUCHANEK, F. 2009. Automated construction and growth of a large ontology. Ph.D. thesis, Saarland University.

SUCHANEK, F. M., KASNECI, G., AND WEIKUM, G. 2007. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*. 697–706.

VERCOUSTRE, A.-M., THOM, J. A., AND PEHCEVSKI, J. 2008. Entity ranking in Wikipedia. In *Proceedings of the ACM Symposium on Applied Computing*. 1101–1106.

VOORHEES, E. M. 2003. Overview of the trec 2003 question answering track. In *Proceedings of thee 12th Text Retrieval Conference*. 54–68.

ZARAGOZA, H., RODE, H., MIKA, P., ATSERIAS, J., CIARAMITA, M., AND ATTARDI, G. 2007. Ranking very many typed entities on Wikipedia. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*. 1015–1018.

ZHOU, M., CHENG, T., AND CHANG, K. C.-C. 2010. Data-oriented content query system: Searching for data into text on the web. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. 121–130.

Received December 2010; revised March 2011; accepted May 2011