Position-Based Ranking for Entity-Relationship Queries

Xiaonan Li
Department of Computer Science and Engineering
University of Texas at Arlington
xiaonan.li@mavs.uta.edu

Chengkai Li
Department of Computer Science and Engineering
University of Texas at Arlington
cli@uta.edu

Cong Yu
Yahoo! Research New York
congyu@yahoo-inc.com

ABSTRACT

Existing techniques for ranking entity query results only focus on queries with single predicates. This paper studies how to rank the answers to general entity-relationship queries with multiple predicates. We propose a position-based Bounded Cumulative Model that aims at maintaining the accuracy beyond the top-few answers, which is the focus of previous works. Experiments on INEX benchmark queries and manually designed queries validate the effectiveness and accuracy of our method.

1. INTRODUCTION

The continuous evolution of the Web has made itself an information repository full of *entities*. In discovering and exploring the entities that fascinate them, Web users are in need of structured querying facilities that explicitly deal with the entities, their properties, and relationships.

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 the above tasks. First, our tasks focus on *typed* entities such as PERSON and COMPANY and, in database terminology, their "join" relationships. Second, our tasks often involve synthesizing information scattered across different places, therefore a simple list of Web 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. Therefore, while conceptually simple, with only keyword search, the above task requires the users to perform multiple searches and assemble various information from a potentially large number of documents.

We propose a declarative query mechanism, *entity-relationship queries*, for the aforementioned tasks. (A demo of our prototype system is at **http://idir.uta.edu/erq**.) Evaluating such queries produces entities directly instead of documents. For example, a user can write the following SQL-like query to obtain the answers to Example 1.

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* (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 the requirement that `y` was founded by `x`). On the other hand, the individual predicates are specified by keyword-based constraints. The query semantics dictate that entities satisfy a predicate by a simple and intuitive requirement: the entities co-occur with the keywords in some contexts. For example, predicate p_1 requires every `x` in the query answer to co-occur with "Stanford" and "graduate" in some context. In other words, we aim to capture entity properties and relationships through shallow syntax requirements implied by users at query-time [1], instead of explicitly extracting and reasoning about complex semantic information in text before query-time [4, 3, 8, 10, 5, 11, 9]. Although such syntax clue is by no means rigorous or error-proof, it becomes robust when we take into consideration the repetitive nature of the Web: true facts are more likely to be stated on many different pages. This intuition has been widely used in Web search and mining, e.g., information extraction [4, 3] and entity search and ranking [7].

Challenges: Entity-relationship queries can yield many false answers due to abundant accidental co-occurrences. 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 over individual predicates. In contrast, existing entity search techniques only allow one keyword-based *soft* predicate[2] and are therefore unable to handle queries such as Query 1. *Second*, unlike in document search, users of entity-relationship queries may often expect more than just a few correct answers. For exam-

[1]We acknowledge that, the effectiveness of such entity-relationship queries partially relies on the user's capability in providing proper keyword constraints, just like in IR queries.
[2]Although they may allow multiple hard constraints, e.g., type constraints.

ple, in a query for Turing Award winners, more than 50 correct answers are expected. Therefore, the ranking method must aim at maintaining the accuracy beyond the top-few answers, which is more difficult than getting the initial few answers correct.

To address these challenges, we propose a ranking framework for general entity-relationship queries and a Bounded Cumulative Model (BCM). Since the semantics of such queries centers around the co-occurrence of entities and keywords, BCM integrates three position-based features, namely proximity, ordering pattern, and mutual exclusion. In contrast, the existing works on entity query ranking [6, 7] only exploit proximity. In summary, we make the following **contributions** in this paper:

- We present the general entity-relationship queries, for structured querying of entities directly over text with complex constraints (i.e., multiple predicates).
- We propose a position-based Bounded Cumulative Model to improve ranking accuracy beyond the top-few answers.
- We conduct comprehensive experiments to verify that BCM outperforms existing entity ranking techniques.

To our best knowledge, this paper is the first attempt to rank answers to multi-predicate entity-relationship queries. It is also the first attempt to improve accuracy beyond the top-few answers for such queries.

Related Work: Previous studies on structured querying of the Web focus on DB-based approach that explicitly extracts structured information into databases [4, 3, 8, 10, 5, 11, 9], which can then be queried using structured queries. The DB-based approach 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.

Entity search and ranking has gained significant interests in recent years [12, 1, 2, 14, 13]. For example, the tasks in the INEX Entity Ranking track are to find named entities that are relevant to certain contextual information in natural language text or related to a given entity. The tasks are specified by narrative descriptions. The focus is on accurately understanding the descriptions. They often come with type constraints on the entities. However, they do not have the concept of "predicates" and certainly do not deal with multiple predicates.

The studies most related to ours are [6, 7, 15]. [6] learns an optimal scoring function on proximity feature, but it only scores entities by one evidence and makes no attempt to integrate evidences found in multiple documents to improve ranking. Leveraging the redundancy of the Web, [7] aggregates scores of locally evaluated evidences into global scores. However, neither of the two studies tackles the challenge of improving ranking beyond the first few true answers. Moreover, they only focus on queries comparable to our single-predicate queries and thus do not study multi-predicate queries. [15] proposes a Content Query Language for querying entities, but essentially is also limited to single-predicate queries.

2. POSITION-BASED RANKING

2.1 Problem Statement

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, e.g., 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 keyword phrases[3]. Query 1 is thus formalized as $q_1=\langle V, P\rangle$, where $V=\langle x$:PERSON, y:COMPANY$\rangle$ and $P=\{p_1, p_2, p_3\}$. Among the predicates, $p_1=\langle\{x\}, \{$"Stanford", "graduate"$\}\rangle$ and $p_2=\langle\{y\}, \{$"Silicon Valley"$\}\rangle$ are selection predicates, and $p_3=\langle\{x, y\}, \{$"found"$\}\rangle$ is a relation predicate.

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, e.g., $t=\langle$Jerry Yang, Yahoo!$\rangle$ for Query 1. Given a predicate $p=\langle V_p, C_p\rangle$, we use t_p to represent the sub-tuple 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. $t_{p_1}=\langle$Jerry Yang$\rangle$ 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 of C_p and one entity for each variable in V_p, it is considered an *evidence* for p and these entities in whole are said to satisfy p. Suppose three evidences 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 co-founded Yahoo!.*

Jerry Yang satisfies p_1 by evidence s_1; Yahoo! satisfies p_2 by evidence s_2; and they together satisfy p_3 by s_3. Assembling the information together, the entity tuple $\langle$Jerry Yang, Yahoo!$\rangle$ is composed as an answer to the query since it satisfies all the query predicates.

An **evidence** 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 sub-tuple 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. It is an evidence of $t=\langle$Jerry Yang, Yahoo!$\rangle$ for predicate p_1, where Jerry Yang spans from position 3 to 4 and the two phrases ("Stanford" and "graduate") are at positions 0 and 2. The evidence is represented as $\langle 9, 8, \{\langle 3, 4\rangle\}, \{0, 2\}\rangle$.

Note that there can be multiple evidences of t_{p_1}, each being a sentence containing Jerry Yang, "Stanford", and "graduate". We denote all evidences of t_p by $\phi_p(t)$. Without loss of generality, we use sentence and evidence interchangeably unless distinction is needed, since we only consider sentence as the evidence context.

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 information provided by $\phi=\{\phi_p | p \in P\}$, 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 *position-based* ranking problem, and any ranking technique relying on ϕ is classified as *position-based ranking*.

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 $F^A(t)$ (the an-

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

swer score) aggregates all predicate scores obtained via F^S and F^R. In this framework, the scores for 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 independent from whether she founded any COMPANY (p_3) and certainly irrelevant to whether a COMPANY is in Silicon Valley (p_2).

The rest of this section proposes our position-based ranking method following this framework.

2.2 Position-Based Features

This section studies three position-based features that are derivable from an evidence. These features are the key components in our Cumulative Model (CM) and Bounded Cumulative Model (BCM) that are introduced later.

2.2.1 Proximity

Intuitively, if the entities in t_p and the keyword phrases in C_p are close to each other in an evidence $s \in \phi_p(t)$, they are likely to belong to the same grammatical unit of the corresponding sentence (e.g., a phrase like *Stanford University graduate Jerry Yang*) and thus form a valid 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 smallest scope in s covering all the entities in t_p and all the phrases in C_p (a scope is a consecutive sequence of tokens in s); and consequently $|scope_p(t_p,s)|$ is the total number of tokens in the scope. Note that the proximity value is in the range of [0,1] by this definition.

Different representations may be used in various places to refer to the same entity and may have different number 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 s.

Example 2: The following two sentences are both evidences of the underlined entities for predicate p_1 in Query 1. Evidence s_1 is a valid evidence, supporting a true positive, while s_4 is invalid, 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 with 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 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 a more valid evidence and therefore, Jerry Yang is more likely to satisfy p_1 than Colin Marlow, given no other evidence.

2.2.2 Ordering Pattern

An ordering pattern refers to the order of entities and phrases in an evidence. Consider predicate p_1=⟨{x}, {"Stanford", "graduate"}⟩ in Query 1. Let c_1 be the first phrase ("Stanford") and c_2 the second ("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, i.e., *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 valid evidences than others. For example, to express that somebody is a graduate of Stanford University, valid evidences often follow the pattern c_1c_2x (e.g., s_1). Those following another pattern, c_1xc_2, are unlikely to be valid (e.g., s_4). To distinguish strong patterns (those that tend to indicate valid evidences) from weak ones, we may assign a different weight to each pattern, so that entities supported by evidences following strong patterns are scored higher. However, it is impossible to pre-determine the weights since the goodness of ordering patterns are predicate-dependent. To illustrate, c_1c_2x is a strong pattern for predicate p_1 in Query 1, but may not be equally good for another predicate p_1'=⟨{x:NOVEL}, {"by", "Jane Austen"}⟩, because it is less common to see an 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 evidences for p. Denoting $\phi_p(o)$ as the subset of evidences 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 assumes that strong patterns appear more often than weak ones. Although in theory it may happen that many invalid evidences follow the same pattern, making a weak pattern more common, we do not observe such cases in our experiments.

Another possible direction is leveraging Machine Learning techniques to predict which patterns lead to better results. While we are also exploring this direction as future work, we note here that one significant challenge of the Machine Learning approach is the need to obtain training data, which can be costly in terms of human effort.

2.2.3 Mutual Exclusion

Given a predicate p, multiple evidences in ϕ_p may have the same ⟨$doc, sent$⟩ value (i.e., come from the same sentence). They are evidences of different entities and may follow different ordering patterns. The co-existence of different patterns in one sentence is called *collision* and the patterns are referred to as *colliding patterns*. The mutual exclusion rule dictates that, when collision happens, at most one colliding pattern is effective and the sentence is only considered evidences following that pattern.

Example 3: The following sentence illustrates mutual exclusion rule for p_1 in Query 1. The sentence appears as three evidences, one for each underlined 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 an evidence of Ric Weiland.

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 an evidence of at least one sub-tuple t_p for predicate p) has a total credit of 1, meaning that there is only one effective pattern. Given a predicate p, denote the colliding patterns in s by $O_p(s)$. 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 sub-tuples 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, i.e., $T_p^*(o,s)$=$\arg\max_{t_p \in T_p(o,s)} prox_p(t_p,s)$. We compare the representatives (and thus the patterns that they follow), by how prominent they are, i.e., by their overall numbers of evidences in ϕ_p. The credit of o in sentence 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 evidences 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 sub-tuple 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 patterns o_1 is Ric Weiland) since he is the only PERSON in s following pattern o_1. 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 evidences ($|\phi_{p_1}(t^1)|$=4) and Paul Allen in 2 ($|\phi_{p_1}(t^2)|$=2). Then, $credit_{p_1}(o_1,s)$=$\frac{4}{4+2}$=0.67 and $credit_{p_1}(o_2,s)$=0.33.

Note that the pattern credit here is different from the weight of pattern in Section 2.2.2. The weight of pattern o is a global measure (aggregating over ϕ_p) of how frequent, and thus how reliable, pattern 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.

2.3 Single-Predicate Scoring

So far, we have introduced all the features for evaluating the validity of an individual evidence. Integrating these features together, this section presents Cumulative Model (CM) for scoring an answer on 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 $F_p^S(t)$ and $F_p^R(t)$.

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

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

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

The model divides $\phi_p(t)$, t's evidences for p, into $|O_p|$ groups, $\{\phi_p(t,o) | o \in O_p\}$, so that evidences 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 strong patterns account more in $F_p(t)$. The kernel of the function, $prox_p(t,s)$ $credit_p(o,s)$, evaluates the validity of s being an 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 evidences having higher proximities and pattern credits will accumulate higher scores and thus ranked higher.

Table 1: 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

It is interesting to note that CM can be customized easily by switching on and off its component features, so that we can evaluate the effectiveness of individual features. While detailed evaluations are presented in Section 3, below we list three important customizations.

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

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

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

2.4 Multi-Predicate Scoring

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

With CM, predicate scores are unbounded, i.e., the more evidences the higher scores. When multiple predicate scores are aggregated, some could be so high that they dominate the aggregate score, which is called *predicate dominance*. To alleviate this problem, we propose Bounded Cumulative Model (**BCM**) as an alternative for scoring predicates:

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

BCM uses the same three features as CM does, but differs from CM in the computation of group scores, each of which is computed from a set of evidences $\phi_p(t,o)$. Basically, BCM bounds all group scores in the range [0,1], and consequently it bounds the predicate scores within [0,1]. Note that $\sum_{o \in O_p} f_p(o)$=1 according to Section 2.2.2.

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,

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

where $F_p(t)$ can be 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 the case in Table 1. The table shows four answers to the query of 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 product, but loses

Table 2: Ten Types from Wikipedia

Type	(E)ntities	(O)ccurrences	O/E
AWARD	1,045	626,340	600
CITY	70,893	28,261,278	389
CLUB	15,688	5,263,865	335
COMPANY	24,191	9,911,372	409
FILM	41,344	3,047,576	74
NOVEL	16,729	1,036,596	63
PERSON	427,974	38,228,272	89
PLAYER	95,347	2,398,959	25
SONG	29,934	732,175	24
UNIVERSITY	19,717	6,141,840	311
TOTAL	742,862	95,648,273	129

the comparison by summation. Answer t_3 gains high scores on p_1 and p_2 (Both are indeed satisfied by t_3.), but low score on p_3 (In reality, it does not satisfy p_3.). However, the final score of t_3 by summation is dominated by the high scoring predicates and thus t_3 is mistakenly ranked above t_2.

3. EXPERIMENTS

In this section, we provide empirical results based on our prototype system over Wikipedia. An online demo is accessible at **http://idir.uta.edu/erq**.

3.1 Data and Query Sets

Our prototype system is built with the 2008-07-24 snapshot of Wikipedia[4] which contains pages for named-entities, categories and administrations. After we removed all the category pages and administrative pages, our corpus has about 2.4 million articles. For each article, we removed all its section titles, tables, infoboxes, references, etc., retaining only the main textual content, which is segmented into sentences and then stemmed using the Porter Stemmer. We defined 10 entity types (Table 2) and assigned about 0.75 million articles to these types based on 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. We consider the *internal links*, hyperlinks from one Wikipedia article to other Wikipedia articles, as occurrences of the link targets (entities) and collect nearly 100 million occurrences for the 0.75 million typed entities.

Two query sets are used for experiments, INEX17 and OWN28. INEX17 is adapted from topics used in Entity Ranking track of INEX 2009 [1]. There are 60 topics available in INEX. We adapted those about entities belonging to our predefined 10 types and obtained 17 queries, including 11 single-predicate queries and 6 multi-predicate queries. OWN28 contains 28 manually designed queries, including 16 single-predicate queries and 12 multi-predicate queries.

3.2 Analyzing Alternative Ranking Methods

In this section, we compare and analyze the multiple ranking methods discussed earlier, namely COUNT, PROX, MEX, CM and BCM. All the methods differ in how they compute predicate scores, i.e., $F_p(t)$. For multi-predicate queries, the same aggregate function, product, is used to compute answer scores, $F^A(t)$. We compare these ranking methods using three popular measures: *nDCG*, *MAP*, and *Precision-at-k*.

nDCG (Normalized Discounted Cumulative Gain): The first block in Table 3 shows the average nDCG on single-predicate queries (Single-11), multi-predicate queries (Multi-6), and all queries (All-17) from INEX17. Both MEX and PROX improve over COUNT, by 0.02-0.05 across all three cases. PROX appears to be more effective than MEX. CM and BCM are comparable to PROX on Single-11, but further improve by more than 0.02 on Multi-6. We only observe minor difference between CM and BCM.

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

Table 3: MAP and nDCG on INEX17/OWN28

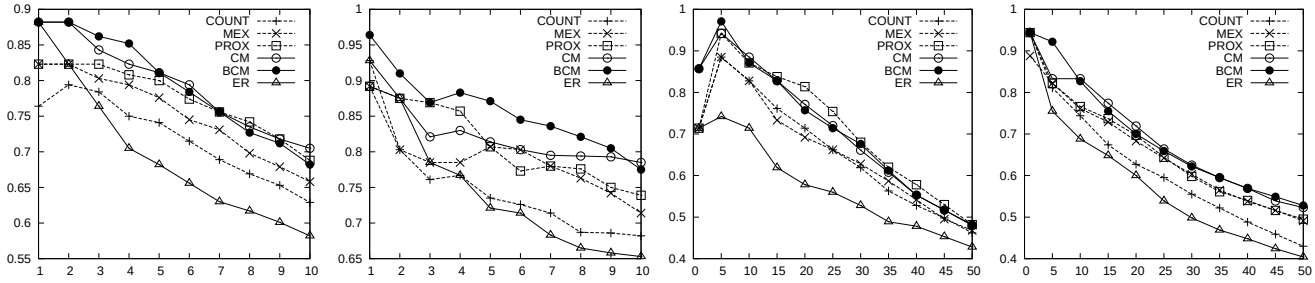
Query	COUNT	MEX	PROX	CM	BCM	ER
nDCG on INEX17						
Single-11	0.889	0.911	0.920	0.920	0.920	0.904
Multi-6	0.880	0.918	0.932	0.954	0.958	0.927
All-17	0.886	0.913	0.924	0.932	0.933	0.912
MAP on INEX17						
Single-11	0.756	0.812	0.843	0.844	0.842	0.779
Multi-6	0.772	0.820	0.852	0.885	0.894	0.809
All-17	0.762	0.815	0.846	0.859	0.860	0.790
nDCG on OWN28						
Single-16	0.917	0.943	0.947	0.953	0.954	0.923
Multi-12	0.800	0.812	0.836	0.844	0.878	0.781
ALL-28	0.867	0.887	0.899	0.906	0.922	0.862
MAP on OWN28						
Single-16	0.758	0.825	0.838	0.858	0.853	0.760
Multi-12	0.579	0.620	0.660	0.684	0.748	0.521
ALL-28	0.681	0.738	0.762	0.783	0.808	0.658

MAP (Mean Average Precision): The second block of Table 3 shows the MAP on INEX17. The observations are mostly similar to those from the nDCG analysis. Note that a larger distinction between CM and BCM is observed on Multi-6, with BCM about 0.01 better than CM.

For further investigation, we repeat the above experiments on OWN28 and provide the results in the bottom half of Table 3. Most results are consistent with INEX17. However, on multi-predicate queries in OWN28 (Multi-12), BCM shows clear advantage over CM in terms of both nDCG (by 0.034) and MAP (by 0.064). The different observations on INEX17 and OWN28 is because, we believe, OWN28 has more multi-predicate queries than INEX17 and the advantage of BCM is more stably observed on OWN28.

Precision-at-k: According to the best reported MRR (Mean Reciprocal Rank) of existing entity search systems [6, 7], the first true answer is typically ranked at top 1-2. To further analyze how different methods perform in detail, especially beyond the top-few answers, we plot precision-at-k curves. Figure 1(a,b) show the results for k=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 nDCG and MAP analysis. BCM is consistently the best among all, while CM has inconsistent performance on INEX17 and OWN28. Figure 1(c,d) show the results for k=50. The curve for each method shows the average precision of the method at rank position k for queries that returned 50 or more answers, including 7 queries in INEX17 and 18 in OWN28. In Figure 1(c), CM and BCM excel before k=10 and BCM is slightly better. PROX is the best after k=10 but is significantly worse than BCM at top ranks. In Figure 1(d), BCM is clearly the best among all, although a little worse than CM between 10 and 25.

In summary, the individual features are effective for entity ranking and they work best in concert when they are integrated into CM and BCM. BCM rivals CM on single-predicate queries, but excels on multi-predicate queries be-



(a) $k = 10$ on INEX17 (b) $k = 10$ on OWN28 (c) $k = 50$ on INEX17 (d) $k = 50$ on OWN28

Figure 1: Precision-at-k on INEX17/OWN28

cause BCM alleviates the predicate dominance problem. Besides, it achieves good precisions consistently across top-50.

3.3 BCM vs. Other Entity Ranking Methods

This section compares BCM with three state-of-the-art entity ranking methods: *EntityRank (ER)*, *INEX* and *INRIA*. All of these systems used Wikipedia as corpus and entity repository, though INEX and INRIA used different snapshots than ours.

EntityRank (ER) [7] focuses on single-predicate queries. It outperforms another closely related method [6] by a large margin, in term of MRR. We re-implemented ER as a plugin for scoring individual predicates in our ranking framework. The same aggregate function, product, is used to compute answer scores (F^A) for multi-predicate queries.

The detailed performance of ER is shown in Table 3 and Figure 1. In Table 3, both CM and BCM outperform ER by large margins. The peak margin (0.22) in terms of MAP is observed on Multi-12 from OWN28, between BCM and ER. In Figure 1, ER rivals PROX, CM, and BCM at top-2, verifying the high MRR reported in [7]. However, it deteriorates very fast when $k > 2$, dropping below 0.7 around k=5, while BCM remains above 0.7 even at k=10.

INEX Entity Ranking track [1] focuses 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, thus their MAPs may tend to be low. The MAP achieved by the best system participating in the 2009 track is 0.517. To avoid the overhead of assessing participating systems, INEX used a sampling strategy to estimate their MAPs.

INRIA [13] works on the same problem as INEX. Unlike INEX participants, it is not based on co-occurrence of entities and query inputs. Rather, it ranks entities by 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 on INEX17 is 0.860. We acknowledge that this comparison is not strictly fair. First, the results are based on different query sets (INEX17 is a subset of INEX Entity Ranking topics) 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.

4. CONCLUSION

In this paper, we studied the ranking problem for general entity-relationship queries. We introduced a ranking framework and explored three position-based features that are integrated in our Cumulative Model (CM) and Bounded Cumulative Model (BCM) for ranking. CM and BCM outperform state-of-the-art techniques by providing more accurate answers beyond top-few answers.

5. REFERENCES

[1] INEX 2009 entity-ranking track.
[2] TREC 2009 entity track: Searching for entities and properties of entities.
[3] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *DL*, 2000.
[4] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB*, 1998.
[5] M. J. Cafarella, C. Ré, D. Suciu, O. Etzioni, and M. Banko. Structured querying of Web text. In *CIDR*, pages 225–234, 2007.
[6] S. Chakrabarti, K. Puniyani, and S. Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *WWW*, 2006.
[7] T. Cheng, X. Yan, and K. C.-C. Chang. EntityRank: searching entities directly and holistically. In *VLDB*, pages 387–398, 2007.
[8] E. Chu, A. Baid, T. Chen, A. Doan, and J. Naughton. A relational approach to incrementally extracting and querying structure in unstructured data. In *VLDB*, pages 1045–1056, 2007.
[9] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured Web community portals: a top-down, compositional, and incremental approach. In *VLDB '07*, 2007.
[10] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the Web. *Commun. ACM*, 51(12):68–74, 2008.
[11] E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar semantic search: a database approach to information retrieval. In *SIGMOD*, pages 790–792, 2006.
[12] D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In *CIKM*, 2007.
[13] A.-M. Vercoustre, J. A. Thom, and J. Pehcevski. Entity ranking in wikipedia. In *SAC*, 2008.
[14] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi. Ranking very many typed entities on Wikipedia. In *CIKM '07*, 2007.
[15] M. Zhou, T. Cheng, and K. C.-C. Chang. Data-oriented content query system: searching for data into text on the web. In *WSDM*, 2010.