Concept Expansion Using Web Tables

Chi Wang, Kaushik Chakrabarti, Yeye He,
Kris Ganjam, Zhimin Chen, Philip A. Bernstein
Microsoft Research, Redmond, WA
{chiw, kaushik, yeyehe, krisgan, zmchen, philbe}@microsoft.com

ABSTRACT

We study the following problem: given the name of an ad-hoc concept as well as a few seed entities belonging to the concept, output all entities belonging to it. Since producing the exact set of entities is hard, we focus on returning a ranked list of entities. Previous approaches either use seed entities as the only input, or inherently require negative examples. They suffer from input ambiguity and semantic drift, or are not viable options for ad-hoc tail concepts. In this paper, we propose to leverage the millions of tables on the web for this problem. The core technical challenge is to identify the "exclusive" tables for a concept to prevent semantic drift; existing holistic ranking techniques like personalized PageRank are inadequate for this purpose. We develop novel probabilistic ranking methods that can model a new type of table-entity relationship. Experiments with real-life concepts show that our proposed solution is significantly more effective than applying state-of-the-art set expansion or holistic ranking techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.2.8 [**Database Management**]: Database applications—*Data mining*

Keywords

Concept Expansion; Web Table; Graph-based Ranking; Entity Extraction; Ontology Learning

1. INTRODUCTION

We study the following problem: *given the name of a concept as well as a few seed entities belonging to the concept, output all entities belonging to the concept.* This is referred to as the *concept expansion* problem. This problem has many applications.

• *Knowledgebase expansion*: A knowledgebase (e.g., Freebase [7], YAGO [25], Probase[32]) contains concepts (e.g., country), entities belonging to those concepts (e.g., USA, China, India for the concept 'country') and attributes of those concepts (e.g., population, capital city for the concept 'country'). They are used by web search engines for a variety of purposes: to answer search queries, understand short texts and annotate web tables [1, 27, 28]. While these knowledgebases have large numbers of entities for head concepts (e.g., country, city), they have very few entities for tail concepts (e.g., 'low-fertility country', 'emerging economies') [32, 33, 31]. There is a growing interest to extend the knowledgebase to cover tail concepts, entities and attributes.

• *Ad-hoc list creation*: Users often create lists for concepts of interest (e.g., dog breeds, beach vacation spots) using a spreadsheet (Excel, Google Sheets) or a note-taking application (e.g., OneNote, Evernote). Manually creating such lists is labor-intensive; approaches that automatically populate such lists or at least suggest some lists can be immensely beneficial.

Since producing the exact set of entities belonging to a concept is hard, a realistic goal is to return a ranked list of entities instead. This requires human effort to examine and admit the top-ranked entities, but still significantly less burdensome than manual population. Better the ranking, lower the human effort.

Prior work and limitation. Though abundant work has been done for the general concept expansion or set expansion problem, they either perform poorly or are not viable options for tail concepts. Set expansion approaches take a small set of "seed entities" as input, to discover 'similar' entities that appear in similar context [18, 29, 30, 16, 24, 21]. When expanding tail concepts, this approach is subject to the ambiguity in the input: given the seed entities 'Canon', 'Sony' and 'Nikon', it is difficult to know which concept the user has in mind, 'camera brands' or 'Japanese companies' or something else. It tends to mix entities belonging to different concepts during the expansion. This is known as the 'semantic drift' issue [11]. To prevent concept ambiguity and semantic drift, a common strategy is to populate negative examples from mutually exclusive concepts according to a reference ontology. This idea is used by Snowball [4], KnowItAll [15], NELL [10] etc. A similar setting is hard to set up for ad-hoc concepts. There is no reference ontology to provide comprehensive mutual-exclusiveness for constraining most tail concepts.

Our solution. To overcome these limitations, we take both a concept name and a small set of seed entities as input, but do not require negative examples or knowledge of other concepts. Furthermore, we propose to leverage *both structured data and text data* associated with millions of tables on the web. This setting has several unique benefits for expansion of tail concepts. First, a web table is often structured as rows about entities pertaining to a coherent, fine-grained concept. Second, the text mentioned in the caption, nearest headings etc. of a web table can help identify the tables about the user-input concept, providing opportunities for indexing and searching them with a web table search system (WTS) like Microsoft's Excel Power Query [3] or Google's Web Tables [2].

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741644.

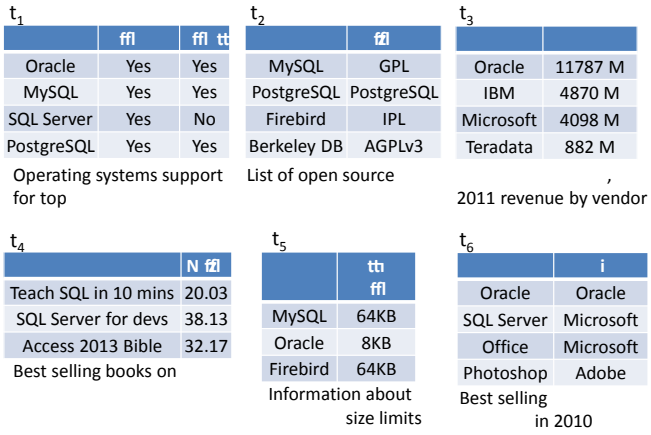


Figure 1: Tables returned by WTS for 'database software'. The text below the tables represent the caption/surrounding text/nearest headings

Once we retrieve the web tables from a WTS, we have two kinds of information at hand: (i) the seed entities and (ii) a ranked list of tables returned by the WTS that *roughly* reflects their relevance to the concept query. A viable approach of utilizing this information is to build a bipartite graph of all retrieved tables and the entities in them, and then run a graph-based ranking method like generalized Co-HITS [13]. Such a method can utilize prior knowledge on both the entity and table side, as well as the link structure. However, a traditional ranking method may fail tail concept expansion due to the abundance of *non-exclusive* tables.

EXAMPLE 1. *Consider the concept 'database software'. Figure 1 shows 6 tables returned by the WTS. The matching keywords found in column headers, captions, surrounding text etc., are shown in bold. The ranking order is* $t_1, t_2, t_3, t_4, t_5, t_6$*:* t_1, t_2, t_3 *and* t_4 *are ranked higher as they match both keywords while* t_5 *and* t_6 *are ranked lower as they match with only one keyword.*

Each table in such a system has a subject column. This column contains the set of entities the table is about, while the other columns represent binary relations or attributes of those entities; previous techniques can be used to accurately identify the subject column (the leftmost column in all the 6 tables in this case) [27, 28].

In this paper, we refer to an entity that belongs to the concept as relevant and others as irrelevant. A WTS returns two types of tables: those that exclusively contain relevant entities in their subject column (t_1*,* t_2*,* t_5*) and those that do not (*t_3*,* t_4*,* t_6*). We refer to the former tables as exclusive and the latter as non-exclusive.*

A bipartite graph can be built as in Figure 2. Suppose there are 4 seed entities (shown in bold). A generalized Co-HITS algorithm ranks entities and tables in the following order: Oracle, MySQL, SQL Server, Teradata, PostgreSQL, Firebird, Microsoft, IBM, Photoshop, Office, Berkeley DB...; $t_3, t_1, t_6, t_2, t_5, t_4$*. Note that irrelevant entities like Microsoft, IBM, Photoshop and Office are ranked above a relevant entity like Berkeley DB.*

Non-exclusive tables cause existing random-walk ranking algorithms to fail. Large non-exclusive tables about overlapping concepts (e.g., t_6*) can accumulate high scores. Irrelevant entities that are popular in such non-exclusive tables will get higher scores than relevant but less popular entities, such as Berkeley DB. These relevant but less popular entities are often desirable for the application of concept expansion, since popular entities have a high chance to exist in a knowledgebase already. This analysis applies to all graph-based ranking methods that linearly aggregate scores from all neighboring vertices.*

Our insight to resolving this semantic drift issue, is to model the *exclusivity* of tables. Using only exclusive tables to find relevant entities vastly boosts precision, without substantial loss of recall because: (i) it is common to retrieve from the WTS a large number of exclusive web tables for a tail concept such as t_1, t_2 and t_5 in Example 1; and (ii) most relevant entities appear in exclusive tables. *The core technical challenge is to identify exclusive tables.*

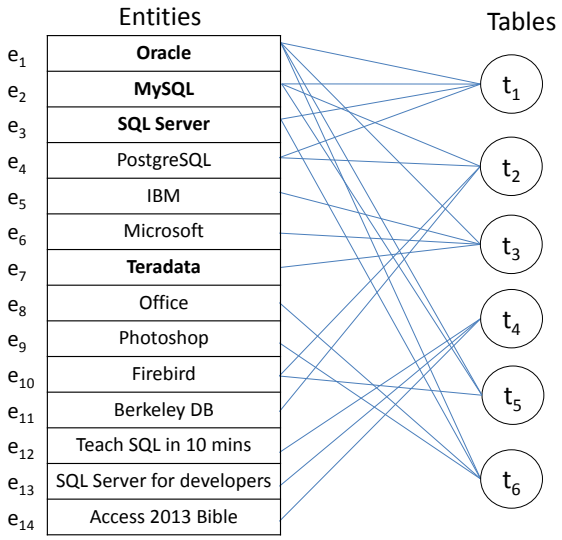


Figure 2: Bipartite graph of entities and tables. The seed entities are shown in bold

To infer exclusive tables and find relevant entities, our solution stems from the simple yet special entity-table relationship: *a table is exclusive if and only if all its subject entities are relevant, and an entity is relevant if it appears in at least one exclusive table.* Incorporating uncertainty, we propose a novel probabilistic model that holistically ranks the entities and tables. Our ranking method models the score of a table as 'how likely the entire table is exclusive' instead of 'the fraction of the table that is relevant,' and respects the asymmetric entity-table relationship: an exclusive table must contain only relevant entities while a relevant entity can appear in a non-exclusive table. The score from table to entity and entity to table is propagated in an asymmetric way, and aggregated nonlinearly to emulate taking 'all or none' from exclusive or non-exclusive tables. That is a key difference of our approach.

Our contributions can be summarized as follows:

- We propose a new problem setting to address the challenges of tail concept expansion using both the concept name and seed entities. We propose to leverage structured data and text data of millions of web tables for this problem (Section 2).
- We propose a novel holistic, probabilistic ranking model to rank entities based on their likelihood of belonging to the concept, and rank tables based on their likelihood of being exclusive to the concept. Specifically, we present properties that the score aggregation functions must satisfy in order to model the special table-entity relationship (Section 3). We develop a novel algorithm to perform the ranking, with guaranteed convergence for a broad class of aggregation functions (Section 4). We present concrete choices of aggregation functions and prior knowledge (Section 5).
- We perform extensive experiments on expanding a large scale concept-entity database. Our experiments show that our method produces significantly better quality results than three existing algorithms: a competitive set expansion algorithm using web tables, a probabilistic graphical model that propagates table scores via column similarities, and a generalized algorithm of personalized PageRank and Co-HITS. Our technique can increase the number of entities per concept by 1 to 2 orders of magnitude with close to 100% precision (Section 6).

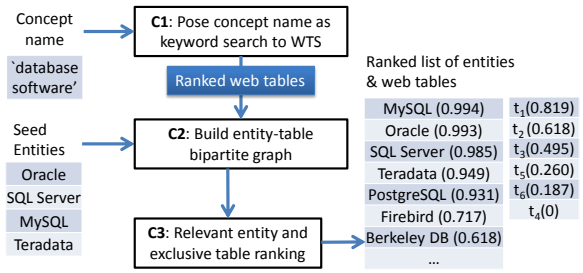


Figure 3: System architecture

Table 1: Example concepts and returned entities

asian country	third world country	trading partner
internet company	multiplayer game	national newspaper
adverse side effect	fitness class	flu-like symptom
antifungal agent	bacterial species	cytotoxic agent
grape variety	mammal species	memory card

Concept	Seed entities	Returned entities
adverse side effect	addiction, depression	headache, nausea, diarrhea, insomnia, dizziness, constipation...
grape variety	nebbiolo, barbera	ramisco, cornifesto, tinta carvalha, espadeiro, tinta barroca, tinta francisca...

2. PROBLEM STATEMENT AND SYSTEM ARCHITECTURE

2.1 System task

The end-to-end system task is: *given a concept name, seed entities and web tables, return a ranked list of relevant entities*. Recall that we refer to the entities belonging to the concept as relevant entities, and exclusive tables are defined as tables that contain only relevant entities in their subject columns.

This task is challenging for an ad-hoc tail concept. A tail concept can be defined as concepts with web mention frequency below a cut point, which partitions the sum of concept frequencies roughly into two equal parts. For example, the most frequent 350 concepts in Probase occupy 40% of the total frequency of all concepts. Any concept with lower frequency than them can be deemed a tail concept. This definition of tail concepts is similar to that of tail attributes by Yahya et al. [33].

Formally, we let C denote the concept name, S the set of seed entities, $T = \{t_j\}_{j=1}^{m}$ the set of web tables, $E = \{e_i\}_{i=1}^{n}$ the set of entities appearing in the subject columns of these web tables, and $\{x_i\}_{i=1}^{n}$ the entity ranking score indicating their relevance.

For convenience, we also use C to denote the set of relevant entities in the concept, and t_j the set of entities mentioned in the subject column of the table. The context disambiguates which meaning is used. For example, $e_i \in C$ means that entity e_i belongs to concept C, $e_i \in t_j$ means that entity e_i is mentioned in t_j's subject column, and $t_j \subset C$ means that t_j is an exclusive table.

We assume the seed entity set S is a subset of E; we ignore any seed entity not present in E.

2.2 System architecture

Figure 3 shows the architecture of the CONCEPTEXPAND system. It has 3 main components, namely C1, C2 and C3.

(C1) Pose concept name to Web Table Search system: This component retrieves the set of web tables from which the relevant entities can be identified. A quality set of retrieved tables should comprise (i) most of the tables containing relevant entities and (ii) few tables that contain no relevant entities at all.

One option is to develop custom algorithms for this purpose. A plausible algorithm is to obtain the tables whose subject column name matches with the concept name. While this works well for broad concepts (e.g., 'city', 'country'), it misses most tables for specific concepts. For example, we tried this algorithm for the 15 concepts shown in Table 1 (with both singular and plural variants). In a large fraction of the web snapshot, we found only 73 tables.

We resort to the rich content signals associated with web tables. For tables we aim to retrieve, the concept name typically appears in captions, nearest headings (h2, h3, h4), page titles, surrounding text, incoming anchor text and queries for which the page was clicked. These are the general criteria used by a web table search system (WTS) to return tables by keyword queries [27, 9]. Hence, we pose the ad-hoc concept name to a WTS and use the returned list of tables for further investigation. To obtain a reasonable amount of exclusive tables, we retrieve a large number of tables from WTS (such as 1,000). WTS rank positions are based on relevance, and they can be used as distant prior knowledge for exclusivity-aimed ranking. In the following, we assume t_j refers to the table ranked at position j by this WTS. We assume the WTS outputs the identity of the subject column for each returned table. All WTSs internally identify it by certain means [27, 28].

(C2) Build an entity-table bipartite graph: The inputs to this component are the set of seed entities and the tables returned by component C1. It identifies the set of distinct entities among the entity mentions in the subject columns of the input tables. It also identifies the seed entities among those distinct entities. In general, this is the entity resolution problem [6]. Since this is not the focus of the paper, we identify the distinct entities by simply grouping the entity mentions in the subject columns using a string matching module. We identify the seed entities amongst the distinct entities in the web tables using the same string matching module. The output of this component is a bipartite graph. It comprises the set of distinct entities on one side (with the seed entities identified) and the set of tables returned by WTS on the other side. An edge between an entity and a table indicates that the former is mentioned in the subject column of the latter. The bipartite graph for the example in Figure 1 is shown in Figure 2. Let $T(e_i) = \{t_j \ni e_i\}$ be the set of tables that are linked to e_i. $T(e_i)$ is referred to as *support table set*, and $|T(e_i)|$ as *support* of e_i. For example, $T(e_1) = \{t_1, t_3, t_5, t_6\}$.

(C3) Relevant entity and exclusive table ranking: The input to this component is the bipartite graph output by component C2, with seed entities and WTS table ranking included. The output is a ranked list of entities according to their relevance as well as a ranked list of tables according to their exclusivity. We perform ranking for both entities and tables because they are highly interdependent, as we will discuss in Section 3. The ranked list of tables can also complement the ranked list of entities and facilitate the investigation. Furthermore, the tables provide opportunities of adding relational attributes to the knowledgebase or spreadsheets, which is an important goal of knowledgebase expansion and ad-hoc list creation. It is formally defined as follows.

DEFINITION 1. *Given a concept C, ordered web tables $T = \{t_j\}_{j=1}^{m}$ that cover entities $E = \{e_i\}_{i=1}^{n}$, and a seed entity set $S \subset E$, rank these entities with score $\{x_i\}_{i=1}^{n}$, and the tables with score $\{y_j\}_{j=1}^{m}$, according to entities' relevance and tables' exclusivity, i.e., whether $e_i \in C$ and $t_j \subset C$.*

This component is the focus of the rest of the paper.

3. RANKING MODEL

We first state the principles for entity and table ranking. We then propose a probabilistic model following the principles.

Finding relevant entities and exclusive tables are highly interdependent tasks and can mutually enhance each other. It is due to the following relationship between them:

PRINCIPLE 1. *A table t_j is exclusive if and only if all the entities $e_i \in t_j$ are relevant.*

PRINCIPLE 2. *An entity e_i is relevant if it appears in at least one table t_j such that t_j is exclusive.*

Principle 2 is actually a straightforward corollary of Principle 1. We restate it in order to emphasize the assymetric relationship: An exclusive table must contain only relevant entities, but a relevant entity can appear in non-exclusive tables.

If we know either the complete set of relevant entities or the complete set of exclusive tables, we can leverage the two hard principles to obtain the other set. For example, if we know which entities belong to the concept C, we can deduce which tables belong to concept C using Principle 1. Conversely, we can deduce which entities belong to concept C using Principle 2 when the knowledge of tables is given. Note that in the latter case, false negatives are possible because some relevant entities may only appear in non-exclusive tables. Yet we assume this rarely happens given the large sample size of web tables and a reasonable performance of WTS

In reality, we do not have the complete knowledge on either side. The input provides partial prior information on both sides. If $e_i \in S$ is a seed entity, we have strong prior knowledge for e to belong to the concept C. If table t_j is ranked high (j is small) by WTS, we have weak prior knowledge for t_j to belong to concept C. The table prior is weak because the WTS does not rank tables according to their exclusivity. We can only assume that the WTS ranking is overall positively correlated with the exclusivity.

With the unavoidable uncertainty, we model the problem as a holistic entity and table ranking task that incorporates *soft counterparts* of the above principles.

Probabilistic ranking model: Let $x_i = p(e_i \in C) \in (0, 1]$ denote the likelihood of entity e_i belonging to concept C, and $y_j = p(t_j \subset C) \in [0, 1]$ denote the likelihood of table t_j belonging to concept C. We let $x_i > 0$ because we cannot assert an entity does not belong to concept C for sure, as we do not have negative evidence. We then model their relationship with soft counterparts of Principle 1 and 2. According to Principle 1, we model each y_j as an aggregation function $f_{E\to T}$ of $\{x_i, e_i \in t_j\}$. According to Principle 2, we model each x_i as an aggregation function $f_{T\to E}$ of $\{y_j, t_j \ni e_i\}$. Then, we solve the following equations to perform holistic entity and table ranking.

$$\begin{aligned} x_i &= f_{T\to E}(\{y_j, t_j \ni e_i\}) \\ y_j &= f_{E\to T}(\{x_i, e_i \in t_j\}) \end{aligned} \tag{1}$$

It is easy to incorporate the prior knowledge on entity and table side by adding pseudo tables and entities in the bipartite graph, which will be discussed in Section 5.2.

As we discussed in Section 1, symmetric, linear aggregation functions, like the random walk employed by PageRank, Co-HITS, etc. suffer from the semantic drift issue. The reason is that they do not consider the special relationship we are modeling between relevant entities and exclusive tables through Principle 1 and 2. We now discuss the design principle of the two aggregation functions $f_{E\to T}$ and $f_{T\to E}$, both of which should produce values between 0 and 1.

3.1 Entity to table aggregation function

Given x_i for $e_i \in E$, we model $y_j = p(t_j \subset C)$ as an aggregation function $f_{E\to T}$ of $\{x_i | e_i \in t_j\}$. This is the soft counterpart of the relationship expressed in Principle 1.

Principle 1 suggests that as long as one entity $e_i \in t_j$ is not in concept C, the whole table t_j is not in concept C. In our probabilistic model, we would like $f_{E\to T}$ to reflect the likelihood that all the entities in t_j belong to C. It should produce a low value if any $x_i | e_i \in t_j$ is low. Also, it should well distinguish tables that have small difference. We abstract the following two axiomatic properties:

1. Consider a table t_j. For any entity $e_i \in t_j$, its likelihood of belonging to C is an explanatory variable of the likelihood t_j belonging to C. When it is almost certain that the entity e_i does not belong to C, the table t_j is also almost certain to be non-exclusive. In contrast, if all the entities in t_j are certain to be relevant, the table t_j is also certain to be exclusive.

   PROPERTY 1 (ASYMPTOTIC PRINCIPLE 1). *$\forall e_i \in t_j$, $\lim_{x_i \to 0} y_j = 0$; Let $\mathbf{x} = [x_i | e_i \in t_j]$, $y_j = 1$ iff $\mathbf{x} = \mathbf{1}$.*

2. If two tables t_a and t_b are identical except two entities with equal support, the table containing the entity with a higher likelihood should have a higher likelihood of belonging to the concept (unless the likelihood for both is equal to zero).

   PROPERTY 2 (MONOTONICITY). *If $t_a = t \cup \{e_1\}, t_b = t \cup \{e_2\}$, $|T(e_1)| = |T(e_2)|$ and $x_1 < x_2$, then $0 \le y_a \le y_b \le 1$ ($y_a = y_b$ happens only when $y_b = 0$).*

As one example, the minimum value from $\{x_i | e_i \in t_j\}$, i.e.,

$$f_{E\to T}(\{x_i | e_i \in t_j\}) = \min_{e_i \in t_j} x_i \tag{2}$$

satisfies Property 1, but not Property 2. It is determined by a single element x^* in the set $\{x_i | e_i \in t_j\}$ and insensitive to other elements as long as they are not smaller than x^*.

EXAMPLE 2. *Consider the two tables t_1 and t_3 in Figure 1. t_1 contains 4 entities $e_1 = Oracle$, $e_2 = MySQL$, $e_3 = SQL\ Server$ and $e_4 = PostgreSQL$ and t_3 contains 4 entities $e_1 = Oracle$, $e_5 = IBM$, $e_6 = Microsoft$ and $e_7 = Teradata$. Suppose $x_1 = x_2 = x_3 = 0.9, x_4 = x_5 = x_6 = x_7 = 0.1$. Then Equation (2) produces identical y_j value for these two tables. But t_1 should be deemed more likely to belong to C than t_3 as it has more relevant entities.*

As another example, the random walk aggregation function, i.e.,

$$f_{E\to T}(\{x_i | e_i \in t_j\}) = \sum_{e_i \in t_j} \frac{x_i}{|T(e_i)|} \tag{3}$$

violates Property 1.

3.2 Table to entity aggregation function

Given y_j for $t_j \in T$, we model $x_i = p(e_i \in C)$ as an aggregation function $f_{T\to E}$ of $\{y_j | t_j \ni e_i\}$. This is the soft counterpart of the relationship expressed in Principle 2.

Principle 2 suggests that if an entity appears in one exclusive table, it is relevant. In our probabilistic model, we would like $f_{T\to E}$ to reflect the likelihood that at least one table $t_j \ni e_i$ in e_i's support set belongs to C. It should produce a high value if any $y_j (t_j \ni e_i)$ is high. Also, it should well distinguish entities that have slightly different support table sets. We abstract the following two axiomatic properties:

1. Consider an entity e_i and its arbitrary support table t_j. When the table t_j is certain to belong to C, the entity e_i is also certain to belong to C.

   PROPERTY 3 (PRINCIPLE 2). *$x_i = 1$ if $\exists t_j \ni e_i, y_j = 1$.*

   Note that Property 3 is unlike Property 1 in two aspects: (i) the limit ($\lim_{\mathbf{y}\to\mathbf{0}} x_i = 0$) is not true, because a relevant entity could occur in non-exclusive tables; and (ii) $x_i = 1$ if but not only if $\exists t_j \ni e_i, y_j = 1$, for the same reason.

2. If the support table sets of two entities e_a and e_b are identical except two equal sized tables, the entity contained in the table with a higher likelihood should have a higher likelihood of belonging to the concept (unless the likelihood for both is 1).

   PROPERTY 4 (MONOTONICITY). *If $T(e_a) = T_0 \cup \{t_1\}$, $T(e_b) = T_0 \cup \{t_2\}$, $|t_1| = |t_2|$ and $y_1 < y_2$, then $0 < x_a \leq x_b \leq 1$ ($x_a = x_b$ only when $x_a = 1$).*

As one example, the maximal value from $\{y_j | t_j \ni e_i\}$, i.e.,

$$f_{E\to T}(\{y_j | t_j \ni e_i\}) = \max_{t_j \ni e_i} y_j \tag{4}$$

satisfies Property 3 but violates Property 4.

As another example, the random walk aggregation function, i.e.,

$$f_{T\to E}(\{y_j | t_j \ni e_i\}) = \sum_{t_j \ni e_i} \frac{y_j}{|t_j|} \tag{5}$$

violates Property 3.

The concrete design of the aggregation functions following these principles will be presented in Section 5.1.

4. RANKING ALGORITHM

Based on the model proposed in Section 3, we develop an algorithm to perform joint inference for entity likelihood x_i and table likelihood y_j.

A simple idea is to find a solution that satisfies Equation (1), by iteratively applying the aggregation functions in Equation (1). This simple algorithm may fail to rank the relevant entities and exclusive tables ahead of irrelevant and non-exclusive ones when the following are both true: (i) there are a large number of tables in our bipartite graph that belong to a different concept C'; and (ii) these tables contain a similar set of entities. For example, in Figure 1, if there are many tables like t_4, the entities and tables about database books can amass high scores from each other.

To address this issue, we develop an algorithm, called CONCEPTEXPAND (Section 4.1). We give a formal theorem about the convergence condition of our algorithm in Section 4.2.

4.1 CONCEPTEXPAND algorithm

The main idea is to perform restricted propagation on a subset of entities and tables, instead of propagating scores among all entities and tables. We consider a two-phase algorithm. In the first phase, we identify such a subset. We begin with the seed entities, which are most likely to be relevant. We then gradually expand the set by iteratively adding the most plausible tables based on current estimation. We stop adding tables when the remaining tables all have low estimated likelihood of belonging to C. We collect entities with enough support, and remove tables with few entities in this collection. In the second phase, the iterative propagation will be performed within this set. The two-staged method prevents adding too many irrelevant tables all at once and impairing score propagation. During the first phase, only those high-confidence tables contribute to entities' likelihood computation. And the second phase will have a chance to refine the earlier estimation which was based on incomplete information.

As the starting point of the probabilistic reasoning, the score for the seed entities can be set according to the prior knowledge (i.e., they are very likely to be in C). For all the other entities, x_i is unknown, and they will not be computed until a table containing them is added to the reasoning set. To compute the score for a table t_j with missing entity score, we use the table prior π_j to replace the missing entities' score and feed to the aggregation function $f_{E\to T}$. The more knowledge we have of entities in a table, the less important the table prior is.

Algorithm 1: CONCEPTEXPAND

Input: Entity set E, table set T, seed entity set S, table likelihood threshold α
Output: entity score $\{x_i | e_i \in E\}$, table score $\{y_j | t_j \in T\}$

1 Initialize the support table set $T_0 \leftarrow \emptyset$;
2 Initialize the excluded entity set $U \leftarrow E \setminus S$;
3 Initialize seed entity score x_i using prior, for $e_i \in S$;
4 **repeat**
5 Update $y_j \leftarrow f_{E\to T}(\{x_i | e_i \in t_j, e_i \notin U\} \cup \{\pi_j | e_i \in t_j \cap U\})$ for $t_j \in T$;
6 Choose the optimal $(y^*, t^*) = \max_{t_j \notin T_0, |t \setminus U| > 0} y_j$;
7 $T_0 \leftarrow T_0 \cup \{t^*\}$;
8 Update $x_i \leftarrow f_{T\to E}(\{y_j | t_j \ni e_i, t_j \in T_0\})$ for $e_i \in t^*$;
9 $U \leftarrow U \setminus t^*$;
10 **until** $y^* < \alpha$ *or* $T_0 = T$;
11 **repeat**
12 $T_0 \leftarrow T_0 \setminus \{t \in T_0, |t \setminus U| < |t \cap U|\}$;
13 $U = U \cup \{e \in E \setminus U, |T(e) \cap T_0| \leq 1\}$;
14 **until** U *stops growing*;
15 **repeat**
16 Update $y_j \leftarrow f_{E\to T}(\{x_i | e_i \in t_j\})$ for $t_j \in T_0$;
17 Update $x_i \leftarrow f_{T\to E}(\{y_j | t_j \ni e_i\})$ for $e_i \in E \setminus U$;
18 **until** x_i*'s converge*;

We develop the CONCEPTEXPAND algorithm based on the above intuition. The pseudocode is shown in Algorithm 1. Lines 4–14 identify the table set and entity set for reasoning, by adding the most plausible table one by one, updating the estimation of both tables and entities, and removing the ones with low support. When adding a table, we require that the table contains some entity in the reasoning set (Line 6). This reduces the chance of adding a table simply because it has high prior. When removing tables and entities, we keep checking if any entity has no more than one support table (Line 13), and if any table has more than half of entities missing in the reasoning set (Line 12). Lines 15–17 solves the equations within the reasoning set, using a fixed-point iteration.

The parameter α can be set according to the precision-recall requirement. Lower α leads to higher recall but lower precision in general. When $\alpha = 0$, all the tables will be added to the reasoning set. $\alpha = 1$ filters out all tables. In general, one can set α automatically according to table prior scores $\{\pi_j\}$ (e.g., the median of $\{\pi_j\}$). See Section 6.3 for a study of its effect.

The algorithm has a complexity $O(mL)$, where m is the number of tables and L the total size of tables (i.e., the number of links in the table-entity bipartite graph). We show a running example of the algorithm in Section 5.3.

4.2 Convergence of algorithm

Given that the CONCEPTEXPAND algorithm requires iterative computation of y_j and x_i, a desirable property of the computation is that it converges after a limited number of iterations. In the following, we provide a condition for aggregation functions $f_{E\to T}$ and $f_{T\to E}$ which guarantees convergence.

PROPERTY 5 (DIMINISHING RETURN). *Let $\mathbf{x} = [x_1, x_2, ..., x_n]$ be an n-dimensional vector with entity score x_i as components. Let $g(\mathbf{x})$ be the entity score after one iteration of updates using $f_{E\to T}$ and $f_{T\to E}$. The aggregation generates diminishing return if (i) there exists a compact set $\mathscr{C} \in [0,1]^n$ such that $g(\mathscr{C}) \subseteq \mathscr{C}$, and (ii) $\forall \mathbf{x}, \mathbf{z} \in \mathscr{C}(\mathbf{x} \neq \mathbf{z}), \max_{i\in[n]} |x_i - z_i| > \max_{i\in[n]} |g(\mathbf{x})_i - g(\mathbf{z})_i|$. $\mathscr{C}$ is called the diminishing region of g.*

Discussions. The first part of Property 5 states that the aggregation has to produce proper probabilities in a compact region in $[0,1]^n$ if input are proper probabilities in the same region. The compact region can be $[0,1]^n$ itself, or a subset of it. This is a natural requirement since we want to output probabilities, and in some cases regularize them. For example, random walk requires the probabilities to be within a simplex $\sum_{i=1}^n x_i = 1$.

The second part of the property states that the maximum difference of probability for the same entity in $\mathbf{x}$ and $\mathbf{z}$ always diminishes after one iteration of $f_{E\to T}$ and $f_{T\to E}$. In particular, when we boost the initial belief of one entity by δ, the new belief will increase by less than δ after one iteration. Intuitively, this means that the initial boost will be "averaged out" by other entities and tables after iterative propagations. It is also a natural requirement because the change of inferred confidence should not surpass the change of original confidence.

We can in fact show that the condition holds in many common aggregation functions, when prior knowledge is incorporated as pseudo entities and pseudo tables. One example is to use arithmetic mean as aggregation functions for both $f_{E\to T}$ and $f_{T\to E}$, and add a pseudo entity with likelihood π_j to table t_j as prior knowledge (e.g., based on the ranking position in WTS).

PROPOSITION 1. *Let $f_{E\to T}(\{x_i|e_i \in t_j\}) = \frac{\pi_j + \sum_{e_i\in t_j} x_i}{|t_j|+1}$, and $f_{T\to E}(\{y_j|t_j \ni e_i\}) = \frac{\sum_{t_j\ni e_i} y_j}{|T(e_i)|}$, i.e., both $f_{E\to T}$ and $f_{T\to E}$ are arithmetic mean, then the aggregation generates diminishing return in $[0,1]$.*

There are other combinations of common aggregation functions that also enjoy Property 5, e.g., the random walk aggregation functions as in Equations (3) and (5). ∎

Property 5 guarantees convergence.

THEOREM 1. *If aggregation functions satisfy Property 5 with diminishing region $\mathscr{C}$, then Algorithm 1 converges to the unique solution to Equation* (1) *in $\mathscr{C}$, with initial $\mathbf{x} \in \mathscr{C}$.*

To prove Theorem 1, we can essentially view $g(\mathbf{x})$ as a composite function from $f_{E\to T}$ and $f_{T\to E}$, and apply Banach's Fixed Point theorem [17] to ensure that a unique solution can be found for the system of Equation (1) using iterative computation.

Theorem 1 requires fairly simple properties to guarantee the convergence of CONCEPTEXPAND algorithm. Proposition 1 implies that common aggregation functions, such as smoothed arithmetic mean, guarantees convergence. Therefore, our computational framework is generic – any reasonable aggregation functions satisfying the required properties can be plugged in Equation (1) to model a particular application, and Algorithm 1 will converge to the solution of it.

5. RANKING DESIGN

We now discuss concrete choices of aggregation functions and prior knowledge encoding, which are functions that we plug into the generic framework outlined in Algorithm 1. We design these functions specifically for our problem. At the end of this section we will also present a running example.

5.1 Aggregation function

Entity to table aggregation. As discussed in Section 3.1, we need a function $f_{E\to T} : [0,1]^n \to [0,1]$ that satisfies the desirable Properties 1 and 2.

Property 2 (monotonicity) requires the function to be responsive to the change of every input element positively. Property 1 (asymptotic Principle 1) requires it to approach 0 when any input element approaches 0, and to approach 1 when all input elements approach 1. The three Pythagorean means, arithmetic mean, geometric mean, and harmonic mean satisfy the monotonicity, and the latter two satisfy Principle 1 asymptotically.

Between these two, we choose the one that fits Principle 1 most closely. We reemphasize that the table ranking score reflects whether *all* entities in the subject column of a table are within the concept. We prefer the score to be low even when there are only a small number of irrelevant entities in the table. Since the harmonic mean of a list of numbers tends strongly toward the least elements of the list, it tends (compared with the arithmetic and geometric mean) to mitigate the impact of large numbers (relevant entities) and magnify the impact of small ones (irrelevant entities). As such, we choose the harmonic mean as the aggregation function:

$$f_{E\to T}(\{x_i|e_i \in t_j\}) = \frac{|t_j|}{\sum_{e_i\in t_j} \frac{1}{x_i}} \tag{6}$$

Table to entity aggregation. As discussed in Section 3.2, we need a function $f_{T\to E} : [0,1]^n \to [0,1]$ that satisfies the desirable Properties 3 and 4.

Property 4 (monotonicity) requires the function to be responsive to the change of every input element positively. Property 3 (Principle 2) requires it to approach 1 when any input element approaches 1. The three Pythagorean means satisfy the monotonicity, but none of them satisfy Principle 2.

We derive a meaningful aggregation function from the probabilistic meaning of the score. If we assume the events '$t_j \subset C$' are independent for tables $t_j \in T(e_i)$ in the support set of e_i, we can use the 'noisy-or' model for table-to-entity score aggregation: $e_i \notin C$ only if none of its support tables belong to C.

$$\begin{aligned} f_{T\to E}(\{y_j|t_j \ni e_i\}) &= p(\vee_{t_j\ni e_i} t_j \subset C) \\ =1 - \prod_{t_j\ni e_i} p(t_j \not\subset C) &= 1 - \prod_{t_j\ni e_i} (1 - y_j) \end{aligned} \tag{7}$$

The problem with this model is that it can accumulate weak evidences quickly to produce false positives, as illustrated by the following example.

EXAMPLE 3. *Consider two entities* Office *and* Berkeley DB. Berkeley DB *appears in one table t_2, with $y_2 = 0.9$. Suppose there are 6 other tables identical to t_6 (say, $t_7, \ldots, t_{12}$) and they all contain* Office. *Suppose $y_6 = \cdots = y_{12} = 0.3$. This is realistic because tables about popular yet non-target concepts (e.g., general software in this case) can dominate the WTS results. Equation (7) produces higher likelihood for* Office *than for* Berkeley DB *(0.92 vs 0.9) which is undesirable. Note that there is no table that strongly supports* Office *to be relevant; on the contrary, there is a table (t_2) that strongly supports* Berkeley DB *to be relevant.*

A similar issue is also discussed by Downey et al. [14], in a different scenario. They addressed the problem that when an entity is involved in a pattern for *multiple* times, the 'noisy-or' model will accumulate weak evidence. They proposed a *Urn* model for repetitive sampling of entities that allows replacement. In our case, each entity appears once in each table, so this model does not apply to our problem.

We propose a new heuristic solution. Given the collection of $\{y_j|t_j \ni e_i\}$, a few large values in it should contribute more to the aggregated value x_i than many small values. Based on that intuition, we sort the y_j's in a descending order $y_{j_1} \geq y_{j_2} \geq \cdots$, and assign a decreasing series of weights $1 \geq w_1 > w_2 > \ldots$ to them.

$$f_{T\to E}(\{y_j|t_j \ni e_i\}) = 1 - \prod_{t_{j_u} \ni e_i, y_{j_u} \geq y_{j_{u+1}}} (1 - y_{j_u})^{w_u} \quad (8)$$

So, the small values in $\{y_j|t_j \in T(e_i)\}$ have relatively small contribution to x_i. In other words, the number of support tables of e has a diminishing marginal utility. The more support tables we observe, the less important those small values are. When w_i's are all equal to 1, this function reduces to the 'noisy-or' model.

To determine w_i, we introduce a notion of *effective support*. For simplicity let us assume an entity appears in m tables with equal likelihood q. Then Equation (8) becomes: $f_{T\to E}(\{y_j|t_j \ni e_i\}) = 1 - (1-q)^{\sum_{i=1}^{m} w_i}$. We call $\sum_{i=1}^{m} w_i$ the effective support of x_i. In the 'noisy-or' model, the effective support is equal to the actual support m. By assigning the decaying weight $1 \geq w_1 > w_2 > \ldots$, we desire the effective support to grow sublinearly with m. The question is how slowly the effective support should grow with respect to m. A fast growing example is $\Theta(m^{\frac{m-1}{m}})$, which is close to linear when m is large. A extremely slow growing example is $\Theta(\ln(m))$, which almost does not grow in our scale with thousands of tables. A medium example is $\Theta(\sqrt{m})$, where support 100 is converted to effective support 10. In general, we can set $w_i = \frac{1}{i^p}, 0 < p < 1$, to obtain a asymptotic growth rate $\Theta(m^{1-p})$ of effective support ($p = 0$ and $p = 1$ correspond to linear and logarithmic growth respectively). The sequence $\{w_i\}$ in this case is referred to as p-series.

In this paper, we use the $\frac{1}{2}$-series as the decaying weight w_i. It resolves the issue in Example 3 to a large extent: the undesirable accumulation is much slower, with a square root growth rate of effective support.

PROPOSITION 2. *The rectified 'noisy-or' function with decaying weight, as in Equation* (8)*, satisfies both Property 3 and 4.*

We omit the proof due to limit of space.

5.2 Prior knowledge

Now we discuss how to incorporate prior knowledge for the specific task of concept expansion by adding pseudo entities or tables.

Prior on entity side: For seed entities, we have strong prior knowledge that they belong to concept C. We add one pseudo table for each seed entity $e_i \in S$ and link the pseudo table to the corresponding entity. The table has a fixed confidence score $1 - \epsilon_i$, where ϵ_i is a small error rate depending on the source of the entities. If the entities are provided by human curators, ϵ_i should reflect the error rate of the curators which is typically very low. If the entities come from a non-curated database such as web-extracted concept-entity pairs, ϵ_i should be set according to the precision of the database [32]. Furthermore, the different seed entities may have different confidence of belonging to C. For example, in Probase, an entity e_i has an integer support s_i about its occurrence with C. The larger s_i is, the more confident it is to be a correct seed entity for C. So we can set $\epsilon_i = \epsilon_C^{s_i}$, where ϵ_C is a small constant like 0.1.

Prior on table side: All WTSs return a ranked list of tables for a query. We assume the relevance of a table to the concept C degrades with their position in the ranked list returned by a WTS. Therefore, the prior estimation of the likelihood of a top ranked table belonging to C is higher than that of a lower ranked table. Since we retrieve a constant number of tables from WTS, we may retrieve tables that are partial matches to the concept name (like table t_6 in Figure 1). To account for that, we penalize tables that do not match all the tokens in the concept name. We define the table prior π_j as:

$$\pi_j = (\frac{1}{2})^{\text{\#missing tokens of } C \text{ in } t_j} \frac{1}{j+1} \quad (9)$$

Recall that j is the rank position of table t_j. When a table contains all tokens in C, its prior is $\frac{1}{j+1}$, which decays fast as the ranking position increases. Otherwise, its prior is penalized exponentially with respect to the number of missing tokens. We incorporate the prior by adding a pseudo entity with confidence π_j to link to each table t_j. In this way, the prior is just one value that participates in the aggregation function $f_{E\to T}$. The more knowledge of entities in a table we have, the less we rely on the table prior.

While it is possible to design more sophisticated prior score, we use this simple treatment that is general and easy to compute. The imperfect but reasonable prior score tests the ability of our reasoning framework of handling noise. Better prior score can be deigned in future work to exploit more information in web tables.

It can be verified that the chosen aggregation functions and prior score satisfy the convergence condition in Theorem 1. We leave detailed proofs to the full version of this paper.

5.3 Running example

EXAMPLE 4. *Consider the example in Figure 2. The prior of tables t_1 to t_6 are: 0.5, 0.33, 0.25, 0.2, 0.08, 0.07. For simplicity we set $\alpha = 0, \epsilon_i = 0.1$, and skip Line 13 of entity removal because in larger samples these entities occur more than once.*

Initially, $E \setminus U$ consists of $e_1 = Oracle$, $e_2 = MySQL$, $e_3 = SQLServer$ and $e_7 = Teradata$ and $T_0 = \emptyset$. In the first round, $y_1 = 0.68, y_3 = 0.35, y_5 = 0.15, y_6 = 0.11$. t_1 is selected and added to T_0. x_1 to x_4 are updated: $x_1 = 0.98, x_2 = 0.972, x_3 = 0.971, x_4 = 0.68$. $e_4 = PostgreSQL$ is removed from U. In the second round, t_2 is selected. t_3 is added in round 3, and t_5 in round 4. In round 5, t_6 is the last table to add into T_0, because t_4 has no entity in $E \setminus U$. With that we enter the second phase.

After the second phase, the scores converge to a fixed point. The entity scores from high to low are: MySQL (0.992), Oracle (0.991), SQL Server (0.984), Teradata (0.949), PostgreSQL (0.904), Firebird (0.683), Berkeley DB (0.608), IBM (0.495), Microsoft (0.495), Office (0.187), Photoshop (0.187). And the table scores from high to low are: $y_1 = 0.814, y_2 = 0.608, y_3 = 0.495, y_5 = 0.258, y_6 = 0.187, y_4 = 0$. The entity ranking is desired, while the table ranking has one mistake: t_5 is ranked after t_3. The main reason is that the table prior plays an important role when the table is small: $\pi_3 = 0.25 > \pi_5 = 0.08$. The real table size is typically much larger, and therefore the table prior would not dominate the table ranking as in this example.

6. EXPERIMENTAL EVALUATION

We present an experimental evaluation of the proposed algorithm. The goals of the experimental study are (i) to compare the ranking quality of different approaches in terms of precision and recall, and (ii) to understand the sensitivity of the proposed approach to different parameter settings.

6.1 Experimental Setup

6.1.1 Dataset

We use a production web table search engine (WTS) that indexes over 500M web tables extracted from a recent snapshot of the Microsoft Bing search engine. This is the same WTS that powers web table search in Excel PowerQuery [3]. We use concepts in Probase [32] as the testbed for expansion. We choose Probase instead of knowledgebases like YAGO [25] or Freebase [7] for the sake of better concept coverage, especially for tail concepts [32].

To prepare the set of concept queries, we acquired from Probase 9,926 concepts with sufficient popularity. In Probase, each concept-entity pair has a popularity value indicating the number of different domains it is found on the web. We chose concepts with popularity above 150 so that the concept is not too niche and the web-extracted concept-entity pairs are relatively reliable. Seed entities were sampled based on their popularity. We retrieved 1,000 tables for each concept, ending up with a dataset of 10M web tables.

6.1.2 Compared methods

• **Number of occurrences (# Occur).** In this baseline approach, we rank entities by their number of occurrences in the top ranked tables returned by WTS. Entities that occur more often in top tables are ranked higher.

• **Weighted number of occurrences (Weighted # Occur).** This approach enhances the previous one by weighting each occurrence of entity using the table prior. In addition, in order to utilize seed information, we only consider tables with 2 or more seeds. Entities are finally ranked by their aggregate weight across these tables.

• **SEISA [18].** Set expansion aims to automatically discover entities in the target concept using seed entities but not concept name as input. We implement the Dynamic Thresholding algorithm in the SEISA system [18], which is shown a competitive technique for set expansion using web lists.

• **Graphical Model [?].** We implement a graphical model-based table ranking method proposed in [**?**]. It takes column keywords as input and outputs a ranked list of relevant tables. The goal is similar with our table ranking when the input is a single concept name. We use the table prior as a feature for node potential and the column value overlap as edge potential in this graphical model.

• **Generalized Co-HITS (PageRank) [13].** Personalized PageRank and its variations have been adopted in set expansion systems like SEAL [29]. For our bipartite graph, the most appropriate variation to use is a generalized Co-HITS algorithm [13]. For brevity we still refer to it as PageRank. We set high restart probability on entity prior and low on table prior.

• **CONCEPTEXPAND.** This is our proposed method Algorithm 1, using aggregation functions discussed in Section 5.1. The threshold α for table filtering is simply set to 0, without any tuning.

• **CONCEPTEXPAND_PageRank.** To study the value of aggregation functions, we include a variation of CONCEPTEXPAND that uses random walk aggregation functions discussed in Section 3.

6.1.3 Evaluation

Ground truth preparation. Probase has an incomplete coverage of entities per concept, especially for tail concepts. We randomly checked 100 fine grained concepts and found that more than 90% of time the first relevant entity found in the web tables does not exist in Probase. We did a study for YAGO's fine grained concepts and found similar results. So simply using the existing Probase entities as ground truth leads to rather incomplete judgment. To obtain a reliable set of ground truth, we first identified 90 concepts for which Probase covers at least 4 entities in top-10 WTS tables.

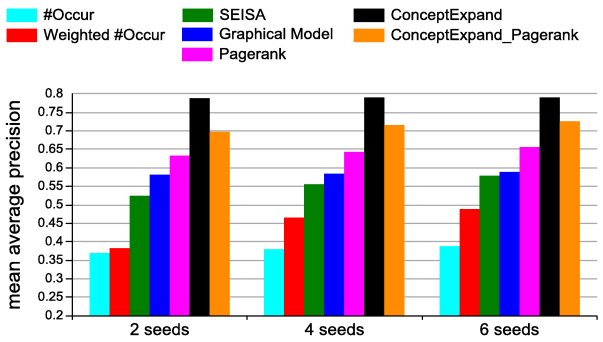


Figure 4: Mean average precision. Methods from left to right follow the order introduced in Section 6.1.2

Then we manually labeled a sample for 54 unambiguous concepts of them. Finally, we merge our labels and Probase entities to form a ground truth set. The final ground truth for each concept may still be incomplete, but is largely augmented upon Probase entities. This set has 18,232 relevant entities in total, with an average number of 338 entities per concept. The number of Probase entities per concept ranges from 4 to 60, with a mean of 23.

The labeled concepts span across multiple domains, such as geography, economy, business, healthcare and technology. Sample concepts and output from the algorithm are listed in Table 1.

Metrics. We use two metrics for evaluation:

• Mean average precision (MAP). This is the standard information retrieval measure for overall ranking performance.

• Relative recall[1] at a certain precision level. This is a metric more concerned with the practical value of the concept expansion task. It reflects the best recall one algorithm can achieve before falling below a precision level. Typically, the requirement for the precision in a knowledgebase is very high. Although we do not expect the expansion algorithm to return 100% correct results to directly populate into a knowledgebase, a high precision level such as 0.9 is helpful for easing the job of human curators in noise filtering.

6.2 Results

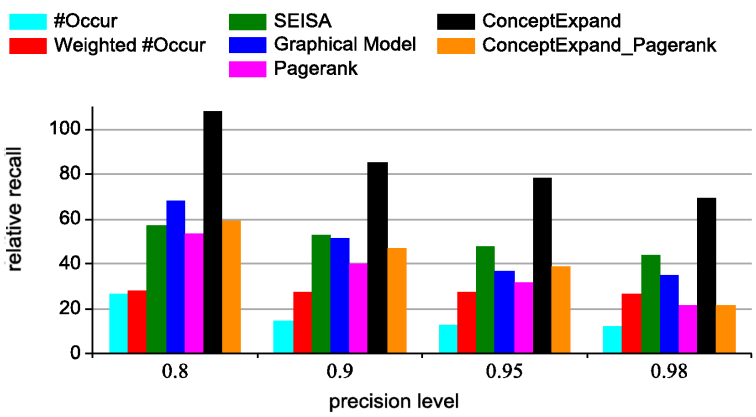
In this section we compare the 7 methods. Figure 4 and Figure 5 plot the mean average precision and relative recall at different precision levels. In both figures, the methods are sorted according to the order they are introduced in Section 6.1.2.

MAP performance. The overall ranking performance measured by MAP is consistent when using different number of seeds (Figure 4). The first six methods can be roughly partitioned into three groups according to their MAP: (i) baseline algorithms, # Occur and Weighted # Occur; (ii) existing methods adapted to this task, SEISA, Graphical Model and PageRank; and (iii) our proposed method CONCEPTEXPAND. The gap is large across groups. CONCEPTEXPAND outperforms existing algorithms by 20-50%. Note that the filtering parameter α for CONCEPTEXPAND has not been tuned in this experiment.

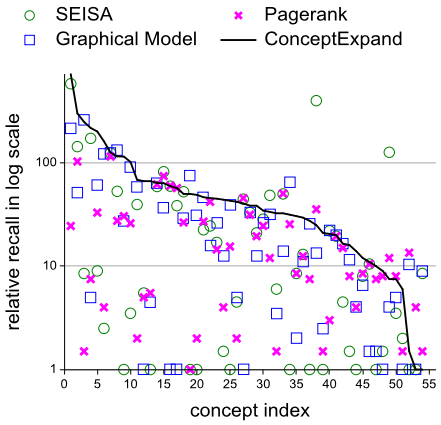
The last method CONCEPTEXPAND_PageRank is about halfway between CONCEPTEXPAND and PageRank. It shows that both the inference algorithm and the aggregation functions contribute to the superior performance of CONCEPTEXPAND.

CONCEPTEXPAND is not sensitive to the number of seeds. Without parameter tuning, it achieves a mean average precision of close to 80% with only two seeds per concept.

[1] $r = \frac{\#\text{ returned relevant entities}}{\#\text{ seed entities}}$



(a) Methods from left to right follow the order introduced in Section 6.1.2



(b) Scatter plot for precision level 0.98

Figure 5: Relative recall at certain precision levels (two seeds)

High precision performance. Regarding the relative recall at high precision levels, there is an even larger gap between CONCEPTEXPAND and competing algorithms (Figure 5). Though all methods can increase the entity coverage before the precision falls below a tolerable threshold, the degree of increase differs across the methods. For example, at precision level 0.98, PageRank increases the coverage by 20 times, SEISA does a better job and achieves 44 times expansion, none of which is comparable to the 70 times gain of CONCEPTEXPAND (Figure 5a). Figure 5b further visualizes the individual performance for these 54 concepts. With a few outliers, CONCEPTEXPAND dominates in the relative recall. When the requirement to precision gradually lowers down, CONCEPTEXPAND achieves even higher recall (108 times at precision level 0.8). Again, without parameter tuning, CONCEPTEXPAND shows strong superiority in its practical value.

When connecting the overall ranking performance (Figure 4) to the performance at high precision (Figure 5), we have two interesting observations.

First, CONCEPTEXPAND_PageRank improves the overall performance of PageRank algorithm by removing low confidence entities and tables, but the improvement is not much on the high precision end. In Figure 5a, CONCEPTEXPAND_PageRank's recall is as one third to one half as CONCEPTEXPAND's. Therefore, the high recall at high precision of CONCEPTEXPAND is mainly attributed to its asymmetric, non-linear score propagation following the special entity-table relationship, rather than the filtering.

Second, there is no clear winner among the three existing algorithms SEISA, Graphical Model and PageRank. Though PageRank has a better MAP than the other two, it poorly performs at high precision levels (lower than baseline at p-level 0.98). In contrast, SEISA does a good job at precision levels above 0.9, but underperforms Graphical Model at p-level 0.8, and eventually has a lowest MAP among the three. This observation reflects the limitations of existing algorithms. We take the concept 'public university' as an example to illustrate. The precision-recall curve is shown in Figure 6. Set expansion techniques like SEISA find most similar entities in the beginning, yet shift to different concepts (e.g., from public university to other universities) without constraining the expansion. Concept-name-based table ranking like Graphical Model does not utilize seed entities, and suffers from imperfect table ranking. For example, several tables about public university

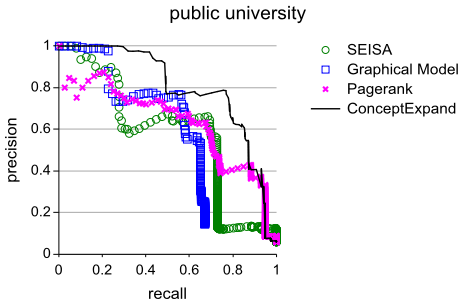


Figure 6: Precision-recall curve case study

cost, programs, salaries, faculty numbers are ranked among top 20. PageRank takes advantage of both concept names and seed information, but it tends to select popular entities from different tables and mix them because of the linear score aggregation. Popular irrelevant entities (e.g., New York University) are ranked high and inserted between relevant ones, thus it cannot produce contiguous high precision results.

6.3 Parameter study

In this section we study how the algorithm performs in response to varying parameter values.

Our algorithm has only one parameter, the stopping criterion α. Recall that this parameter controls the precision-recall trade-off, as it determines when the core table set stops expansion. We used $\alpha = 0$ for all previous experiments, which means all tables will be added to the table set. As we increase α, fewer tables will be added, which helps eliminating irrelevant tables. When α is equal or larger than 1, only one table will be added, leading to poor results.

In our experiments, we sort all table prior scores in descending order and set α to the score at certain positions of the ranked scores. For example, we use the lower quartile (denoted as lower 1/4), median (1/2), upper quartile (upper 1/4) and so on. Figure 7 shows the mean average precision and relative recall at different precision levels. It turns out that CONCEPTEXPAND works robustly in a wide range of α. In fact, the MAP has very small change when α varies from 0 to upper 1/16 quantile of table prior. The most sensitive metric is recall at precision level 0.8. It reaches the peak around 110 at

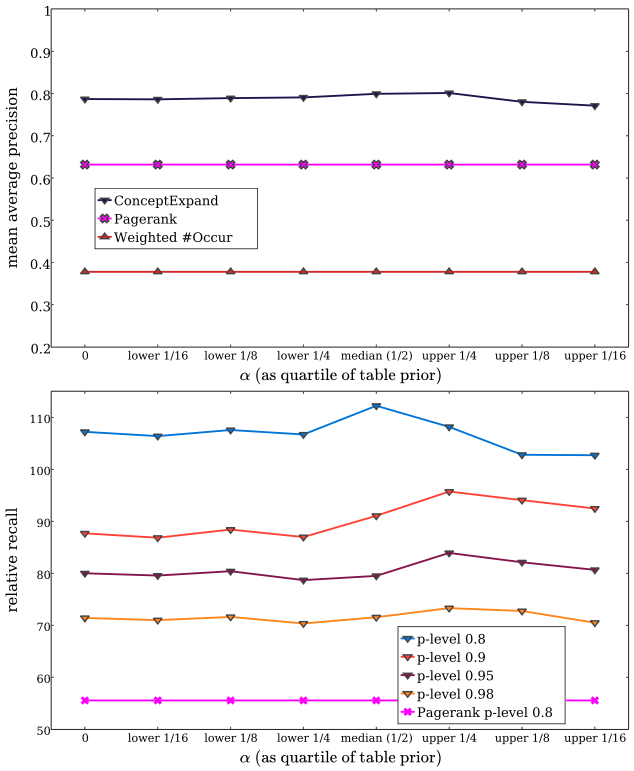


Figure 7: Vary filtering threshold α

median and declines when α further increases, but still above 100 when α is equal to upper quartile of table prior, where other three curves reach a high point. In the worst case in our test (α = upper 1/16 quantile), the recall at precision 0.98 is still much higher than PageRank's recall at precision 0.8. These results indicate a stop criterion ranging from median to upper quartile of table prior is a good choice, depending on the goal of precision. For higher precision a slightly higher α is preferred. As long as α is not too large to leave enough tables, stable performance can be expected thanks to the strong reasoning power of our method.

7. RELATED WORK

The concept expansion problem has been studied mainly in two settings: set expansion and multi-concept expansion. The former takes a set of seed entities as input to expand a single implicit concept, and the latter takes entities in two or more mutually exclusive concepts together and classify other entities.

Google Sets was an early set expansion system. It spawned quite a few set expansion techniques, such as Bayesian Sets [16], SEAL [30], SEISA [18] and others [24, 29, 21]. They mainly leverage the similarity between entities measured by their co-occurrences in web text, wrappers and lists. For example, SEISA employs iterative similarity aggregation and SEAL employs PageRank. Multi-concept expansion proposes to constrain the expansion by using negative examples from mutually exclusive concepts or type constraints [15, 10, 26]. This works well when expanding several mutually exclusive concepts in a reference ontology. None of the above methods are designed for ad-hoc tail concept expansion.

To put this problem in a broader context, the problem of knowledgebase expansion or completion has attracted a bulk of interest due to its practical value. Concept expansion is one main category, focusing on extending the number of entities (a.k.a. instances) belonging to each concept (a.k.a. type or class). Another main category is attribute completion, focusing on completing attributes (a.k.a. facts or relations) of entities. It is introduced as an annual competition in 2008 to the Text Analysis Conference. One can refer to Ji and Grishman [19] for an overview.

To differentiate the paradigm, West *et al.* [31] discusses the 'pull' mode and the 'push' mode for attribute completion. The 'pull' mode refers to extracting values for specific entities and attributes, while the 'push' mode refers to processing a large number of documents in batch and populates whatever facts it can find. A similar categorization applies to concept expansion. Set expansion works in the 'pull' mode and multi-concept expansion works in the 'push' mode. The 'pull' mode is found to better suit tail domains and tail entities. Our approach works in the 'pull' mode. Similar to West *et al.* [31] that leverages a web-search based question-answering system, we leverage a web table search system.

With regard to information source from which to harvest knowledge, the most popular sources are unstructured text [15, 21, 33, 31, 19], and semi-structured web wrappers, lists and tables [26, 18, 29]. Typically, textual patterns have good coverage for head entities, and the context helps identifying the concept. Semi-structured lists and tables contain both head entities and tail ones. A good strategy is to leverage both the semantics in text and the co-occurrence information in lists. For example, NELL [10] is a system for multi-concept expansion that combines textual signals and structure signals from separate sources respectively. In comparison, our work proposes to simultaneously utilizes text and structure from one source.

Our work is related to the holistic table ranking approach proposed by Pimplikar and Sarawagi [22], which takes concept keywords (column names) as input and returns relevant web tables. Though our method can rank tables as well, the ranking criteria is based on exclusivity rather than relevance. Other closely related studies about web tables include table column annotation [12, 20, 23, 27], and entity attribute discovery [8, 34, 35].

There is more work to which the CONCEPTEXPAND algorithm can be related. For example, CONCEPTEXPAND adds tables one by one to expand the viewpoint bipartite graph of reasoning. Therefore, techniques related to viewpoint network expansion, such as [5], can be considered as alternative ways for extracting a relevant viewpoint neighborhood.

8. CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of finding entities belonging to a concept, given an ad-hoc concept name and a few seed entities. We resort to exclusive web tables and leverage both text signals and structure information. We formulate the task as a holistic ranking problem and develop a novel solution for probabilistic reasoning. It is the first method addressing the challenges for expansion of tail concepts, and experiments demonstrate its superiority to state-of-the-art approaches. Besides, the convergence guarantee of our simple unified framework works with many common aggregation functions (including generalized PageRank as special cases), rendering it generalizable to different problems.

This work can be extended in multiple directions. There is much more information in the knowledgebase and web tables that can be leveraged for concept expansion as well as attribute completion. First, one can utilize the relational attributes in web tables; tables belonging to the same concept often have similar attributes, and the missing attributes in knowledgebases can be potentially populated from them. Second, we distinguish exclusive tables and non-exclusive tables in this work since the latter is more risky to use for expansion. In future work, one can study how to select relevant portions in non-exclusive tables, e.g., via the attributes.

References

[1] Google Knowledge Graph. http://www.google.com/insidesearch/features/search/knowledge.html.

[2] Google Web Tables. http://research.google.com/tables.

[3] Microsoft Excel Power Query. http://office.microsoft.com/powerbi.

[4] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, 2000.

[5] S. Asur and S. Parthasarathy. A viewpoint-based approach for interaction graph analysis. In *KDD*, 2009.

[6] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *PVLDB*, 18(1):255–276, 2009.

[7] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[8] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[9] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

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

[11] J. R. Curran, T. Murphy, and B. Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *PAACL*, 2007.

[12] D. Deng, Y. Jiang, G. Li, J. Li, and C. Yu. Scalable column concept determination for web tables using large knowledge bases. In *PVLDB*, volume 6, pages 1606–1617, Aug. 2013.

[13] H. Deng, M. R. Lyu, and I. King. A generalized co-hits algorithm and its application to bipartite graphs. In *KDD*, 2009.

[14] D. Downey, O. Etzioni, and S. Soderland. Analysis of a probabilistic model of redundancy in unsupervised information extraction. *Artif. Intell.*, 174(11):726–748, July 2010.

[15] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[16] Z. Ghahramani, Z. Ghahramani, and K. A. Heller. Bayesian sets. In *NIPS*, 2005.

[17] A. Granas and J. Dugundji. *Fixed Point Theory*. Springer-Verlag, 2003.

[18] Y. He and D. Xin. SEISA: set expansion by iterative similarity aggregation. In *WWW*, 2011.

[19] H. Ji and R. Grishman. Knowledge base population: Successful approaches and challenges. In *ACL*, 2011.

[20] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. In *VLDB*, 2010.

[21] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *EMNLP*, 2009.

[22] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *PVLDB*, 5(10):908–919, 2012.

[23] G. Quercini and C. Reynaud. Entity discovery and annotation in tables. In *EDBT*, 2013.

[24] L. Sarmento, V. Jijkoun, M. de Rijke, and E. Oliveira. "more like these": growing entity classes from seeds. In *CIKM*, 2007.

[25] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *WWW*, 2007.

[26] P. Talukdar and F. Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *ACL*, 2010.

[27] P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. In *PVLDB*, volume 4, pages 528–538, 2011.

[28] J. Wang, H. Wang, Z. Wang, and K. Zhu. Understanding tables on the web. In *International Conference on Conceptual Modeling*, 2012.

[29] R. Wang and W. Cohen. Iterative set expansion of named entities using the web. In *ICDM*, 2008.

[30] R. C. Wang and W. W. Cohen. Language-independent set expansion of named entities using the web. In *ICDM*, 2007.

[31] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin. Knowledge base completion via search-based question answering. In *WWW*, 2014.

[32] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

[33] M. Yahya, S. E. Whang, R. Gupta, and A. Halevy. Renoun: Fact extraction for nominal attributes. In *EMNLP*, 2014.

[34] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. *SIGMOD*, 2012.

[35] M. Zhang and K. Chakrabarti. Infogather+: Semantic matching and annotation of numeric and time-varying attributes in web tables. In *SIGMOD*, 2013.