LIEGE: Link Entities in Web Lists with Knowledge Base

Wei Shen[1], Jianyong Wang[1], Ping Luo[2], Min Wang[2]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]HP Labs China, Beijing, China
[1]chen-wei09@mails.tsinghua.edu.cn, jianyong@tsinghua.edu.cn
[2]{ping.luo, min.wang6}@hp.com

ABSTRACT

A critical step in bridging the knowledge base with the huge corpus of semi-structured Web list data is to link the entity mentions that appear in the Web lists with the corresponding real world entities in the knowledge base, which we call list linking task. This task can facilitate many different tasks such as knowledge base population, entity search and table annotation. However, the list linking task is challenging because a Web list has almost no textual context, and the only input for this task is a list of entity mentions extracted from the Web pages. In this paper, we propose LIEGE, the first general framework to Link the entIties in wEb lists with the knowledGe basE to the best of our knowledge. Our assumption is that entities mentioned in a Web list can be any collection of entities that have the same conceptual type that people have in mind. To annotate the list items in a Web list with entities that they likely mention, we leverage the prior probability of an entity being mentioned and the global coherence between the types of entities in the Web list. The interdependence between different entity assignments in a Web list makes the optimization of this list linking problem NP-hard. Accordingly, we propose a practical solution based on the iterative substitution to jointly optimize the identification of the mapping entities for the Web list items. We extensively evaluated the performance of our proposed framework over both manually annotated real Web lists extracted from the Web pages and two public data sets, and the experimental results show that our framework significantly outperforms the baseline method in terms of accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Experimentation

KDD'12, August 12–16, 2012, Beijing, China.

Keywords

List linking, Knowledge base, Similarity metric, Iterative substitution

1. INTRODUCTION

A large number of Web pages contain data structured in the form of lists. A Web list on some Web page may enumerate a list of famous American basketball players, a list of best-selling albums in the United States or a list of films directed by James Cameron. In addition, table which is another major source of structured data on the Web can be regarded as a set of lists/columns. Therefore, lists are considered as a general data structure to express entity references. Items in Web lists often refer to entities, however, as being free text form, the list items are potentially ambiguous: the same textual name may refer to several different real world entities. For instance, the entity mention of "Michael Jordan" can refer to the famous basketball player, the computer science professor who is one of the keynote speakers in KDD'12, or some other persons.

Recently, the success of Wikipedia and the development of information extraction techniques have facilitated the automated construction of large scale machine-readable knowledge base about the world's entities, their semantic classes and their mutual relationships. Such kind of notable endeavors include DBpedia [2], KnowItAll [9], YAGO [21, 20] and KOG [23, 24]. Bridging these knowledge bases with the semi-structured Web lists is beneficial for the exploitation of this huge corpus of structured data on the Web. Additionally, the list linking task enables powerful join and union operations that can integrate information across different lists, tables and pages.

The list linking task is of practical importance and can be used in various applications. For instance, 75% of the tables on the Web typically have a column that is the subject of the table, and the subject column contains the set of entities the table is about [22]. Linking this subject column with the knowledge base is significantly helpful for the task of table annotation [16] and recovering the semantics of tables [22]. As another example, linking the Web lists or table columns with a knowledge base can enrich the existing knowledge base and impulse the trend to advance the traditional keyword-based search to the semantic entity-based search.

In recent years, considerable progresses have been made in linking entities in free text with a knowledge base [3, 6, 7, 13, 15, 19]. When linking entities in free text, the textual context is considered as the key evidence for disambigua-

tion. The essential step in these previous approaches is to define a similarity measure between the text around the entity mention and the document associated with the entity. While in our setting, lists never have any header text information and textual context. The only input for the list linking task is a list of entity mentions extracted from the Web pages, such as the Web list on the left of Figure 1. On the other hand, most of the methods addressing the problem of linking entities in free text exploit the observation that entities mentioned from a single document are likely to be semantically related [6, 15, 19]. However, the entities in a Web list can be any set of entities that have the same conceptual type, and they are semantically similar, rather than semantically related. Accordingly, this observation for linking entities in free text clearly cannot be applied to the Web list data. From these two dimensions mentioned above, it can be seen that the list linking task is challenging and different from the task of linking entities in free text. Moreover, we are interested in linking these Web list items in a fully automatic manner, and avoid tedious tuning or parameter settings.

Given a knowledge base about the world's entities, their semantic classes and their mutual relationships, for each Web list in the list corpus, our goal is to link each list item in the Web list with the real world mapping entity existing in the knowledge base. In this paper, we propose LIEGE, a general framework to annotate the list items in the Web lists with entities based on the assumption that entities in a list can be any set of entities that have the same conceptual type. Specifically, we propose to measure the *linking quality* of the candidate mapping entities in a straightforward and comprehensive way. Intuitively, a candidate mapping entity is "good" for some list item if it has two key properties: (1) the prior probability of the entity being mentioned is high; (2) the type of the candidate mapping entity is coherent with the types of the other mapping entities in the same list, in the sense that they represent a list of entities having the same conceptual type. If the type of an entity is coherent with the type of another entity, we say these two entities are semantically similar. To calculate the semantic similarity between entities, we leverage two categories of information: (1) type hierarchy based similarity; (2) distributional context similarity. To combine these signals to give the *linking quality* for each candidate mapping entity, we use the max-margin technique to automatically learn the proper weights for different features. The interdependence between different entity assignments in a Web list makes the inference of this problem NP-hard. Thus, we develop a practical and effective solution based on the iterative substitution to jointly optimize the entity assignments for each Web list.

Contributions. The main contributions of this paper are summarized as follows.

- We propose LIEGE, the first framework that can effectively link list items in the Web lists with the mapping entities in the knowledge base to the best of our knowledge.
- We propose to use the max-margin technique to automatically learn the proper weights for different features to calculate the *link quality* for each candidate mapping entity, which allows LIEGE to run without tedious parameter tuning.
- We propose a practical and effective algorithm, called *iterative substitution algorithm*, to jointly optimize the identification of the mapping entities for the list items of each Web list.
- To verify the effectiveness of our framework, we evaluated our proposed framework LIEGE over both manually annotated real Web list data set and two public data sets, and the experimental results show that our framework significantly outperforms the baseline method in terms of accuracy.

The remainder of this paper is organized as follows. Section 2 introduces some preliminaries and notations. We present the framework of LIEGE in Section 3. Section 4 presents our experimental results and Section 5 discusses the related work. Finally, we conclude this paper in Section 6.

2. PRELIMINARIES AND NOTATIONS

In this section, we begin by describing the knowledge base and the task of list linking. Next, we introduce the generation of candidate mapping entities for each list item.

Knowledge base. The knowledge base consists of a type hierarchy and entities that are instances of types. The set of types is denoted by T, and each $t \in T$ is a type, such as the type *American singers*. Types are connected by the subtype relation $t_1 \subset t_2$, which means t_1 is a subtype of t_2. For example, *American singers* is a subtype of *Singers*. We use $t_1 \subset^* t_2$ to denote that t_2 is an ancestor node of t_1 in the type hierarchy. The set of entities existing in the knowledge base is denoted by E, and each $e \in E$ is an entity, such as the entity *Michael Jackson*. We say entity e is an instance of one type t, denoted by $e \in t$. For instance, the entity *Michael Jackson* is an instance of the type *American singers*. We also use $e \in^* t$ to denote that type t is an ancestor node of entity e in the hierarchy, thus, we can say, *Michael Jackson* $\in^*$ *Singers*. Let $T(e) = \{t | e \in t\}$ be the set of types of which entity e is the instance. Likewise, let $E(t) = \{e | e \in^* t\}$ be the set of entities having type t as an ancestor node in the hierarchy. In this paper, the knowledge base we adopt is YAGO [21, 20], an open-domain ontology combining Wikipedia and WordNet [10] with high coverage and quality. In YAGO, they use unique canonical strings from Wikipedia as the entity names. Currently, YAGO contains about 250,000 types and over one million entities.

List linking. The input of our framework LIEGE is a collection of Web lists, and let L be a Web list with $|L|$ items. We use $1 \leq i \leq |L|$ to index the list item in L, and the list item with index i is denoted by l_i. Each list item $l_i \in L$ is a token sequence of an entity mention that is potentially linked with an entity in the knowledge base (see Figure 1 for an example). For this list L, it is noted that we do not have a name for it (i.e., the type of entities this list L enumerates). And there are no textual context and header text about this list L. Here, we formally state the list linking task as follows. Given the set of entities E in the knowledge base and the Web list L, the goal is to identify the mapping entity list M, with the same size as L, such that each $m_i \in M$ is the mapping entity for the corresponding list item $l_i \in L$. In this paper, we assume that the knowledge base contains all the mapping entities for all the list items, i.e., $M \subset E$.

Candidate mapping entity. For each $l_i \in L$, the mapping entity m_i should be the entity that may be referred by

Table 1: A part of the dictionary D

K (Mention form)	$K.value$ (Mapping entity)
IBM	*IBM*
HP	*Hewlett-Packard*
Michael Jordan	*Michael Jordan* *Michael I. Jordan* *Michael Jordan (mycologist)* *Michael Jordan (footballer)* ...
Bill Hewlett	*William Reddington Hewlett*

the token sequence of l_i. Therefore, we firstly identify all the entities that may be referred by l_i, and denote this set of entities as the candidate mapping entity set R_i for the list item l_i. Let $R = \{R_i | l_i \in L\}$ be the set of candidate mapping entity sets for all the list items in the Web list L.

To identify R_i for each $l_i \in L$, we need to build a dictionary D that contains vast amount of information about various mention forms of the named entities, like name variations, abbreviations, confusable names, spelling variations, nicknames, etc. The structure of the Wikipedia provides a set of useful features for the creation of such a dictionary D. In addition, Wikipedia has high coverage of named entities [25], which is profitable for constructing the dictionary D as well. The dictionary D is a <key, value> mapping, where the column of the key K is a list of mention forms and the column of the mapping value $K.value$ is the set of named entities which are referred by the key K. We construct the dictionary D by leveraging the following four structures of Wikipedia:

Entity page: Each entity page in Wikipedia describes a single entity, and generally, the title of each entity page is the most common name for that entity, e.g., the page title "IBM" for that giant American company headquartered in Armonk. Thus, we add the title of the entity page to the key K, and add the entity described in this page to $K.value$.

Redirect page: A redirect page exists for each alternative name which can be used to refer to an existing entity in Wikipedia. For example, the redirect page titled "HP" contains a pointer to the entity page titled "Hewlett-Packard". Henceforth, we add the title of the redirect page to the key K, and add the pointed entity to $K.value$.

Disambiguation page: When multiple entities in Wikipedia could be given the same name, a disambiguation page is created to separate them and contains a list of references to those entities. For instance, the disambiguation page for the name "Michael Jordan" lists eight associated entities having the same name of "Michael Jordan", including the famous NBA player and the Berkeley professor. For each disambiguation page, the title of this page is added to the key K, and the entities listed in this page are added to $K.value$.

Hyperlink in Wikipedia article: The article in Wikipedia often contains some hyperlinks each of which links to the page of the corresponding entity mentioned in this article. For example, in the entity page titled "Hewlett-Packard", there is a hyperlink pointing to the entity *William Reddington Hewlett* whose anchor text is "Bill Hewlett". Then we add the anchor text of the hyperlink to the key K, and add the pointed entity to $K.value$.

Using the four structures of Wikipedia described above, we construct the dictionary D. A part of the dictionary D is shown in Table 1. For each list item $l_i \in L$, we look up the dictionary D and search for l_i in the column of the key K. If a hit is found, i.e., $l_i \in K$, we add the set of the entities $l_i.value$ to the candidate mapping entity set R_i. We denote the size of R_i as $|R_i|$, and use $1 \leq j \leq |R_i|$ to index the candidate entity in R_i. The candidate mapping entity with index j in R_i is denoted by $r_{i,j}$.

In most cases, the size of the candidate mapping entity set is larger than one. Figure 1 shows an example. The Web list in Figure 1 enumerates 6 best-selling single-volume books on the left. For each list item, we show its candidate mapping entity set generated from the dictionary D on the right of the figure, and the real mapping entity is underlined. It is noted that in Figure 1 each candidate entity has the name which is the unique canonical string for that entity in Wikipedia, and the size of the candidate mapping entity set for most of the list items (5 out of 6) is larger than 1.

3. LIST LINKING

In this section, we introduce how to pick the proper entity from R_i as the mapping entity m_i for the list item l_i, when $|R_i| > 1$. We firstly illustrate how to define the *linking quality* metric for each candidate mapping entity in Section 3.1. Next, we show how to learn the proper weights for different features to calculate the *link quality* for each candidate mapping entity in Section 3.2. Finally, we present an effective algorithm, called *iterative substitution algorithm*, to jointly optimize the identification of the mapping entities for the list items of each Web list in Section 3.3.

3.1 Linking quality metric

Since each list item can refer to several candidate entities, it is nontrivial to select the mapping entity of the list item from the candidate mapping entity set. In this section, we propose an intuitive and comprehensive metric to measure the *linking quality* of the candidate mapping entity.

Prior probability: The first observation we have is that, each candidate mapping entity $r_{i,j} \in R_i$ having the same mention form l_i has different popularity, and some entities are very obscure and rare for the given mention form l_i. For the example in Figure 1, for the list item "A Tale of Two Cities", the candidate entity *A Tale of Two Cities (musical)*, the stage musical by Jill Santoriello, is much rarer than the candidate entity *A Tale of Two Cities*, the novel by Charles Dickens, and in most cases when people mention "A Tale of Two Cities", they mean the novel rather than the stage musical whose name is also "A Tale of Two Cities". We formalize this observation via taking advantage of the count information from Wikipedia, and define the *prior probability* $P_{pr}(r_{i,j})$ for each candidate mapping entity $r_{i,j} \in R_i$ with respect to the list item l_i as the proportion of the links with the mention form l_i as the anchor text which point to the candidate mapping entity $r_{i,j}$:

$$P_{pr}(r_{i,j}) = \frac{count(r_{i,j})}{\sum_{u=1}^{|R_i|} count(r_{i,u})} \quad (1)$$

where $count(r_{i,j})$ is the number of links which point to entity $r_{i,j}$ and have the mention form l_i as the anchor text.

For the example in Figure 1, with respect to the list item "A Tale of Two Cities", the *prior probability* of the entity *A Tale of Two Cities*, the novel, is 0.6528, while the *prior probability* of the entity *A Tale of Two Cities (musical)*, the

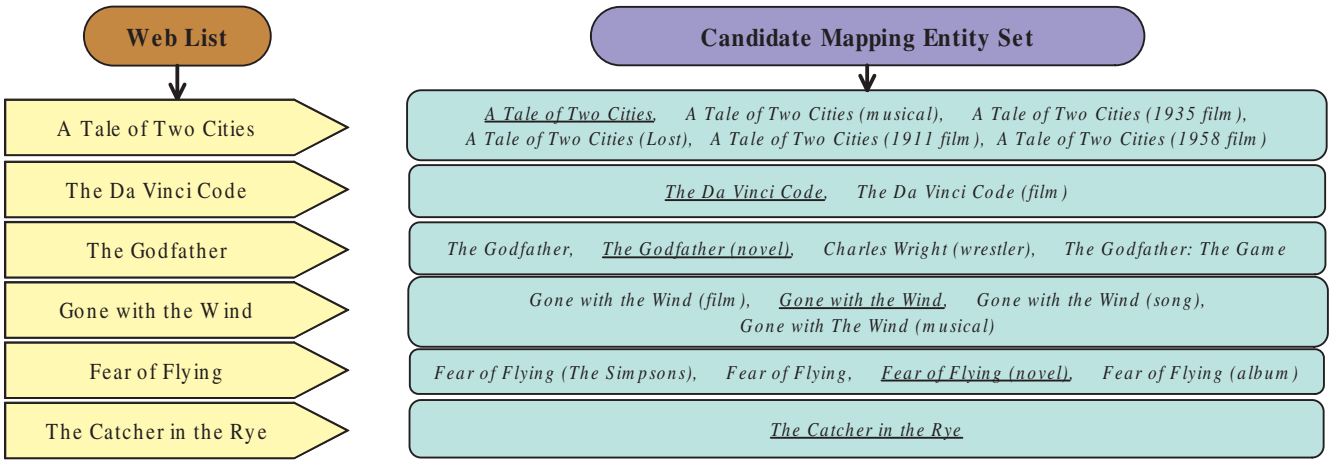

Figure 1: The candidate mapping entities for an example Web list

stage musical, is 0.1319. And the *prior probabilities* of all other candidate entities for the list item "A Tale of Two Cities", e.g., *A Tale of Two Cities (1935 film)*, *A Tale of Two Cities (Lost)*, are much smaller than that for the entity *A Tale of Two Cities*. Moreover, in Figure 1, the candidate mapping entities for each list item are ranked by their *prior probabilities* shown on the right of the figure in decreasing order. It can be seen that the notion of *prior probability* suitably expresses the popularity of the candidate mapping entity being mentioned given a mention form.

Coherence: A Web list usually enumerates some entities which have the same conceptual type. As an example, the Web list in Figure 1 enumerates some best-selling single-volume books. We observe that the definition of *prior probability* does not capture the contextual meaning the Web list represents. For the example in Figure 1, for the list item "The Godfather", the entity *The Godfather*, an American epic crime film, which has the highest *prior probability*, is not the corresponding mapping entity for this list item in this Web list. The same situation occurs for the list items "Gone with the Wind" and "Fear of Flying" in Figure 1. Accordingly, to choose the mapping entity for a list item, we should select the candidate entity whose type is coherent with the types of the other mapping entities in the same Web list. In other words, the mapping entity for some list item should be semantically similar to the other mapping entities in the same Web list. Therefore, we have the intuition that the more semantically similar the candidate mapping entity is to the other mapping entities in the same Web list, the more likely the candidate entity is the correct mapping entity for the list item. To capture this intuition, we formally define the notion of *coherence* $Coh(r_{i,j})$ for each candidate mapping entity $r_{i,j} \in R_i$ as:

$$Coh(r_{i,j}) = \frac{1}{|L|-1} \sum_{u=1,u\neq i}^{|L|} Sim(r_{i,j}, m_u) \quad (2)$$

where m_u is the mapping entity for the list item $l_u \in L$, and $Sim(e_1, e_2)$ is the function that measures the semantic similarity between entities e_1 and e_2, $e_1, e_2 \in E$.

To calculate the semantic similarity between entities, we leverage two categories of information in this paper: (1) type hierarchy based similarity; (2) distributional context similarity.

Type hierarchy based similarity: To measure the semantic similarity between entities based on type hierarchy, we have the assumption that two entities are semantically similar if they are in close places in the type hierarchy. Given two entities $e_1, e_2 \in E$, $T(e_1) = \{t|e_1 \in t\}$ is the set of types of which entity e_1 is the instance in the knowledge base, likewise, $T(e_2) = \{t|e_2 \in t\}$. To measure the semantic similarity $Sim_{hr}(e_1, e_2)$ between e_1 and e_2 based on type hierarchy, we firstly define how to calculate the semantic similarity between the sets of types $T(e_1)$ and $T(e_2)$. Since the sizes of $T(e_1)$ and $T(e_2)$, and the elements in $T(e_1)$ and $T(e_2)$ are likely to be different, we start by defining the correspondence between the elements of types from one set to another set. For each type $t_1 \in T(e_1)$, we assign a target type $\varepsilon(t_1)$ in another set $T(e_2)$ as follows: $\varepsilon(t_1) = \arg\max_{t_2 \in T(e_2)} Sim_{hr}(t_1, t_2)$, where $Sim_{hr}(t_1, t_2)$ is the semantic similarity between two types t_1 and t_2 based on type hierarchy, and $\varepsilon(t_1)$ is the type in $T(e_2)$ which maximizes the semantic similarity between these two types.

To compute $Sim_{hr}(t_1, t_2)$, we adopt the approach introduced in [17] which is an information-theoretic method. Assuming $t \in T$ is a type in the knowledge base, the amount of information contained in the statement "$x \in t$" is $-log(P(t))$, where $P(t)$ is the probability that a randomly selected entity e belongs to $E(t)$, which is the set of entities having the type t as an ancestor node in the hierarchy. We also assume that t_0 is the lowest common ancestor node for type nodes t_1 and t_2 in the hierarchy. The following is the definition of the semantic similarity between types t_1 and t_2 based on type hierarchy:

$$Sim_{hr}(t_1, t_2) = \frac{2 \times log(P(t_0))}{log(P(t_1)) + log(P(t_2))} \quad (3)$$

Next, we can calculate the semantic similarity from one set of types $T(e_1)$ to another set of types $T(e_2)$:

$$Sim_{hr}(T(e_1) \rightarrow T(e_2)) = \frac{\sum_{t_1 \in T(e_1)} Sim_{hr}(t_1, \varepsilon(t_1))}{|T(e_1)|} \quad (4)$$

Similarly, we can calculate the semantic similarity from $T(e_2)$ to $T(e_1)$, i.e., $Sim_{hr}(T(e_2) \rightarrow T(e_1))$, in the similar way

to that in Formula 4. Based on the definitions mentioned above, we can define the semantic similarity between entities e_1 and e_2 based on type hierarchy as the average of the semantic similarity from $T(e_1)$ to $T(e_2)$ and that from $T(e_2)$ to $T(e_1)$:

$$Sim_{hr}(e_1, e_2) = \frac{Sim_{hr}(T(e_1) \to T(e_2)) + Sim_{hr}(T(e_2) \to T(e_1))}{2} \quad (5)$$

For the example in Figure 1, the semantic similarity between entities *A Tale of Two Cities* and *The Da Vinci Code*, a mystery-detective novel, based on the type hierarchy is 0.9527, while the semantic similarity between *A Tale of Two Cities (musical)* and *The Da Vinci Code* is much smaller, i.e., 0.3637.

Distributional context similarity: To measure the semantic similarity between entities based on the contexts where they appear in the external document corpus (e.g. the Wikipedia article corpus), we have the assumption that entities that occur in similar contexts are semantically similar, which is the extension of the distributional hypothesis [14]. We define the document context η_e for each entity $e \in E$ as the set of windows of words (removing all punctuation symbols) before and after each occurrence of the entity e in the external document corpus. Assume that an entity e of length $|e|$ words appears in a document at position p. The size-k document context η_e of entity e should contain these two windows of words, i.e., $w_{p-k}, \ldots, w_{p-1}$ and $w_{p+|e|}, \ldots, w_{p+|e|+k-1}$, around the occurrence of e. For instance, the entity *A Tale of Two Cities* occurs in a document containing such a sentence, "Simplified versions of *A Tale of Two Cities*, a novel by Charles Dickens, have been published ...". When the size k is set to 4, we should add the windows "Simplified versions of" and "a novel by Charles" to the document context of the entity *A Tale of Two Cities*. To compute the n-gram vector V_e for each entity $e \in E$, we firstly add all n-grams in the size-k document context η_e of entity e together with their frequencies, and then discard the top W most frequent n-grams which appear in this vector V_e. In the experiments, we set $k = 4$, $W = 50$. In order to measure the distributional context similarity between entities e_1 and e_2, we calculate the cosine similarity of the two n-gram vectors V_{e_1} and V_{e_2} for entities e_1 and e_2 respectively as follows:

$$Sim_{ds}(e_1, e_2) = \frac{\sum_{i=1}^{g} a_i * b_i}{\sqrt{\sum_{i=1}^{g} a_i^2} * \sqrt{\sum_{i=1}^{g} b_i^2}} \quad (6)$$

where g is the number of distinct n-grams in the union of the n-gram vectors for entities e_1 and e_2, each $a_i(b_i)$ is the frequency of the corresponding n-gram which appears in the vector $V_{e_1}(V_{e_2})$, and $V_{e_1} = \langle a_1, a_2, \ldots, a_g \rangle$, $V_{e_2} = \langle b_1, b_2, \ldots, b_g \rangle$.

For the example in Figure 1, the distributional context similarity between entities *A Tale of Two Cities* and *The Da Vinci Code* is 0.0934, while the distributional context similarity between *A Tale of Two Cities (musical)* and *The Da Vinci Code* is much smaller, i.e., 0.0137.

In this paper, we only leverage these two categories of information to calculate the semantic similarity between entities, the function Sim in Formula 2 could be either one of these two semantic similarity metrics, i.e., Sim_{hr} and Sim_{ds}, or their weighted sum. In addition, we emphasize that our framework is general and extensible enough that any other semantic similarity methods can be easily plugged in to be used.

Linking quality: Based on the observation that both *prior probability* and *coherence* contribute to the *linking quality* of candidate mapping entity, we define the *linking quality* $LQ(r_{i,j})$ for each candidate mapping entity $r_{i,j} \in R_i$ as the weighted sum of *prior probability* and *coherence* as follows:

$$LQ(r_{i,j}) = \alpha * P_{pr}(r_{i,j}) + (1 - \alpha) * Coh(r_{i,j}) \quad (7)$$

where α is a weight factor that balances the importance between *prior probability* and *coherence*.

With the definition of *linking quality* for each candidate mapping entity, we also define the *linking quality* for the mapping entity list M as the sum of the *linking qualities* of the mapping entities for all the list items with respect to the Web list L:

$$LQ(M) = \alpha * \sum_{s=1}^{|L|} P_{pr}(m_s) + (1 - \alpha) * \sum_{s=1}^{|L|} Coh(m_s) \quad (8)$$

where $m_s \in M$ and $Coh(m_s)$ can be calculated in the way introduced in Formula 2. Thus, we have

$$LQ(M) = \alpha * \sum_{s=1}^{|L|} P_{pr}(m_s) + \frac{1 - \alpha}{|L| - 1} * \sum_{s=1}^{|L|} \sum_{u=1, u \neq s}^{|L|} Sim(m_s, m_u) \quad (9)$$

Now, we formally state the list linking task as follows:

List linking: *Given the set of entities E in the knowledge base, the Web list L and the definition of linking quality in Formula 9, the goal is to identify the mapping entity list $M \subset E$, with size $|L|$, such that the objective function $LQ(M)$ is maximized.*

However, the inference problem of identifying the mapping entity list $M \subset E$, with size $|L|$, which maximizes the objective function $LQ(M)$ is NP-hard. The hardness can be shown using a reduction from the maximal clique problem[11]. Details of proof are omitted to save space.

3.2 Weight learner

To measure the *linking quality* $LQ(r_{i,j})$ for each candidate mapping entity $r_{i,j} \in R_i$, we use the definitions introduced in Formulae 7 and 2. Thus, we have

$$LQ(r_{i,j}) = \alpha * P_{pr}(r_{i,j}) + \frac{1 - \alpha}{|L| - 1} * \sum_{u=1, u \neq i}^{|L|} Sim(r_{i,j}, m_u) \quad (10)$$

where the function Sim could be the weighted sum of the two semantic similarity metrics, i.e., Sim_{hr} and Sim_{ds}, introduced in Formulae 5 and 6 respectively. Thus,

$$LQ(r_{i,j}) = \alpha * P_{pr}(r_{i,j}) + \beta * \frac{1}{|L| - 1} \sum_{u=1, u \neq i}^{|L|} Sim_{hr}(r_{i,j}, m_u) + \gamma * \frac{1}{|L| - 1} \sum_{u=1, u \neq i}^{|L|} Sim_{ds}(r_{i,j}, m_u) \quad (11)$$

where $\alpha + \beta + \gamma = 1$, and α, β and γ are weight factors that give different weights for different feature values in the calculation of the *linking quality* $LQ(r_{i,j})$. Here, we denote $\overrightarrow{w}$ as the weight vector, and $\overrightarrow{w} = \langle \alpha, \beta, \gamma \rangle$. In this section, we introduce how to learn the weight vector $\overrightarrow{w}$ based on the training data set.

To learn $\overrightarrow{w}$, we use a max-margin technique based on the training data set. Given the ground truth mapping entity $m_i \in M$ for each list item $l_i \in L$, we assume that the *linking quality* of the mapping entity $LQ(m_i)$ is larger than the *linking quality* of any other candidate entity $LQ(r_{i,j})$ with a margin, where $r_{i,j} \in R_i$ and $r_{i,j} \neq m_i$. This gives us the usual SVM linear constraints for all list items:

$$\forall l_i, \forall r_{i,j} \neq m_i \in R_i : LQ(m_i) - LQ(r_{i,j}) \geq \xi_{l_i} \qquad (12)$$

and we maximize the objective $C_{l_i} \sum_{l_i} \xi_{l_i}$ where C_{l_i} is a constant with respect to ξ_{l_i}, to learn the weight vector $\overrightarrow{w} = \langle \alpha, \beta, \gamma \rangle$ such that $\alpha + \beta + \gamma = 1$.

If the assumption is violated by some list item l_i, that is to say, $LQ(m_i)$ is smaller than $LQ(r_{i,j})$ for all $r_{i,j} \neq m_i \in R_i$, we have $\xi_{l_i} < 0$. Therefore, for the list item l_i which violates the assumption, i.e., $\xi_{l_i} < 0$, we give some penalty to it and set $C_{l_i} = \theta$, where θ is a constant that is larger than 1, called penalty parameter; Otherwise, for the list item l_i, $\xi_{l_i} \geq 0$, we set $C_{l_i} = 1$. θ is the only parameter we have to set in our framework, and in the experiments, we show that the choice of θ does not affect the results of our framework greatly. Accordingly, our framework can run without tedious parameter tuning.

3.3 Iterative substitution algorithm

As the inference problem of identifying the mapping entity list $M \subset E$, with size $|L|$, which maximizes the objective function $LQ(M)$ is NP-hard, we propose an effective algorithm, called *iterative substitution algorithm*, to jointly optimize the identification of the mapping entity list M for each Web list L in this section.

The *iterative substitution algorithm* starts with a good guess of the mapping entity list from the candidate mapping entity sets, and then iteratively refines the mapping entity list to improve the *linking quality* as defined in Formula 9, until the algorithm converges and a local maximum of the *linking quality* of the mapping entity list is reached. To improve the *linking quality* of the mapping entity list in each iteration, the algorithm replaces the mapping entity of the previous iteration with the candidate mapping entity which maximizes the improvement of the overall *linking quality* of the mapping entity list. The details of the *iterative substitution algorithm* are described in Algorithm 1.

The *iterative substitution algorithm* depicted in Algorithm 1 takes the Web list L and the candidate mapping entity sets R as input, and outputs the mapping entity list M for the Web list L. We firstly calculate the *prior probability* $P_{pr}(r_{i,j})$ for each candidate entity $r_{i,j}$ in the candidate entity set R_i of each list item l_i, as defined in Formula 1, and pick the candidate entity which has the maximum *prior probability* as the initial estimate of the mapping entity $m_i^{(0)}$ for the list item l_i (line 1-line 3). Let $M^{(0)}$ be the initial mapping entity list for the Web list L (line 4). In the subsequent iterations in the while loop, we iteratively identify the new mapping entity list $M^{(iter)}$ based on the mapping entity list of previous iteration $M^{(iter-1)}$, and gradually improve the *linking quality* of the mapping entity list until the algorithm converges and a local maximum is reached (line 6-line 20). Specifically, in each iteration *iter*, for each candidate mapping entity $r_{i,j} \neq m_i^{(iter-1)} \in R_i$ of each list item $l_i \in L$, where $m_i^{(iter-1)}$ is the estimate of the mapping entity for l_i of previous iteration, we compute the new mapping entity list $M_{r_{i,j}}^{(iter)}$ assuming that we substitute $r_{i,j}$ for $m_i^{(iter-1)}$ to choose $r_{i,j}$ as the mapping entity for l_i in iteration *iter* (line 9), and then calculate the improvement of the *linking quality* of the mapping entity list $IncreLQ_{r_{i,j}}$ if this substitution is operated (line 10). Let $r_{i,j}^{max}$ be the candidate entity which achieves the maximum improvement of the *linking quality* of the mapping entity list (line 13). If $IncreLQ_{r_{i,j}^{max}} > 0$, we substitute $r_{i,j}^{max}$ for $m_i^{(iter-1)}$ to generate the mapping entity list $M^{(iter)}$ of iteration *iter*, and continue the iteration(line 14-line 16). Otherwise, if $IncreLQ_{r_{i,j}^{max}} \leq 0$, the algorithm has converged and reached a local maximum, thus, we stop the algorithm (line 18) and return $M^{(iter-1)}$ as the result of the mapping entity list M for L (line 21).

Algorithm 1 *Iterative Substitution Algorithm*

Input: Web list L, candidate mapping entity sets R.
Output: mapping entity list M.

1: **for** each $l_i \in L$ **do**
2: $\quad m_i^{(0)} = \arg\max_{r_{i,j}} P_{pr}(r_{i,j})$, $r_{i,j} \in R_i$
3: **end for**
4: $M^{(0)} = \{m_i^{(0)} | l_i \in L\}$
5: $iter = 1$
6: **while** true **do**
7: $\quad$ **for** each $l_i \in L$ **do**
8: $\quad\quad$ **for** each $r_{i,j} \neq m_i^{(iter-1)} \in R_i$ **do**
9: $\quad\quad\quad M_{r_{i,j}}^{(iter)} = (M^{(iter-1)} - \{m_i^{(iter-1)}\}) \bigcup \{r_{i,j}\}$
10: $\quad\quad\quad IncreLQ_{r_{i,j}} = LQ(M_{r_{i,j}}^{(iter)}) - LQ(M^{(iter-1)})$
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: $\quad r_{i,j}^{max} = \arg\max_{r_{i,j}} IncreLQ_{r_{i,j}}, \quad r_{i,j} \in R_i, R_i \in R$
14: $\quad$ **if** $IncreLQ_{r_{i,j}^{max}} > 0$ **then**
15: $\quad\quad M^{(iter)} = (M^{(iter-1)} - \{m_i^{(iter-1)}\}) \bigcup \{r_{i,j}^{max}\}$
16: $\quad\quad iter++$
17: $\quad$ **else**
18: $\quad\quad$ break
19: $\quad$ **end if**
20: **end while**
21: $M = M^{(iter-1)}$

We state that the computation of the mapping entity list M in Algorithm 1 is guaranteed to **converge**, and the *iterative substitution algorithm* depicted in Algorithm 1 is bound to terminate. Here, we give out the proof of the convergence of the *iterative substitution algorithm*. We note that the *linking quality* of the mapping entity list $LQ(M^{(iter)})$ of each iteration *iter* is greater than the *linking quality* of the mapping entity list $LQ(M^{(iter-1)})$ of the previous iteration $(iter - 1)$. Thus, we can say that the *linking quality* $LQ(M^{(iter)})$ is monotonically increasing with the number of iteration *iter*. Moreover, since $M \subset E$, the number of different mapping entity lists is finite. The linking quality $LQ(M)$ has an upper bound. Accordingly, we know that the *iterative substitution algorithm* is bound to stop and the computation of M in Algorithm 1 converges. Though we do not know the upper bound on the number of iterations Algorithm 1 may run until it terminates, in our experiments, we observe that it converges quickly and typically takes only a few iterations.

Table 2: Summary of the data sets

	# tables	# lists	# list items
Wiki_Manual	36	66	1691
Web_Manual	371	550	9239
Web_Lists	N/A	50	2520

4. EXPERIMENTS

To evaluate the effectiveness of LIEGE, we present a thorough experimental study in this section. We firstly describe the experimental setting in Section 4.1 and then show the experimental results in Section 4.2.

4.1 Experimental setting

To the best of our knowledge, there is no publicly available data set for the list linking task. However, we found that the table annotation task introduced in [16] has a subtask, whose goal is to annotate the table cells in Web tables with the mapping entities in the knowledge base. Though this subtask is not the same as our list linking task, we could regard each column of each table in the data sets as a Web list, and used the ground truth annotation in [16] to evaluate the performance of our framework. Specifically, we used two Web table data sets, i.e., Wiki_Manual and Web_Manual, introduced in [16]. The Wiki_Manual data set contains 36 (non-Infobox) tables selected from Wikipedia article text, and the Web_Manual data set consists of 371 Web tables similar to the Web tables in Wiki_Manual. The main difference between Wiki_Manual and Web_Manual is that the cells in the latter are more noisy. In both of these two data sets, the table cells whose mapping entities do not exist in the knowledge base have been dropped from the labeling task [16]. A summary of the data sets used in this paper is shown in Table 2. From the summary in Table 2, we can see that the average number of annotated columns for each table is less than 2 in both Wiki_Manual and Web_Manual data sets.

In addition, to verify the effectiveness of our framework over the real Web list data set, we collected 50 Web lists from the Web and manually annotated the list items with the mapping entities in the knowledge base, which we refer as the Web_Lists data set. The Web lists in our Web_Lists data set enumerate various kinds of entities, such as a list of American basketball coaches, a list of best-selling single-volume books and a list of best films winning the Empire Award. In our experiments, we evaluate the performance of LIEGE using only the list items whose mapping entities exist in the knowledge base. The summary of this data set is also shown in Table 2.

In the following experiments, we used YAGO(1)[1] of version 2008-w40-2 as the knowledge base, which is the same version as that used in [16]. We downloaded the May 2011 version of Wikipedia to construct the dictionary D introduced in Section 2. Then we used the dictionary D to generate the candidate mapping entity set R_i for each list item $l_i \in L$. If the size of the candidate mapping entity set $|R_i|$ of some list item l_i equals 0, we considered that the textual form of this list item has some noise. We processed this kind of list items by some methods, such as eliminating appositives (either within parentheses or following a comma), removing the sequence number at the beginning of the textual form, eliminating the redundant white spaces in the textual form and correcting the spelling errors using the query spelling correction supplied by Google. Then we generated the candidate mapping entity sets for these list items using the dictionary D again. To generate the document context for each entity to calculate the distributional context similarity, we employed these downloaded 3.5 million Wikipedia pages as the external document corpus, where each entity in the Wikitext has been converted to its canonical representation. To learn the weight vector introduced in Section 3.2, we set $\theta = 100$. In the next section, we will show that the results of our framework LIEGE is insensitive to the parameter θ and the choice of θ does not affect the results greatly. To evaluate the results of our framework, we calculated the *accuracy* as the number of correctly linked list items divided by the total number of all list items.

[1] http://www.mpi-inf.mpg.de/yago-naga/yago/

Table 3: Experimental results over Wiki_Manual

Approach	# correctly linked	Accuracy
TableAnno	1419	0.8392
LIEGE$_{\beta=0,\gamma=0}$	1461	0.8640
LIEGE$_{\beta=0}$	1519	0.8983
LIEGE$_{\gamma=0}$	1498	0.8859
LIEGE$_{full}$	**1536**	**0.9083**

Table 4: Experimental results over Web_Manual

Approach	# correctly linked	Accuracy
TableAnno	7518	0.8137
LIEGE$_{\beta=0,\gamma=0}$	7925	0.8578
LIEGE$_{\beta=0}$	8219	0.8896
LIEGE$_{\gamma=0}$	8097	0.8764
LIEGE$_{full}$	**8249**	**0.8928**

4.2 Experimental results

We compared our framework LIEGE with the algorithm proposed in [16], which we refer as *TableAnno*, over the Wiki_Manual and Web_Manual data sets. To give a fair comparison, we used the Wiki_Manual data set to learn the weight vector $\overrightarrow{w}$ and testing was done over the Wiki_Manual and Web_Manual data sets, which is the same way as that proposed in [16]. The experimental results of the baseline method *TableAnno* performing collective inference in the full model over the Wiki_Manual and Web_Manual data sets introduced in [16] are shown in Tables 3 and 4, respectively. The experimental results of our framework LIEGE over the Wiki_Manual and Web_Manual data sets are also presented in Tables 3 and 4 respectively. In these two tables, besides the accuracy, we also show the number of correctly linked list items. For our framework LIEGE, we do not only show the performance of LIEGE leveraging all the features introduced in this paper, which we refer as LIEGE$_{full}$, but also present the performance of LIEGE leveraging a subset of the features. When we calculated the *linking quality* using the Formula 11, if we set $\beta = 0$ and $\gamma = 0$, it means we only leveraged the feature of *prior probability* to address the list linking task, which we refer as LIEGE$_{\beta=0,\gamma=0}$. If we only set $\beta = 0$, it means that we leveraged the features of *prior probability* and distributional context similarity, which we refer as LIEGE$_{\beta=0}$. Likewise, if we only set $\gamma = 0$, it means that we leveraged the features of *prior probability* and type hierarchy based similarity, which we refer as LIEGE$_{\gamma=0}$. From

Table 5: Experimental results over Web_Lists

Approach	# correctly linked	Accuracy
LIEGE$_{\beta=0,\gamma=0}$	1969	0.7813
LIEGE$_{\beta=0}$	2329	0.9242
LIEGE$_{\gamma=0}$	2265	0.8988
LIEGE$_{full}$	**2352**	**0.9333**

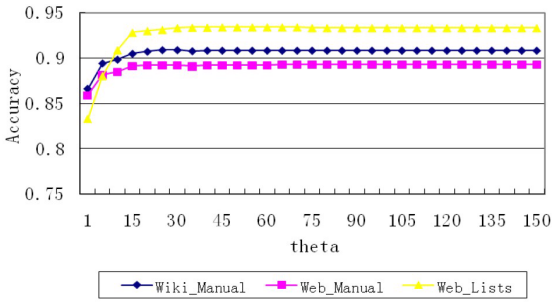

Figure 2: Sensitivity analysis to θ

the experimental results shown in Tables 3 and 4, we obtain the same conclusion that our framework leveraging different subsets of features achieves significantly higher accuracy than the baseline method *TableAnno*, which performs collective inference using entities, types and relationships. It can be also seen from the results in Tables 3 and 4 that every feature has a positive impact on the performance of our framework, and with the combination of all features LIEGE can obtain the best results.

The baseline method *TableAnno* is proposed to deal with the table annotation task [16], while for the Web list data set, *TableAnno* cannot perform collective inference using entities, types and relationships. Moreover, the approach LIEGE$_{\beta=0,\gamma=0}$, only leveraging the feature of *prior probability*, significantly outperforms the approach *TableAnno* over both Wiki_Manual and Web_Manual data sets. Thus, we did not compare our framework LIEGE with the *TableAnno* method over the Web_Lists data set. To learn the weight vector $\overrightarrow{w}$ on the Web_Lists data set, we used 2-fold cross validation. We show the experimental results of LIEGE leveraging different subsets of features over the Web_Lists data set in Table 5. From the results in Table 5, it can be seen that the list linking task over the Web_Lists data set is more challenging compared with that over the Wiki_Manual and Web_Manual data sets, as the accuracy achieved by the approach LIEGE$_{\beta=0,\gamma=0}$ over the Web_Lists data set (78.13%) is much smaller than those over the Wiki_Manual (86.40%) and Web_Manual (85.78%) data sets. However, the approach LIEGE$_{full}$ leveraging all the features obtains very high accuracy over the Web_Lists data set, which demonstrates the effectiveness of our framework LIEGE over the Web list data set.

To better understand the performance characteristics of our proposed LIEGE framework, we conducted sensitivity analysis to understand the influence of the parameter θ to the results of LIEGE. Here, we used our framework leveraging all the features LIEGE$_{full}$ as the example. Figure 2 depicts the performance of LIEGE$_{full}$ with varied parameter θ over these three data sets, i.e., Wiki_Manual, Web_Manual and Web_Lists, respectively. Recall that the parameter θ is called the penalty parameter in Section 3.2, which gives some penalty to the list item which violates the assumption. From the trends plotted in Figure 2, it can be seen that when θ is set to be larger than 40, the accuracies achieved by LIEGE$_{full}$ are stable with varied θ and remain the best over these three data sets, which demonstrates that the results of our framework are insensitive to the parameter θ, and the choice of θ does not affect the results greatly. Thus, we set $\theta = 100$ in all experiments introduced above. While we only report the performance of LIEGE$_{full}$ over these three data sets, similar trends are observed for the approaches LIEGE$_{\beta=0}$ and LIEGE$_{\gamma=0}$ over all these three data sets.

5. RELATED WORK AND DISCUSSION

As more and more knowledge bases like DBpedia [2], KnowItAll [9], YAGO [21, 20] and KOG [23, 24] are available publicly, considerable progresses have been made in linking entities in free text with a knowledge base [3, 6, 7, 13, 15, 19]. Entity linking is the task to link a textual entity mention in the unstructured free text, with the corresponding real world entity in an existing knowledge base. Bunescu and Pasca [3] firstly tackled this problem by exploiting a set of useful features derived from Wikipedia for entity detection and disambiguation. They leveraged the bag of words model to measure the cosine similarity between the context of the mention and the text of the Wikipedia article, and selected the entity whose Wikipedia article is most similar to the context where the mention appears as the mapping entity for this mention. Cucerzan [6] proposed a solution which is the first system to recognize the global document-level topical coherence of the entities. The system addresses the entity linking problem through a process of maximizing the agreement between the context of the entity mention and the contextual information extracted from the Wikipedia, as well as the agreement among the categories associated with the candidate entities. The learning based solution in [7] focuses on the classification framework to resolve entity linking. It develops a comprehensive feature set based on the entity mention, the contextual document and the knowledge base entry, and then uses a SVM ranker to score each candidate entity. Han and Sun [13] proposed a generative probabilistic model for the entity linking task. This model incorporates multiple types of heterogenous entity knowledge, i.e., popularity knowledge, name knowledge and context knowledge. Our previous work [19] proposed LINDEN to deal with the entity linking task. LINDEN is a novel framework to link named entities in text with a knowledge base unifying Wikipedia and WordNet, by leveraging the rich semantic knowledge embedded in the Wikipedia and the taxonomy of the knowledge base. In all these work, the essential step is to define a similarity measure between the text around the entity mention and the document associated with the entity. While in our list linking task, lists never have any header text information and textual context. The only input for the list linking task is a list of entity mentions extracted from the Web pages. Therefore, these methods cannot be applied to our list linking task.

Kulkarni *et al.* [15] proposed a general collective disambiguation approach based on the observation that coherent documents refer to entities from one or a few related topics or domains, and entities mentioned from a single document are likely to be semantically related. In that paper, they gave precise formulations for the trade-off between lo-

cal compatibility between mention and entity and measures of global coherence between entities. However, this observation cannot be applied to the Web list data, since the entity mentions in a Web list can be any set of entities that have the same conceptual type, that is to say, they are semantically similar, rather than semantically related. Accordingly, it can be seen that the list linking task is challenging and greatly different from the traditional entity linking task.

Another thread of research related to our work is managing structured data on the Web [1, 4, 5, 18, 8, 12, 22, 16]. These solutions were proposed to find, extract and integrate structured data on the Web. The work most related to our list linking task is the model proposed in [16] to address the problem of table annotation. In that paper, the table annotation task is defined as three subtasks: annotating each column of the table with one or more types, annotating pairs of columns with a binary relation in the knowledge base and mapping table cells with entities that they likely mention in the knowledge base. They proposed a new probabilistic graphical model for simultaneously associating entities for cells, types for columns and binary relations for column pairs. Specifically, they modeled the table annotation problem as a joint distribution over variables, and the goal is to find optimal values to the variables that maximize the joint probability. However, for the subtask of annotating table cells with entities, they did not consider the prior probability of entities being mentioned and the semantic similarity between each pair of entities in the same column or list, which have been shown to be significantly important and effective in our experiments for the list linking task.

6. CONCLUSION

In this paper, we have studied the problem of list linking. We propose LIEGE, the first framework that can effectively link list items in the Web lists with a knowledge base to the best of our knowledge. LIEGE is based on the observation that the type of the mapping entity should be coherent with the types of the other mapping entities in the same Web list. To evaluate the effectiveness of LIEGE, a thorough experimental study was conducted, and the experimental results demonstrate that our framework significantly outperforms the baseline method in terms of accuracy. Moreover, LIEGE can run in a fully automatic manner without tedious parameter tuning.

7. ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China under Grant No. 60833003, National Basic Research Program of China (973 Program) under Grant No. 2011CB302206, and an HP Labs Innovation Research Program award.

8. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348, 2003.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of ISWC*, pages 11–15, 2007.

[3] R. Bunescu and M. Pasca. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proceedings of EACL*, pages 9–16, 2006.

[4] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1:538–549, August 2008.

[5] M. J. Cafarella, A. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Webtables: exploring the power of tables on the web. In *Proceedings of WebDB*, 2008.

[6] S. Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of EMNLP-CoNLL*, pages 708–716, 2007.

[7] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *Proceedings of COLING*, pages 277–285, 2010.

[8] H. Elmeleegy, J. Madhavan, and A. Halevy. Harvesting relational tables from lists on the web. *Proc. VLDB Endow.*, 2:1078–1089, August 2009.

[9] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of WWW*, pages 100–110, 2004.

[10] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[12] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proc. VLDB Endow.*, 2:289–300, August 2009.

[13] X. Han and L. Sun. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of ACL*, pages 945–954, 2011.

[14] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[15] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of SIGKDD*, pages 457–466, 2009.

[16] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3:1338–1347, September 2010.

[17] D. Lin. An information-theoretic definition of similarity. In *Proceedings of ICML*, pages 296–304, 1998.

[18] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. *Proc. VLDB Endow.*, 1:1241–1252, August 2008.

[19] W. Shen, J. Wang, P. Luo, and M. Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of WWW*, pages 449–458, 2012.

[20] F. Suchanek, G. Kasneci, and G. Weikum. Yago: A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217, 2008.

[21] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *Proceedings of WWW*, pages 697–706, 2007.

[22] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4:528–538, June 2011.

[23] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *Proceedings of CIKM*, pages 41–50, 2007.

[24] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of WWW*, pages 635–644, 2008.

[25] T. Zesch, I. Gurevych, and M. Mühlhäuser. Analyzing and Accessing Wikipedia as a Lexical Semantic Resource. In *Biannual Conference of the Society for Computational Linguistics and Language Technology*, 2007.