Exploiting Collective Hidden Structures in Webpage Titles for Open Domain Entity Extraction

Wei Song[†]*, Shiqi Zhao[‡], Chao Zhang[‡], Hua Wu[‡], Haifeng Wang[‡], Lizhen Liu[†], Hanshi Wang[†]
[†]College of Information Engineering, Capital Normal University, Beijing, China
[‡]Baidu Inc., Beijing, China
{wsong, lzliu, hswang}@cnu.edu.cn
{zhaoshiqi, zhangchao01, wu_hua, wanghaifeng}@baidu.com

ABSTRACT

We present a novel method for open domain named entity extraction by exploiting the collective hidden structures in webpage titles. Our method uncovers the hidden textual structures shared by sets of webpage titles based on generalized URL patterns and a multiple sequence alignment technique. The highlights of our method include: 1) The boundaries of entities can be identified automatically in a collective way without any manually designed pattern, seed or class name. 2) The connections between entities are also discovered naturally based on the hidden structures, which makes it easy to incorporate distant or weak supervision. The experiments show that our method can harvest large scale of open domain entities with high precision. A large ratio of the extracted entities are long-tailed and complex and cover diverse topics. Given the extracted entities and their connections, we further show the effectiveness of our method in a weakly supervised setting. Our method can produce better domain specific entities in both precision and recall compared with the state-of-the-art approaches.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition*; I.2.7 [**Artificial Intelligence**]: Natural language processing; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Experimentation

Keywords

Open domain entity extraction;Multiple sequence alignment;URL pattern;Template generation;Weak supervision

*This work was done when the first author was visiting Baidu.

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741107.

1. INTRODUCTION

A named entity (entity for short) is a contiguous sequence of textual tokens, which represents the name of an object of a certain class, such as a person or an organization. Entity extraction is a key subtask of Information Extraction (IE), and also a fundamental component for many Natural Language Processing (NLP) and Information Retrieval (IR) tasks. For example, relation extraction is based on identifying entities in advance. Word segmentation and parsing would be improved if one system already knows that a piece of text is an entity. Besides, search engines have been paying much attention on semantic and knowledge driven search beyond traditional keyword based search paradigm. In such scenario, mining and integrating new and long-tailed entities would benefit knowledge base construction and help understanding user intent.

The main challenge of entity extraction is to precisely determine the boundaries of entities [14]. It is challenging, especially for long and complex entities like book names and restaurant names. The forms of these names might be flexible, containing multiple words and even other entities. Almost all existing approaches attempt to solve this problem in different ways, among which supervised models are based on various syntactic and contextual features extracted from manually annotated training data [5, 11, 22], while semi (weakly)-supervised methods make use of class names, seeds or manually designed syntactic patterns to induce templates for determining the boundaries of entities [12, 15, 26]. Another alternative way is to make use of explicit structured data such as HTML tags [1, 19] and HTML tables of webpages [13, 17].

However, most of the above work relies on domain knowledge or human labor. Due to the necessity of training data, supervised methods only work well on predefined categories [18]. Semi-supervised methods simplify the training process, but don't get rid of prior knowledge either. The manual labor scales with the number of classes of interest. Because HTML patterns are usually website specific, it is impossible for people to enumerate HTML patterns or set seeds for every relevant website. Therefore, search engines and seeds are usually needed for retrieving webpages and inducing wrappers [19, 31]. Besides, some open entity extraction methods are proposed [14, 21], which attempt to extract entities without human intervention. But existing systems commonly utilize language dependent heuristics, which are difficult to be applied to other languages.

In this paper, we propose an open entity extraction method, which extracts entities and their connections from the web

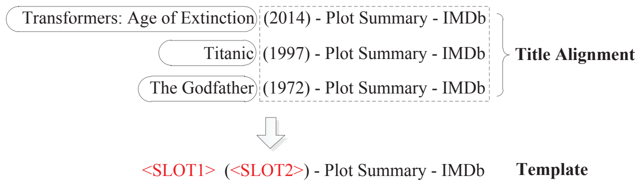
Transformers: Age of Extinction
(2014) - Plot Summary - IMDb
Titanic
(1997) - Plot Summary - IMDb
The Godfather
(1972) - Plot Summary - IMDb
Title Alignment
<SLOT1> (<SLOT2>) - Plot Summary - IMDb
Template

Figure 1: Webpage title alignment for inducing templates with substitutable slots.

without requiring prior knowledge, manual labor or sophisticated NLP techniques, and is language independent.

The key point is that we induce templates and extract entities by collectively identifying and aligning webpage titles with the same hidden textual structure. Our method is based on the following observation: **Parallel webpage titles that share the same hidden textual structures may contain certain classes of entities**. The term *hidden* here means that there is no explicit structure in an isolated webpage title, but if we align these titles together, the common structures among titles emerge naturally, such as the example shown in Figure 1. We can see that entities usually play a role as substitutable slot fillers within these titles. Therefore, the task to determine the boundaries of entities is actually casted to the task of inducing templates within parallel webpage titles and identifying substitutable slots in the templates.

Although the observation is obvious, the main challenges are two-fold: how to identify parallel webpage titles with the same hidden structures and how to induce and utilize the hidden structures for entity extraction.

For the first challenge, we identify webpage titles with the same or similar hidden structures by exploiting **generalized URL patterns**. We notice that webpage titles with similar URL morphology tend to share similar textual structures. Based on this strategy, we can collect large scale of parallel webpage titles, which are used as the data resource for entity extraction.

For the second challenge, we first apply the idea of **Multiple Sequence Alignment** (MSA) to align these webpage titles for inducing templates automatically. Then, we utilize search engine user behaviors and heuristic constraints to filter out invalid templates which fail to produce valid entities. Finally, for each target webpage title, we rank candidate templates based on the probability that they could correctly determine the entity boundary in this title and choose the most appropriate one for entity extraction.

To the best of our knowledge, we are the first to exploit webpage titles for entity extraction in an unsupervised manner. Figure 2 shows the working flowchart of the proposed approach. The system takes webpages as input and no additional manual labor is necessary. The system output includes extracted open domain entities and their connections. The connections are formed by recording from which templates each entity is extracted. Since the entities extracted by the same template are often of the same class, the connections reveal topic relatedness among entities and make it easy to incorporate distant or weak supervision.

In summary, the main contributions of this paper include:

- We exploit the hidden textual structures in webpage titles for open domain entity extraction. We propose algorithms for both discovering and utilizing these structures, which make our method get rid of manual labor.
- We demonstrate experimentally that: 1) Our method can extract large scale of open domain entities with high precision and topic coverage, many of which are long-tailed entities. 2) Based on the extracted entities and their connections, we can easily incorporate weak supervision for mining more accurate and complete domain specific entities. In both settings, our method outperforms the state-of-the-art approaches based on search engine query logs.

The rest of this paper is organized as follows. In Section 2, we propose our method. We show our experiments and results on open domain entity extraction in Section 3, and demonstrate its effectiveness on domain specific entity extraction in Section 4. We introduce the related work in Section 5 and conclude this paper in Section 6.

2. PROPOSED METHOD

2.1 Motivation

Our idea comes from the following simple observations:

- There are extensive entities of various topics existing in webpage titles, especially from vertical domain websites.
- The webpage titles, which are from the same website and contain the same class of entities, have similar hidden textual structures.
- Within the textual structures, entities play a role as substitutable slot fillers.

Numerous webpages are about entities. Because webpage titles summarize the key concepts of the content, if a webpage introduces an entity, it is natural that the title contains the entity. Considering the redundancy of the web, for one entity, it is quite likely that there is at least one webpage title containing it. Therefore, *webpage titles are valuable resources for entity extraction.*

The second and the third observations are also understandable. The webpages introducing the same classes of entities or their attributes are often organized in the same way within a website. As shown in Figure 1, it is easy to see that three titles share the same hidden structure:

$< movie name >$ ($< digit >$) - Plot Summary - IMDB

The hidden structure forms a **template**. The field like $< movie name >$ or $< digit >$ is substitutable, noted as a **slot**. The **slot fillers** are the text fragments which could be placed at the slot positions and expected to be some class of entities. This phenomenon becomes obvious when these titles are aligned together. Motivated by this, we aim to learn an extractor by aligning webpage titles with the same hidden textual structures collectively.

2.2 Generalized URL Patterns

A generalized URL pattern is a regular expression which could recognize a set of URLs. The webpage titles of URLs which could be recognized by the same generalized URL pattern are named as parallel webpage titles. We expect

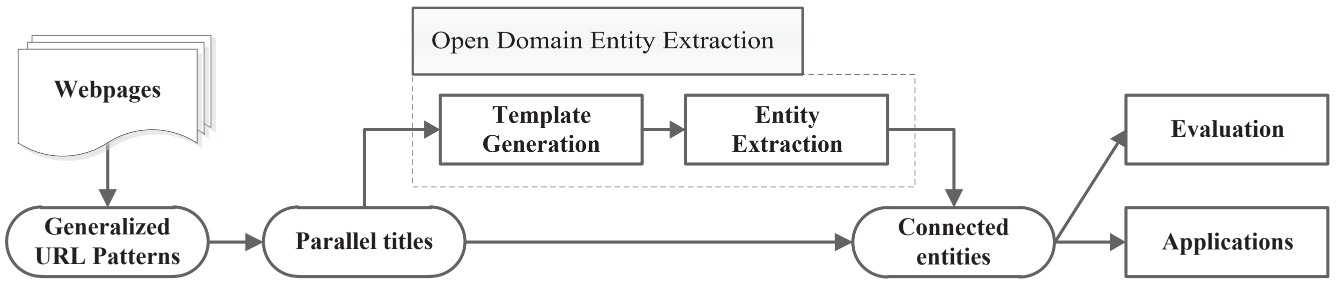

Figure 2: The flowchart of the proposed system.

that parallel webpage titles should have the same or similar hidden textual structures. The example below shows a generalized URL pattern and 2 URL instances.

$Pattern$: $http://www.imdb.com/title/tt\backslash d+/\$$

$URL1$: $http://www.imdb.com/title/tt0000001/$

$URL2$: $http://www.imdb.com/title/tt0000002/$

Exploiting generalized URL patterns for identifying parallel webpage titles is motivated by the idea that topic related webpages often form clusters and share the same URL pattern in a website [6, 32]. We learn the generalized URL patterns from a large scale URL database. The process is as follows:

Step 1: Segment the non-domain part of each URL with "/". We will generate candidate URL patterns by replacing one segment with a regular expression each time. For example, the candidate patterns for URL1 are $http://www.imdb.com/REG/tt0000001/\$$ and $http://www.imdb.com/title/REG/\$$, where REG is a special symbol to represent regular expressions.

Step 2: To determine whether a segment should be generalized, we accumulate all candidate patterns over the URL database. The ones with a frequency above a threshold are retained, such as
$http://www.imdb.com/title/REG/\$$.

Step 3: For each retained candidate pattern, the final generalized URL pattern is got by replacing the symbol REG with a regular expression based on the fillers of this segment. The resulting pattern for URL1 becomes $http://www.imdb.com/title/tt\backslash d+/\$$.

Given one generalized URL pattern, we can get corresponding parallel webpage titles which share the same or similar textual structures. These structures are difficult to be discovered if we deal with each title separately. We will show the structures could be discovered automatically by aligning them together in a collective way.

2.3 Open Domain Entity Extraction

Given parallel webpage titles of a generalized URL pattern, we aim to generate templates to uncover the hidden textual structures for entity extraction.

2.3.1 MSA based Template Generation

We view parallel titles as a set of word sequences and conduct alignment to induce templates. The textual structures of parallel titles in real data often have variations. There might be multiple templates, each of which matches a subset of the examined parallel titles. Therefore, we first adapt the Needleman-Wunsch algorithm [23] for pair-wise alignment to induce all possible templates.

Pairwise Alignment. We define the element set as $\Sigma = \mathcal{V} \bigcup \{\_\}$, where $\mathcal{V}$ refers to the word vocabulary, and "_" is used to represent an indel (suppose "_" $\notin \mathcal{V}$). Given two word sequences $A \in \mathcal{V}^*$ and $B \in \mathcal{V}^*$, an alignment can be represented as a 2-dimensional array $Align^{A,B}_{2\times I}$ that every word in one sequence is aligned to one word in the other sequence or to an indel which is caused by **inserting** a word into one sequence or **deleting** a word from the other. I is the number of aligned element pairs. A *substitution matrix* R is used to assign alignment scores, in which $R(a,b)$ represents the alignment score between a pair of elements $(a,b) \in \Sigma\times\Sigma$. A well aligned pair will be rewarded with a higher score. We define an overall score function $AlignScore(A,B)$ as the sum of the scores over all aligned element pairs in two sequences:

$$AlignScore(A,B) = \sum_{i=1}^{I} R(Align^{A,B}_{0,i}, Align^{A,B}_{1,i}) \quad (1)$$

Therefore, the task, as shown in Equation 2, is to find an optimal alignment with the best overall score.

$$Align^* = \arg\max_{Align^{A,B}_{2\times I}} AlignScore(A,B) \quad (2)$$

To achieve this, we should first define a proper substitute matrix R. In detail, $R(a,b)$ is defined as follows.

- Exact Match: $a \in \Sigma$ and $b \in \Sigma$ are exact matched if $a == b$. Define $R(a,a) = \alpha$, $\alpha > 0$, so that identical match is awarded. If two words are both delimiters, they are considered as an exact match as well.

- Mismatch: Include two types: normal word mismatch and gap penalty. If $a \in \mathcal{V}$ and $b \in \mathcal{V}$ are different normal words, define $R(a,b) = \beta$, $\beta < 0$. If $a \in \mathcal{V}$ and $b ==$ "_", define $R(a,b) = d$, $d < 0$.

- Forbiden Match: A normal word a is not allowed to be aligned with a delimiter b, for which we define $R(a,b) = -\infty$.

We construct a delimiter dictionary manually including "[", "|", "-" and so on. The delimiters and normal words together form the vocabulary $\mathcal{V}$. We specially consider delimiters, because delimiters are often used for segmenting different blocks in titles. Generally, delimiters shouldn't be part of an entity so that they shouldn't be aligned with normal words. Normal word mismatches are penalized by assigning a negative score, while gap mismatches are actually to penalize long distance alignment.

Given the substitute matrix R, we compute a matrix F by dynamic programming as described in Algorithm 1. The entries of F give the maximum scores among all possible alignments. Once the matrix F is computed, we backtrack the coming path starting from the bottom right corner of F and get the optimal alignment.

Algorithm 1 Computing Scoring Matrix for All Possible Alignments

```
Input:
    Sequences $A$ and $B$;
    Substitution Matrix $R$ and gap penalty $d$;
Output:
    Score Matrix $F$;
 1: /* Initialize matrix $F_{(length(A)+1)\times(length(B)+1)}$ with a linear gap penalty. */
 2: for $i = 0$ to $length(A)$ do
 3:     $F_{i,0} = i * d$;
 4: end for
 5: for $j = 0$ to $length(B)$ do
 6:     $F_{0,j} = j * d$;
 7: end for
 8: /* Compute optimal scores. */
 9: for $i = 1$ to $length(A)$ do
10:     for $j = 1$ to $length(B)$ do
11:         MATCH $\longleftarrow F_{i-1,j-1} + R(A_i, B_j)$;
12:         DELETE $\longleftarrow F_{i-1,j} + d$;
13:         INSERT $\longleftarrow F_{i,j-1} + d$;
14:         $F_{i,j} = \max\{$MATCH, DELETE, INSERT$\}$;
15:     end for
16: end for
17: return $F$;
```

Template Induction. Given a pairwise sequence alignment, we can induce a template for each sequence accordingly. We replace consecutive mismatch words in each sequence with a <*SLOT*> symbol which represents a template slot. The matched parts are considered as the backbone of the template. If one template does not contain <*SLOT*> or the template itself is <*SLOT*>, it is discarded, because this indicates two sequences are the same or completely different. An example is shown below.

Given two sequences A = "$w_1[w_2w_3]$", B = "$v_1(w_2w_3)$" respectively, the alignment result and corresponding templates are :

A:	w_1	_	[	w_2	w_3	]
B:	_	v_1	(	w_2	w_3	)
		$\Downarrow$				
Template 1:		<SLOT>	[	w_2	w_3	]
Template 2:		<SLOT>	(	w_2	w_3	)

We accumulate all templates generated by pairwise alignment and remove the templates with low frequencies.

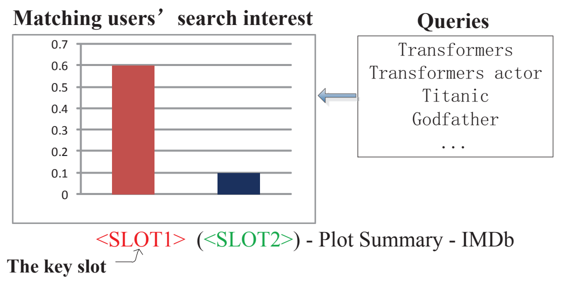

Figure 3: User behavior based slot selection, in which we can see that the slot for movie names is selected as the key slot, since it matches users' search interest when clicking on the corresponding titles.

User Behavior based Slot Selection. Some of the retained templates contain multiple slots. Such as the example shown in Figure 1, the slot fillers for movie names are what we really want. To select the right slot in a template, which is termed **key slot** in what follows, we take advantage of user click-through behaviors in search engines. Our assumption is that the key slot of a template is more likely to match users' search interest. In other words, for titles matched by the template, the fillers of the key slot are more likely to match users' search queries that click on these titles. A motivating example is shown in Figure 3.

We consider templates and user queries together through clicked URLs in query logs. Suppose a template T has m slots. One title x is matched by T and the corresponding slot fillers are $\{f_1(x), ..., f_m(x)\}$, where $f_i(x)$ is the slot filter of the ith slot s_i of the template T. If the title x is clicked by a searcher submitting query q, a title-query pair (x, q) forms. By accumulating all title-query pairs $\{(x, q)\}$ related to template T, the weight of the ith slot s_i to user queries could be measured as:

$$weight(s_i, T) = \sum_{(x,q)} Jaccard(f_i(x), q) \qquad (3)$$

where $Jaccard(.,.)$ is the jaccard similarity between two word sets. We choose $s^* = \arg\max_i weight(s_i, T)$ as the key slot of template T, if $weight(s^*, T) > 0$. So if one template has more than one slot, we use all template slots for generalization, but extract slot fillers as candidate entities only from the key slot. This constraint asks help from searchers to guarantee the quality of entities.

Template filtering. We further filter out invalid templates by assessing the quality of extracted candidate entities. To this end, we first identify high-confident valid and invalid entity sets in a heuristic way and then design two indicators for template filtering.

- Lexicon based indicator: According to our observation, invalid candidate entities are often extracted from the titles of news or forum webpages. To filter out such candidates, we construct a stopword list including interrogative words and colloquial words like "where", "who", "ah" and some punctuation. If a candidate entity contains these words, it is considered as invalid.

- Redundancy based indicator: We figure out a set of valid entities based on redundancy. A candidate entity is more likely to be valid if it appears many times on the web. In this work, a candidate entity is considered as valid if it can be extracted more than 5 times by different templates.

Finally, a template is considered as a valid template, if it could produce at least J valid entities determined by the redundancy based indicator and less than K invalid entities determined by the lexicon based indicator.

2.3.2 *Template based Entity Extraction*

We apply the generated valid templates for entity extraction. One issue is that one title could be matched by more than one template at different granularity. For example, the title below can be matched by the following three templates.

Title: Stir Fried Beef Recipe - Website name

Template 1: <SLOT> Recipe - Website name

Template 2: <SLOT> - Website name

Template 3: <SLOT> Beef Recipe - Website name

To choose the best template for each title, we view it as a template ranking problem. Formally, given the titles $\mathcal{X} = \{x_1, ..., x_{|\mathcal{X}|}\}$ for one URL pattern, and a template set $\mathcal{T} = \{t_1, ..., t_{|\mathcal{T}|}\}$ associated with $\mathcal{X}$, we have to select the best template $t \in \mathcal{T}_x$ for each title $x \in \mathcal{X}$ for entity extraction, where $\mathcal{T}_x$ is a subset of $\mathcal{T}$ that can match x.

At first glance, some simple heuristics could be used for ranking:

- Common-First: Choose the template with the best coverage over all titles.
- Alignment-First: Choose the template with the best alignment score with the title.

The first heuristic will work for most titles, but may produce *super-entities* in some cases. For example, applying template 2 in the above example will extract "*Stir Fried Beef Recipe*" as a candidate entity. In contrast, alignment-first heuristic should be preferred for extracting clean entities. But sometimes, it may result in *sub-entities* such as "*Stir Fried*" which is extracted by applying template 3. Such templates are generated when two entities with common words happen to appear in paired titles.

To balance between the two extremes, we propose an **adaptive ranking (AdaRank)** function combining both alignment preference and popularity. Given title x, the probability of choosing $t \in \mathcal{T}_x$ as the perfect template can be represented as:

$$p(t|x) = \frac{p(x|t)p(t)}{p(x)} \tag{4}$$

$p(x|t)$ represents the likelihood that title x is generated using template t. Here, we approximate it using $\frac{1}{alRank(t,x)^\rho}$, where $alRank(t, x)$ is the rank of template t among $\mathcal{T}_x$ according to alignment scores. This means that templates with larger alignment scores with the title are more likely to generate the title. ρ is a positive value to control the punishment degree for the low rank templates. A larger value of ρ favors top ranked templates more. $p(t)$ can be seen as the prior probability of template t being a good template. We estimate $p(t)$ based on the global alignment information:

$$p(t) = \frac{\#(alRank(t,\cdot) == 1)}{|\mathcal{X}|} \tag{5}$$

where $\#(alRank(t,\cdot) == 1)$ represents the number of titles in $\mathcal{X}$ that template t has the largest alignment score with them. According to the probability ranking principle and removing constant factors, we compute the ranking score of the template t for the title x as:

$$rank(t|x) \equiv \frac{\#(alRank(t,\cdot) == 1)}{alRank(t,x)^\rho} \tag{6}$$

In summary, AdaRank integrates both the local alignment score and the global score of a template and achieves the best balance between them. It prefers the templates with both good alignment score and a certain size. Based on Equation 6, we use the top 1 ranked template for entity extraction and store the pair of the entity and the template for extracting it. The same procedure is applied to all titles as summarized in Algorithm 2.

Algorithm 2 Extracting Entities with Templates

```
Input:
    A set of webpage titles $\mathcal{X}$ of certain URL pattern;
    A set of templates $\mathcal{T}$;
Output:
    A set of entity-template pairs, $ET$;
1: Initiate $ET = \oslash$;
2: for each $x$ in $\mathcal{X}$ do
3:     Get templates $\mathcal{T}_x$ which can match $x$;
4:     Rank $\mathcal{T}_x$ to get the best template $t^*$ for $x$;
5:     Extract the slot filler $e$ from $x$ which fills the key slot of $t^*$;
6:     Add $(e, t^*)$ to $ET$;
7: end for
8: return $ET$;
```

3. EVALUATING OPEN DOMAIN ENTITY EXTRACTION

In this section we evaluate the quality of extracted entities. We first describe the data, then introduce the baseline our method is compared with. Finally, we report the experimental results on system comparisons and take a deep insight into the extraction results.

3.1 Experimental Settings

3.1.1 *Data*

The data we used for experiments include query logs and a webpage dataset. We use query logs from Baidu, which is the largest Chinese search engine in the world. In all, there are more than 1.9 billion queries and corresponding user clicked URLs. In our work, the query logs are used for selecting key slots. While the baseline (see Section 3.1.2) extracts entities directly from the queries.

Different methods extract entities from different resources. We constructed a webpage dataset according to the scale of query logs. First, generalized URL patterns were learned using the strategy described in Section 2.2 on 30 billion webpages. Then, we retained generalized URL patterns which

Table 1: Statistics of the data.

Webpage dataset	Volume
#generalized URL patterns	918,541
#webpages	11,056,478,017
query logs	**Volume**
#queries	1,929,617,042

Table 2: Performance on open entity extraction.

Method	#Entity	Precision	Coverage
OEQ (10, 0.1)	12419342	0.62	0.31
OEQ (100, 0.1)	930133	0.67	0.025
Ours	12375492	0.81	0.43

could recognize at least one user clicked URL in the query logs. Finally, we collected all webpages whose URLs can be recognized by one of the retained generalized URL patterns. We only used the titles of these webpages for entity extraction in our method. The statistics of query logs and webpage dataset are shown in Table 1.

Because we deal with Chinese, all webpage tiltes are segmented into words for further processing such as sequence alignment. Note that since online encyclopedias have URL patterns as well, our method could exhaust entries from them. To eliminate this effect of these websites, our method doesn't use URL patterns of three main Chinese online encyclopedias: Baidu Baike[1], Hudong Baike[2] and Chinese Wikipedia[3].

3.1.2 *Baseline*

The most common data resources for entity extraction are web documents and query logs. Extracting open domain entities from web documents without any supervision is hard and computationally expensive. As reported in both [21] and [25], entity extraction from query logs outperforms systems based on documents. In this work, we aim to show that by exploiting the hidden textual structures, entity extraction from webpage titles can achieve superior performance compared with approaches based on query logs.

We implemented a method which extracts open entities from queries [21], noted as OEQ. We adapt OEQ to Chinese: 1) We extract all text fragments segmented by blank characters from queries, since an entity should not cross 2 fragments separated by blank characters in Chinese. 2) We extract all k-grams in text fragments and count their frequencies. 3) We retain all k-grams, if they satisfy the representation and stand-alone constraints. Representation constraint is that the number of entity occurrence must exceed a threshold. Stand-alone means that one entity forms a query by itself, and stand-alone constraint is that the ratio of an entity occurring in a stand-alone way must exceed a threshold. Refer to [21] for more details.

3.1.3 *Parameters*

Our method has several parameters to be set. Due to the large size of dataset, we select a small portion of data to tune the parameters. For pairwise alignment, we define the substitute matrix by setting $\alpha = 2$, $\beta = -1$ and $d = -2$. The templates matching less than 20 titles were eliminated after template induction. For template filtering, we set parameter K to 20, and J to 30. For adaptive template ranking, ρ is set to 2.

For the baseline, we set the threshold for stand-alone score to 0.1 which is the same in [21], and tried several values of representation score threshold. We would report the performance when it is set to 10, with which OEQ produces a similar scale of entities to us. The quantity of produced entities reduces as the representation score threshold gets larger. We also report the results when the threshold is set to 100 as a reference.

[1]http://baike.baidu.com/
[2]http://www.baike.com/
[3]http://zh.wikipedia.org/

3.2 Experimental Results

3.2.1 *Overall Performance*

Precision and Coverage. We randomly sampled 500 entities produced by each system respectively, and asked two annotators to judge whether the extracted ones are correct. The annotation standard is according to [28]. Entities that are names of specific objects or general concepts such as "*低碳经济(low-carbon economy)*" are all considered as correct. Inner-annotator agreement measured by Kappa coefficient [7] is 0.79. We adopt *Precision* as an evaluation metric to measure the ratio of correct entities, which is defined as $\frac{\#\text{correct extracted entities}}{\#\text{all extracted entities}}$. Because it is impossible to evaluate the recall, we evaluate the *Coverage* by comparing the system outputs against entries in Baidu Baike—the largest Chinese online encyclopedia, which has more than 6.5 million entries. We didn't consider entity alias so that the *Coverage* is measured based on exact string matching.

Table 2 shows the experimental results. We can see when producing similar quantity of entities, our method has higher precision compared with OEQ. This indicates that our method could produce large scale of entities with an acceptable precision. OEQ achieves higher precision when setting a larger threshold (100) for representation score at the price of a sharp decrease of coverage. But the precision is still obviously lower than our method. On coverage, our method outperforms OEQ as well. Because both methods could be applied to larger dataset, the coverage could be further improved.

Topic Analysis. We examine the topic distribution of extracted entities to see whether they cover a wide range of topics. To this end, we extract all entities with category tags from Baidu Baike. In detail, Baidu Baike maintains an entity category taxonomy, based on which entities can be assigned a category tag by users. The taxonomy contains 23 general topical categories and 90 sub-categories. In this experiments, if a Baidu Baike entity already has a subcategory tag, we will automatically assign its higher level general category to it. In this way, we collected about 1.1 million entities in Baidu Baike with general category labels. In Figure 4, the black bars show the number of entities each category has. We can see the distribution is unbalanced, person (PER), literature work (LITERATURE) and organization (ORG) are the most popular categories.

Among the 1.1 million entities, more than 0.59 million entities could be extracted by our method. The distribution over categories of our extracted entities are shown in Figure 4 as well. We can see that the distribution is consistent with that of the entities in Baidu Baike. The coverage ranges

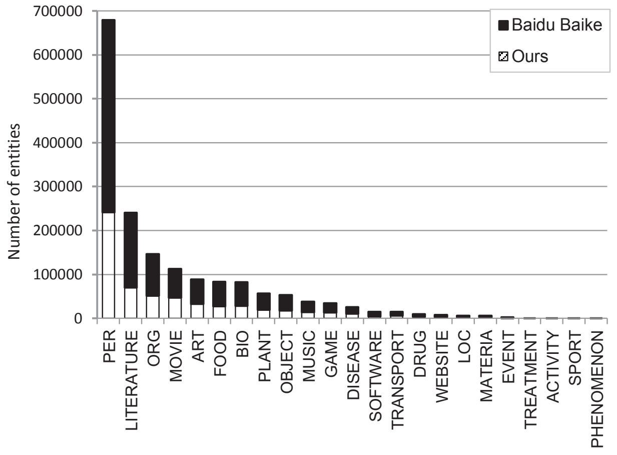

Figure 4: Entity category distributions.

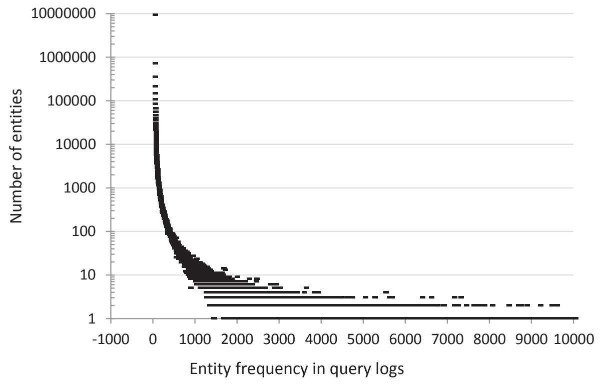

Figure 5: Number of entities with different frequencies in query logs.

from 41% (on LITERATURE) to 74% (on SPORT) across all categories. This indicates that the entities extracted by our method cover diverse topics.

Frequency Analysis. We analyse the frequencies of our extracted entities in query logs. Figure 5 illustrates the number of entities with different frequencies. We can see that about 9 million entities never occur in query logs which make up 73% of all entities. This indicates that our method is good at extracting long-tailed entities that are unable to be extracted from queries.

Length Analysis. Discovering complex multi-word entities is important and challenging. Thus we analyze the length (the number of words) of the extracted entities. Figure 6 shows the distribution of entities extracted by OEQ and by ours across different length intervals. We can see that OEQ extracts more simple entities with length no longer than 3, while our method extracts more long entities. From the outputs of our method and OEQ, we sampled 100 entities whose lengths are longer than 10 and evaluated the precision. Our method achieves a *Precision* of 77%, while OEQ gets a *Precision* of 52%. Many errors resulted by OEQ are long queries revealing some hot information need. Although they frequently appear on web, they are not entities.

Error Analysis. We also analyze the incorrect entities extracted by our method. The errors could be categorized

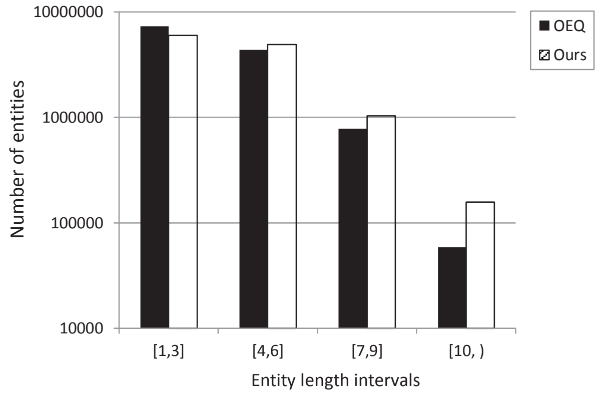

Figure 6: Number of entities across different length intervals.

into 3 types: 1) Too general classes. The extracted entities play a navigational function in webpage titles and usually are related to general information needs, such as "*关于爱国的作文(essays about patriotism)*". 2) Containing entity attribute or intent. The extracted candidates describe attributes or aspects of entities, such as "*北京景点(Attractions in Beijing)*". 3) User generated errors. Some webpage titles are generated by users. In some cases, although the titles share the same hidden structure, the slot fillers are noisy due to informality of users' edit.

Discussion. Based on the above analysis, we show that our method could extract massive, long-tailed entities with high precision and topic coverage. These entities have unique characteristics compared with the ones extracted from search queries and those already existing in encyclopedias: 1) The entities extracted from webpage titles are more formal. For example, we can extract complete full names of objects such as books, research papers and organizations. In contrast, many entities extracted from queries are shorten forms or alias of full names. 2) Large scale of entities are long-tailed. Less popular entities, which are difficult to be extracted based on redundancy, can be extracted by templates induced based on hidden structures. 3) Entities of some fine-grained domains can be extracted. For example, we can find names of rare plants, because our method can automatically discover and extract entities from vertical websites. These long-tailed fine-grained entities are potentially useful for development of vertical applications, for which we expect to exhaust all entities belonging to the application domain. On the other hand, although the entities in encyclopedias cover a wide range of domains, the coverage on long-tailed entities of certain domains is far from enough for satisfying the requirement of practical applications.

On the other side, we find that our method does not work well on extracting common concepts such as *climate*. The reason may be that few website collects and organizes common concepts together.

3.2.2 Evaluating Generalized URL Patterns

We randomly sampled 100 from the retained generalized URL patterns introduced in Section 3.1.1 and judged whether the corresponding titles have hidden structures. The results show that 78 out of 100 generalized URL patterns have hidden structures (though they might not produce correct enti-

Table 3: Performance on template ranking.

Method	Precision@1
Common-first	0.79
Alignment-first	0.84
AdaRank	0.93

ties). The parallel webpage titles that have no hidden structure can be divided into 2 categories: 1) All titles are the same. 2) All titles are totally different. Fortunately, our extractor can deal with both cases so that they cannot affect entity extraction. 26 among the 78 generalized URL patterns have a single hidden structure each, while the others have more hidden structures. This analysis indicates that generalized URL patterns are useful for identifying webpage titles with hidden textual structures.

After template filtering, 92,318 generalized URL patterns remain, which account for 10% of all. Each remained generalized URL pattern has 24 entity extraction templates on average. However, the most frequent template for each pattern can cover nearly 73% webpage titles. The number increases to 93%, if we take top-5 templates into account for each URL pattern. This further reveals that webpage titles with the same URL pattern truly have highly similar hidden textual structures.

3.2.3 Evaluating Entity Extractor

Evaluating Template Generation

Data and metrics. Here we also evaluate the entity extraction templates. Specifically, we evaluate whether we have correctly detected the key slot. For this purpose, we randomly sampled 200 templates, and asked two annotators to label the selected key slot as correct or not. To facilitate the annotation, we also provide 10 fillers for each slot in each template, so that the annotators can judge whether each examined slot can be used to extract correct entities. We use $Precision$ as an evaluation metric, which is defined as $\frac{\text{\#correctly selected key slots}}{\text{\#selected key slots}}$.

The experimental results show that our method achieves a $Precision$ of 81.5%. This indicates that user behavior based slot selection is effective for selecting high quality slots in templates for entity extraction.

Evaluating Template Ranking

Data and metrics. We randomly sampled 50 generalized URL patterns with more than 10 entity extraction templates after template generation phase (Section 2.3.1). From each URL pattern, we sampled 10 titles which could be matched by at least 2 templates. The resulting dataset consists of 500 titles which have 16 templates on average. For each title, we rank the templates using three ranking strategies introduced in Section 2.3.2: Common-first, Alignment-first and AdaRank. We use $Precision$@1 as a metric to see whether the top one template can correctly determine the boundary of the entity for each title.

Results. Table 3 shows the results. We can see that AdaRank obviously outperforms the other two strategies. This indicates that considering both local alignment and global popularity is effective for ranking entity extraction templates. Alignment-first heuristic performs better than common-first. All errors of alignment-first heuristic come from incorrect word alignment in multi-word entities, in which single words within two entities are aligned so that entities are incorrectly split into fragments.

4. INCORPORATING SUPERVISION

The extracted entity-template pairs can be viewed as a data resource for more specific tasks. Actually, the entity-template pairs form a bipartite graph. Therefore, the distant supervision (e.g. existing entries in knowledge bases) and weak supervision (e.g. manually provided seeds with labels) can be easily incorporated by propagating information through this graph. In this paper, we apply this data resource for set expansion of domain entities.

4.1 Set Expansion of Domain Entities

We focus on evaluating it in a weakly supervised setting as in [26]: **a few seeds S of a domain is given, the system should return more entities of the same domain.** This setting is useful for knowledge base expansion based on available resources.

We generate a subset of entities that are potentially relevant to the seeds from the entity-template pairs we have extracted and then rank these entities. This process involves the following steps:

Step 1: Extract all templates $\mathcal{T}$ that contain at least one of the seeds.

Step 2: Build a bipartite graph $\mathcal{G} = \{\mathcal{T} \cup \mathcal{E}, \mathcal{L}\}$, where $\mathcal{E}$ represents the set of entities that can be extracted by templates in $\mathcal{T}$. If an entity $e \in \mathcal{E}$ is extracted by a template $t \in \mathcal{T}$, there is an edge $l \in \mathcal{L}$ between them.

Step 3: An iterative algorithm is adopted to rank entities in this graph.

We use $TScore(t)$ to represent the relevance of a template t, use $EScore(e)$ to represent the relevance of an entity e, and use $Pri(e)$ to represent the prior relevance of e to the examined domain. We set the $Pri(e)$ for seed nodes as $\frac{1}{|S|}$ and 0 for all other entities. The iterative algorithm is motivated by the personalized PageRank [20]. It starts by initializing the $TScore^0(t)$ for all templates and the $EScore^0(e)$ for all entities as 0. Then in each iteration, the relevance scores of entities and templates in this graph are updated alternatively as follows.

$$EScore^{i+1}(e) = \mu Pri(e) + (1-\mu)\sum_t p(e|t)TScore^i(t) \quad (7)$$

$$TScore^{i+1}(t) = (1-\mu)\sum_e p(t|e)EScore^i(e) \quad (8)$$

where $p(e|t)$ is the probability that a template t extracts an entity e. If t extracts e, $p(e|t) = \frac{1}{\text{\#entities extracted by t}}$, otherwise $p(e|t) = 0$. Similarly, $p(t|e)$ is set to $\frac{1}{\text{\#templates that extract e}}$, if t extracts e, otherwise $p(t|e) = 0$. μ is a real value between 0 and 1 for a linear combination of the prior relevance of an entity and the sum of the relevance scores of the templates that can extract it. The algorithm will iterate for a predefined number of iterations.

We can find from the whole process that the hidden structures represented by templates play important roles for propagating information.

4.2 Experimental Settings

4.2.1 Baseline

We compare our domain specific entity expansion method with the method proposed in [26], which is the state-of-the-art weakly supervised method for domain specific entity extraction based on web search logs, noted as WSLOG. This algorithm makes use of seeds of a given domain to find patterns from queries of a search engine and further mines and ranks entities matching the patterns.

4.2.2 Data and Metrics

We conducted experiments on 10 domains that are the same as those examined in [26]. For each domain, 5 seeds are given. We re-implemented WSLOG and conducted experiments using the query log data introduced in Section 3.1.1. Our method runs on the entity-template pairs extracted by our open entity extraction module. The parameter μ is set to 0.5, and the algorithm iterates 10 times.

For each domain, we chose the top ranked 500 expanded entities from each method and pooled them together. We asked two annotators to judge the quality of these pooled entities of each domain. One entity that can be annotated as correct must satisfy: 1) It is a correct entity. 2) It belongs to the examined domain. The Kappa value of annotation is 0.82. We use $Precision(P)$, $Recall(R)$ and $F1$ as evaluation metrics. We define $P = \frac{\#\text{correct entities extracted by the system}}{\#\text{all extracted entities by the system}}$, $R = \frac{\#\text{correct entities extracted by the system}}{\#\text{all correct entities among pooled entities}}$ and $F1 = \frac{2 \times P \times R}{P+R}$.

4.3 Experimental Results

Table 4 shows the performance of our method and WSLOG. In most domains, our method outperforms WSLOG in both precision and recall.

The contextual templates extracted by WSLOG based on seeds sometimes cannot correctly determine the boundaries of entities in queries. For example, many templates for domain *University* can extract text fragments like "*2014北京大学(2014 Peking University)*" as candidate entities. In contrast, our method extracts entities collectively from parallel webpage titles. With the support of weak supervision, our method can better avoid incorrect entity boundaries.

In addition, WSLOG favors popular entities which appear enough times in query logs, but it can hardly mine long-tailed entities well. For the domain *Food*, WSLOG and our method both achieve good precision but low recall. It indicates that the correct entities extracted by two methods have a low overlap. WSLOG provides more short food names such as "*鱼(fish)*" and "*猪肉(pork)*", because these appear more often in queries. Even we use complex seeds like "*宫保鸡丁(Kung Pao Chicken)*", the simple food names still rank higher. In contrast, our method can provide more complex entities like "*西红柿炒鸡蛋(Scrambled Egg with Tomato)*".

Finally, the templates extracted by WSLOG are ambiguous in some cases. For example, the templates like "*<drug> 过敏(allergy)*" and "*<drug> 说明书(instruction)*" are important templates for domain *Drug*. However, these templates can extract entities of other domains such as *Cosmetics* as well. Actually, our method also faces the challenge on distinguishing entities from highly related domains. This is because the entities can be grouped together according to the purpose of the designers of the websites. For example, newspapers, TV programs and magazines are often organized together in some websites. That is why our performance on domain *Newspaper* is not so good. Extracting entities of fine-grained domains is necessary to be studied further.

Table 4: Comparison with WSLOG on set expansion of domain entities.

Domain	WSLOG			Ours		
Metrics	P	R	$F1$	P	R	$F1$
City	0.81	0.69	0.75	0.91	0.77	0.83
Country	0.1	0.31	0.15	0.30	0.94	0.46
Drug	0.49	0.52	0.5	0.75	0.81	0.78
Food	0.98	0.57	0.72	0.88	0.52	0.65
Location	0.84	0.66	0.74	0.99	0.78	0.87
Movie	0.64	0.54	0.59	0.92	0.77	0.84
Newspaper	0.11	0.46	0.18	0.17	0.72	0.27
Person	0.77	0.35	0.48	0.98	0.45	0.61
University	0.53	0.45	0.49	0.99	0.85	0.91
VideoGame	0.82	0.73	0.77	0.91	0.81	0.85
Average	0.61	0.53	0.54	0.78	0.74	0.71

5. RELATED WORK

In this section, we discuss the related work close to ours and highlight the main differences.

5.1 Entity Extraction

Traditional entity extraction is to identify and classify entities within domain-specific texts into predefined categories [5, 11, 12, 18, 22]. Our work is more related to entity discovery from massive web and user generated data. The differences among these studies lie in using different types of data resources and requiring different degrees of supervision.

5.1.1 Entity Extraction with Supervision

Extensive work extracts entities from different data resources with (weak) supervision.

Extraction from Documents. The mainstream work is to extract entities from web documents [8, 15]. For example, Etzioni et al. present a system that creates keyword queries for a given class and extracts entities from search result pages using predefined extracting rules. Although the system doesn't have to train a classifier using training data, it still needs domain knowledge as system input.

Extraction from Structured Data. Our work is also related to approaches that extract records from structured web data [1, 13, 17, 19, 24]. In [30], a semi-supervised method is proposed for combining information from both free texts and web tables. These methods identify tables for extraction based on explicit HTML tags or visually structured features. Instead, we exploit hidden collective textual structures. Besides, the outputs of record extraction usually are tuples containing various types of data, whereas our method focuses on extracting entities.

Extraction from Search Queries. Another body of research focuses on extracting entities from user search queries. For example, in [26], a weakly supervised method is proposed. A set of seeds are provided for an examined class. First, context patterns are learned from queries of a search engine based on the seeds. The class is represented with these patterns. Then these patterns are used to extract more entities from queries. Each entity is represented with a set of patterns which extract it. Finally, the entities are ranked according to the distance between the entity representation and the class representation.

5.1.2 *Open Entity Extraction*

Open Information Extraction (OIE) attempts to extract entities or relations without human intervention [2, 16].

Extraction from Documents. Banko et al. propose an open domain information extraction paradigm for entity extraction [2]. They first train a classier on a small amount of samples to classify a candidate as trustful or not, and then make a data driven pass over the corpus to extract candidate entities. Downey et al. also present a method to locate complex entities in texts [14]. They make use of capitalization information and measuring collocation to extract multi-word expressions as entities. However, the strategies for generating candidates are based on either sophisticated NLP techniques such as dependency parsing and noun phrase chunking, or language dependent heuristics.

Extraction from Query Logs. Parameswaran et al. propose a concept extraction algorithm based on statistical measures of support and confidence on k-grams [25]. But the complexity of this algorithm is high when k is large. So it could only extract concepts with limited length. Jain et al. present a method for open entity extraction from queries [21]. Like previous open entity extraction work on texts, this method also makes use of language dependent capitalization information to identify candidate entities. According to their experimental results, query log based methods outperform document based methods.

Our method follows the open entity extraction paradigm. It doesn't need any human labor or deep NLP techiniques. Unlike previous open entity extraction work, we focus on the most important field of a document, the title, instead of using document contents. Specifically, we identify and utilize the hidden structures in webpage titles and process them collectively. In this way, we significantly simplify the task of determining the boundaries of entities. In addition, we also discover the connections between entities based on the hidden structures, which provide new resources for potential applications.

5.2 Multiple Sequence Alignment (MSA)

Multiple sequence alignment technique is commonly used in computational biology for determining the commonalities within a collection of biological sequences, generally protein, DNA, or RNA [9, 23]. MSA is also applied to natural language processing for constructing concept mapping dictionary [3], identifying sentence level paraphrases [4] and modeling the organization of student essays [27]. Motivated by these work, we adapt MSA for one key component in our framework to infer the hidden structures within parallel webpage titles. In this way, textural templates can be learned automatically and unsupervisedly for extracting entities from large scale web data.

Some previous studies share similar ideas to ours. In [29], entities are mined from comparable news articles. We can view it as an alignment between different sources. But their approach is different from ours, and difficult to identify complex multi-word entities. In [1], record tuples are extracted from structured data by aligning HTML labels. Related work on template induction for IE (e.g., [31]) could be seen as a kind of alignment as well. In [10], clustering is adopted to learn templates automatically for event extraction.

Our method differs from previous work in the following aspects: 1) In previous work, the data to be aligned is provided in advance or retrieved based on seeds, and search engines are often used for searching for related data, while we identify data to be aligned automatically in web scale. 2) Instead of using explicit structured data, we deal with plain texts with collective hidden structures in webpage titles.

6. CONCLUSIONS

In this paper we put forward a novel method to extract entities by exploiting the hidden structures within webpage titles and applying the multiple sequence alignment technique for automatic template generation and entity extraction. Experimental results show that our method could extract large scale open domain entities with high precision. The extracted entities cover a wide range of topics, a large ratio of which is long-tailed and complex. Through the whole process, manual labor is unnecessary.

In addition, the generated title templates imply relatedness connections between entities. The extracted entity-template pairs can be used as a data resource, which can produce more precise domain specific entities by incorporating weak supervision.

Acknowledgments

This research is supported by the National Basic Research Program of China (973 Program, Grant No.2014CB340505), the National Natural Science Foundation of China (Grant No.61402304, Grant No.61303105) and the Beijing Municipal Natural Science Foundation (Grant No.4154065).

7. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD 2003*, pages 337–348. ACM, 2003.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[3] R. Barzilay and L. Lee. Bootstrapping lexical choice via multiple-sequence alignment. In *EMNLP 2002*, pages 164–171. Association for Computational Linguistics, 2002.

[4] R. Barzilay and L. Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *NAACL 2003 - Volume 1*, pages 16–23. Association for Computational Linguistics, 2003.

[5] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics, 1997.

[6] L. Blanco, N. Dalvi, and A. Machanavajjhala. Highly efficient algorithms for structural clustering of large websites. In *WWW 2011*, pages 437–446. ACM, 2011.

[7] J. Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.

[8] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM 2010*, pages 101–110. ACM, 2010.

[9] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.

[10] N. Chambers and D. Jurafsky. Template-based information extraction without the templates. In *ACL 2011:Volume 1*, pages 976–986. Association for Computational Linguistics, 2011.

[11] H. L. Chieu and H. T. Ng. Named entity recognition: a maximum entropy approach using global information. In *COLING 2002:Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[12] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110. Citeseer, 1999.

[13] B. B. Dalvi, W. W. Cohen, and J. Callan. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *WSDM 2012*, pages 243–252. ACM, 2012.

[14] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *IJCAI*, volume 7, pages 2733–2739, 2007.

[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] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

[17] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *WWW 2007*, pages 71–80. ACM, 2007.

[18] R. Grishman and B. Sundheim. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471, 1996.

[19] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2(1):289–300, 2009.

[20] T. Haveliwala, S. Kamvar, and G. Jeh. An analytical comparison of approaches to personalizing pagerank. 2003.

[21] A. Jain and M. Pennacchiotti. Open entity extraction from web search query logs. In *COLING 2010*, pages 510–518, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[22] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[23] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[24] Z. Nie, F. Wu, J.-R. Wen, and W.-Y. Ma. Extracting objects from the web. In *ICDE 2006.*, pages 123–123. IEEE, 2006.

[25] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2):566–577, 2010.

[26] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *CIKM 2007*, pages 683–690. ACM, 2007.

[27] I. Persing, A. Davis, and V. Ng. Modeling organization in student essays. In *EMNLP 2010*, pages 229–239. Association for Computational Linguistics, 2010.

[28] S. Sekine, K. Sudo, and C. Nobata. Extended named entity hierarchy. In *LREC*, 2002.

[29] Y. Shinyama and S. Sekine. Named entity discovery using comparable news articles. In *COLING 2004*, page 848. Association for Computational Linguistics, 2004.

[30] P. P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP 2008*, pages 582–590. Association for Computational Linguistics, 2008.

[31] R. C. Wang and W. W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *EMNLP 2009: Volume 3*, pages 1503–1512. Association for Computational Linguistics, 2009.

[32] C. Zhang, S. Zhao, and H. Wang. Bootstrapping large-scale named entities using url-text hybrid patterns. In *IJCNLP 2013*, pages 293–301, 2013.