Knowledge-Based Systems 115 (2017) 80–86

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Towards a language-independent solution: Knowledge base completion by searching the Web and deriving language pattern

Lidong Bing[a,d,*], Zhiming Zhang[b], Wai Lam[c], William W. Cohen[d]

[a] AI Platform Department, Tencent Inc., Shenzhen, China
[b] Web Data Mining Department, Baidu Inc., Shenzhen, China
[c] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, China
[d] Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA

ARTICLE INFO

Article history:
Received 10 January 2016
Revised 29 September 2016
Accepted 4 October 2016
Available online 20 October 2016

Keywords:
Knowledge base completion
Language pattern
Language-independent solution

ABSTRACT

Knowledge bases (KBs) such as Freebase and Yago are rather incomplete, and the situation is more serious in non-English KBs, such as Chinese KBs. In this paper, we present a language-independent framework to tackle the slot-filling task by searching the Web with high-precision queries, and deriving lightweight extraction patterns. The patterns are based on string matching, and since they make no use of complex NLP resources, which may be unavailable in some languages, they are very language-independent.

We use a traditional bootstrapping approach for extraction, but also use a novel approach to suppress the noise associated with distant supervision: in particular, we use a pseudo-testing method to validate the patterns derived from different sentences. Experiments show that our framework achieves very encouraging results.

1. Introduction

Knowledge bases (KBs) are widely used in the tasks of information retrieval, natural language processing, etc. Some KBs, such as Yago [18], DBPedia [24], and Freebase [1], are constructed largely by processing a small number of sources (e.g., as Wikipedia), while other projects such as NELL [9], Open IE [12], DeepDive [26] and Knowledge Vault [10] aim at automatically extracting knowledge from the general Web and/or other sources. In automatic KB construction (AKBC), one major subtask is to identify individual entities and organize them into existing or new semantic categories, either by using predefined patterns [16], or, in later work, by using techniques like automatic wrapper induction or employing conditional random fields (CRFs) [4,38]. Another major subtask, called slot-filling [22], is to extract knowledge of entities, including attribute values and their relations to other entities (e.g. country of birth and spouse relationships). In this paper, we focus on language-independent approaches to the slot-filling task.

Slot-filling is important because even large KBs are usually incomplete: e.g., a recent work showed that more than 57% of the nine most commonly used attributes and relations are missing for the top 100K most frequent PERSON entities in Freebase [41]. This situation is even more serious in non-English KBs, such as Chinese KBs, and existing slot-filling methods used for English KBs cannot always be adapted to different languages, due to lack of language-specific resources and tools. Here we present a language-independent framework to tackle the slot-filling task by searching the Web and deriving language-specific extraction patterns. Our method is a precision-oriented strategy, as high-precision KBs are often the focus of AKBC, at least in earlier stages of KB construction.

In order to achieve language independence, we focus on "lightweight" approaches, and in particular, we find "shallow" surface patterns for relations using distant training data, and these shallow patterns are employed to extract more slots of other entities. For instance, for the *hasWife* relation, we find patterns such as "**B** S 's wife O" and "**B** S and O's oldest child", where S and O are placeholders for subject and object, and **B** marks the left boundary of a text fragment. We note that even in English, surface patterns are frequently useful, e.g., in Q&A systems [31]. Our language patterns differ from the patterns used in Snowball [2], LEILA [35], and [7], where their patterns include the output of NER or dependency parsing. Our patterns are also different from the patterns in

* Corresponding author at: AI Platform Department, Tencent Inc., Shenzhen, China.
E-mail addresses: lyndonbing@tencent.com, binglidong@gmail.com (L. Bing), zhangzhiming@baidu.com (Z. Zhang), wlam@se.cuhk.edu.hk (W. Lam), wcohen@cs.cmu.edu (W.W. Cohen).

[1] https://www.freebase.com/.

http://dx.doi.org/10.1016/j.knosys.2016.10.014
0950-7051/

SEAL [38,39] and DIPRE [6], which use patterns of character-level, and the characters are from both HTML code and visible text of Web pages. Our language patterns are based on string-matching, and are specifically designed to be applicable to free text in any natural language, and to not suffer from a lack of availability of NLP tools and resources for non-English languages.

We also propose a novel approach to solve the inborn weakness of distant supervision (DS) because of noisy data. DS can produce a large amount of training data, but the data can be quite noisy [3,5]. One potential problem is that a sentence that has been distantly labeled as an instance of a relation R may instead express a relation R', or no relation at all. We propose a pseudo-testing method to validate the patterns from different sentences, in which a held-out portion of training data is used for validation purpose, and all patterns are jointly evaluated with a machine learning approach. Thus, our framework is more robust on noisy data. Our work thus differs from most existing machine learning based methods, such as those based on CRFs [4] and factor graphs {CITENiuBlhypdeepdive:web-scale,hoffmann2011knowledge, which do not make patterns. The work also differs from prior template-based methods [2,35,38], which do not valid all templates jointly with a machine learning approach.

In outline, our framework is based on searching a large Web page repository by formulating queries on-the-fly to fetch targeted facts. The traditional way of collecting facts is to process a large corpus to perform named-entity disambiguation followed by relation extraction, as is done in NELL [9] and ReVerb [13]. We call this model a "trawling" model. In contrast, we adopt a "harpoon" model, whereby we "harpoon" the facts for relations by searching the Web with high-precision queries. This focuses attention on the retrieved information sources, which are usually high-quality and up-to-date—particularly the snippets of search results. This approach has been used occasionally in the past—e.g., Krause et al. also adopted a harpoon-like approach and employed a search engine to collect a large set of Web pages for extracting instances of a particular relation [23]. Here, however, we exploit the snippets of search results, but not whole Web pages, and our search queries are made more specific by using relation-specific keywords. Also, some prior works rely on Q&A systems to identify instances of targeted relations [41]. However, such Q&A systems are not available for most non-English languages, and in any case Q&A systems are not particularly designed for KB construction purpose.

In summary, the contributions of our work are as follows: (1) Our framework generates "shallow" surface patterns with distant training data, so that they are applicable to free text in any natural language. (2) We propose a novel approach to solve the inborn weakness of distant supervision because of noisy data, which is a pseudo-testing method to validate the patterns from different sentences jointly. (3) Our framework searches a large Web page repository by formulating queries on-the-fly to fetch targeted facts from search result snippets and the queries are tailor-made with relation-specific keywords.

In the reminder of this paper, the details of our framework are first presented in Section 2. In Section 3, its performance in completing a non-English KB is examined. After more related works are reviewed in Section 4, we conclude the paper in Section 5.

2. Our framework

2.1. Overview

Let $\mathbf{P}$ denote a set of relation predicates and p be a single predicate. Let $\mathbf{N}_j = \{s\}$ denote a set of entities for each of which the objects o's satisfying the triple (s, p_j, o) are known. For example, suppose $p_j = hasWife$, we have ("Barack Obama", *hasWife*, "Michelle Obama"). Let $\mathbf{U}_j = \{s\}$ denote a set of subjects for which the objects satisfying p_j are unknown. Given $\{\mathbf{N}_j\}$, the known facts of the predicates in $\mathbf{P}$, as the training data, the goal of our system is to output the objects o''s satisfying (s, p_j, o') for a subject in a testing set of $\{\mathbf{U}_j\}$. We use "subject" and "entity", or "object" and "entity", interchangeably, if the context is clear.

We developed a framework as depicted in Fig. 1 which has two stages, namely, training stage in the upper part, and KB completion stage in the lower part. In the training stage, our framework employs $\{\mathbf{N}_j\}$ and Baidu search engine [2] to find shallow patterns that describe each predicate p_j. The training procedure is illustrated with an example of *hasWife* predicate in the callouts. It has five steps: Given some training subjects and their object values of the targeted predicate, T1 generates search queries; T2 issues those queries to Baidu search engine, and collects a set of text fragments from the result snippets; T3 extracts a set of keywords from those fragments according to term frequency, which will be used in the second stage; T4 generates extraction patterns for each predicate which are used to extract the missing object values for testing subjects; These patterns are weighted with a pseudo-testing method in T5.

After the patterns are generated and weighted, they are utilized to extract slot values for other subjects in the KB completion stage, which also has five steps: C1 and C2 are similar to T1 and T2, except the only difference that C1 uses the subject name and the keywords to form search queries; C3 matches the patterns generated above with the collected text fragments to generate candidate answers; These candidates will be scored in C4 according to the pattern weights; Finally, C5 links the answers to existing entities in KB or mint new entities. The details are described step by step in the following sections. The example is described in English, and the parts that need to be tailor-made for other languages are discussed where necessary.

2.2. Training stage

Suppose the currently targeted predicate p_j is *hasWife*. $\mathbf{N}_j$ is split into two parts: $\mathbf{N}_j^1$ which is used to generate the extraction patterns, and $\mathbf{N}_j^2$ which is used to validate the patterns.

T1: Query Generation. In the first step of training (T1), the queries are generated with subject-object pairs of *hasWife*. For example, with the training set {Andy LAU, Leehom WANG, ...}, the queries such as "Andy LAU Carol CHU" and "Leehom WANG Jinglei Lee" are generated by concatenating the wife name, i.e, "Carol CHU" or "Jinglei Lee", to the subject name.

T2: Text Fragment Collection. In the second step (T2), each generated query is issued to Baidu search engine to collect a set of snippets such as "Andy Lau's wife Carol Chu has been sighted sporting..." and "Leehom Wang and wife Lee JingLei will be expecting...".

After the snippets are collected, we segment each snippet into text fragments with a sentence detector. The text fragments that do not contain both the subject string and the object string are filtered out, as required in MultiR [19] and MIML [37], in order to achieve high precision.

T3: Keyword Collection. In the third step (T3), the terms that are frequent, say top K, in text fragments are collected as keywords of this predicate. For *hasWife*, such keywords include "wife", "marry", etc. For some languages such as Chinese, Japanese, and Korean, we need to segment a text fragment into individual words since word boundaries are not marked by space. Stopwords are removed with a tailor-made stopword list so that the words, such as "and (和 in Chinese)" and "'s (的 in Chinese)", that are useful for

[2] http://www.baidu.com.

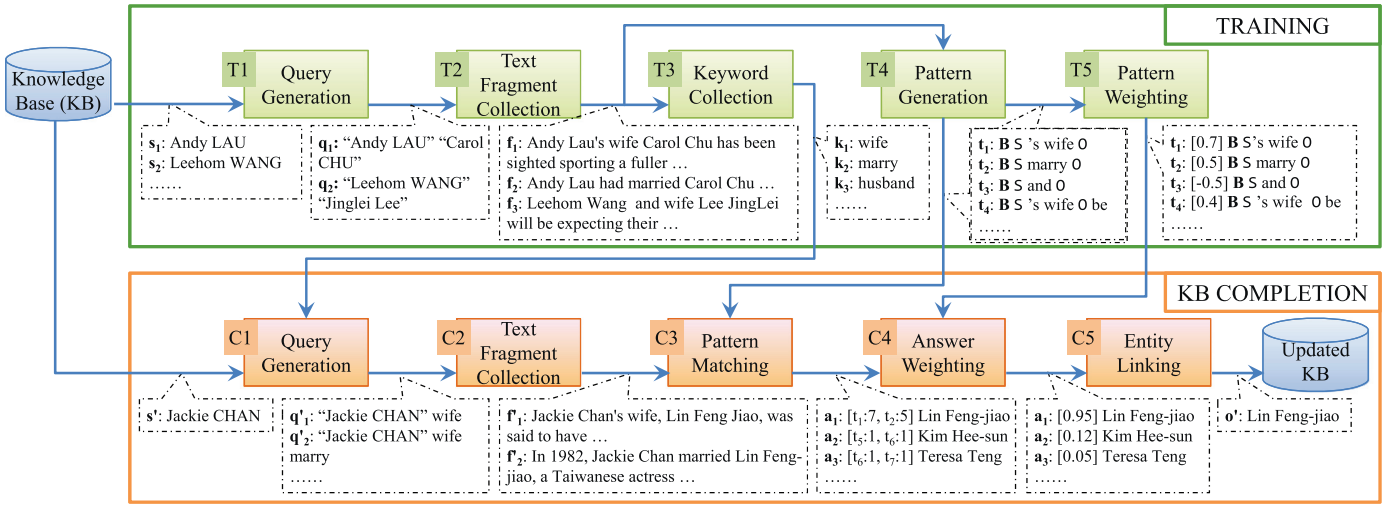

Fig. 1. Architecture of the framework. The work procedure is illustrated with *hasWife* as an example relation in callouts. In the training stage, a set of entities with their wife values are given, such as "Andy Lau" (his wife is Carol Chu) and "Leehom Wang" (his wife is Jinglei Lee). In the KB completion stage, the wife values are extracted for other entities.

our task are kept. **T4: Pattern Generation**. In the fourth step (T4), we generate a set of patterns from the text fragments collected in T2. The subject and object strings are replaced with placeholders S and O respectively. Each text fragment is employed to generate a pattern in the form of "[**B**] $w[0 \sim k]$ S $w[1 \sim k]$ O $w[0 \sim k]$ [**E**]" or "[**B**] $w[0 \sim k]$ O $w[1 \sim k]$ S $w[0 \sim k]$ [**E**]", where **B** and **E** are symbols for the beginning and the end of a fragment. w is called pattern word and $[0 \sim k]$ means the legal values of w. We adopt an overcomplete strategy in pattern generation by enumerating the sequences of pattern words with a window size k on each end. From "Andy Lau's wife Carol Chu has been sighted sporting...", we generate patterns such as "**B** S 's wife O", "**B** S 's wife O be", "**B** S 's wife O be sight", "**B** S 's wife O be sight sport", suppose $k = 3$. One pattern could be generated from multiple text fragments. In very minor cases, one fragment contains multiple appearances of S or O, if so, one pattern is generated from each adjacent pair of (S, O) or (O, S). Let $\mathbf{T}_j$ denote the pattern set of p_j.

T5: Pattern Weighting. One essential step in training is to validate the generated patterns. The details of this step (T5) are discussed in Section 2.4 after the KB completion stage since we develop an elegant validation method which employs some steps in the completion.

2.3. KB completion stage

C1: Query Generation. In the first step of completion (C1), we generate queries for a testing subject in $\mathbf{U}_j$. Each query has three terms, namely, the subject name, the relation name, and a keyword from T3. For the subject "Jackie CHAN", the queries are "Jackie CHAN wife", "Jackie CHAN wife marry", etc. Note that the relation name is always used in the queries, and we have a synonym set of names for a relation, such as {"father", "dad", ...} for *hasFather*.

C2: Text Fragment Collection. The second step (C2), similar to T2, uses the queries from C1 to collect text fragments. We filter out the fragments that do not contain the testing subject name.

C3: Pattern Matching. In the third step (C3), after the subject name in a text fragment is replaced with the placeholder S, the fragment is matched against the patterns from T4 to extract candidate answers. To speed up the matching, an inverted index is built for all patterns of a predicate with the pattern words as the vocabulary. A fragment is only matched against the patterns that are retrieved with the terms in the fragment. The matching operation of multiple patterns for one fragment follows the descending order of their length. Once a pattern is successfully matched, the matching is terminated. After matching, we obtain a set of candidate answers, and each of them is associated with the information of matched patterns and matching frequency such as "[t_1:7, t_2:5] Lin Feng-jiao". Our matching algorithm first anchors a pattern onto the fragment by aligning the placeholder S in them. Then, if the pattern words between S and O are sequentially and successfully matched with the corresponding words in the fragment, the pattern words on each end are examined. If all pattern words, including **B** and **E**, are successfully matched, the pattern is successfully matched with the fragment. The term in the fragment that is aligned with the placeholder O is extracted as a candidate answer.

C4: Answer Weighting. The fourth step (C4) is discussed in Section 2.4 after discussing pattern weighting.

C5: Entity Linking. In the fifth step (C5), we employ existing techniques for linking the answers to existing entities [15] or detecting new entities [17]. It is not the main focus of this paper, therefore no further details are discussed.

2.4. Weighting strategy

Pattern Weighting (T5). We propose a novel pattern weight learning method with a pseudo-testing strategy. After the patterns are generated with $\mathbf{N}_j^1$, $\mathbf{N}_j^2$ is applied to learn the weight. To do so, each subject in $\mathbf{N}_j^2$ is regarded as a pseudo-testing subject and employed to generate one positive weighting vector, standing for p_j, and a few negative weighting vectors, standing for $p' \in \mathbf{P} \backslash \{p_j\}$. Initially, each weighting vector has a dimensionality of $|\mathbf{T}_j|$, where each dimension corresponds to one pattern. Specifically, we use a subject s in $\mathbf{N}_j^2$ to go through C1, C2, and C3. While matching the text fragments for s in C3, if a pattern extracts the correct answer for $(s, p_j, \_)$ from a fragment, its dimension in the positive vector is added with 1. If an answer satisfying $(s, p' \in \mathbf{P} \setminus \{p_j\}, \_)$ is extracted, the dimension of this pattern in the negative vector corresponding to p' is added with 1. We follow the local closed world assumption [10] to generate an additional negative vector to capture those general incorrect answers that do not satisfy any $p' \in \mathbf{P} \backslash \{p_j\}$. Thus, after scanning all fragments for s, a positive vector and a few negative vectors are generated. The number of negative vectors from s varies, since the answers satisfying p' for s may not appear. After the weighting vectors from all subjects in $\mathbf{N}_j^2$ are generated, we compress the vectors by removing the dimensions that

Table 1
The details of our data set.

	Fa	Mo	Hu	Wi	Da	So	ES	EB	YS	YB
$\mathbf{N}^1$	1000	1000	1000	1000	1000	1000	267	372	387	327
$\mathbf{N}^2$	3700	3725	1460	1935	995	3215	268	373	388	328
U	940	945	492	587	399	843	107	149	155	131

have 0 value in all vectors. Accordingly, the corresponding patterns are removed. Then, we employ Logistic Regression to this binary classification problem with value 1 being positive label and 0 being negative label. Thus, a weighting model for the patterns of p_j is learnt.

Our weighting method has two advantages. First, when weighting the patterns of one predicate, the training data of other predicates is employed for providing clues to distinguish predicate-specific patterns such as "**B** S 's wife O" and general patterns such as "**B** S and O expect" as well as noisy patterns. As a result, it is very effective for simultaneously extracting multiple relations for persons since false positive answers are frequently due to related relations. Second, all patterns of one predicate are simultaneously weighted so that the correlations among them are explored.

Answer Weighting (C4). To weight the candidate answers extracted in C3, the matching information associated with each candidate is employed to generate a feature vector in the same approach as in T5. With this vector and the weighting model from T5, we can predict a value in [0, 1] for each candidate indicating its probability to be correct.

3. Experiments

We evaluate the performance of our framework in completing Chinese KB.

3.1. Experimental setting

Data Set. Our data set contains 10 predicates of family relation, namely, "hasFather (父亲)" ("Fa" for short), "hasMother (母亲)" (Mo), "hasHusband (丈夫)" (Hu), "hasWife (妻子)" (Wi), "hasDaughter (女儿)" (Da), "hasSon (儿子)" (So), "hasElderSister (姐姐)" (ES), "hasElderBrother (哥哥)" (EB), "hasYoungerSister (妹妹)" (YS), and "hasYoungerBrother (弟弟)" (YB). The details are given in Table 1, and we will make the data set publicly available later. One sixth of the data for each predicate is used for testing (i.e., **U**). At most 1000 entities or half of the training data are employed to generate patterns (i.e., $\mathbf{N}^1$), and the remaining are used for pattern weighting (i.e., $\mathbf{N}^2$). Note that in some languages, such as Chinese and Arabic, speakers usually distinguish "elder sister" and "younger sister". Therefore, one-time effort is needed to tailor-make the predicates when applying our framework to such languages. In the same way, it can be applied to other predicates, such as hasNationality and hasBirthPlace.

Evaluation Metrics. We evaluate the output of C4, instead of C5. The reason is that entity linking is not our focus, and in addition, our Chinese KB is rather incomplete and our annotation of ground truth results for the testing subjects is in the form of strings. Thus, it is more reliable to examine the performance by evaluating the output of C4. We evaluate the results with the metrics of MRR and MAP as used in TAC-KBP (http://www.nist.gov/tac/2016/KBP/). We also report P@1 to examine whether our precision-oriented strategy results in higher P@1.

Comparison Systems. Our comparison system, called *CRF-Sys*, is implemented based on CRFs. It employs the training set in Table 1, i.e., $\mathbf{N}^1$ + $\mathbf{N}^2$, to generate the training sequences with steps T1 and T2, and employs the testing set to generate the testing sequences with steps C1 and C2. After each fragment is processed with POS tagger, NER, and dependency parser, they are fed into CRF++ (https://code.google.com/p/crfpp/) to predict three labels, namely, "S_" labeling subjects, "O_" labeling objects, "N_" labeling others, for each term in a text fragment. Similar to our system, the text fragments that do not contain both subject and object are removed from the training set of CRF-Sys. But for testing, CRF-Sys does not require that a fragment contains the subject since its target is to predict "O_". Each predicted label is associated with a probability. To aggregate the appearances of the same candidate answer in different fragments, CRF-Sys adopts a normalized weighted frequency method. Specifically, the weighted frequency of an answer is calculated as the summation of probabilities of the appearances. Then, it is normalized with the summed weighted frequency of all candidates and the result is employed as the weight of this candidate. For our framework, candidate answers are also aggregated in the same way. We also compare against a latent factor model, *MultiR* [19], with distance supervision for relation extraction, which models each relation mention separately and aggregates their labels using a deterministic OR. We used the publicly available code from the authors [3] for the experiments on our dataset. Its only parameter, namely the number of training iterations, is tuned with our validation set $\mathbf{N}^2$ and set to be 30. The same NLP tools of CRF-Sys are used for MultiR to generate features.

3.2. Experimental results

The results are given in Table 2. Under MRR and MAP metrics, our framework can outperform CRF-Sys and MultiR in 7 predicates. On average, 13% and 18% improvements are achieved respectively. This demonstrates that our framework is very effective although it does not apply those complex parsers. One reason is that our framework exploits the language patterns, which is more effective than modeling individual terms as did in CRF-Sys or MultiR. Moreover, we apply a learning method to jointly weight the patterns and the robustness is further enhanced. Under P@1 metric, our framework can outperform CRF-Sys and MultiR for all predicates. This demonstrates that our precision-oriented strategy achieves promising results. This can save significant manpower for KB construction in the beginning stage, because the initial accuracy of KB is extremely important for achieving better quality in later iterations.

The performance for "*hasMother*" is much lower than the performance for others. The first reason is that we find for about 26% of the testing cases, the correct answers are not covered in the snippets. The second reason is that the filtering step in C2 causes a loss of 40% of correct answers. Although CRF-Sys and MultiR are not affected by the second reason, their results on "*hasMother*" are still lower than our framework.

[3] http://aiweb.cs.washington.edu/ai/raphaelh/mr/.

Table 2
Results under MRR, MAP, and P@1.

		Fa	Mo	Hu	Wi	Da	So	ES	EB	YS	YB	Average
MRR	Our	**0.466**	**0.218**	**0.616**	**0.514**	0.343	**0.349**	0.347	**0.306**	**0.367**	**0.328**	**0.386**
	CRF-Sys	0.350	0.159	0.477	0.447	0.379	0.334	**0.416**	0.268	0.323	0.255	0.341
	MultiR	0.365	0.167	0.413	0.420	**0.385**	0.346	0.403	0.271	0.308	0.263	0.334
MAP	Our	**0.466**	**0.218**	**0.598**	**0.476**	0.299	0.286	0.333	**0.286**	**0.344**	**0.298**	**0.361**
	CRF-Sys	0.350	0.159	0.471	0.413	0.310	0.277	**0.379**	0.239	0.302	0.231	0.313
	MultiR	0.365	0.167	0.403	0.388	**0.319**	**0.287**	0.367	0.240	0.289	0.243	0.307
P@1	Our	**0.759**	**0.529**	**0.799**	**0.755**	**0.672**	**0.622**	**0.663**	**0.758**	**0.626**	**0.901**	**0.708**
	CRF-Sys	0.490	0.293	0.659	0.613	0.481	0.466	0.402	0.376	0.406	0.496	0.468
	MultiR	0.522	0.305	0.661	0.637	0.521	0.485	0.449	0.409	0.465	0.535	0.499

Table 3
Different parameter settings.

	Snippet #	Window size k	Keyword #	Whether to use relation name
(a)	{10, 20, 30, 40, 50}	3	10	Yes
(b)	30	{1, 2, 3, 4, 5}	10	Yes
(c)	30	3	{0, 5, 10, 15, 20}	Yes
(d)	30	3	10	{Yes, No}

Table 4
Effect of different parameters.

(a)	Snippet #	10	20	30	40	50
	AVG MRR	0.364	0.383	0.382	0.381	0.375
(b)	Window size k	1	2	3	4	5
	AVG MRR	0.346	0.374	0.382	0.382	0.383
(c)	Relation keyword #	0	5	10	15	20
	AVG MRR	0.288	0.358	0.382	0.382	0.384
(d)	Whether to use relation name	Yes			No	
	AVG MRR	0.382			0.365	

One point we would emphasize is that our framework does not use NLP parsers such as POS tagging, NER, and dependency parsing, that are used by CRF-Sys, MultiR, and other well-known systems, because we are committed to make our framework easily adaptable to more languages that might be of poverty in such tools. In fact, if word segmentation tool is not available, our framework can still work by regarding each Chinese character as a word.

3.3. Parameter setting

Key Parameters. In our framework, four parameters might have effect on the performance, namely, snippet #, window size k, relation keyword #, and whether to use relation name. To examine the effect of a particular parameter, we empirically set other parameters as given in Table 3. For example, we test snippet # of 10, 20, 30, 40, and 50 with the other parameters set as those values in the first row. Although exhaustive grid search is the perfect approach, our strategy is still workable because the parameters are perpendicular to some extent. The results of those settings in Table 3 are given in Table 4. For snippet #, 20, 30, and 40 perform very similar. 10 and 50 are less effective, because of missing some answers or involving more noisy data. A larger window size can generate a superset of patterns of a smaller window size, theocratically its performance should be better. However, we find that window size 3, 4, and 5 achieve almost the same result. It is because there are seldom cases that are matched with a pattern with 4 or 5 terms between S and O. The larger number of keywords is used, the better performance is achieved. It is because more comprehensive snippets are collected. Adding the relation name in C1 always improves the results. We also attempted to add some augmentation (or profile) terms as did in [41], but no improvement was obtained.

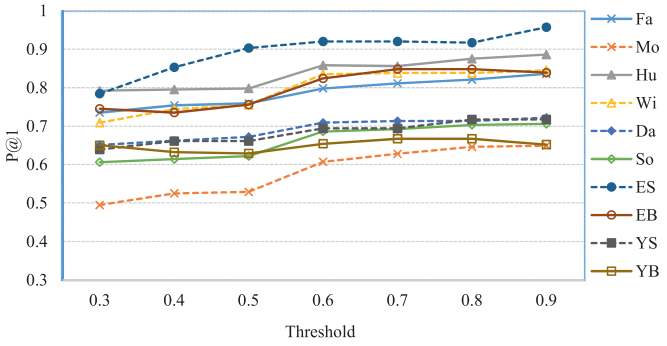

Fig. 2. P@1 with different thresholds.

Threshold. The P@1 results of our framework in Table 2 are calculated on the candidate answers having a confidence no less than 0.5, because 0.5 indicates likely to be correct in Logistic Regression. The effect of the threshold parameter is given in Fig. 2. In general, the larger the threshold is, the better P@1 result is. We can see that 0.6 outperforms 0.5 with a relatively larger gap, which might be a better threshold for practical use. After 0.6, increasing the threshold does not improve the performance much.

Pattern Pruning. The first row of Table 5 gives the number of patterns generated in T4, which tends to be too many, from 12 K to 162 K. The reason is that in the text fragments of a particular training entity, there are many entity-specific terms or infrequently used terms. For example, "Andy Lau together with his wife Carol Chu and daughter Hanna Lau was photographed by paparazzi.." produces a pattern "**B** S together with his wife O and daughter Hanna Lau", which is not useful for other entities because of the

Table 5
Pattern # before and after pruning.

	Fa	Mo	Hu	Wi	Da	So	ES	EB	YS	YB
Before	58,077	25,357	146,523	162,256	58,043	80,922	16,543	15,132	13,533	12,065
After	770	385	1229	1293	584	402	286	324	239	263

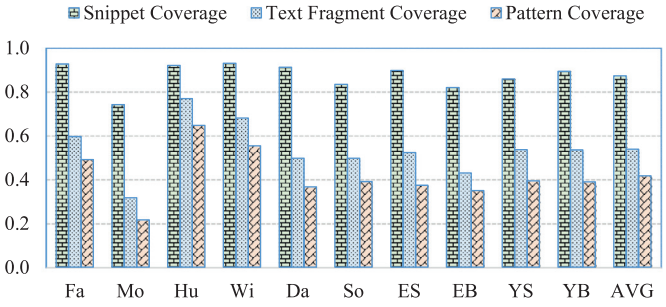

Fig. 3. Coverage of true answers in key steps.

term "Hanna Lau". After the pattern weighting step T5, the number of patterns is significantly pruned to 1K or a few hundreds, as given in the second row of Table 5. Those patterns that cannot match with fragments of any entity in $\mathbf{N}_j^2$ are removed. In KB completion, we only match the text fragments with the retained patterns.

3.4. Step by step loss analysis

To have a clearer understanding about what affects much on the performance, we have a closer look at each step. One factor that affects the results is that the ground truth answers are not included in the search snippets retrieved in C2. We calculate "***snippet coverage***" as the percentage of the testing cases whose true answers are covered in the snippets. For some testing cases, even the snippets contain the true answers, it is still possible that no fragments are retained due to the filtering requirement in C2. Similarly, we calculated a "***text fragment coverage***" for this. Another step is C3 that may cause loss of true answers because the patterns may not be able to successfully extract the answer from text fragments, although the ground truth answer is covered by the fragments. Similarly, we calculated a "***pattern coverage***". The calculated coverage results are given in Fig. 3. On average, snippet retrieval and pattern matching cause 12% and 13% losses, respectively. The largest loss, about 33% of true answers, is caused by the fragment filtering in C2 since they do not contain the corresponding subject names. Therefore, one future direction to enhance our framework is to relax the filtering constraint in C2 and improve the matching capability in C3 by allowing some ambiguity, and meanwhile making sure that the precision is not affected much. We would like to mention that CRF-Sys and MultiR does not have fragment filtering as we do in C2, and our framework still performs better than them.

4. Related work

The KB completion task has grown in popularity, and it is introduced as an annual competition in Text Analysis Conference (TAC) [4] and workshops [1]. Good summaries of the standard approaches to this task are given by Ji and Grishman [21] and Weikum et al. [34,40]. KB completion is different from building KB from scratch in which an initial ontology are manually defined with a bunch of initial classes and relations [9]. One major task in KB completion is to gather individual entities and categorize them into existing classes or generate new classes with them. Another major task is to extract attribute values or relation facts, called slot-filling, for the entities in KB.

Entity set expansion takes a few seed entities of a particular class as input, and aims at collecting more entities of the same class. SEAL [38] extracts named entities with wrappers [39], each of which is a pair of letter-level prefix and suffix, to expand a given entity set. [14] takes several user input data records, including entity names and attribute values, as seeds to discover more entities as well as values of specified attributes by CRFs-based extractor [33]. Some other works focus on a more general problem setting for the task of entity set expansion [27,28,30]. They first obtain a set of candidate entities with some methods. Then the similarity of a candidate with the seeds is calculated using their context distributions on the Web or Web queries. Entity set acquisition systems [8,11] do not have input seeds and they leverage patterns, such as "is a" and "such as", to collect the instances of a given class.

For the task of slot-filling, some works [20,36,42,43] train extractors on the free text of Wikipedia articles that are automatically annotated with the corresponding articles' infoboxes. This approach suppresses the noise due to distant labeling to some extent, since the labeled text is tightly related to the labeling information, i.e., infoboxes. To overcome the noise in general distant labeling, [32] introduced a "at least one" heuristic, where instead of taking all mentions for a pair as correct examples, only at least one of them is assumed to express that relation. MultiR [19] and Multi-Instance Multi-Label Learning (MIML) [37] extend this approach to support multiple relations expressed by different sentences in a bag. Some other approaches propagate the distant labels in an appropriate graph to improve the quality of the distant training data [3,5] To grow the skeleton of a KB, researchers have investigated new relations discovery and new attribute acquisition. Discovering relations between noun categories considers every pair of categories in an ontology to search for evidence of a frequently discussed relation between members of the category pair [25]. The methods of weakly-supervised attribute acquisition [27,29] can be applied in identifying important attributes for the categories.

5. Conclusions and future work

In this paper, we present a language-independent framework to tackle the slot-filling task by searching the Web and deriving language patterns. The patterns are based on string matching and they are very language-independent. To solve the inborn weakness of distant supervision because of noisy data, we exploited a pseudo-testing method to validate the patterns from different sentences. Encouraging experimental results in completing Chinese KB show that our framework can outperform to comparison methods, i.e. CRF-Sys and MultiR. For the future work, one direction is to relax the filtering constraint and improve the matching capability to recall more correct answers. Another direction is to conduct more experiments on completing KB in other languages.

[4] http://www.nist.gov/tac/2015/KBP/.

Acknowledgments

This work was substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055034). This work was also funded by a grant from Baidu USA and by the NSF under research grant IIS-1250956.

References

[1] AKBC '13: Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, ACM, New York, NY, USA, 2013.
[2] E. Agichtein, L. Gravano, Snowball: extracting relations from large plain-text collections, in: DL '00, 2000, pp. 85–94.
[3] L. Bing, S. Chaudhari, R.C. Wang, W.W. Cohen, Improving distant supervision for information extraction using label propagation through lists, in: EMNLP, 2015, pp. 524–529.
[4] L. Bing, W. Lam, T.-L. Wong, Wikipedia entity expansion and attribute extraction from the Web using semi-supervised learning, in: WSDM, 2013, pp. 567–576.
[5] L. Bing, M. Ling, R.C. Wang, W.W. Cohen, Distant IE by bootstrapping using lists and document structure, in: AAAI, 2016, pp. 2899–2905.
[6] S. Brin, Extracting patterns and relations from the world wide Web, in: WebDB '98, 1999, pp. 172–183.
[7] R.C. Bunescu, R.J. Mooney, A shortest path dependency kernel for relation extraction, in: HLT '05, 2005, pp. 724–731.
[8] M.J. Cafarella, D. Downey, S. Soderland, O. Etzioni, Knowitnow: fast, scalable information extraction from the Web, in: HLT, 2005, pp. 563–570.
[9] A. Carlson, J. Betteridge, B. Kisiel, B. Settles Jr., E.R. H., T.M. Mitchell, Toward an architecture for never-ending language learning, AAAI, 2010.
[10] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, W. Zhang, Knowledge vault: a Web-scale approach to probabilistic knowledge fusion, in: KDD, 2014, pp. 601–610.
[11] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld, A. Yates, Web-scale information extraction in knowitall (preliminary results), in: WWW, 2004, pp. 100–110.
[12] O. Etzioni, A. Fader, J. Christensen, S. Soderland, M. Mausam, Open information extraction: the second generation, in: IJCAI, 2011, pp. 3–10.
[13] A. Fader, S. Soderland, O. Etzioni, Identifying relations for open information extraction, EMNLP, 2011.
[14] R. Gupta, S. Sarawagi, Answering table augmentation queries from unstructured lists on the Web, Proc. VLDB Endow. 2 (2009) 289–300.
[15] X. Han, L. Sun, J. Zhao, Collective entity linking in Web text: a graph-based method, in: SIGIR, 2011, pp. 765–774.
[16] M.A. Hearst, Automatic acquisition of hyponyms from large text corpora, in: COLING, 1992, pp. 539–545.
[17] J. Hoffart, Y. Altun, G. Weikum, Discovering emerging entities with ambiguous names, in: WWW, 2014, pp. 385–396.
[18] J. Hoffart, F.M. Suchanek, K. Berberich, G. Weikum, Yago2: a spatially and temporally enhanced knowledge base from wikipedia, Artif. Intell. 194 (2013) 28–61.
[19] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, D.S. Weld, Knowledge-based weak supervision for information extraction of overlapping relations, in: ACL '11, 2011, pp. 541–550.
[20] R. Hoffmann, C. Zhang, D.S. Weld, Learning 5000 relational extractors, in: ACL, 2010, pp. 286–295.
[21] H. Ji, R. Grishman, Knowledge base population: successful approaches and challenges, in: HLT '11, 2011, pp. 1148–1158.
[22] H. Ji, J. Nothman, B. Hachey, Overview of tac-kbp2014 entity discovery and linking tasks, TAC, 2014.
[23] S. Krause, H. Li, H. Uszkoreit, F. Xu, Large-scale learning of relation-extraction rules with distant supervision from the Web, in: ISWC'12, 2012, pp. 263–278.
[24] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia, Semant. Web J. 6 (2015) 167–195.
[25] T.P. Mohamed, E.R. Hruschka Jr., T.M. Mitchell, Discovering relations between noun categories, in: EMNLP '11, 2011, pp. 1447–1455.
[26] F. Niu, C. Zhang, C. Ré, J. Shavlik, Deepdive: Web-scale knowledge-base construction using statistical learning and inference, 2012,
[27] M. Paşca, Organizing and searching the world wide Web of facts – step two: harnessing the wisdom of the crowds, in: WWW, 2007, pp. 101–110.
[28] M. Paşca, Weakly-supervised discovery of named entities using Web search queries, in: CIKM, 2007, pp. 683–690.
[29] M. Paşca, B.V. Durme, Weakly-supervised acquisition of open-domain classes and class attributes from Web documents and query logs., in: ACL, 2008, pp. 19–27.
[30] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, V. Vyas, Web-scale distributional similarity and entity set expansion, in: EMNLP, 2009, pp. 938–947.
[31] D. Ravichandran, E. Hovy, Learning surface text patterns for a question answering system, in: ACL, 2002, pp. 41–47.
[32] S. Riedel, L. Yao, A. McCallum, Modeling relations and their mentions without labeled text, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2010, pp. 148–163.
[33] S. Sarawagi, W.W. Cohen, Semi-markov conditional random fields for information extraction, in: NIPS, 2004, pp. 1185–1192.
[34] F. Suchanek, G. Weikum, Knowledge harvesting in the big-data era, in: SIGMOD '13, 2013, pp. 933–938.
[35] F.M. Suchanek, G. Ifrim, G. Weikum, Combining linguistic and statistical analysis to extract relations from Web documents, in: KDD '06, 2006, pp. 712–717.
[36] F.M. Suchanek, M. Sozio, G. Weikum, Sofie: a self-organizing framework for information extraction, in: WWW, 2009, pp. 631–640.
[37] M. Surdeanu, J. Tibshirani, R. Nallapati, C.D. Manning, Multi-instance multi-label learning for relation extraction, in: EMNLP-CoNLL '12, 2012, pp. 455–465.
[38] R.C. Wang, W.W. Cohen, Language-independent set expansion of named entities using the Web, ICDM, 2007.
[39] R.C. Wang, W.W. Cohen, Character-level analysis of semi-structured documents for set expansion, in: EMNLP, 2009, pp. 1503–1512.
[40] G. Weikum, M. Theobald, From information to knowledge: harvesting entities and relationships from Web sources, in: PODS '10, 2010, pp. 65–76.
[41] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, D. Lin, Knowledge base completion via search-based question answering, in: WWW, 2014, pp. 515–526.
[42] F. Wu, D.S. Weld, Autonomously semantifying wikipedia, in: CIKM, 2007, pp. 41–50.
[43] F. Wu, D.S. Weld, Open information extraction using wikipedia, in: ACL, 2010, pp. 118–127.