Mining Name Translations from Entity Graph Mapping*

Gae-won You[†] **Seung-won Hwang**[†] **Young-In Song**[‡] **Long Jiang**[‡] **Zaiqing Nie**[‡]

[†]Pohang University of Science and Technology, Pohang, Republic of Korea
{gwyou,swhwang}@postech.ac.kr

[‡]Microsoft Research Asia, Beijing, China
{yosong,longj,znie}@microsoft.com

Abstract

This paper studies the problem of mining entity translation, specifically, mining English and Chinese name pairs. Existing efforts can be categorized into (a) a transliteration-based approach leveraging phonetic similarity and (b) a corpus-based approach exploiting bilingual co-occurrences, each of which suffers from inaccuracy and scarcity respectively. In clear contrast, we use unleveraged resources of monolingual entity co-occurrences, crawled from entity search engines, represented as two entity-relationship graphs extracted from two language corpora respectively. Our problem is then abstracted as finding correct mappings across two graphs. To achieve this goal, we propose a holistic approach, of exploiting both transliteration similarity and monolingual co-occurrences. This approach, building upon monolingual corpora, complements existing corpus-based work, requiring scarce resources of parallel or comparable corpus, while significantly boosting the accuracy of transliteration-based work. We validate our proposed system using real-life datasets.

1 Introduction

Entity translation aims at mapping the entity names (*e.g.*, people, locations, and organizations) in source language into their corresponding names in target language. While high quality entity translation is essential in cross-lingual information access and translation, it is non-trivial to achieve, due to the challenge that entity translation, though typically bearing pronunciation similarity, can also be arbitrary, *e.g.*, Jackie Chan and 成龙 (pronounced Cheng Long). Existing efforts to address these challenges can be categorized into transliteration- and corpus-based approaches. Transliteration-based approaches (Wan and Verspoor, 1998; Knight and Graehl, 1998) identify translations based on pronunciation similarity, while corpus-based approaches mine bilingual co-occurrences of translation pairs obtained from parallel (Kupiec, 1993; Feng et al., 2004) or comparable (Fung and Yee, 1998) corpora, or alternatively mined from bilingual sentences (Lin et al., 2008; Jiang et al., 2009). These two approaches have complementary strength– transliteration-based similarity can be computed for any name pair but cannot mine translations of little (or none) phonetic similarity. Corpus-based similarity can support arbitrary translations, but require highly scarce resources of bilingual co-occurrences, obtained from parallel or comparable bilingual corpora.

In this paper, we propose a holistic approach, leveraging both transliteration- and corpus-based similarity. Our key contribution is to replace the use of scarce resources of bilingual co-occurrences with the use of untapped and significantly larger resources of monolingual co-occurrences for translation. In particular, we extract monolingual co-occurrences of entities from English and Chinese Web corpora, which are readily available from entity search engines such as PeopleEntityCube[1], deployed by Microsoft Research Asia. Such engine

*This work was done when the first two authors visited Microsoft Research Asia.

[1]http://people.entitycube.com

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 430–439,
MIT, Massachusetts, USA, 9-11 October 2010.

automatically extracts people names from text and their co-occurrences to retrieve related entities based on co-occurrences. To illustrate, Figure 1(a) demonstrates the query result for "Bill Gates," retrieving and visualizing the "entity-relationship graph" of related people names that frequently co-occur with Bill in English corpus. Similarly, entity-relationship graphs can be built over other language corpora, as Figure 1(b) demonstrates the corresponding results for the same query, from Renlifang[2] on Chinese Web corpus. From this point on, for the sake of simplicity, we refer to English and Chinese graphs, simply as G_e and G_c respectively. Though we illustrate with English-Chinese pairs in the paper, our method can be easily adapted to other language pairs.

In particular, we propose a novel approach of abstracting entity translation as a graph matching problem of two graphs G_e and G_c in Figures 1(a) and (b). Specifically, the similarity between two nodes v_e and v_c in G_e and G_c is initialized as their transliteration similarity, which is iteratively refined based on relational similarity obtained from monolingual co-occurrences. To illustrate this, an English news article mentioning "Bill Gates" and "Melinda Gates" evidences a relationship between the two entities, which can be quantified from their co-occurrences in the entire English Web corpus. Similarly, we can mine Chinese news articles to obtain the relationships between "比尔·盖茨" and "梅琳达·盖茨". Once these two bilingual graphs of people and their relationships are harvested, entity translation can leverage these parallel relationships to further evidence the mapping between translation pairs, as Figure 1(c) illustrates.

To highlight the advantage of our proposed approach, we compare our results with commercial machine translators (1) Engkoo[3] developed in Microsoft Research Asia and (2) Google Translator[4]. In particular, Figure 2 reports the precision for two groups– "heads" that belong to top-100 popular people (determined by the number of hits), among randomly sampled 304 people names[5] from six graph pairs of size 1,000 each, and the remaining "tails". Commercial translators such as Google, leveraging bilingual co-occurrences that are scarce for tails, show significantly lower precision for tails. Meanwhile, our work, depending solely on monolingual co-occurrences, shows high precision, for both heads and tails.

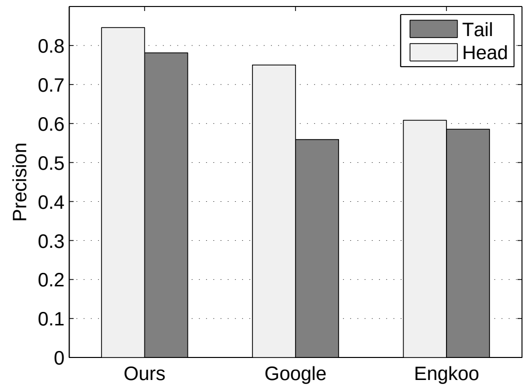


Figure 2: Comparison for Head and Tail datasets

Our focus is to boost translation accuracy for long tails with non-trivial Web occurrences in each monolingual corpus, but not with much bilingual co-occurrences, *e.g.*, researchers publishing actively in two languages but not famous enough to be featured in multi-lingual Wikipedia entries or news articles. As existing translators are already highly accurate for popular heads, this focus well addresses the remaining challenges for entity translation.

To summarize, we believe that this paper has the following contributions:

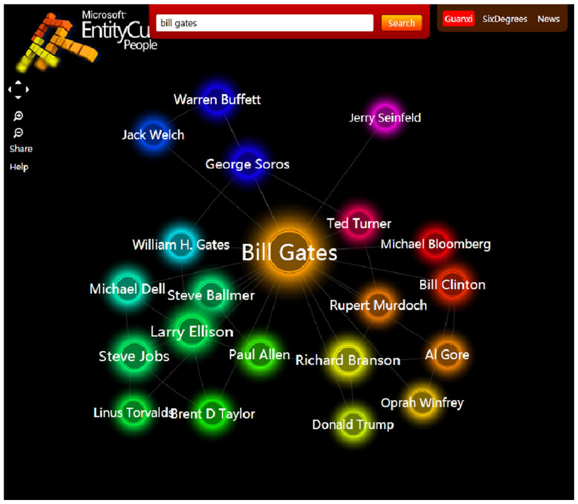
- We abstract entity translation problem as a graph mapping between entity-relationship graphs in two languages.

- We develop an effective matching algorithm leveraging both pronunciation and co-occurrence similarity. This holistic approach complements existing approaches and enhances the translation coverage and accuracy.

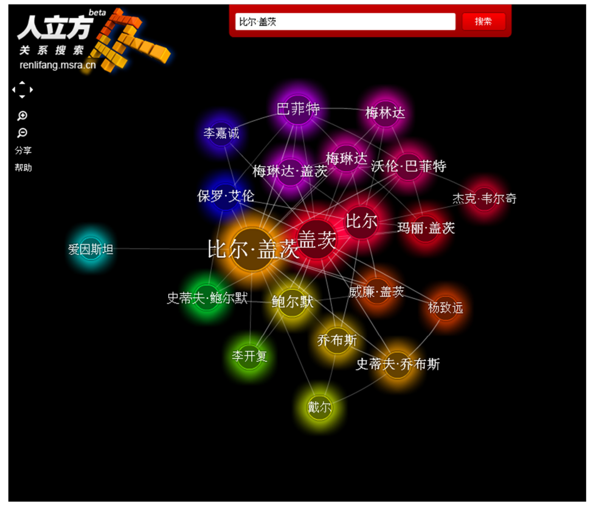
- We validate the effectiveness of our approach using various real-life datasets.

The rest of this paper is organized as follows. Section 2 reviews existing work. Section 3 then develops our framework. Section 4 reports experimental results and Section 5 concludes our work.

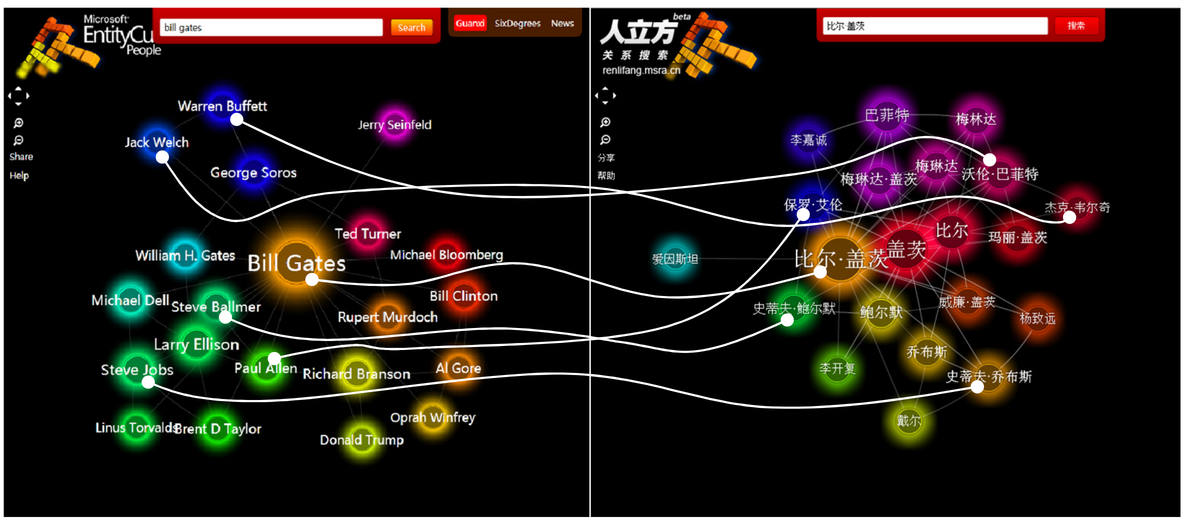
[2]http://renlifang.msra.cn
[3]http://www.engkoo.com
[4]http://translate.google.com
[5]See Section 4 for the sampling process.



(a) English PeopleEntityCube G_e



(b) Chinese Renlifang G_c



(c) Abstracting translation as graph mapping

Figure 1: Illustration of entity-relationship graphs

2 Related Work

In this section, we first survey related efforts, categorized into transliteration-based and corpus-based approaches. Our approach leveraging both is complementary to these efforts.

2.1 Transliteration-based Approaches

Many name translations are loosely based on phonetic similarity, which naturally inspires transliteration-based translation of finding the translation with the closest pronunciation similarity, using either rule-based (Wan and Verspoor, 1998) or statistical (Knight and Graehl, 1998; Li et al., 2004) approaches. However, people are free to designate arbitrary bilingual names of little (or none) phonetic similarity, for which the transliteration-based approach is not effective.

2.2 Corpus-based Approaches

Corpus-based approach can mine arbitrary translation pairs, by mining bilingual co-occurrences from parallel and comparable bilingual corpora. Using parallel corpora (Kupiec, 1993; Feng et al., 2004), *e.g.*, bilingual Wikipedia entries on the same person, renders high accuracy but suffers from high scarcity. To alleviate such scarcity, (Fung and Yee, 1998; Shao and Ng, 2004) explore a more vast resource of comparable corpora, which share no parallel document- or sentence-alignments as in parallel corpora but describe similar contents in two languages, *e.g.*, news articles on the same event. Alternatively, (Lin et al., 2008) extracts bilingual co-occurrences from bilingual sentences, such as annotating terms with their corresponding translations in English inside parentheses. Similarly, (Jiang et al., 2009) identifies potential translation pairs from bilingual sentences using lexical pattern analysis.

2.3 Holistic Approaches

The complementary strength of the above two approaches naturally calls for a holistic approach, such as recent work combining transliteration- and corpus-based similarity mining bilingual co-occurrences using general search engines. Specifically, (Al-Onaizan and Knight, 2002) uses transliteration to generate candidates and then web corpora to identify translations. Later, (Jiang et al., 2007) enhances to use transliteration to guide web mining.

Our work is also a holistic approach, but leveraging significantly larger corpora, specifically by exploiting monolingual co-occurrences. Such expansion enables to translate "long-tail" people entities with non-trivial Web occurrences in each monolingual corpus, but not much bilingual co-occurrences. Specifically, we initialize name pair similarity using transliteration-based approach, and iteratively reinforces base similarity using relational similarity.

3 Our Framework

Given two graphs $G_e = (V_e, E_e)$ and $G_c = (V_c, E_c)$ harvested from English and Chinese corpora respectively, our goal is to find translation pairs, or a set S of matching node pairs such that $S \subseteq V_e \times V_c$. Let R be a $|V_e|$-by-$|V_c|$ matrix where each R_{ij} denotes the similarity between two nodes $i \in V_e$ and $j \in V_c$.

Overall, with the matrix R, our approach consists of the following three steps, as we will discuss in the following three sections respectively:

1. **Initialization:** computing base translation similarities R_{ij} between two entity nodes using transliteration similarity

2. **Reinforcement model:** reinforcing the translation similarities R_{ij} by exploiting the monolingual co-occurrences

3. **Matching extraction:** extracting the matching pairs from the final translation similarities R_{ij}

3.1 Initialization with Transliteration

We initialize the translation similarity R_{ij} as the transliteration similarity. This section explains how to get the transliteration similarity between English and Chinese names using an unsupervised approach.

Formally, let an English name $N_e = (e_1, e_2, \cdots, e_n)$ and a Chinese name $N_c = (c_1, c_2, \cdots, c_m)$ be given, where e_i is an English word and N_e is a sequence of the words, and c_i is a Chinese character and N_c is a sequence of the characters. Our goal is to compute a score indicating the similarity between the pronunciations of the two names.

We first convert N_c into its Pinyin representation $PY_c = (s_1, s_2, \cdots, s_m)$, where s_i is the Pinyin representation of c_i. Pinyin is the romanization representation of pronunciation of Chinese character. For example, the Pinyin representation of N_e = ("Barack", "Obama") is PY_c =("ba", "la", "ke", "ao", "ba", "ma"). The Pinyin representations of Chinese characters can be easily obtained from Chinese character pronunciation dictionary. In our experiments, we use an in-house dictionary, which contains pronunciations of $20,774$ Chinese characters. For the Chinese characters having multiple pronunciations, we only use the most popular one.

Calculation of transliteration similarity between N_e and N_c is now transformed to calculation of pronunciation similarity between N_e and PY_c. Because letters in Chinese Pinyins and English strings are pronounced similarly, we can further approximate pronunciation similarity between N_e and PY_c using their spelling similarity. In this paper, we use Edit Distance (ED) to measure the spelling similarity. Moreover, since words in N_e are transliterated into characters in PY_c independently, it is more accurate to compute the ED between N_e and PY_c, *i.e.*, $ED_{name}(N_e, PY_c)$, as the sum of the EDs of all component transliteration pairs, *i.e.*, every e_i in N_e and its corresponding transliteration (s_i) in PY_c. In other words, we need to first align all s_j's in PY_c with corresponding e_i in N_e based on whether they are translations of each other. Then based on the alignment, we can calculate $ED_{name}(N_e, PY_c)$ using the following formula.

$$ED_{name}(N_e, PY_c) = \sum_i ED(e_i, es_i) \quad (1)$$

where es_i is a string generated by concatenating all s_i's that are aligned to e_i and $ED(e_i, es_i)$ is the Edit Distance between e_i and es_i, *i.e.*, the minimum number of edit operations (including insertion, deletion and substitution) needed to transform e_i into es_i. Because an English word usually consists of multiple syllables but every Chinese character consists of only one syllable, when aligning e_i's with s_j's, we add the constraint that each e_i is allowed to be aligned with 0 to 4 s_i's but each s_i can only be aligned with 0 to 1 e_i. To get the alignment between PY_c and N_e which has the minimal $ED_{name}(N_e, PY_c)$, we use a Dynamic Programming based algorithm as defined in the following formula:

$$\begin{aligned}
&ED_{name}(N_e^{1,i}, PY_c^{1,j}) = \min(\\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j}) + Len(e_i), \\
&\quad ED_{name}(N_e^{1,i}, PY_c^{1,j-1}) + Len(s_j), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-1}) + ED(e_i, s_j), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-2}) + ED(e_i, PY_c^{j-1,j}), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-3}) + ED(e_i, PY_c^{j-2,j}), \\
&\quad ED_{name}(N_e^{1,i-1}, PY_c^{1,j-4}) + ED(e_i, PY_c^{j-3,j}))
\end{aligned}$$

where, given a sequence $X = (x_1, x_2, \cdots)$ such that x_i is a word, $X^{i,j}$ is the subsequence $(x_i, x_{i+1}, \cdots, x_j)$ of X and $Len(X)$ is the number of letters except spaces in the sequence X. For instance, the minimal Edit Distance between the English name "Barack Obama" and the Chinese Pinyin representation "ba la ke ao ba ma" is 4, as the best alignment is: "Barack" $\leftrightarrow$ "ba la ke" (ED: 3), "Obama" $\leftrightarrow$ "ao ba ma" (ED: 1). Finally the transliteration similarity between N_c and N_e is calculated using the following formula.

$$Sim_{tl}(N_c, N_e) = 1 - \frac{ED_{name}(N_e, PY_c)}{Len(PY_c) + Len(N_e)} \quad (2)$$

For example, Sim_{tl}("Barack Obama", "巴拉克·奥巴马") is $1 - \frac{4}{11+12} = 0.826$.

3.2 Reinforcement Model

From the initial similarity, we model our problem as an iterative approach that iteratively reinforces the similarity R_{ij} of the nodes i and j from the matching similarities of their neighbor nodes u and v.

The basic intuition is built on exploiting the similarity between monolingual co-occurrences of two different languages. In particular, we assume two entities with strong relationship co-occur frequently in both corpora. In order to express this intuition, we formally define an iterative reinforcement model as follows. Let R_{ij}^t denote the similarity of nodes i and j at t-th iteration:

$$R_{ij}^{t+1} = \lambda \sum_{(u,v)_k \in B^t(i,j,\theta)} \frac{R_{uv}^t}{2^k} + (1-\lambda) R_{ij}^0 \quad (3)$$

The model is expressed as a linear combination of (a) the relational similarity $\sum R_{uv}^t / 2^k$ and (b) transliteration similarity R_{ij}^0. (λ is the coefficient for interpolating two similarities.)

In the relational similarity, $B^t(i,j,\theta)$ is an ordered set of the best matching pairs between neighbor nodes of i and ones of j such that $\forall (u,v)_k \in B^t(i,j,\theta), R_{uv}^t \geq \theta$, where $(u,v)_k$ is the matching pair with k-th highest similarity score. We consider (u,v) with similarity over some threshold θ, or $R_{uv}^t \geq \theta$, as a matching pair. In this neighbor matching process, if many-to-many matches exist, we select only one with the greatest matching score. Figure 3 describes such matching process more formally. $N(i)$ and $N(j)$ are the sets of neighbor nodes of i and j, respectively, and H is a priority queue sorting pairs in the decreasing order of similarity scores.

Meanwhile, note that, in order to express that the confidence for matching (i,j) progressively converges as the number of matched neighbors increases, we empirically use decaying coefficient $1/2^k$ for R_{uv}^t, because $\sum_{k=1}^{\infty} 1/2^k = 1$.

3.3 Matching Extraction

After the convergence of the above model, we get the $|V_e|$-by-$|V_c|$ similarity matrix R^∞. From this matrix, we extract one-to-one matches maximizing the overall similarity.

More formally, this problem can be stated as the *maximum weighted bipartite matching* (West,

```
B^t(i, j, θ) ← {}
∀u ∈ N(i), ∀v ∈ N(j) : H.push(u, v; R^t_uv)
while H is not empty do
  (u, v; s) ← H.pop()
  if s < θ then
    break
  end if
  if neither u nor v are matched yet then
    B^t(i, j, θ) ← B^t(i, j, θ) ∪ {(u, v)}
  end if
end while
return B^t(i, j, θ)
```

Figure 3: How to get the ordered set $B^t(i, j, \theta)$

2000)– Given two groups of entities V_e and V_c from the two graphs G_e and G_c, we can build a *weighted bipartite graph* is $G = (V, E)$, where $V = V_e \cup V_c$ and E is a set of edges (u, v) with weight R^{∞}_{uv}. To filter out null alignment, we construct only the edges with weight $R^{\infty}_{uv} \geq \delta$. From this bipartite graph, the maximum weighted bipartite matching problem finds a set of pairwise non-adjacent edges $S \subseteq E$ such that $\sum_{(u,v)\in S} R^{\infty}_{uv}$ is the maximum. Well-known algorithms include Hungarian algorithm with time complexity of $O(|V|^2 \log |V| + |V||E|)$ (West, 2000).

In this paper, to speed up processing, we consider a greedy alternative with the following steps– (1) choose the pair with the highest similarity score, (2) remove the corresponding row and column from the matrix, and (3) repeat (1) and (2) until their matching scores are over a specific threshold δ.

4 Experiments

This section reports our experimental results to evaluate our proposed approach. First, we report our experimental setting in Section 4.1. Second, we validate the effectiveness and the scalability of our approach over a real-life dataset in Section 4.2.

4.1 Experimental Settings

This section describes (1) how we collect the English and Chinese EntityCube datasets, (2) how to build ground-truth test datasets for evaluating our framework, and (3) how to set up three parameters λ, θ, and δ.

First, we crawled $G_e = (V_e, E_e)$ and $G_c = (V_c, E_c)$ from English and Chinese EntityCubes. Specifically, we built a graph pairs (G_e, G_c) expanding from a "seed pair" of nodes $s_e \in V_e$ and $s_c \in V_c$ until the number of nodes for each graph becomes 1,000[6]. More specifically, when we build a graph G_e by expanding from s_e, we use a queue Q. We first initialize Q by pushing the seed node s_e. We then iteratively pop a node v_e from Q, save v_e into V_e, and push its neighbor nodes in decreasing order of co-occurrence scores with v_e. Similarly, we can get G_c from a counterpart seed node v_c. By using this procedure, we built six graph pairs from six different seed pairs. In particular, the six seed nodes are English names and its corresponding Chinese names representing a wide range of occupation domains (*e.g.*, 'Barack Obama,' 'Bill Gates,' 'Britney Spears,' 'Bruno Senna,' 'Chris Paul,' and 'Eminem') as Table 1 depicts. Meanwhile, though we demonstrate the effectiveness of the proposed method for mining name translations in Chinese and English languages, the method can be easily adapted to other language pairs.

Table 1: Summary for graphs and test datasets obtained from each seed pair

i	$\|V_e\|, \|V_c\|$	$\|T_i\|$	English Name	Chinese Name
1	1,000	51	Barack Obama	巴拉克·奥巴马
2	1,000	52	Bill Gates	比尔·盖茨
3	1,000	40	Britney Spears	布兰妮·斯皮尔斯
4	1,000	53	Bruno Senna	布鲁诺·塞纳
5	1,000	51	Chris Paul	克里斯·保罗
6	1,000	57	Eminem	艾米纳姆

Second, we manually searched for about 50 "ground-truth" matched translations for each graph pair to build test datasets T_i, by randomly selecting nodes within two hops[7] from the seed pair (s_e, s_c), since nodes outside two hops may include nodes whose neighbors are not fully crawled. More specifically, due to our crawling process expanding to add neighbors from the seed, the nodes close to the seed have all the neighbors they would have in the full graph, while those far from the node may not. In order to pick the nodes that well represent the actual

[6]Note, this is just a default setting, which we later increase for scalability evaluation in Figure 6.

[7]Note that the numbers of nodes within two hops in G_e and G_c are 327 and 399 on average respectively.

neighbors, we built test datasets among those within two hops. However, this crawling is used for the evaluation sake only, and thus does not suggest the bias in our proposed framework. Table 1 describes the size of such test dataset for each graph pair.

Lastly, we set up the three parameters λ, θ, and δ using 6-fold cross validation with 6 test datasets T_i's of the graphs. More specifically, for each dataset T_i, we decide λ_i and θ_i such that average MRR for the other 5 test datasets is maximized. (About MRR, see more details of Equation (4) in Section 4.2.) We then decide δ_i such that average F1-score is maximized. Figure 4 shows the average MRR for λ_i and θ_i with default values $\theta = 0.66$ and $\lambda = 0.2$. Based on these results, we set λ_i with values $\{0.2, 0.15, 0.2, 0.15, 0.2, 0.15\}$ that optimize MRR in datasets $T_1, \ldots T_6$, and similarly θ_i with $\{0.67, 0.65, 0.67, 0.67, 0.65, 0.67\}$. We also set δ_i with values $\{0.63, 0.63, 0.61, 0.61, 0.61, 0.61\}$ optimizing F1-score with the same default values $\lambda = 0.2$ and $\theta = 0.66$. We can observe the variances of optimal parameter setting values are low, which suggests the robustness of our framework.

4.2 Experimental Results

This section reports our experimental results using the evaluation datasets explained in previous section. For each graph pair, we evaluated the effectiveness of (1) reinforcement model using MRR measure in Section 4.2.1 and (2) overall framework using precision, recall, and F1 measures in Section 4.2.2. We also validated (3) scalability of our framework over larger scale of graphs (with up to five thousand nodes) in Section 4.2.3. (In all experimental results, **Bold numbers** indicate the best performance for each metric.)

4.2.1 Effectiveness of reinforcement model

We evaluated the reinforcement model over MRR (Voorhees, 2001), the average of the reciprocal ranks of the query results as the following formula:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank_q} \tag{4}$$

Each q is a ground-truth matched pair (u, v) such that $u \in V_e$ and $v \in V_c$, and $rank_q$ is the rank of the similarity score of R_{uv} among all R_{uk}'s such that $k \in V_c$. Q is a set of such queries. By comparing MRRs for two matrices R^0 and R^∞, we can validate the effectiveness of the reinforcement model.

- Baseline matrix (R^0): using only the transliteration similarity score, *i.e.*, without reinforcement
- Reinforced matrix (R^∞): using the reinforced similarity score obtained from Equation (3)

Table 2: MRR of baseline and reinforced matrices

Set	MRR	
	Baseline R^0	Reinforced R^∞
T_1	0.6964	**0.8377**
T_2	0.6213	**0.7581**
T_3	0.7095	**0.7989**
T_4	0.8159	**0.8378**
T_5	0.6984	**0.8158**
T_6	0.5982	**0.8011**
Average	0.6900	**0.8082**

We empirically observed that the iterative model converges within 5 iterations. In all experiments, we used 5 iterations for the reinforcement.

Table 2 summarizes our experimental results. As these figures show, MRR scores significantly increase after applying our reinforcement model except for the set T_4 (on average from 69% to 81%), which indirectly shows the effectiveness of our reinforcement model.

4.2.2 Effectiveness of overall framework

Based on the reinforced matrix, we evaluated the effectiveness of our overall matching framework using the following three measures–(1) **precision**: how accurately the method returns matching pairs, (2) **recall**: how many the method returns correct matching pairs, and (3) **F1-score**: the harmonic mean of precision and recall. We compared our approach with a baseline, mapping two graphs with only transliteration similarity.

- Baseline: in matching extraction, using R^0 as the similarity matrix by bypassing the reinforcement step
- Ours: using R^∞, the similarity matrix converged by Equation (3)

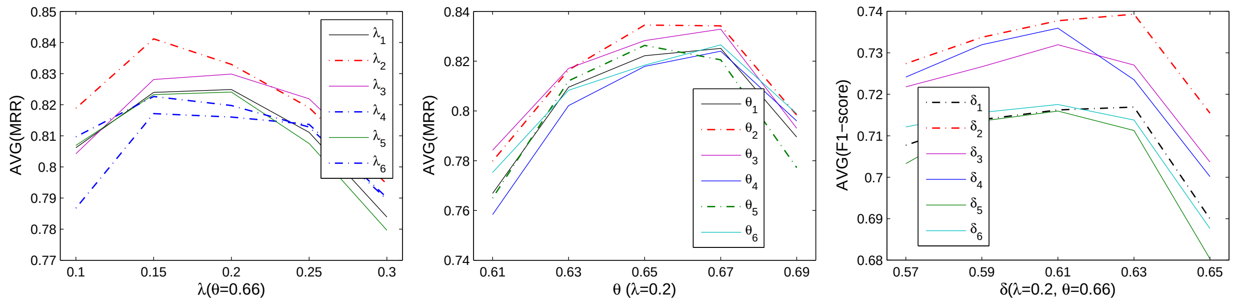


Figure 4: Parameter setup for λ, θ, and δ

In addition, we compared ours with the machine translators of Engkoo and Google. Table 3 summarizes our experimental results.

As this table shows, our approach results in the highest precision (about 80% on average) without compromising the best recall of Google, *i.e.*, 61% of Google vs. 63% of ours. Overall, our approach outperforms others in all three measures.

Meanwhile, in order to validate the translation accuracy over popular head and long-tail, as discussed in Section 1, we separated the test data into two groups and analyzed the effectiveness separately. Figure 5 plots the number of hits returned for the names from Google search engine. According to the distribution, we separate the test data into top-100 popular people with the highest hits and the remaining, denoted *head* and *tail*, respectively.

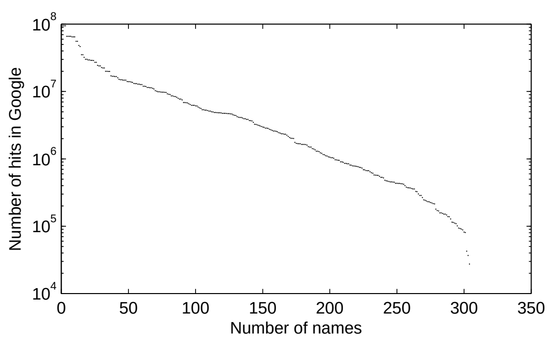


Figure 5: Distribution over number of hits

Table 4 shows the effectiveness with both datasets, respectively. As difference of the effectiveness between tail and head (denoted *diff*) with respect to three measures shows, our approach shows stably high precision, for both heads and tails.

4.2.3 Scalability

To validate the scalability of our approach, we evaluated the effectiveness of our approach over the number of nodes in two graphs. We built larger six graph pairs (G_e, G_c) by expanding them from the seed pairs further until the number of nodes reaches 5,000. Figure 6 shows the number of matched translations according to such increase. Overall, the number of matched pairs linearly increases as the number of nodes increases, which suggests scalability. The ratio of node overlap in two graphs is about between 7% and 9% of total node size.

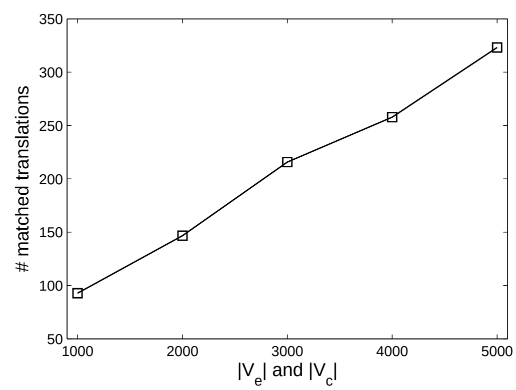


Figure 6: Matched translations over $|V_e|$ and $|V_c|$

5 Conclusion

This paper abstracted name translation problem as a matching problem of two entity-relationship graphs. This novel approach complements existing name translation work, by not requiring rare resources of parallel or comparable corpus yet outperforming the state-of-the-art. More specifically, we combine bilingual phonetic similarity and monolingual Web co-occurrence similarity, to compute a holistic notion of entity similarity. To achieve this goal, we de-

Table 3: Precision, Recall, and F1-score of Baseline, Engkoo, Google, and Ours over test sets T_i

Set	Precision				Recall				F1-score			
	Engkoo	Google	Baseline	Ours	Engkoo	Google	Baseline	Ours	Engkoo	Google	Baseline	Ours
T_1	0.5263	0.4510	0.5263	**0.8974**	0.3922	0.4510	0.1961	**0.6863**	0.4494	0.4510	0.2857	**0.7778**
T_2	0.7551	0.75	0.7143	**0.8056**	0.7115	**0.75**	0.2885	0.5577	0.7327	**0.75**	0.4110	0.6591
T_3	0.5833	0.7925	0.5556	**0.7949**	0.5283	**0.7925**	0.1887	0.5849	0.5545	**0.7925**	0.2817	0.6739
T_4	0.5	0.45	**0.7368**	0.7353	0.425	0.45	0.35	**0.625**	0.4595	0.45	0.4746	**0.6757**
T_5	0.6111	0.3137	0.5	**0.7234**	0.4314	0.3137	0.1765	**0.6667**	0.5057	0.3137	0.2609	**0.6939**
T_6	0.5636	**0.8947**	0.6	0.8605	0.5438	**0.8947**	0.1053	0.6491	0.5536	**0.8947**	0.1791	0.74
AVG	0.5899	0.6086	0.6055	**0.8028**	0.5054	0.6086	0.2175	**0.6283**	0.5426	0.6086	0.3155	**0.7034**

Table 4: Precision, Recall, and F1-score of Engkoo, Google, and Ours with head and tail datasets

Method	Precision			Recall			F1-score		
	head	tail	diff	head	tail	diff	head	tail	diff
Engkoo	0.6082	0.5854	**0.0229**	0.59	0.4706	0.1194	0.5990	0.5217	0.0772
Google	0.75	0.5588	0.1912	**0.75**	0.5588	0.1912	**0.75**	0.5588	0.1912
Ours	**0.8462**	**0.7812**	0.0649	0.66	**0.6127**	**0.0473**	0.7416	**0.6868**	**0.0548**

veloped a graph alignment algorithm that iteratively reinforces the matching similarity exploiting relational similarity and then extracts correct matches. Our evaluation results empirically validated the accuracy of our algorithm over real-life datasets, and showed the effectiveness on our proposed perspective.

Acknowledgments

This work was supported by Microsoft Research Asia NLP theme funding and MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement) (IITA-2009-C1090-0902-0045).

References

Yaser Al-Onaizan and Kevin Knight. 2002. Translating Named Entities Using Monolingual and Bilingual Resources. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*, pages 400–408. Association for Computational Linguistics.

Donghui Feng, Yajuan Lü, and Ming Zhou. 2004. A New Approach for English-Chinese Named Entity Alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*, pages 372–379. Association for Computational Linguistics.

Pascale Fung and Lo Yuen Yee. 1998. An IR Approach for Translating New Words from Nonparallel,Comparable Texts. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, pages 414–420. Association for Computational Linguistics.

Long Jiang, Ming Zhou, Lee feng Chien, and Cheng Niu. 2007. Named Entity Translation with Web Mining and Transliteration. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 1629–1634. Morgan Kaufmann Publishers Inc.

Long Jiang, Shiquan Yang, Ming Zhou, Xiaohua Liu, and Qingsheng Zhu. 2009. Mining Bilingual Data from the Web with Adaptively Learnt Patterns. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL'09)*, pages 870–878. Association for Computational Linguistics.

Kevin Knight and Jonathan Graehl. 1998. Machine Transliteration. *Computational Linguistics*, 24(4):599–612.

Julian Kupiec. 1993. An Algorithm for finding Noun Phrase Correspondences in Bilingual Corpora. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 17–22. Association for Computational Linguistics.

Haizhou Li, Zhang Min, and Su Jian. 2004. A Joint Source-Channel Model for Machine Transliteration. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL'04)*, pages 159–166. Association for Computational Linguistics.

Dekang Lin, Shaojun Zhao, Benjamin Van Durme, and Marius Pasca. 2008. Mining Parenthetical Translations from the Web by Word Alignment. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*, pages 994–1002. Association for Computational Linguistics.

Li Shao and Hwee Tou Ng. 2004. Mining New Word Translations from Comparable Corpora. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 618–624. Association for Computational Linguistics.

Ellen M. Voorhees. 2001. The trec question answering track. *Natural Language Engineering*, 7(4):361–378.

Stephen Wan and Cornelia Maria Verspoor. 1998. Automatic English-Chinese Name Transliteration for Development of Multilingual Resources. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, pages 1352–1356. Association for Computational Linguistics.

Douglas Brent West. 2000. *Introduction to Graph Theory*. Prentice Hall, second edition.