Cross-Language Latent Relational Search between Japanese and English Languages Using a Web Corpus

NGUYEN TUAN DUC, DANUSHKA BOLLEGALA and MITSURU ISHIZUKA
The University of Tokyo

Latent relational search is a novel entity retrieval paradigm based on the proportional analogy between two entity pairs. Given a latent relational search query {(Japan, Tokyo), (France, ?)}, a latent relational search engine is expected to retrieve and rank the entity “Paris” as the first answer in the result list, because the relation between Japan and Tokyo is highly similar to that between France and Paris. A latent relational search engine extracts entities and relations between those entities from a corpus, such as the Web. Moreover, from some supporting sentences in the corpus, (e.g., “*Tokyo is the capital of Japan*” and “*Paris is the capital and biggest city of France*”), the search engine must recognize the relational similarity between the two entity pairs. In cross-language latent relational search, the entity pairs as well as the supporting sentences of the first entity pair and of the second entity pair are in different languages. Therefore, the search engine must recognize similar semantic relations across languages. In this paper, we study the problem of cross-language latent relational search between Japanese and English using Web data. We represent the relations between two entities in an entity pair using the lexical patterns of the context surrounding the two entities. To perform cross-language latent relational search in high speed, we propose a multi-lingual indexing method for storing entities and relations extracted from Web corpora. We then propose a hybrid lexical pattern clustering algorithm to capture the semantic similarity between lexical patterns across languages. Using this algorithm, we can precisely measure the relational similarity between entity pairs across languages, thereby achieving high precision in the task of cross-language latent relational search. Experiments show that the proposed method achieves a Mean Reciprocal Rank (MRR) of 0.605 on Japanese-English cross-language latent relational search query sets. In addition, when using the English and Japanese Wikipedia data dumps as a corpus to create an index, the proposed search engine was able to answer several sophisticated questions in the INEX 2008 Entity Ranking task.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; *Retrieval models*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: latent relational search, cross-language relational search, analogical search, latent relational analysis

ACM Reference Format:
Duc, N. T., Bollegala, D. and Ishizuka, M. 2012. Cross-Language Latent Relational Search between Japanese and English Languages Using a Web Corpus. ACM Trans. Asian Language Inform. Process. V, N, Article A (January YYYY), 32 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

We would like to thank Peter D. Turney for precious suggestions on this work. We thank Hong Anh for creating many queries in the experiment on the INEX query set. We thank the anonymous reviewers for valuable comments. This work is partially supported by the IPA Mitoh project of the Information-technology Promotion Agency (IPA), Japan and a grant from Google Research Award.
Author’s addresses: N. T. Duc, D. Bollegala and M. Ishizuka, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan.

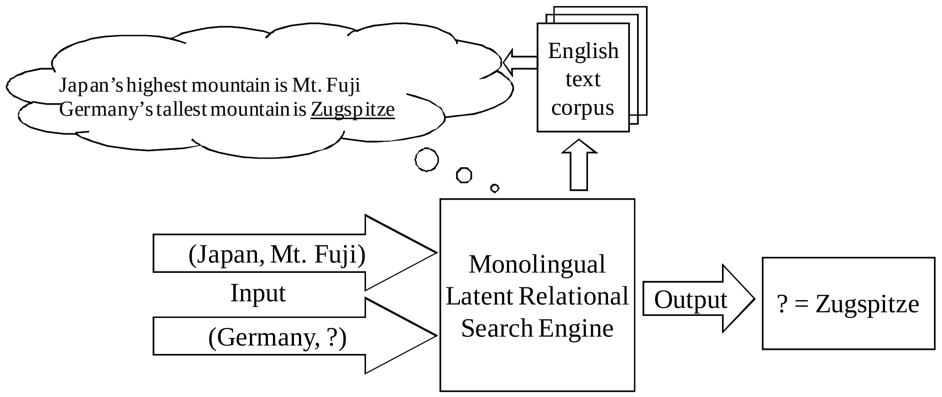

Japan's highest mountain is Mt. Fuji
Germany's tallest mountain is Zugspitze
English text corpus
(Japan, Mt. Fuji)
Input
(Germany, ?)
Monolingual Latent Relational Search Engine
Output
? = Zugspitze

Fig. 1. Example of a monolingual latent relational search query

1. INTRODUCTION

The World Wide Web consists of a huge number of Web pages referring to numerous entities and relations between those entities. With enormous number of entities and relations extracted from the Web, relying only on traditional keyword-based Web search engines are inadequate to fulfill users' information needs. This motivates us to explore the latent relational search approach [Kato et al. 2009; Goto et al. 2010; Duc et al. 2010], in which the retrieval process is based on semantic relations between entities, rather than only on keywords.

Latent relational search is a novel entity retrieval paradigm based on the analogy between entity pairs. An example of a monolingual latent relational search query is shown in Fig. 1. In this figure, the search engine is given three entities: two entities in an entity pair *(Japan, Mt. Fuji)* and a third entity (*Germany*) with a question mark (*?*). The search engine relies on some sentences in its corpus such as "Japan's highest mountain is Mt. Fuji." and "Germany's tallest mountain is Zugspitze." to output an appropriate entity (*Zugspitze*) to fill in the position of the question mark. We denote this query as {(Japan, Mt. Fuji), (Germany, ?)}. Therefore, a latent relational search query has the form of {(A, B), (C, ?)}, in which A, B, C are input entities. We call the entity pair (A, B) as the "source entity pair" (or the "source pair") and the entity C as the "key entity". The objective of latent relational search is to retrieve a ranked list **L** of entities so that for each entity $D \in \mathbf{L}$, the semantic relation between A and B is similar to that between C and D. We call the entity pair (C, D) as the "target entity pair" (or the "target pair").

When the semantic relation between two entities A and B is highly similar to that between two entities C and D, we say that the entity pairs (A, B) and (C, D) have a high degree of *relational similarity* or they are analogous [Turney 2005; 2006]. A latent relational search engine works by recognizing the analogy between two entity pairs, the source pair and the target pair. In the above example, the entity "Zugspitze" is retrieved because the relation between "Japan" and "Mt. Fuji" in the source pair is highly similar to that between "Germany" and "Zugspitze" in the target entity pair (Mt. Fuji is the highest mountain in Japan, where as, Zugspitze is the highest mountain in Germany).

Latent relational search can be used for mapping knowledge between different domains [Duc et al. 2010] as can be seen in the above example. The user has knowledge about the highest mountain in Japan (the source domain). Using this knowledge, the

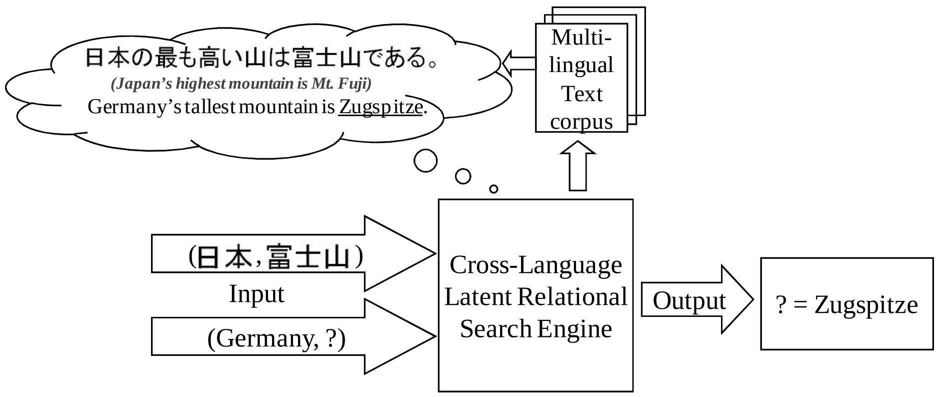

日本の最も高い山は富士山である。
(Japan's highest mountain is Mt. Fuji)
Germany's tallest mountain is Zugspitze.
Multi-lingual Text corpus
(日本,富士山)
Input
(Germany, ?)
Cross-Language Latent Relational Search Engine
Output
? = Zugspitze

Fig. 2. An example cross-language latent relational search query, the input pair is in Japanese, meaning (Japan, Mt. Fuji).

user can formulate the query {(Japan, Mt. Fuji), (Germany, ?)} to query for the name of the highest mountain in Germany (the target domain). The search engine uses relational similarity to map the knowledge from the source domain (Japan's mountains) to the target domain (Germany's mountains) and retrieve the answer. That is, the knowledge of a familiar domain is mapped to a novel domain to discover new knowledge in this novel domain. Therefore, latent relational search is useful for knowledge discovery and knowledge acquisition. Especially, when a user does not know which keywords are appropriate to formulate a query, latent relational search can be used very effectively. Using latent relational search, a user can find an appropriate keyword (e.g., "Zugspitze") and can then use this keyword to formulate queries to a keyword-based search engine to retrieve related documents.

Because a large portion of the Web is non-English, cross-language information retrieval and cross-language question answering using Web data have become important than ever before. Many attempts have been made in the field of Web-based question answering [Kwok et al. 2001; Dumais et al. 2002] and cross-language question answering [Isozaki et al. 2005; Peñas et al. 2009] to overcome this language barrier in information retrieval. However, cross-language information retrieval and question answering systems rely heavily on machine translation, which might produce poor results because of the noise in Web text and the lack of resources such as parallel corpora or translation dictionaries for some language pairs [Ferrandez et al. 2009]. Consequently, in this paper, we propose *cross-language latent relational search*, in which only simple phrases are required to be translated, to alleviate the adverse effects attributable to machine translation in cross-language information retrieval and question answering systems.

An example of cross-language latent relational search is answering the question "ドイツの最も高い山の名前は何ですか。" (meaning "What is the name of the highest mountain in Germany?" in Japanese) when a user only knows that the highest mountain in Japan is "富士山" ("Mt. Fuji") and there are not enough Japanese web pages concerning the highest mountain in Germany for a Web-based question answering system to find the answer. In this situation, the user can formulate the query {(日本, 富士山), (Germany, ?)} (the first entity pair is (Japan, Mt. Fuji) written in Japanese), to obtain the answer "Zugspitze", as shown in Fig. 2. This kind of queries might be useful when a Japanese user is traveling to Germany and wants to visit some places like Mt. Fuji in Japan. In

Fig. 2, the search engine relies on some supporting sentences in Japanese (to identify the relation between 日本 (Japan) and 富士山 (Mt. Fuji)) and some other sentences in English (to identify the relation between Germany and Zugspitze) to output the answer “Zugspitze”.

Another real-world situation for using cross-language latent relational search is that when a Japanese user wants to find a list of recording companies that sell the *Kingston Trio*'s songs. This is the topic number 23 in the TREC 2010 Entity Ranking track[1] [Balog et al. 2010]. An English user can easily form a query such as {(Lady Gaga, UMG), (Kingston Trio, ?)} to retrieve the list. However, because many Japanese users are not familiar with the UMG (Universal Music Group), they can not imagine such query. The user might use a Japanese-to-Japanese monolingual query such as {(*Hamasaki Ayumi*, *Eibekkusu*), (*Kingusuton Torio*, ?)} (all entities are written in Japanese) to retrieve the list. However, there is much little information concerning the Kingston Trio in Japanese than in English. As another option, the user might use traditional cross-lingual IR systems to input the query in Japanese and retrieve the answers in English. However, the user must formulate some queries such as “*Kingusuton Torio hanbai kaisha*” (meaning “companies selling Kingston Trio” in Japanese). Moreover, the above query would not return a comprehensive list of recording companies that sell songs of the Kingston Trio because there are many sentences that describe the target relationship but do not co-occur with the term “*hanbai*” (selling), even when we translate the term into English. For example, the sentence “Sony BMG records and distributes Kingston Trio's songs.” mentions a company that sells the Kingston Trio's songs, but it does not strongly match the query. In this case, the user can use a Japanese-to-English cross-language latent relational search query (e.g., {(*Hamasaki Ayumi*, *Eibekkusu*), (Kingston Trio, ?)}) to get the answers because the pair (*Hamasaki Ayumi*, *Eibekkusu*) (meaning (Ayumi Hamasaki, Avex)) co-occurs with many lexical patterns that exactly describe the desired semantic relation.

There are several methods for answering latent relational search queries [Kato et al. 2009; Duc et al. 2010; Halskov and Barriere 2008; Goto et al. 2010]. However, these methods focus on monolingual latent relational search, as they represent the semantic relations between two entities in an entity pair by terms or lexico-syntactic patterns from the context surrounding the two entities and compare them in the same language. Consequently, if there are not any sentence pairs that mention the source pair and the target pair in the same language, then these search engines do not have sufficient contexts to measure the relational similarity between the two entity pairs. Moreover, even while searching for an entity in another language, users often easily imagine a source entity pair in their own languages. For example, if a non-native Japanese speaker can only write down the source pair in English (not in Japanese), but the target entity is mainly mentioned in Japanese web pages, then the user can not use monolingual latent relational search to retrieve the answer. With the growing number of non-English documents on the Web, Web search engine users frequently encounter such situations. Therefore, there is a strong requirement for latent relational search across languages.

Cross-language latent relational search extends the capability of latent relational search from cross-domain knowledge mapping to cross-domain and cross-language knowledge mapping [Duc et al. 2011]. In this paper, we focus on Japanese-English cross-language latent relational search because the Japanese-English pair is one of the most difficult language pairs for machine translation [Sumita 2001]. We believe that if we can get a reasonable result in Japanese-English cross-language latent rela-

[1]http://trec.nist.gov/data/entity/10/10.entity%5Ftopics

tional search then there is a high probability that we can achieve better performance for other language pairs.

The contributions of this paper are as follows:

— We propose cross-language latent relational search, an advanced latent relational search paradigm in which the source entity pair and the target entity pair are in different languages, possibly with different writing systems. The evidences (supporting sentences) that the search engine can rely on are also in different languages. This extends the capability of latent relational search from cross-domain knowledge mapping to cross-domain and cross-language knowledge mapping.
— We propose a multi-lingual indexing method to index entity pairs and relationships in multiple languages and to utilize identical surface forms of named entities across languages for cross-language relational similarity measuring.
— Following previous work on relational similarity measuring algorithms [Turney 2005; 2006; Bollegala et al. 2009b; 2009a] and monolingual latent relational search [Kato et al. 2009; Goto et al. 2010; Duc et al. 2010], we represent the semantic relations between two entities in an entity pair by lexical patterns of the context surrounding the two entities. We propose a novel two-phase clustering algorithm to capture the semantic similarity of paraphrased lexical patterns across two languages, Japanese and English. Using the result of this algorithm, we can measure the relational similarity between two entity pairs written in two different languages to rank the result list of an English-Japanese cross-language latent relational search query. Moreover, we present a method for combining the proposed lexical pattern clustering algorithm with cross-language Latent Relational Analysis [Turney 2005] to improve the overall performance of cross-language latent relational search systems.
— We evaluate the proposed method using a corpus containing 1.6 GB of web pages in English and Japanese. Our experiments show that, the proposed method achieves a mean reciprocal rank (MRR) of 0.605 for cross-language latent relational search queries. The percentage of queries with correct answer in the Top 10 results is 78.5%. Moreover, we also evaluate the system with questions in the INEX 2008 Entity Ranking task to show that the system can be used for answering sophisticated questions concerning various relation types.

The remainder of this paper is organized as follows. We review related work in Section 2. We present an overview of the proposed method in Section 3. Section 4 presents the proposed multi-lingual entity pair indexing method and the relation representation method. To recognize paraphrased lexical patterns across languages, we propose a hybrid (hard and soft) lexical pattern clustering algorithm in Section 5. The proposed method has been extensively experimented with Japanese-English web corpora. We report the evaluation results with these experiments in Section 6. Finally, we conclude the paper in Section 7.

2. RELATED WORK

Analogical reasoning is an important topic in Artificial Intelligence. Many models have been proposed to solve analogy questions, such as the Structure Mapping Engine (SMT) [Gentner 1983] and the Latent Relational Mapping Engine [Turney 2008b].

Hofstadter et al. propose "a model of high-level perception and analogical thought in which perceptual processing is integrated with analogical mapping" [Chalmers et al. 1992; Hofstadter and FARG 1995]. Using this model, they can solve the analogy puzzles such as "If **abc** changes to **abd**, what does **iijjkkll** change to?" when the computer knows only the orders of characters in the alphabet, and has no other knowledge. They describe that the analogical reasoning process is influenced by belief, goals and external contexts. Therefore, analogical mapping models must take into account these factors. A user of a latent relational search system might also be affected by this behavior. For example, given the query $\{$(Google, YouTube), (Yahoo, ?)$\}$, the answer might be any of the companies that Yahoo has acquired. Depend on the pragmatic understanding of the user, the best answer might be different (for example, if the goal of the user is to find a video sharing service that Yahoo has acquired, the best answer should be "Broadcast.com"). In this research, we rely on the frequencies of common (shared) lexical patterns to determine the relational similarity between entity pairs, as seen in [Turney 2005; Bollegala et al. 2009b]. Therefore, we would not directly take into account the characteristics of entities and the pragmatic understanding of a specific user, but we do consider the pragmatic understanding of the majority of users, who write text on the Web, as the proposed search engine analyzes text on the Web to create its knowledge base (the index). For example, if the term YouTube co-occurs with the term "video sharing" with high frequency, then we would extract many lexical patterns that contain this term for the pair (Google, YouTube).

The idea of latent relational search has been discussed by Veale [2003] and Bollegala et al. [2009a]. Veale [2003] proposes a search paradigm in which queries such as "Muslim church" or "Greek A" can be answered. In these queries, the second concept (e.g., "church") is in different domain with the first concept (church is a concept in the domain of "Christian", not "Muslim"). The answer of this kind of queries should be the corresponding concept in the domain of the first concept. For example the answer for the query "Muslim church" should be "mosque" because mosque is to Muslim as church is to Christian. Similarly, the answer for the query "Greek A" is "α" (the Greek letter Alpha).

Veale also presents a method for answering such kind of queries by adding fine-grained concepts into WordNet[2]. Although the method relied on WordNet is intuitive to understand, it can not handle almost all named entities on the Web because WordNet does not cover all named entities. Because named entities are often interesting to search engine users, we need a method that allows the retrieval of these kinds of entities.

Research on measuring relational similarity between two word pairs [Turney 2005; 2006; Bollegala et al. 2009b; 2009a] suggests a method for ranking candidate entity pairs in latent relational search. Latent relational search aims to retrieve a list of entities $\mathbf{L}$ as the result list for the query $\{(A, B), (C, ?)\}$ in which each entity $D \in L$ satisfies that the degree of relational similarity between (C, D) and (A, B) is high. Consequently, the result list $\mathbf{L}$ could be ranked by the relational similarity measure between the source pair and candidate target pairs. In these studies, the relation between two entities in an entity pair is represented by lexical patterns, i.e., the context where the two entities appear. In this research, we adapt the relational similarity measuring algorithm in [Bollegala et al. 2009b] for measuring the relational similarity between entity pairs across languages.

Latent relational search can be considered as a kind of template filling task which is one of the targets of MUC [Grishman and Sundheim 1996] and TREC [Balog et al. 2009]. Likewise, the Kiwi/Tonguen systems [Tanaka-Ishii and Nakagawa 2005; Tanaka-Ishii and Ishii 2007] fill templates provided by users. However, in latent relational search, to find the answer, the relation provided in the source pair must be extracted or represented by some means. For example, we might represent the explicit semantic relation between the two entities in the source pair by some statistical proxy for the relationship, such as by a set of lexical patterns of the context surrounding the two entities [Turney 2005; Bollegala et al. 2009a].

[2]http://wordnet.princeton.edu/

WWW2REL [Halskov and Barriere 2008] is a system that can fill templates of the forms $R(arg_1, ?)$ or $R(?, arg_2)$, in which R is a relation and arg_1, arg_2 are arguments of the relation. It first discovers the lexical patterns that represent the relation R using 40 seed pairs that hold the relation R. It then uses the given argument in the query and the extracted lexical patterns to find the answer (e.g., for the query *INDUCES*$(aspirin, ?)$, it uses the word *aspirin* and the lexical pattern "*may cause*" to find the answer *apoptosis*). Therefore, the method requires that the thesaurus contains several instances of the relation in the query. If the thesaurus did not contain the relation's instances or there was no thesaurus, WWW2REL would give low precision or even would not find the answer.

One implementation of monolingual latent relational search is described in [Kato et al. 2009]. They use two phases to find and rank candidates for the query {(A, B), (C, ?)}. Suppose that the answer for this query is D. First, they build a "relation extractor" E(A, B) for the pair (A, B) using a Web search engine. The relation extractor E(A, B) extracts terms or lexico-syntactic patterns that represent the relations between A and B (this term or pattern set is denoted by T). E(A, B) can be built by querying a Web search engine for terms or lexical patterns that are likely to appear only in documents which contain both A and B. Then, in the second phase, they use the keyword C and the term or lexical pattern $t \in T$ for querying the Web search engine for documents that contain both C and t. The candidate set for the answer D is then the set of terms that are likely to appear only in these documents. Therefore, this method represents the relations between two words in a given word pair by using a bag-of-words model. The method in [Kato et al. 2009] has advantages such as it can find a large range of term D (because it finds all terms D that are likely to appear with C and $t \in T$), it does not require a local index for searching (it uses an existing keyword-based Web search engine to find E(A, B) and D). However, it does not use an explicit relational similarity measure for ranking the result list, but instead it uses the likelihood of co-occurrence of term D and the pair (C,t) in a document for ranking. Term D might appear in a document with the terms C, t but the relation between C and D might not be expressed by the term t. For example, if C is *Microsoft* and t is *CEO*, the term *Windows* might co-occur with C and t with high frequency. In these cases, the term *Windows* is in documents which contain both *Microsoft* and *CEO* but it might not be in a same sentence such as *"Microsoft's CEO is . . . "* . Therefore, to achieve high precision, the relational similarity between (A, B) and (C, D) should be measured using a well-defined method such as [Bollegala et al. 2009a; Turney 2006], in which the relation between C and D is represented by lexical patterns that are in the same sentence with the pair (C, D).

Goto et al. [2010] propose a method for improving performance of latent relational search by exploiting the symmetries of relational similarity. Because the degree of relational similarity between (A, B) and (C, D) is similar to that between (B, A) and (D, C), Goto et al. [2010] propose to incorporate the score of D in the reversed query {(B, A), (?, C)} when ranking results of the query {(A, B), (C, ?)}. We also use the reversed query {(B, A), (?, D)} in this research to calculate the rank in the final result list.

In our previous work on monolingual latent relational search [Duc et al. 2010], we propose a method for indexing entity pairs and relations to perform latent relational search in high speed. Moreover, we apply the method in [Bollegala et al. 2009b] to recognize paraphrased lexical patterns in the same language. This helps us to precisely measure the relational similarity between two entity pairs, and therefore achieve a high precision in the task of monolingual latent relational search.

In cross-lingual latent relational search, we must recognize semantically similar lexical patterns across languages. Specifically, in this work, we propose a method to recognize paraphrased lexical patterns across languages by a hybrid (hard/soft) clustering algorithm of lexical patterns. In a previous research, Davidov and Rappoport [2010] propose an approach for automated translation of semantic relationships across languages. Therefore, there is a common problem that the research in [Davidov and Rappoport 2010] and this research try to solve: finding out semantically similar lexical patterns for representing semantic relationships among different languages. Consequently, we present a comparison between the proposed method in this work and the method in [Davidov and Rappoport 2010] in the next paragraphs.

Given a lexical pattern cluster that represents a semantic relation in a source language (e.g., English), Davidov and Rappoport [2010] first use a Web search engine to retrieve a set of entity pairs that might hold this semantic relation. For example, from the pattern cluster {"X, CEO of Y", "X is the CEO of Y"}, the method formulates some queries such as "$*$, CEO of $*$", "$*$ is the CEO of $*$" and queries a keyword-based Web search engine to retrieve text snippets such as "Steve Jobs, CEO of Apple" or "Steve Ballmer is the CEO of Microsoft". From these text snippets, they can retrieve entity pairs (e.g., (Steve Jobs, Apple), (Steve Ballmer, Microsoft), ...) that hold the semantic relation (the *CEO-COMPANY* relation). They then use Web hit count ratios to rank the entity pairs according to their degree of representativeness (*"specificity"*) for the given semantic relation. For example, the specificity of the pair (Steve Jobs, Apple) against the CEO-COMPANY relation might be calculated as the hit count ratio between the number of hits for the query "Steve Jobs, CEO of Apple" and the number of hits for the query "Steve Jobs $* * *$ Apple". At this step, an entity pair that holds more than one semantic relations (such as the pair (Steve Jobs, Apple) holds both the CEO-COMPANY relation and the FOUNDER-COMPANY relation) might be excluded because the specificity of the pair is low (i.e., this pair is not a representative pair for the CEO-COMPANY relation). The reason that causes the specificity to be low is that when an entity pair holds different semantic relationships, the hit count ratio between the number of hits for the query that matches the relation (e.g., "Steve Jobs, CEO of Apple") and the query that matches only the entities ("Steve Jobs $* * *$ Apple") will be small. After calculating the specificities, entity pairs with highest specificities are selected and are translated into the target language (e.g., Japanese) using dictionaries. From a translated entity pair (A, B), they formulate queries such as "A $*$ B" and "A $*$ $*$ B" to retrieve text snippets, which are the contexts in which this pair occurs. For example, from the entity pair (*Sutivu Baluma, Maikurosofuto*) (in Japanese Katakana, which means (Steve Ballmer, Microsoft)), they can retrieve some text snippets such as "*X ha Y no shacho*", "*X ga Y no daihyou torishimari yaku*", etc. which represent the *CEO-COMPANY* relation in Japanese. They then rank the translated lexical patterns by using a confident score which represents the salience of each pattern to the translated entity pair set. Only patterns with high salient scores are selected in the final translated pattern cluster.

The method in [Davidov and Rappoport 2010] does not require a machine translation system and it works well for many language pairs. Therefore, it can be effectively used for translating lexical patterns in cross-lingual latent relational search when we can not utilize machine translation.

However, the final target of our research is not limited to translation of lexical patterns between languages, but to optimize the performance of cross-lingual latent relational search. In cross-lingual latent relational search, incorrectly translated lexical patterns might result in retrieving inappropriate candidate entity pairs, which do not hold the same semantic relation with the input pair. Consequently, we try to achieve highest accuracy in recognizing paraphrased lexical patterns across languages by exploiting both parallel entity pairs and parallel lexical patterns. Specifically, the most different point between the proposed method in this work and in [Davidov and Rap-

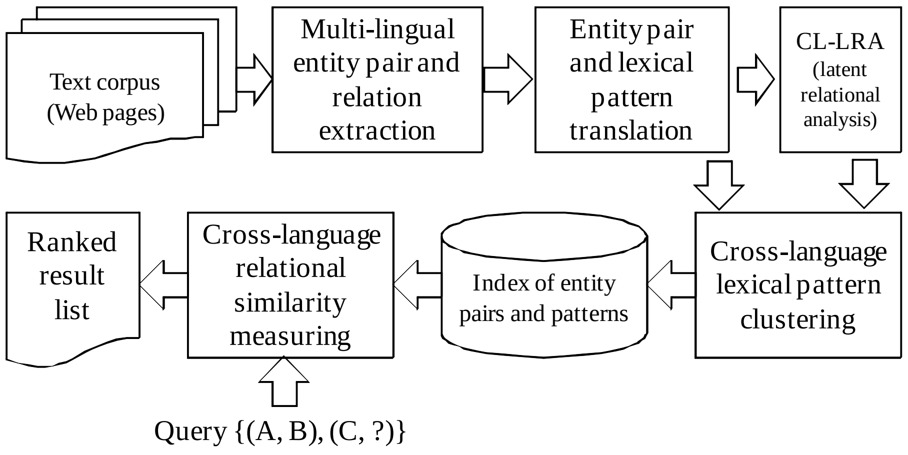

Text corpus (Web pages)
Multi-lingual entity pair and relation extraction
Entity pair and lexical pattern translation
CL-LRA (latent relational analysis)
Cross-language lexical pattern clustering
Index of entity pairs and patterns
Cross-language relational similarity measuring
Ranked result list
Query {(A, B), (C, ?)}

Fig. 3. Overview of the proposed method

poport 2010] is the starting point. Our approach starts from parallel entity pairs to measure the semantic similarity between lexical patterns across languages. Furthermore, the proposed method also *directly* translates lexical patterns to obtain several high-confident parallel patterns for pattern clustering. A lexical patterns often contains a short and simple sequence of words. Consequently, it can be translated by a machine translation system with high accuracy. Moreover, with the verification scheme (will be described later in Section 4.2), we can filter out incorrect translation results and keep only correctly translated patterns.

On the other hand, the method proposed by Davidov and Rappoport [2010] starts from pattern clusters in the source language to achieve some representative entity pairs that hold the relation in the source language. It then translates these entity pairs into the target language (using dictionary lookups) and then infers the translated lexical patterns from the contexts in which the translated entity pairs occur. Consequently, the method in [Davidov and Rappoport 2010] *indirectly* translates lexical patterns. A future research direction would be to use those indirectly translated lexical patterns to verify the result of the pattern clustering algorithm proposed in this work to further improve the accuracy while recognizing semantically similar lexical patterns across languages.

3. METHOD OVERVIEW

Fig. 3 shows an overview of the proposed method. The first step in our method is to extract from a given Japanese-English text corpus (a corpus containing Japanese and English documents, not necessarily an aligned or a parallel corpus) all entity pairs and lexical patterns that might represent the semantic relations between two entities in each pair.

We improve the quality of our cross-language relational similarity measuring algorithm by trying to find parallel pattern pairs in the two languages. We do this by a pattern translation step as shown in Fig. 3. Translating short lexical patterns can be done with a good precision in a high speed. We then assign a same pattern ID to two patterns that are parallel of each other. Similarly, we assign a same entity pair ID for parallel entity pairs to create a matrix in the next step.

Next, we create a matrix $\mathbf{A}$ whose row i corresponds to the lexical pattern p_i and column j corresponds to the entity pair w_j. The element $\mathbf{A}_{ij}$ of $\mathbf{A}$ represents the degree of association between p_i and w_j. We then apply Singular Value Decomposition to the matrix $\mathbf{A}$ as can be seen in Latent Relational Analysis (LRA) [Turney 2005] to reduce the dimension of the feature vector for each lexical pattern.

We propose a two-phase lexical pattern clustering algorithm to 1) capture the semantic similarity between paraphrased lexical patterns in the same language and 2) capture the semantic similarity between similar patterns across languages. The second phase helps us to transfer semantic relations across languages. We then use the result of the clustering algorithm to measure the relational similarity between the source entity pair and each candidate target pair. We rank the candidate answer set using the relational similarity scores that we obtained by the cross-language relational similarity measuring algorithm to achieve a ranked result list for a query.

4. RELATION EXTRACTION AND MULTI-LINGUAL INDEXING METHOD

4.1. Multi-lingual entity pair and lexical pattern indexing

To efficiently perform cross-language latent relational search, we propose a method for building a multilingual index of entity pairs and lexical patterns.

We use a single-pass extraction method [Bollegala et al. 2010] to extract entity pairs and lexical patterns which represent the semantic relation between two entities in a pair from a given multi-lingual text corpus. To extract entity pairs from a document in the text corpus, we first identify the language of the document. Methods with high accuracy in linear time have been proposed in previous work on document language identification [Cavnar and Trenkle 1994; Takci and Sogukpinar 2004; McNamee 2005]. After identifying the language of a document, we use a Sentence Boundary Detector to split the document into sentences. Japanese language has an unambiguous sentence-ending marker, the Unicode character “u3002” (the “Ku-ten” symbol). However, there is not an unambiguous sentence boundary marker in English, as the period symbol (“.”) can be used for abbreviations (such as U.S.) and other purposes. In our algorithm, we use the Sentence Boundary Detector in the Stanford POS Tagger[3] to identify sentence boundaries in an English document. We then use the Stanford POS Tagger and Stanford Named Entity Recognizer[4] to split an English sentence into words and recognize named entities containing in the sentence. We use the MeCab POS Tagger[5] for Japanese word segmentation and named entity recognition. While Open IE systems require deep linguistic analysis such as dependency parsing [Banko et al. 2007] or co-reference resolution [Shinyama and Sekine 2006], the proposed system requires only shallow linguistic processing tools, namely, Sentence Boundary Detectors, POS Taggers and Named Entity Recognizers. This helps us to achieve short pre-processing time and scale the proposed method to large corpora.

After splitting a sentence into tokens (words) and recognizing named entities, we extract all named entity pairs that preserve the order of each entity in the sentence. For example, from the sentence “Microsoft acquired San Francisco based company Powerset for $100M.”, we extract three entity pairs (Microsoft, San Francisco), (San Francisco, Powerset) and (Microsoft, Powerset). Other combinations such as (Powerset, Microsoft) are not extracted because they do not preserve the order of the entities. We only extract entity pairs that preserve the order of each entity in a sentence because the relation between two entities in a pair might not be symmetric, and hence the relation in the reversed pair is not described in the sentence (e.g., “Microsoft acquired

[3]http://nlp.stanford.edu/software/tagger.shtml
[4]http://nlp.stanford.edu/software/CRF-NER.shtml
[5]http://mecab.sourceforge.net/

Table I. The lexical pattern extraction process for the entity pair (Sarkozy, France).

POS, NE tagged sentence	Sarkozy who is the current president of France was born in Budapest.
Stemming	Sarkozy who is the current presid of France wa born in Budapest.
Variable substitution	X who is the current presid of Y wa born in Budapest.
Window extraction	X who is the current presid of Y wa born in
Generating n-grams	X who is current presid of Y, X * is current presid of Y, X * presid * Y, X * is current presid * Y, X * current * Y, . . .

Powerset" is different from "Powerset acquired Microsoft"). It is important to note that, we extract all entity pairs (that preserve the order) from a sentence, irrespective of the relations that they might hold. Therefore, the proposed method does not require the relation types to be given in advance. We will filter-out noisy entity pairs, that include misspelling etc. by a frequency filter.

The most important task in latent relational search is calculating the relational similarity between two entity pairs. For example, we must recognize that the semantic relation between Tokyo and Japan is highly similar to that between Paris and France. To accomplish this task, one must represent the relation between two entities in an entity pair by some features. To capture the semantic relation between two entities, we extract lexical patterns from the contexts in which those entities co-occur. Using lexical patterns to represent semantic relations between two entities yields high precision in many tasks such as measuring the relational similarity between two entity pairs [Turney 2005; Bollegala et al. 2009a], extracting synonyms, antonyms [Turney 2008a], question answering [Ravichandran and Hovy 2002] and monolingual latent relational search [Duc et al. 2010]. In latent relational search, when two entity pairs actually hold similar semantic relations, we must make the relational similarity between them significantly higher than when they do not actually hold similar relations. That is, we want the two relationally similar entity pairs share many common lexical patterns so that the cosine between their feature vectors is large. Consequently, we use the extraction algorithm in our previous work [Duc et al. 2010] for lexical pattern extraction of two entities in a sentence, as shown in Table I.

The first step of the algorithm is to identify the positions of the two entities in the pair that we need to extract lexical patterns. In Table I, the two entities are Sarkozy and France. The second step is to stem all words other than named entities in the sentence. We use the Porter Stemmer[6] for stemming English words. For a Japanese word, the MeCab POS tagger provides the primitive form of the word so we can use this primitive form. We found that stemming is an important step to improve both recall and precision of latent relational search. We will compare the experimental results with and without stemming later in Section 6.5. Intuitively, stemming eliminates the differences between inflected forms of a word. Because different inflected forms of a word are considered as equal after stemming, various lexical patterns (that contains inflected forms of a same word) are considered as equal. This makes the association between a stemmed lexical pattern and the semantic relation that it represents stronger. Moreover, in monolingual latent relational search, the recall level will be improved after stemming, because stemming makes the probability that two entity pairs share some common lexical patterns (which are not the same before stemming) higher.

In the next step, we replace the two entities with their symbolic representations, to make the extracted lexical patterns independent from the entity pairs with which they co-occur. In Table I, the first entity is replaced with the variable X, the second entity is replaced with the variable Y. We then consider only a window of text surrounding the entity pair for extraction because we want prevent an explosion in the number

[6]http://nltk.googlecode.com/svn/trunk/doc/api/nltk.stem.porter-module.html

Table II. The pattern vs. entity pair matrix $\mathbf{M}$

Pattern vs. Entity pair	(Google, YouTube)	(グーグル, ユーチューブ)†	(Microsoft, Powerset)	(日産, 横浜)‡	(Yahoo, Sunnyvale)
X acquires Y	80	0	70	0	0
XがYを買収	6	75	2	0	0
X bought Y	67	0	60	0	0
X is headquartered in Y	0	0	0	0	105
XはYに本社を置く	0	0	0	90	0

Note: Each Japanese lexical pattern has the same meaning with the corresponding pattern above it.
† meaning (Google, YouTube)
‡ meaning (Nissan, Yokohama)

Table III. The pattern vs. entity pair matrix after translation and merging ($\mathbf{A}$).

Pattern vs. Entity pair	(Google, YouTube) = (グーグル, ユーチューブ)	(Microsoft, Powerset)	(日産, 横浜)‡	(Yahoo, Sunnyvale)
X acquires Y = XがYを買収	161	72	0	0
X bought Y	67	60	0	0
X is headquartered in Y = XはYに本社を置く	0	0	90	105

Note: Parallel patterns and entity pairs are marked with the equal (=) symbol.
‡ meaning (Nissan, Yokohama)

of extracted lexical patterns. Finally, we use our previously proposed lexical pattern extraction algorithm [Duc et al. 2010] to generate all n-grams[7] from the text window as lexical patterns. This algorithm allows extracting discontinuous segments from a text window. This makes the probability that two entity pairs have common lexical patterns higher because we do not need a complete match between the sequences in the gaps of the entity pairs. For example, consider the two sentences: “Obama is the 44th and current president of the U.S” and “Sarkozy is the current president of France”. If we use three discontinuous fragments “X”, “current president of” and “Y” (i.e., we generate the pattern “X $*$ current president of $*$ Y” (here, $*$ is the wildcard operator, meaning zero or more words)) then we have a common pattern between two pairs *(Obama, U.S)* and *(Sarkozy, France)* [Duc et al. 2010; 2011]. This will not only improve the relational similarity measure between two entity pairs in the same language but also alleviate the data sparseness problem while we perform cross-lingual lexical pattern clustering as described later.

At this step, we can create a matrix M of co-occurrences between lexical patterns and entity pairs. The value of each element M_{ij} of the matrix M can be the number of co-occurrences between the pattern p_i and the entity pair w_j, as shown in Table II. We can also use the point wise mutual information (PMI) between p_i and w_j as the value of M_{ij}. PMI has been successfully used for assessing the strength of the association between an entity pair and a lexical pattern in previous research [Pantel and Ravichandran 2004; Pantel and Pennacchiotti 2006]. We experimentally compare the method using numbers of co-occurrences and the method using PMIs in Section 6.6 to show that PMI actually improves the performance of the proposed latent relational search system. Following Pantel and Ravichandran [2004], we define the PMI between

[7]The n-grams are allowed to contain wildcards. Therefore, they should be “skip”-ngrams, but we use the term “n-gram” in this paper for convenience.

an entity pair w and a lexical pattern p as follows:

$$\mathrm{pmi}(p,w) = \left(\frac{\mathrm{f}(w,p)}{\mathrm{f}(w,p)+1} \times \frac{\min\left(\mathrm{f}(w),\mathrm{f}(p)\right)}{\min\left(\mathrm{f}(w),\mathrm{f}(p)\right)+1}\right) \log\left(\frac{\frac{\mathrm{f}(w,p)}{N}}{\frac{\mathrm{f}(w)}{N}\frac{\mathrm{f}(p)}{N}}\right) \tag{1}$$

where $\mathrm{f}(w,p)$ is the number of co-occurrences between w and p, whereas, N is the total number of co-occurrences between all entity pairs and all lexical patterns. PMI is known to has a bias towards infrequent entity pairs and lexical patterns. The first factor in Equation 1 is to prevent this bias [Pantel and Ravichandran 2004].

It is worth noting that the index that we build is a multilingual index. We do not differentiate between entity pairs that appear in Japanese documents and those in English documents. In many situations, Japanese also use an identical orthography to represent an entity name. For example, in many sentences, Japanese write the name of the Google Inc. as “Google”, instead of the Katakana expression グーグル. Our indexing method exploits this phenomenon to partially capture the semantic similarity between lexical patterns in different languages. As “グーグル” is also written as “Google”, we can extract some Japanese lexical patterns of the pair (Google, YouTube) and associate them with other English patterns of the same pair (see the column (Google, YouTube) in Table II). Because these patterns express the semantic relations of the same entity pair (Google, YouTube), there is a high probability that they are semantically similar. These associations can be used in the next step, cross-language lexical pattern clustering, which groups semantically similar patterns in different languages into a pattern cluster, as described in Section 5. However, if we fail to get some associations at this step, the proposed method still works because we will use machine translation to identify parallel patterns as shown in Section 4.2. Because in the proposed indexing method, lexical patterns that are associated with an entity pair might be in different languages, we call this indexing method “multi-lingual entity pair and lexical pattern indexing”.

4.2. Entity pair and lexical pattern translation

Although we can extract potentially similar lexical patterns in different languages with multi-lingual entity pair and lexical pattern indexing, we might also extract noisy patterns that do not describe the semantic relation between two entities in an entity pair. To make the lexical pattern clustering process more accurate, we propose an entity pair and lexical pattern translation method that can find parallel patterns and entity pairs with high precision.

To find parallel entity pairs, such as (Japan, Mt. Fuji) and (日本, 富士山), we first translate all entities that are extracted. We use a Statistical Machine Translation (SMT) system[8] to translate or transliterate an entity pair from a source language into a target language. An SMT system is able to find the transliteration of an entity because it can find the alignment between the source entity and the target entity in its corpus. For example, using the SMT system, we can transliterate the entity “Google” in English into “グーグル” in Japanese and vice versa. Similarly, we can translate the entity “Japan” in English into “日本” in Japanese. After translating all entities, we can combine the translation results to find the translation of an entity pair. If an entity pair (A,B) is translated into (A',B') by the SMT system, we look up the target entity pair (A',B') in the index to verify the translation result. If we can actually find the pair (A',B') in the index, then we record that the entity pair (A',B') is a parallel entity pair of the entity pair (A,B). We then merge two columns of the matrix M that

[8]We experimentally use Google Translate and Moses

correspond to these entity pairs. The resulting column corresponds to both (A, B) and (A', B') (i.e., two entity pairs are now assigned a same column ID). The effect of merging parallel entity pairs is that two similar lexical patterns will have higher similarity. For example, if we merge the first column and the second column of the matrix M in Table II then the cosine similarity between the second row (XがYを買収, meaning "X acquires Y") and the third row (X bought Y) is increased.

Parallel lexical patterns play an important role in the process of transferring semantic relation across two languages. To find the parallel pattern of a lexical pattern, we use the information concerning POS tags (named entity types) of the two variables X and Y in the pattern to replace each variable with a well-known entity. For example, if the lexical pattern is from an English document and the tag of X is ORGANIZATION then we replace X with the entity "Microsoft". We remember that the transliteration of "Microsoft" in Japanese is "*Maikurosofuto*" (the lexical pattern must actually be written in Japanese characters (Katakana) but for convenience, we write the pattern using the English alphabet) to use later. We do not translate the entity markers X and Y in lexical patterns, but replace X and Y with well-known entities to let the SMT system easily find the corresponding entities in the target language. After substituting two variables X and Y with well-known entities A and B, if the pattern does not contain any wildcard operator "$*$", we input the pattern into the SMT system to get the translation result. We only translate patterns without any wildcard operator because the result of translating a non-complete pattern is not reliable. Moreover, we need to limit the number of patterns to translate to reduce the pre-processing time. Suppose that we already know that A is translated into A' and B is translated into B' in the target language (we remember the translation of an entity before substituting it). Then we search for the string A' and B' in the translation result. If we can find both of these strings in the translation result, we replace them with the variables X and Y to obtain a translated lexical pattern, otherwise we assume that the translation process failed and we omit the result. Lexical patterns in the index are usually written by humans, not generated by machine translation systems. Consequently, if we can find the translated lexical pattern in the index, then there is a high probability that the SMT system has produced a correct result (which is used by humans). Therefore, if we can find the translated pattern in our index, we record that the two patterns are parallel and merge two rows in the matrix M that correspond to these lexical patterns, as shown in the last row of Table III. From the matrix M in Table II, after merging both parallel entity pairs and lexical patterns, we obtain a matrix **A** as shown in Table III.

We denote $\mathbb{P}(w)$ as the set of all lexical patterns with which the entity pair w co-occurs:

$$\mathbb{P}(w) = \{p_1, p_2, \ldots, p_\ell\} \tag{2}$$

Moreover, to efficiently retrieve candidate entity pairs that co-occur with lexical patterns of the source pair, we also store an inverted index from a lexical pattern to the set of entity pairs that the pattern co-occurs with [Duc et al. 2010]. We denote $\mathbb{W}(p)$ as the set of all entity pairs with which the pattern p co-occurs:

$$\mathbb{W}(p) = \{w_1, w_2, \ldots, w_t\} \tag{3}$$

If the matrix **A** has size $m \times n$, then the pattern frequency (or PMI) vector $\mathbf{\Psi}(w_j)$ of the entity pair w_j is defined as the jth column of **A**:

$$\mathbf{\Psi}(w_j) = (\mathbf{A}_{1j}, \mathbf{A}_{2j}, \ldots, \mathbf{A}_{mj})^{\mathrm{T}} \tag{4}$$

Similarly, the entity pair frequency (or PMI) vector $\mathbf{\Phi}(p_i)$ of the pattern p_i is defined as the transpose of the ith row of **A**:

$$\mathbf{\Phi}(p_i) = (\mathbf{A}_{i1}, \mathbf{A}_{i2}, \ldots, \mathbf{A}_{in})^{\mathrm{T}} \tag{5}$$

The similarity between two lexical patterns p_i and p_j is defined as the cosine of $\mathbf{\Phi}(p_i)$ and $\mathbf{\Phi}(p_j)$, as frequently used in the Vector Space Model (VSM):

$$\mathrm{sim}_{\mathrm{VSM}}(p_i, p_j) = \mathrm{cosine}(\mathbf{\Phi}(p_i), \mathbf{\Phi}(p_j)) \tag{6}$$

5. MEASURING LATENT RELATIONAL SIMILARITY ACROSS LANGUAGES

5.1. Multi-lingual Latent Relational Analysis

The dimensions of the row and column vectors of the matrix $\mathbf{A}$ are very large because they are the numbers of different entity pairs and lexical patterns. Moreover, the extraction algorithm in Section 4.1 might extract several noisy co-occurrences between entity pairs and lexical patterns. In addition, many lexical patterns actually have the same meaning but they have different surface forms because there are several ways to state a semantic relation in a natural language (e.g., “X is the CEO of Y” and “X, the CEO of Y”). Therefore, it is difficult to precisely measure the semantic similarity between two entity pairs or two lexical patterns if we directly use column vectors or row vectors of the matrix $\mathbf{A}$. Latent Semantic Analysis [Landauer and Dumais 1997] and Latent Relational Analysis [Turney 2005] have successfully used Singular Value Decomposition (SVD) to reduce the number of dimensions of these vectors and to compress semantically similar dimensions into one. The idea of multilingual term-document indexing has been exploited in multi-lingual Latent Semantic Analysis [Dumais et al. 1997] and cross-language sentiment classification [Prettenhofer and Stein 2010]. Therefore, we propose *multi-lingual Latent Relational Analysis*, to measure the similarity between entity pairs and lexical patterns across languages.

For the $m \times n$ matrix $\mathbf{A}$, SVD decomposes $\mathbf{A}$ into three matrices $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{7}$$

where $\mathbf{U}$ is an $m \times m$ matrix, $\mathbf{V}$ is an $n \times n$ matrix in column orthonormal form and $\mathbf{\Sigma}$ is a rectangular diagonal $m \times n$ matrix of *singular values* [Turney 2005; Manning et al. 2008]. We can re-arrange the column vectors of $\mathbf{U}$ and $\mathbf{V}$ such that the elements in the main diagonal of $\mathbf{\Sigma}$, which contains singular values, are sorted from large to small (i.e., the top left element has the largest value). The rank of $\mathbf{A}$ and $\mathbf{\Sigma}$ are equal, $\mathrm{rank}(\mathbf{A}) = \mathrm{rank}(\mathbf{\Sigma}) = r$. If $\mathbf{\Sigma}_k$ ($k < r$) is the diagonal matrix created from the top k singular values from $\mathbf{\Sigma}$ and $\mathbf{U}_k$, $\mathbf{V}_k$ are the matrices formed by selecting the first k columns of $\mathbf{U}$ and $\mathbf{V}$, then $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$ is the matrix of rank k that best approximates the matrix $\mathbf{A}$ (i.e., the Frobenius norm of $(\mathbf{A} - \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T)$ is minimized) [Manning et al. 2008].

We can directly solve the problem of measuring the relational similarity between two entity pairs (corresponding to two columns in $\mathbf{A}$) by calculating the cosine similarity between two corresponding columns in the low rank matrix $\mathbf{\Sigma}_k\mathbf{V}_k^T$, as in LRA [Turney 2005]. LRA compresses many semantically similar lexical patterns into one dimension. Therefore, it yields the most precise result in measuring the relational similarity for monolingual case [Bollegala et al. 2009a]. However, in multi-lingual LRA, the number of common lexical patterns between two entity pairs in two different languages is not large. This implies that we might not achieve a good performance if we use only LRA. Therefore, we propose a novel two-phase lexical pattern clustering algorithm (to be described in the next section) to precisely group semantically similar lexical patterns (across languages) into a cluster. The clustering algorithm helps us to transfer a semantic relation of an entity pair across two languages because lexical patterns of different languages in the same cluster are considered as expressing the same semantic relation. We then use the result of the clustering algorithm to measure the relational similarity between two entity pairs across languages. We use LRA for calculating the semantic similarity between two lexical patterns (corresponding to two rows in the matrix $\mathbf{A}$). The cosine similarity between two lexical patterns p_i and p_j in the dimensionally reduced space is the cosine of two corresponding rows i and j of the low rank matrix $\mathbf{H}$ defined below [Turney 2005].

$$\mathbf{H} = \mathbf{U}_k \mathbf{\Sigma}_k \tag{8}$$

(because $\mathbf{U}_k$ has the size of $m \times k$, $\mathbf{H}$ is also an $m \times k$ matrix, which represents m lexical patterns in the reduced space of k dimensions, $k < r, n$). We denote $\mathbf{\Phi_{LRA}}(p_i)$ as the transpose of the row vector corresponding to the pattern p_i:

$$\mathbf{\Phi_{LRA}}(p_i) = (\mathbf{H}_{i1}, \mathbf{H}_{i2}, \ldots, \mathbf{H}_{ik})^T \tag{9}$$

The cosine similarity between two patterns p_i and p_j in the dimensionally reduced space is then defined as:

$$\mathrm{sim}_{\mathrm{LRA}}(p_i, p_j) = \mathrm{cosine}(\mathbf{\Phi_{LRA}}(p_i), \mathbf{\Phi_{LRA}}(p_j)) \tag{10}$$

5.2. Lexical pattern clustering algorithm

There might be several paraphrases that describe the same semantic relation in a language. For example, the lexical pattern “X acquired Y” is semantically similar to the pattern “X bought Y”, but the surface forms of the patterns are completely different. Likewise, the lexical pattern “*X ga Y wo baishu shita*” (meaning “X acquired Y”) in Japanese is also semantically similar to the pattern “X purchased Y” in English. Therefore, the number of parallel patterns that we can find by translation in previous step is often not large enough for retrieving a candidate set for every query. Moreover, because of this sparseness, the relational similarity between two entity pairs in different languages will be too small that we can not differentiate between two pairs that share only noisy patterns and two pairs that actually hold similar semantic relations. This is the main reason that LRA might not work well in cross-language latent relational search. Previous research [Duc et al. 2010; Bollegala et al. 2009a] has shown that we can improve the precision while measuring the relational similarity between two entity pairs by clustering semantically similar patterns into clusters and consider all patterns in a cluster as equal. Consequently, we propose a two-phase lexical pattern clustering algorithm to 1) capture the semantic similarity between paraphrased lexical patterns in the same language and 2) capture the semantic similarity between similar or translated patterns across languages.

The clustering algorithm is shown in Algorithm 1. In the first phase of our clustering algorithm, we want to capture the semantic similarity between paraphrased lexical patterns in the same language. Therefore, we use the lexical pattern clustering algorithm by Bollegala et al. [2009a; 2009b] in this phase. First, it sorts the pattern set by descending order of frequency to process high frequency patterns first [Bollegala et al. 2009a]. For each pattern, the algorithm finds the cluster whose centroid has maximum similarity with the pattern (line 5). We try two methods for calculating the similarity between two lexical patterns, $\mathrm{sim}_{\mathrm{LRA}}$ or $\mathrm{sim}_{\mathrm{VSM}}$ as defined in Equation 10 and Equation 6. When we use $\mathrm{sim}_{\mathrm{VSM}}$, we do not need to perform LRA (and hence SVD) so the pre-processing time is fast. We denote the method that uses $\mathrm{sim}_{\mathrm{VSM}}$ for calculating the similarity as **HLPC** (hybrid lexical pattern clustering) and the method that uses $\mathrm{sim}_{\mathrm{LRA}}$ as **HLPC+LRA**. If the similarity is above a pattern clustering similarity threshold θ_1 then the pattern is added to the cluster, otherwise, the pattern forms a new singleton cluster itself (line 10–15). Therefore, this algorithm is a hard clustering algorithm (i.e., each pattern can be in only one cluster). Although there are many clustering algorithms which can be used in the first phase (e.g., the SLINK [Sibson 1973], CLINK [Defays 1977] algorithms) we choose the lexical patten clustering algorithm by Bollegala et al. [2009a; 2009b] because the algorithm has the time complexity

ALGORITHM 1: Hybrid Lexical Pattern Clustering (HLPC) of lexical patterns

```
Input: pattern set $\wp$, threshold $\theta_1 > \theta_2 > 0$
Output: cluster set $\mathbf{K}$
 1: $\mathbf{K} \leftarrow \{\}$
 2: /* First phase */
 3: sort( $\wp$ )
 4: for pattern $p \in \wp$ do
 5:     $maxClus \leftarrow \mathrm{argmax}_{c\in\mathbf{K}}\mathrm{sim}(p, \mathrm{centroid}(c))$
 6:     $maxSim \leftarrow -1$
 7:     if $maxClus \neq$ NULL then
 8:         $maxSim \leftarrow \mathrm{sim}(p, \mathrm{centroid}(maxClus))$
 9:     end if
10:     if $maxSim \geq \theta_1$ then
11:         $maxClus.\mathrm{append}(p)$
12:     else
13:         $newClus \leftarrow \{p\}$
14:         $\mathbf{K} \leftarrow \mathbf{K} \cup \{newClus\}$
15:     end if
16: end for
17: /* Second phase */
18: for pattern $p \in \wp$ do
19:     if hasParallel($p$) then
20:         for cluster $c \in \mathbf{K}$ do
21:             if $\mathrm{sim}(p, \mathrm{centroid}(c)) \geq \theta_2$ then
22:                 $c.\mathrm{append}(p)$
23:                 $c.\mathrm{append}(\mathrm{paralellOf}(p))$
24:             end if
25:         end for
26:     end if
27: end for
28: return $\mathbf{K}$
```

of $O(m\log m + m|\mathbf{K}|)$, where m is the number of input lexical patterns, and $|\mathbf{K}|$ is the number of output clusters. Normally, $|\mathbf{K}| << m$, therefore, the amortized time complexity of the algorithm is much smaller than $O(m^2)$ or $O(m^3)$, which are required by hierarchical clustering algorithms. This allows us to perform the clustering process in high speed to reduce pre-processing time.

We need to set θ_1 to a high value to reduce the number of large clusters that might express many different semantic relations. However, we observe that when two semantically similar lexical patterns p and q are in the same language, their semantic similarity is normally higher than when they are in two different languages (e.g., p is in Japanese and q is in English). This happens even when the pattern p (in Japanese) has a parallel pattern p' in English. This is because patterns that have parallel partners are associated with a large number of entity pairs in different languages, as the result of multi-lingual entity indexing, which merges entity pairs of two parallel patterns. For example, after merging, the merged pattern $\{p, p'\}$ now co-occurs with both English and Japanese entity pairs, whereas, q co-occurs with only English entity pairs. This implies that the cosine similarity between two row vectors corresponding to these patterns in $\mathbf{A}$ is low. Therefore, in the second phase, we use a soft clustering algorithm with a lower pattern clustering similarity threshold θ_2 to associate parallel patterns to the pattern clusters that we obtained in the first phase. That is, we consider only patterns that have some parallel partners (e.g., p and p' in the above example) for clustering in the second phase (line 19 of Algorithm 1), and we allow each of these patterns to be associated with many pattern clusters. If the similarity between a pattern that has some parallel partners and the centroid of a pattern cluster is above θ_2, we add the pattern and its parallel partners to the cluster (line 20–24). We need to associate as many parallel patterns as possible to these clusters to increase the recall as well as the precision for cross-language queries. A soft clustering algorithm in this phase accomplishes this goal, because a pattern and its parallel partners are allowed to appear in multiple clusters.

The second phase is an important step in our algorithm because it captures the semantic similarity between patterns across languages. Even when two patterns in two different languages share only a small number of entity pairs so that they failed to be in a cluster in the first phase (because the similarity is much lower than θ_1), they can be grouped in a same cluster in the second phase, because of the low similarity threshold value. Therefore, the second phase is mainly to capture the similarity between paraphrased lexical patterns across languages.

5.3. Retrieving and ranking answers

5.3.1. Retrieving candidate answers. Given the query $\{(A, B), (C, ?)\}$, we denote the source pair as s ($s = (A, B)$) and a candidate target pair as c ($c = (C, X)$). To retrieve a candidate answer set, we add all patterns with which the source pair co-occurs (i.e., the set $\mathbb{P}(s)$) to a new lexical pattern set $\mathbb{G}(s)$, which represents the set of potentially similar lexical pattern between the source pair and the target pair. We then add all patterns that are in the same pattern cluster with at least one pattern in $\mathbb{P}(s)$ to $\mathbb{G}(s)$. Adding patterns in the same cluster with a pattern in $\mathbb{P}(s)$ is an important step for processing a cross-language query, especially when the source pair only co-occurs with patterns in the source language (the language of the source pair). This is because a pattern $p \in \mathbb{P}(s)$ might have some parallel patterns in the target language (the language of the key entity C) or might be in the same cluster with some semantically similar patterns in the target language (in the pattern clustering step, we added all parallel patterns of p into a cluster that contains the pattern p). For each lexical pattern p in $\mathbb{G}(s)$, we enumerate all entity pairs that co-occur with p (i.e., the set $\mathbb{W}(p)$) and append all entity pairs of the form (C, X) into the candidate set. By this method we can ensure that each candidate pair c has at least one lexical pattern in the same cluster with some lexical patterns of the source pair s. This condition also helps to limit the number of candidate pairs and speed up the candidate retrieving process.

5.3.2. Ranking by relational similarity score. To rank the result list, we calculate the relational similarity between two entity pairs s ($s = (A, B)$) and c ($c = (C, X)$) using their lexical pattern frequency (or PMI) vectors $\mathbf{\Psi}(s)$ and $\mathbf{\Psi}(c)$. We define the relational similarity $\mathrm{relsim(s, c)}$ between s and c using a modified version of cosine similarity of their pattern frequency (PMI) vectors by considering two patterns that are in the same cluster as equal. That is, we compress all lexical patterns in a cluster into one dimension. To perform this cosine calculation, for each entity pair w, we identify the elements in $\mathbf{\Psi}(w)$ whose corresponding lexical patterns are in the same cluster. We then take the sum of these elements (as the feature vector value representing the pattern cluster). Finally, we remove all these elements from $\mathbf{\Psi}(w)$, add a new dimension that represents the lexical pattern cluster and set the value for this dimension as the above sum.

We sort the candidate list in order of the relational similarity score in the descending order to obtain the final result list. For each shared pattern (common between s and c) or each paraphrased pattern pair (patterns that are in the same cluster), we retrieve the sentences in which the pattern co-occurs with the source pair s and the candidate target pair c to output a list of sentence as evidences (supporting sentences) for each result.

Table IV. Relation types for evaluation (italic entities are actually in Japanese)

Relation type	Exists between	Example
BIRTHPLACE	A person and his place of birth	(Franz Kafka, Prague), (*Hamasaki Ayumi*, *Fukuoka*), . . .
HEADQUARTERS	A company and its headquarters	(Google, Mountain View), (*Toyota*, *Aichi*) . . .
CEO	A CEO and his company	(Eric Schmidt, Google), (*Toyoda Akio*, *Toyota*), . . .
ACQUISITION	Two companies	(Google, Youtube), (*Panasonikku*, *Sanyo*), . . .
PRESIDENT	A person and the country whose president is this person	(Barack Obama, U.S), (*Sarukozi, Furansu*) . . .
PRIMEMINISTER	A person and the country whose PM is this person	(David Cameron, U.K), (*Kan Naoto, Nihon*) . . .
CAPITAL	A city and the country whose the capital is this city	(Paris, France), (*Tokyo, Nihon*) . . .
SATELLITE	A moon and the planet which the moon orbits	(Ganymede, Jupiter), (*Oberon, Tennosei*)

It is worth noting that, a source entity pair with more than one semantic relationships might yield several appropriate answers. For example, the query {(Google, Larry Page), (Microsoft, ?)} might have multiple answers (e.g., Steve Ballmer for the CEO relation; Bill Gates and Paul Allen for the FOUNDER relation). Those three entities can be considered as correct answers in a certain level. The proposed system tends to rank the answer corresponding to the most salient relation in the corpus (e.g., the CEO relation) at the top of the result list.

6. EVALUATION

In this section, we describe several experiments to evaluate the proposed method. We first determine appropriate values of the clustering similarity thresholds (θ_1 and θ_2) in the clustering algorithm by evaluating the proposed system with a corpus for parameter tuning (the *train dataset*). Using another corpus as the *test dataset*[9], we compare the performance of the proposed method with three baseline methods. We then report the best performance that we achieved with the test dataset in Section 6.4. To verify the effectiveness of the proposed lexical pattern extraction algorithm, we compare the performance of the proposed method with the method based on an existing lexical pattern extraction algorithm in Section 6.5. Moreover, in Section 6.6, we evaluate the effect of using different values (frequency and PMI) as elements of feature vectors. We then compare the performance of the proposed system with that of existing monolingual latent relational search systems to show that the proposed cross-language system achieves a reasonable result compared to monolingual search systems. We also compare the performance of the proposed method when using two different Statistical Machine Translation systems to show that we can achieve a reasonable performance with different SMT systems in Section 6.8. Finally, we compare the proposed method with other entity ranking methods by evaluating the system with questions in the INEX 2008 Entity Ranking track [Demartini et al. 2008]. We report the result of this experiment in Section 6.9.

6.1. Relation types and datasets

The proposed method is extensively experimented with eight relation types as shown in Table IV. These relation types are frequently used in previous research to evaluate relational similarity measuring algorithm [Bollegala et al. 2009a], monolingual latent

[9]The dataset (including the corpus and the query sets) is available at http://www.miv.t.u-tokyo.ac.jp/duc/milresh/
A demo version of the search engine is also available at the above page.

relational search engines [Kato et al. 2009; Duc et al. 2010] or relation extraction systems [Banko et al. 2007; Bunescu and Mooney].

To create a corpus that includes Web pages concerning these semantic relations, we first prepare a set of seed entity pairs. Each seed pair in this set holds a specific relation such as the BIRTHPLACE relation as shown in Table IV. The number of seed entity pairs for each relation is 20 in English and 10 in Japanese. Using two entities in each seed pair and some keywords that describe the relation of the two entities, we formulate some queries for retrieving relevant documents. For example, from the entity pair (Google, YouTube), we formulate the following queries to retrieve documents related to the acquisition of YouTube by Google: "Google buy YouTube", "Google * buy * YouTube", "Google * * buy * * YouTube", "Google bought YouTube", "Google * bought * YouTube", "Google * * bought * * YouTube", "Google acquired YouTube", "Google purchased YouTube", "acquisition * YouTube * Google", ... These queries would retrieve many documents that contain several different paraphrases of the acquisition relation existing between Google and YouTube. Note that for simplicity in the experiments, we use these queries to retrieve documents that are strongly related to the relation types in Table IV, but the proposed system can be run on any corpus, with different relation types. We will present the performance of the proposed search engine with the entire Wikipedia data dump (which include numerous relation types) later in section 6.9. Using the query set, we query Google[10] to retrieve the Top 100 URLs that are relevant to each query. From the URL set, we crawl the HTML page at each URL. After the crawling process, we obtain a *train dataset* and a *test dataset*. The train dataset has size of 1.8GB, of which about 60% are English web pages and 40% are Japanese web pages. The test dataset has size of 1.6GB, about one-half of which are English web pages, the rest are Japanese web pages. These sets of web pages contain a large number of entities and relations of many types (not only those in Table IV because a web page might describe many entities and relations and might contain non-related information such as text from advertisements). We then run the pre-processing phases (including entity and pattern extraction; translation and pattern clustering) on the generated text corpus to build an index for our system.

We create 16 query sets to evaluate the system, eight query sets are English-to-Japanese query sets, the other eight sets are Japanese-to-English. Each query set corresponds to a relation type in Table IV and contains 50 queries. A set of 50 queries is considered to be sufficient to evaluate the performance of an IR system [Manning et al. 2008]. Each query has only one correct answer. For example, we create the query {(?, YouTube), (*Panasonikku*, *Sanyo*)} for the ACQUISITION relation and {(Ganymede, Jupiter), (*Oberon*, ?)} for the SATELLITE relation (entities that are written in *italic* are actually written in the Japanese writing system, we use the English alphabet here for convenience). The criteria for evaluation is the Mean Reciprocal Rank (MRR) of each query set. MRR reflects both recall and precision of a search engine and is frequently used for evaluation of high accuracy retrieval techniques [Radev et al. 2002; Shah and Croft 2004; Najork et al. 2007; Kato et al. 2009].

For convenience, if we use $\mathrm{sim}_{\mathrm{VSM}}$ (Equation 6) in Algorithm 1 to calculate the similarity between two lexical patterns then we call the method as **HLPC** (hybrid lexical pattern clustering). If we use $\mathrm{sim}_{\mathrm{LRA}}$ (Equation 10) then we call the method as **HLPC+LRA**.

6.2. Parameter tuning

We run the proposed extraction algorithm on the train dataset to build an index for the system. The resulting index contains 5,241,627 lexical patterns and 236,923 en-

[10]http://www.google.com

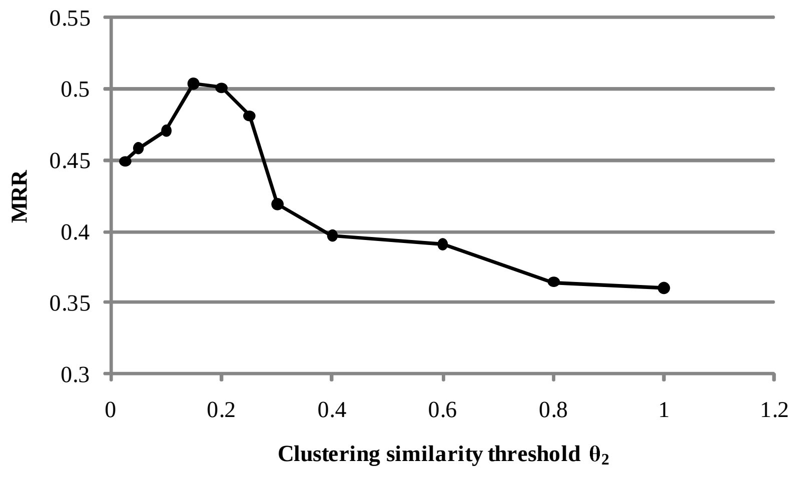


Fig. 4. The relation between MRR and θ_2 of the method **HLPC** (at $\theta_1 = 0.4$)

tity pairs. Bollegala et al. [2010] suggest that we must filter very rare patterns (e.g., patterns that appear only once). This is because rare patterns are normally noisy patterns, which frequently contain strange symbols or misspellings. Therefore, we only use 1,878,463 patterns that appear more than two times to build the matrix **A** for clustering. Of which, only 149,835 patterns that do not contain the wildcard character ("$*$") are considered for translation. We use Google Translate[11] to translate these lexical patterns from English into Japanese and vice versa.

After the pattern translation process, we found 6812 patterns that have parallel patterns. Therefore, the ratio of reliable translation is only 4.55% and only 0.13% of the total number of patterns are translated. Only 4862 entity pairs (2.05%) are translated (i.e., have parallel entity pairs). These very small ratios indicate that if we had relied only on machine translation, then we would not be able to achieve a reasonable recall level. We use SVDLIBC[12] to perform Singular Value Decomposition (SVD) of the matrix **A**. We set the value of k (the number of singular values to be calculated in Section 5.1) to 300, as suggested by Dumais et al. [1997] and Turney [2005]. Because the time complexity of the SVD operation is $O((m+n)k^2)$ ($m \times n$ is the size of the matrix **A**) [Golub and Kahan 1965], we can perform the operation in several hours on an Intel Core i7, 3GHz, 24GB RAM machine. The time complexity of the first phase of the clustering algorithm is $O(mlog(m) + m|\mathbf{K}|)$ ($mlog(m)$ is the cost of the sorting step) where m is the number of lexical patterns (i.e., the number of rows of the matrix **A**) and $|\mathbf{K}|$ is the number of clusters. The time complexity of the second phase is $O(m|\mathbf{K}|)$ (but in practice the number of parallel patterns is much smaller than m so the time complexity is much smaller). Therefore, the total time complexity of the clustering algorithm is $O(mlog(m) + m|\mathbf{K}|)$. We can perform all pre-processing steps in less than one day on the same machine.

In our previous research on monolingual latent relational search [Duc et al. 2010], we perform only the first phase of the clustering algorithm described in Section 5.2 to capture the semantic similarity between paraphrased lexical patterns in the same

[11]http://translate.google.com
[12]http://tedlab.mit.edu/~dr/SVDLIBC/

language (i.e, no second phase and no LRA). In this setting, we found that the appropriate value for the pattern clustering similarity threshold is 0.4 [Duc et al. 2010]. Consequently, we set the value of the parameter θ_1 in our clustering algorithm to 0.4 if we use the method **HLPC** (the method that does not require LRA, only $\mathrm{sim}_{\mathrm{VSM}}$). We then vary the parameter θ_2 to determine the best value. We use four relation types in the first four rows of Table IV for training purpose. In the test phase, we will use all of the eight relation types in Table IV, to avoid the bias to those relations that are optimized in the training phase. Therefore, we use only eight query sets (four English-to-Japanese and four Japanese-to-English query sets) that correspond to the first four relation types in Table IV in this experiment and evaluate the average MRR value of these query sets. Fig. 4 shows the experiment result. At $\theta_2 = 0.15$, we obtain the best value of MRR. Consequently, in all following experiments, we set θ_2 to 0.15.

For two semantically similar lexical patterns p and q, $\mathrm{sim}_{\mathrm{LRA}}(p, q)$ is often larger than $\mathrm{sim}_{\mathrm{VSM}}(p, q)$ because LRA compresses semantically similar dimensions into one and reduces noisy dimensions. Therefore, we can not assume that the appropriate value for θ_1 in the **HLPC** method (which was set to 0.4) is also appropriate for the **HLPC+LRA** method.

We vary the value of θ_1 in the **HLPC+LRA** method to find an appropriate value. At $\theta_1 = 0.8$ we achieve the best performance for the **HLPC+LRA** method. This value is much larger than the appropriate value in the **HLPC** method (which was 0.4). Therefore, in all experiments related to the **HLPC+LRA** method, we set θ_1 to 0.8.

6.3. Comparison with baseline methods for Cross-lingual queries

We compare the performance of the two proposed methods (**HLPC** and **HLPC+LRA**) with that of baseline methods using the test dataset (the 1.6GB corpus). Three baseline methods for comparison are as follows.

— **LPC**: This method uses only the first phase of the lexical pattern clustering algorithm (the first phase in Algorithm 1). This is the method in [Duc et al. 2010] for monolingual latent relational search (however, **LPC** finds parallel patterns by lexical pattern translation).
— **Trans+LPC**: This method first translates all documents in the corpus into English. Then it translates all entities in the query into English and performs monolingual latent relational search, as in [Duc et al. 2010].
— **LRA**: This method does not use clustering, instead it directly calculates the cosine similarity between two entity pairs using the dimensionally reduced vector space after LRA (i.e., the matrix $\mathbf{\Sigma}_k \mathbf{V}_k^T$).

The comparison is performed on eight query sets (four English-to-Japanese and four Japanese-to-English query sets) similar to those in the previous section. The average MRR of these eight query sets is shown in Fig. 5. The proposed methods (**HLPC** and **HLPC+LRA**) outperform the **LPC** method by a wide margin. We verify this difference by using a paired t-test in which the samples are 400 queries derived from eight query sets above (each query set contains 50 queries). For a query q, we use the value of $\frac{1}{r_q}$ in the t-test, where r_q is the rank of the first correct answer (therefore, the mean of these values is the MRR of the query set). Because we use the same 400 queries to evaluate the five methods, we can use a paired t-test in this case. A set size of 400 is large enough for a paired t-test to verify the difference between two means of two sample sets. We found that the difference between the performance of the method **LPC** and **HLPC** is statistically significant (at the significance level 0.01) under the paired t-test. This proves that the proposed second-phase clustering (i.e., soft clustering of parallel

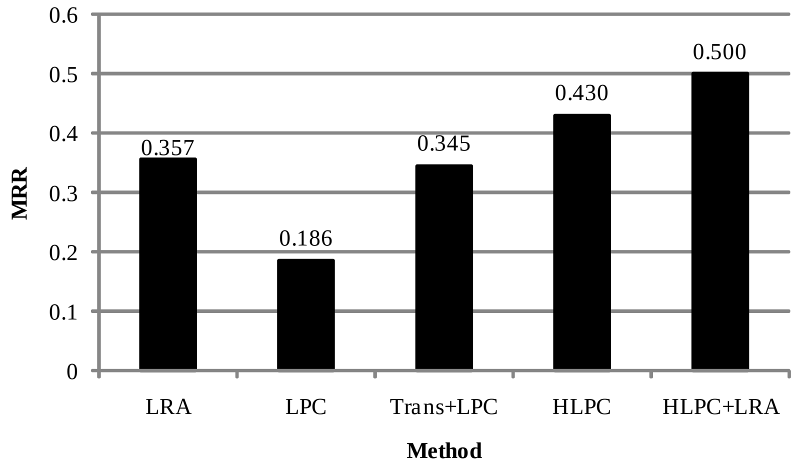


Fig. 5. Comparison between the MRR of the proposed methods (**HLPC** and **HLPC+LRA**) with baseline methods

patterns) successfully captures the semantic similarity between paraphrased lexical pattern across languages.

We also found that the difference between the performance of **Trans+LPC** and **LRA** is not statistically significant under the paired t-test (the value of the t-Statistic is 0.466, corresponding to an one-tail p-value of 0.32, which is much larger than the significance level of 0.05). This indicates that document translation combined with the clustering algorithm proposed by Bollegala et al. [2009b] can achieve a comparable performance to multi-lingual LRA [Turney 2005]. The values of MRR of the two proposed methods, **HLPC** and **HLPC+LRA** are 0.430 and 0.500, respectively. Under the same paired t-test settings, **HLPC** and **HLPC+LRA** significantly outperform **LRA** (the p-value for the **HLPC** vs. **LRA** test is 0.003, whereas, the p-value for the **HLPC+LRA** vs. **LRA** test is 2.5×10^{-8}, which indicates that the differences are statistically significant at the significance level $\alpha = 0.01$). Moreover, **HLPC** and **HLPC+LRA** significantly outperform **Trans+LPC** (at the significance level of 0.01). Finally, **HLPC+LRA** significantly outperforms **HLPC** (the one-tail p-value is 0.0001). This demonstrates that LRA significantly improves the performance of the system (with the cost of the SVD operation on a large matrix).

6.4. Performance on each query set

We use the test dataset (1.6GB corpus) to evaluate the performance of the system on 16 query sets (of eight relation types as describe in Table IV; eight of them are English-to-Japanese query sets, the rest are Japanese-to-English). The evaluation result is shown in Fig. 6. We achieve high MRR on the CAPITAL, PRIMEMINISTER and SATELLITE relation. This is because lexical patterns that represent the CAPITAL, PRIMEMINISTER and SATELLITE relation are simple for translation from English into Japanese and vice versa. Especially, the MRR for the SATELLITE relation is very high because the number of entities in this relation is not large and all entities in this relation are very popular. Therefore, for globally mentioned relations (i.e., relations that are popular across many countries and languages such as the SATELLITE relation), the proposed method achieves high performance. For the CEO relation, although

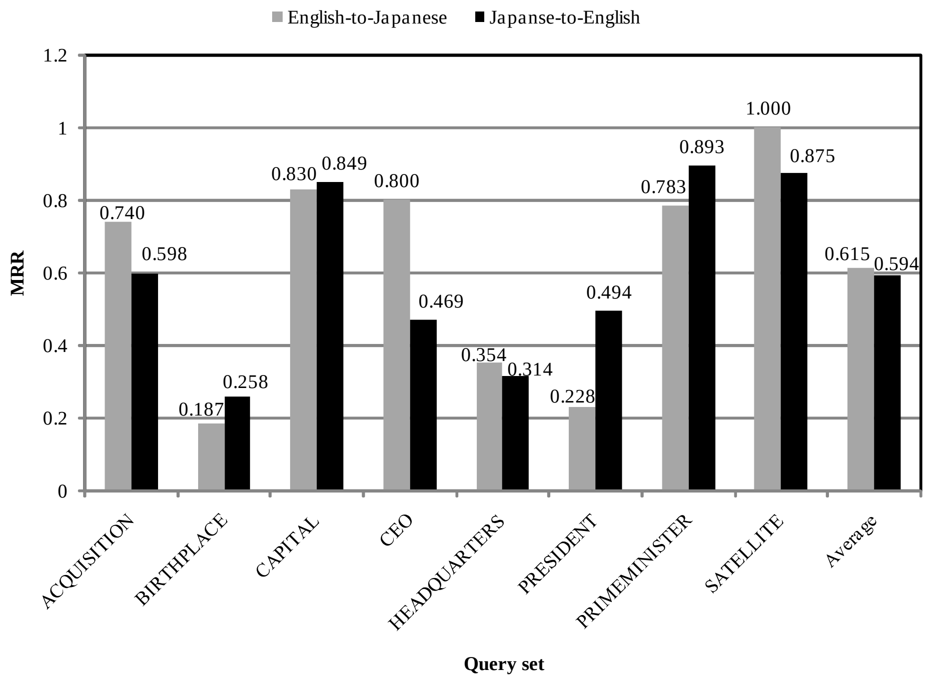


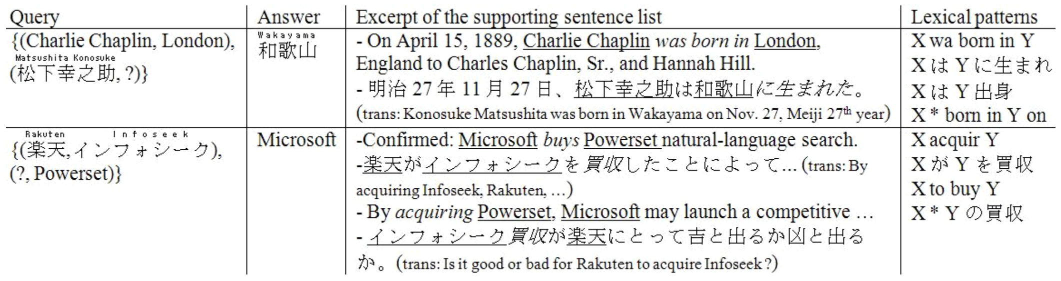
Fig. 6. Performance of the proposed method (**HLPC+LRA**) on cross-language latent relational search queries of eight relation types

the entities might not be popular but in Japanese sometime the phrase for describing the CEO relation is “CEO”, which is identical with the phrase in English. Therefore, we can achieve high performance on the CEO relation in English-to-Japanese query set because we can easily retrieve candidates which have the common lexical pattern “CEO”. However the MRR of the Japanese-to-English CEO query set is not high because there are many paraphrases to describe the CEO relation in Japanese (such as “*X ga Y no daihyo torishimari yaku*”, “*X ga Y no shachou*”, . . .). Moreover, the relation between a CEO and a company might be a local information (i.e., only mentioned in a specific language), if the company is not famous. Similarly, the BIRTHPLACE and HEADQUARTERS relations have many different lexical patterns in Japanese and it is very difficult to exactly translate these patterns into English. More importantly, these relations might be local information and it is difficult to recognize the relational similarity between these relations across languages. Therefore, the performance on these query sets is low.

On average, we achieve an MRR of 0.605 for 16 query sets of eight relation types, as described above. Fig. 7 shows some example queries and results that the search engine retrieved. The time for processing a query of the proposed method is less than 10 seconds, which is acceptable for real-world search sessions. This proves that the proposed indexing method is effective for answering queries in high speed.

6.5. The effectiveness of the proposed extraction algorithm

As described in Section 4.1, in our previous work [Duc et al. 2010;2011], we proposed to make two modifications to the baseline pattern extraction algorithms [Turney 2005;



Query	Answer	Excerpt of the supporting sentence list	Lexical patterns
{(Charlie Chaplin, London), Matsushita Konosuke (松下幸之助, ?)}	Wakayama 和歌山	- On April 15, 1889, Charlie Chaplin *was born in* London, England to Charles Chaplin, Sr., and Hannah Hill. - 明治 27 年 11 月 27 日、松下幸之助は和歌山*に生まれた*。(trans: Konosuke Matsushita was born in Wakayama on Nov. 27, Meiji 27th year)	X wa born in Y X は Y に生まれ X は Y 出身 X * born in Y on
Rakuten Infoseek {(楽天,インフォシーク), (?, Powerset)}	Microsoft	-Confirmed: Microsoft *buys* Powerset natural-language search. -楽天が インフォシーク を*買収*したことによって… (trans: By acquiring Infoseek, Rakuten, …) - By *acquiring* Powerset, Microsoft may launch a competitive … - インフォシーク*買収*が楽天にとって吉と出るか凶と出るか。(trans: Is it good or bad for Rakuten to acquire Infoseek ?)	X acquir Y X が Y を買収 X to buy Y X * Y の買収

Fig. 7. Example queries and results of the proposed search engine (the small string above a Japanese entity is the transliteration of the entity).

Bollegala et al. 2009a]: we stem the input sentence before extraction and we eliminate the condition that a pattern must contain both X and Y (instead we add the prefix "$X*$" if a pattern contains Y, but does not contain X and the subfix "$*Y$" if a pattern contains X, but does not contain Y). However, in our previous work, we have not investigated the effect of these changes on the overall performance of a latent relational search system. To evaluate the effect of these changes, we compare the performance of the search engine when we use our extraction algorithm and the baseline algorithm as described in [Turney 2005; Bollegala et al. 2009a]. The baseline algorithm does not stem the input sentence and does require that a pattern must contain both X and Y. We compare the performance only on monolingual latent relational search query sets because we want to prevent any interference of the lexical pattern translation process, which can blur the difference between the proposed extraction algorithm and the baseline. Therefore, we extract all English documents in the train dataset and use the result of the first phase of the clustering algorithm (i.e., the method described in [Duc et al. 2010]) in this experiment. The similarity of two lexical patterns is measured in the original vector space, without LRA. In accordance with the previous work [Duc et al. 2010], we compare the mean reciprocal rank (MRR) of four monolingual query sets (corresponding to the first four relation types in Table IV) when the system runs with the proposed extraction algorithm and the baseline.

The comparison between MRR of the search engine with each pattern extraction algorithm is shown in Fig. 8. For the BIRTHPLACE and ACQUISITION relation, the proposed extraction algorithm significantly outperforms the baseline algorithm. This is because English phrases that describe these relations contain various inflected forms of a word (e.g., "acquires", "acquired") and the text inside the gap between X and Y is complex (e.g., "X was born and risen up in Y", "X was born in 1948 in Y"). The baseline algorithm could not extract many common patterns in these cases. On the other hand, the proposed algorithm is able to extract many common patterns (e.g., "X was born * Y") in these cases. For the CEO relation, the difference is not significant. This is because the CEO relation is often referred by some patterns such as "X, CEO of Y", "X - CEO of Y", which do not contain inflected forms of words and are not complex. The results prove that the proposed extraction algorithm works well for all types of relations, whereas, the baseline only achieves high performance for relations in which the lexical patterns are not complex.

6.6. PMI vs. Frequency as feature vector values

In this section, we compare the performance of the search engine when using PMI and the number of co-occurrences between entity pairs and lexical patterns as elements of the matrix **A**. We use the same four monolingual query sets as in the previous section

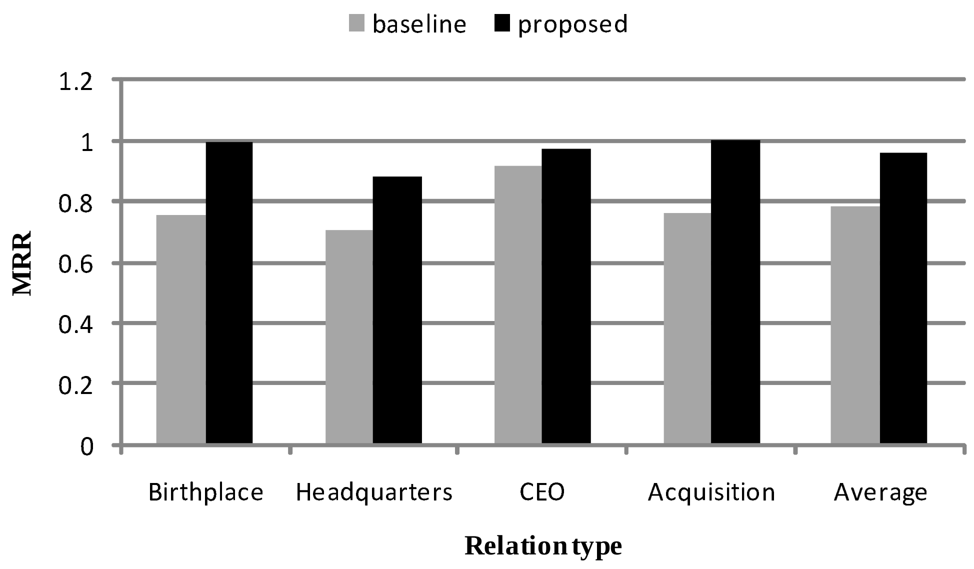


Fig. 8. Comparison between the performance of the search engine on four monolingual query sets while using the proposed lexical pattern extraction algorithm and the baseline algorithm

and we also use only the first phase of the clustering algorithm in this experiment, to prevent any complexity introduced by pattern translation and cross-lingual queries, which might blur the difference between using PMI and Frequency.

When using PMI as features, we achieve an average MRR of 0.989 on four monolingual query sets, whereas, using numbers of co-occurrences, this value is 0.963. Therefore, the value of MRR is only slightly different. However, when we evaluate the system with the Top 10 results of queries that have multiple correct answers, there is a significant difference between the two methods. We use the ACQUIREE query set which contains queries of the form $\{(A, B), (C, ?)\}$ of the acquisition relation for this purpose. Note that the ACQUIREE query set is different from the ACQUISITION query set in previous sections: each query in this set has multiple correct answers. For example, the answers for the query {(Google, YouTube), (Microsoft, ?)} are companies that are acquired by Microsoft. For the ACQUIREE query set, the method based on numbers of co-occurrences achieves a precision of 81.34% in the Top 10 results, whereas, the method based on PMI achieves 88.06%. This shows that using PMI as feature vector elements improves the precision of queries with multiple correct answers.

6.7. Comparison with existing monolingual latent relational search methods

We compare the performance of the proposed methods with that of two existing monolingual latent relational search systems [Kato et al. 2009; Duc et al. 2010]. We evaluate our system with eight Japanese-to-Japanese and eight English-to-English (monolingual) query sets corresponding to eight relation types in Table IV. The comparison result is shown in Table V. The first row in the table shows the result reported in [Kato et al. 2009] on Japanese monolingual query sets (of many common relation types in Table IV). The second row is the result of our previously proposed method (the method **LPC** without pattern translation) for monolingual latent relational search on English

Table V. Comparison between the proposed methods and existing methods
(TopN is the percentage of queries with correct answer in the Top N results)

Method	MRR	Top1	Top5	Top10	Top20
[Kato et al. 2009][JJ]	0.545	43.3	68.3	72.3	76.0
[Duc et al. 2010] [EE]	0.963	95.0	97.8	97.8	97.8
HLPC-EE	0.967	94.1	99.8	99.8	99.9
HLPC-JJ	0.888	87.5	90.0	90.0	90.0
HLPC+LRA-EE	0.971	94.9	99.9	100	100
HLPC+LRA-JJ	0.889	87.0	91.0	91.0	91.0
HLPC-Cross	0.515	37.6	70.1	78.4	82.9
HLPC+LRA-Cross (Freq)	0.579	44.6	74.6	83.6	88.4
HLPC+LRA-Cross (PMI)	0.605	49.8	74.5	78.5	82.0

monolingual query sets (as reported in [Duc et al. 2010]). The third and forth rows are the performance of the **HLPC** method on English and Japanese monolingual query sets, respectively. The fifth and sixth rows are the performance of the **HLPC+LRA** method on the same monolingual query sets. The last three rows are the performance of the proposed methods on 16 cross-language query sets which are described in previous sections. The results of **HLPC-Cross** and **HLPC+LRA-Cross (Freq)** are reported in our recent research concerning cross-language latent relational search [Duc et al. 2011]. The **HLPC+LRA-Cross (PMI)** row shows the performance of the proposed method in this paper, which uses PMI as feature vector values. We found that PMI yields better performance than Frequency (the number of co-occurrences between an entity pair and a lexical pattern) in latent relational search. Because we use the same extraction algorithm with [Duc et al. 2010], the performance on monolingual query sets is at the same level with that of [Duc et al. 2010] (the performance on Japanese query sets is sightly lower because many Japanese lexical patterns do not contain any text in the gap between the two entities, e.g, “*Microsoft Ballmer CEO*”, so many entity pairs share the pattern “XY . . . ”). The performance of the **HLPC+LRA** method on *cross-language* query sets is slightly higher than that of the method in [Kato et al. 2009] on *monolingual* query sets. The gap between **HLPC+LRA-EE** and **HLPC+LRA-Cross** can be explained by the gap between the difficulty of monolingual latent relational search and cross-language latent relational search.

6.8. Dependency on the underlying machine translation system

In our previous work regarding cross-language latent relational search [Duc et al. 2011], we did not investigate the dependency of the proposed method on machine translation. Therefore, in this paper, we study the effect of the Statistical Machine Translation (SMT) system used for translating lexical patterns and entity pairs on the performance of the proposed method. Specifically, we compare the performance of the method when using two different SMT systems. The first SMT system is Google Translate[13], which we used in all previous experiments concerning cross-language queries. The second SMT system is the open source SMT system Moses [Koehn et al. 2007], which is frequently used as a baseline for evaluating machine translation techniques. Moses does not provide default parallel corpus for Japanese-to-English translation. Therefore, we use the Japanese-English bilingual corpus in the Kyoto Free Translation Task (KFTT) [Neubig 2011] as training data to train the Moses system. This corpus is an aligned parallel corpus that contains Wikipedia articles related to Kyoto. The corpus includes about 440,000 parallel sentences which are made up of 12 million Japanese words and 11.5 million English words[14]. With this very small amount

[13]http://translate.google.com/
[14]http://www.phontron.com/kftt/

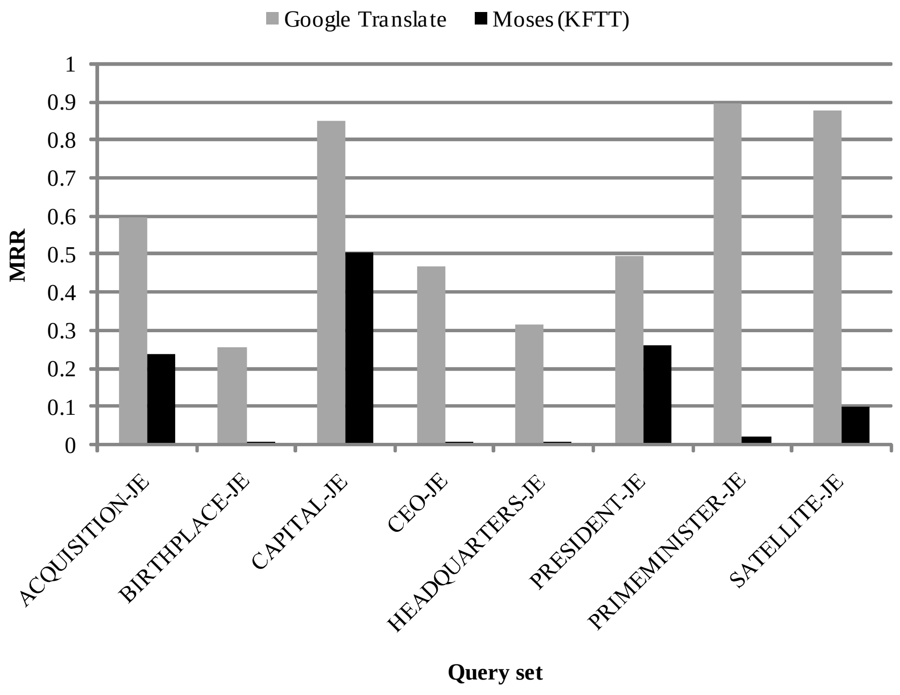


Fig. 9. Comparison between the performance of the search engine on Japanese-to-English cross-language query sets while using the Google Translate and Moses as the underlying SMT system.

of training data, the SMT system trained with this corpus only achieves a reasonable performance if the input sentence is related to cities or locations in Japan, especially Kyoto. We run all pre-processing phases to build an index for our search engine, as in previous experiments, except that the SMT system is now changed from Google Translate to the Moses system training with KFTT. Because we trained Moses for Japanese-to-English translation (i.e., we input Japanese lexical patterns and get English outputs), we only evaluate performance of eight Japanese-to-English query sets in the previous sections. Fig. 9 shows the comparison between the performance of the search engine on these eight cross-lingual (Japanese-to-English) query sets when the search engine uses Google Translate and Moses as the underlying SMT system. From the figure, we can observe that the performance of the system using Moses on the CAPITAL query set is relatively good, despite the fact that the training data for the SMT system is very small. This is because the CAPITAL relation is frequently mentioned in the KFTT bilingual corpus (Kyoto is the former capital of Japan). We do not achieve a comparable performance on other query sets because the KFTT corpus contain little articles concerning these relations, so the precision of Moses on the task of lexical pattern translation is low, compared to Google. This result implies that, with enough training data (bilingual corpus) to train the underlying SMT system, we can achieve a reasonable performance.

6.9. Evaluation with the INEX 2008 Entity Ranking task

In this section, we demonstrate the ability of answering sophisticated queries of many relation types using the proposed system. Specifically, we evaluate the system with the

Table VI. Performance of the system on the INEX 2008 Entity Ranking task (xinfAP is the inferred average precision, there are total of 35 questions)

Criterion	Monolingual (E-E)	Cross-lingual (J-E)
Percentage of answered questions	42.9%	34.3%
xinfAP (over all questions)	0.076	0.054
xinfAP (answered questions only)	0.178	0.156

standard query set from the INEX 2008 Entity Ranking task [Demartini et al. 2008]. We use the INEX 2008 Entity Ranking task as the benchmark for comparison because the entity ranking task is similar to latent relational search and the data for evaluation (the query set and the gold standard answers, as well as the auto-evaluation script) can be freely downloaded from the INEX 2008 homepage[15].

In the INEX 2008 Entity Ranking task, there are 35 topics (questions) of several relation types[16]. The task is to retrieve the Wikipedia page IDs of the answer entities for those questions. In the INEX Entity Ranking task, the corpus that was used is the entire English Wikipedia. Therefore, the corpus is not created by using seed entity pairs (to make Google queries to retrieve documents) as seen in previous experiments. Consequently, the experiment results in this section reflect the performance of the proposed method with a more practical corpus.

To create an index for the search engine, we downloaded the Japanese Wikipedia XML data dump[17] and the English Wikipedia XML data dump[18]. We create a MapReduce application to retrieve text from these dumps and extract entities and relations from the text in parallel. As the result, we retrieved about seven million Wikipedia articles (of which 1.6 million articles are in Japanese Wikipedia). From these articles, we extracted 213,731,476 sentences. After the entity and relation extraction process, we obtained 6,688,119 named entities, which make up 30,778,223 entity pairs. The total number of lexical pattern extracted is 945,857,689. Because the number of lexical patterns is large, we omit the LRA phase (i.e., we use sim_{VSM} to calculate similarity). As noted in Section 6.3, when we omit LRA, the performance will be slightly degraded. However, with this large number of lexical patterns, it is difficult to perform the LRA operation. We were able to complete the pre-processing steps in seven days using five machines (each machine is an Intel Core i7, 3GHz x 6 processors, 24GB of RAM).

We then manually create latent relational search queries to answer the questions in INEX. For example, for the question "I am looking for characters in the Harry Potter universe that are part of Gryffindor house or team and that play Quidditch", we create the monolingual query {(Steve Bruce, Manchester United), (?, Gryffindor Quidditch)} and the Japanese-to-English cross-lingual query {(*Yasuda Michihiro*, *Gamba Osaka*), (?, Gryffindor Quidditch)}. We query the proposed search engine to retrieve answer entities, an we use the mapping from entities to their Wikipedia page IDs to output the result list in INEX/TREC format. We then use the gold standard answers and the evaluation script from INEX[19] to automatically evaluate the result.

The result of this experiment is shown in Table VI. The first row of Table VI shows the percentage of questions that we could retrieve at least an answer (in the gold standard answer list). With monolingual queries (English-to-English), we could answer 15 (out of 35) questions. With cross-lingual queries (Japanese-to-English), we could answer 12 (out of 35) questions. The questions that we could not answer can be classified

[15]http://www.l3s.de/~demartini/XER08/
[16]http://www.l3s.de/~demartini/XER08/inex08-xer-topics-final.xml
[17]http://dumps.wikimedia.org/jawiki/20110921/
[18]http://dumps.wikimedia.org/enwiki/20100817/
[19]http://www.l3s.de/~demartini/XER08/INEX08-XER-testing-final.zip

into two types. The first type contains questions that we could not create latent relational search queries. For example, the question number 126, “I want to compile an exhaustive list of toy train manufacturers that are still in business” is such a question. This question describes a sophisticated relation existing between an ORGANIZATION (company) and a product (toy). We found no way to create a latent relational search query to answer this question. The second type contains questions that we were able to make some reasonable queries, but the proposed system could not retrieve any answer. For example, for the question number 124, “The user wants a list of novels that won the Booker Prize”, we created the query {(A Visit From the Goon Squad, Pulitzer Prize), (?, Booker Prize)}. However, we could not retrieve any answer. This is because the Stanford Named Entity Recognizer does not recognize “A Visit From the Goon Squad” as a named entity, instead it recognizes “Goon Squad” as a named entity.

The second row in Table VI shows the inferred average precision (xinfAP) [Yilmaz et al. 2008] of the entire query sets. This is the evaluation metric that INEX uses to evaluate the performance of an entity retrieval system [Demartini et al. 2008]. For the entire query set, the result of the proposed system is lower than the baseline of INEX 2008 [Demartini et al. 2008]. This is because we failed to create queries for the majority of questions. In addition, for some questions, the system could not answer because the key entities contain names of novels, names of movies or names of awards that the Stanford Named Entity Recognizer is not trained to recognize (it actually mistakenly recognizes some movie names as LOCATIONs or PERSONs, therefore, we could extract some movies, novels or awards).

If we exclude the questions that we could not create appropriate latent relational search queries and the questions that the system could not answer, then the result is as shown in the last row of Table VI. With monolingual search queries, the proposed system slightly outperforms the baseline systems in INEX 2008 (xinfAP is 0.178, compared to 0.111 and 0.159 of the two baselines). The performance is about a half of the best performance in INEX 2008 (0.341). However, note that in the INEX 2008 Entity Ranking task, the participants are allowed to use the information in the title and description of the questions, which include several informative keywords. On the other hand, in the proposed system, we only have the input source entity pair (which is manually created by human), and we need to represent the relation by some lexical patterns. Even in this setting, the proposed system achieves the same level of average precision with baseline systems in INEX. Moreover, with Japanese-to-English cross-lingual queries, we also achieve an xinfAP of 0.156, which is only slightly smaller than that of the monolingual queries. This proves that the proposed method could be used to answer real-world questions, provided that the user of the system could imagine good input source entity pairs to formulate queries. Finding a good source pair as an example is much simpler than enumerating a long list of appropriate keywords.

7. CONCLUSION

We proposed a method for extracting entity pairs and relations from a multi-lingual text corpus for cross-language latent relational search. The method represents the relations between two entities in an entity pair by lexical patterns of the context surrounding the two entities. To effectively recognize paraphrased lexical patterns across languages, we proposed a hybrid lexical pattern clustering algorithm and a method to integrate multi-lingual Latent Relational Analysis (LRA) into the algorithm to improve the overall performance. The proposed method achieves an MRR of 0.605 on the task of English-Japanese cross-language latent relational search. Moreover, we showed that although the proposed method needs a machine translation system to translate lexical patterns, its performance does not strongly depend on the underlying machine translation system. This result suggests that we can make cross-lingual information retrieval less dependent on machine translation by clustering and other dimensionality reduction techniques such as LRA. Finally, we showed that the proposed system can be used to answer many questions in the INEX 2008 Entity Ranking task, which are sophisticated questions. In future, we intend to apply cross-language latent relational search to support human translators by providing nearly parallel supporting sentences in the search result.

REFERENCES

BALOG, K. ET AL. 2009. Overview of the TREC 2009 Entity Track. In *Proc. of TREC'09*.

BALOG, K., SERDYUKOV, P., AND DE VRIES, A. P. 2010. Overview of the TREC 2010 Entity Track. In *Proc. of TREC'10*.

BANKO, M. ET AL. 2007. Open Information Extraction from the Web. In *Proc. of IJCAI'07*. 2670–2676.

BOLLEGALA, D. ET AL. 2009a. Measuring the Similarity between Implicit Semantic Relations from the Web. In *Proc. of WWW'09*. ACM, 651–660.

BOLLEGALA, D. ET AL. 2009b. Measuring the Similarity between Implicit Semantic Relations Using Web Search Engines. In *Proc. of WSDM'09*. ACM, 104–113.

BOLLEGALA, D. ET AL. 2010. Relational Duality: Unsupervised Extraction of Semantic Relations between Entities on the Web. In *Proc. of WWW'10*. ACM, 151–160.

BUNESCU, R. AND MOONEY, R. Learning to Extract Relations from the Web Using Minimal Supervision. In *Proc. of ACL'07*. 576–583.

CAVNAR, W. AND TRENKLE, J. 1994. N-Gram-Based Text Categorization. In *Proc. of the 3rd Annual Symposium on Document Analysis and Information Retrieval, SDAIR'94*. 161–175.

CHALMERS, D. J., FRENCH, R. M., AND HOFSTADTER, D. 1992. High-level Perception, Representation, and Analogy: A Critique of Artificial Intelligence Methodology. *Journal of Experimental & Theoretical Artificial Intelligence 4,* 3, 185–211.

DAVIDOV, D. AND RAPPOPORT, A. 2010. Automated Translation of Semantic Relationships. In *Proc. of COLING'10*. 241–249.

DEFAYS, D. 1977. An Efficient Algorithm for a Complete Link Method. *Computer Journal 20,* 4, 364–366.

DEMARTINI, G., DE VRIES, A. P., IOFCIU, T., AND ZHU, J. 2008. Overview of the INEX 2008 Entity Ranking Track. In *Proc. of INEX'08*. 243–252.

DUC, N. T., BOLLEGALA, D., AND ISHIZUKA, M. 2010. Using Relational Similarity between Word Pairs for Latent Relational Search on the Web. In *Proc. of WI'10*. 196 – 199.

DUC, N. T., BOLLEGALA, D., AND ISHIZUKA, M. 2011. Cross-Language Latent Relational Search: Mapping Knowledge across Languages. In *Proc. of AAAI'11*. 1237–1242.

DUMAIS, S. ET AL. 1997. Automatic Cross-Language Retrieval Using Latent Semantic Indexing. In *Proc. of AAAI Symposium on Cross-Language Text and Speech Retrieval '97*. 18–24.

DUMAIS, S. ET AL. 2002. Web Question Answering: Is More Always Better? In *Proc. of SIGIR'02*. 291–298.

FERRANDEZ, S. ET AL. 2009. Exploiting Wikipedia and EuroWordNet to solve Cross-Lingual Question Answering. *Information Sciences 179,* 20, 3473 – 3488.

GENTNER, D. 1983. Structure-mapping: A Theoretical Framework for Analogy. *Cognitive Science 7,* 2.

GOLUB, G. AND KAHAN, W. 1965. Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis 2,* 2, 205–224.

GOTO, T., DUC, N. T., BOLLEGALA, D., AND ISHIZUKA, M. 2010. Exploiting Symmetry in Relational Similarity for Ranking Relational Search Results. In *Proc. of PRICAI'10*. 595–600.

GRISHMAN, R. AND SUNDHEIM, B. 1996. Message Understanding Conference - 6: A Brief History. In *Proc. of COLING'96*. 466–471.

HALSKOV, J. AND BARRIERE, C. 2008. Web-based Extraction of Semantic Relation Instances for Terminology Work. *Terminology 14,* 1, 20–44.

HOFSTADTER, D. AND FARG. 1995. *Fluid Concepts and Creative Analogies. Computer Models of the Fundamental Mechanisms of Thought*. Basic Books.

ISOZAKI, H. ET AL. 2005. NTT's Japanese-English Cross-Language Question Answering System. In *Proc. of NTCIR'05*.

KATO, M. P. ET AL. 2009. Query by Analogical Example: Relational Search Using Web Search Engine Indices. In *Proc. of CIKM'09*. 27–36.

KOEHN, P. ET AL. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL'07*. Association for Computational Linguistics.

KWOK, C. ET AL. 2001. Scaling Question Answering to the Web. In *Proc. of WWW'01*. 150–161.

LANDAUER, T. AND DUMAIS, S. 1997. Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. *Psychological Review 104*, 211–240.

MANNING, C., RAGHAVAN, P., AND SCHUTZE, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

MCNAMEE, P. 2005. Language Identification: A Solved Problem Suitable for Undergraduate Instruction. *Journal of Computing Sciences in Colleges 20,* 3, 94–101.

NAJORK, M., ZARAGOZA, H., AND TAYLOR, M. J. 2007. Hits on the Web: How Does It Compare? In *Proc. of SIGIR'07*. 471–478.

NEUBIG, G. 2011. The Kyoto Free Translation Task. http://www.phontron.com/kftt.

PANTEL, P. AND PENNACCHIOTTI, M. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proc. of ACL'06*. 113–120.

PANTEL, P. AND RAVICHANDRAN, D. 2004. Automatically Labeling Semantic Classes. In *Proc. of HLT-NAACL'04*. 321–328.

PEÑAS, A. ET AL. 2009. Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In *Proc. of CLEF'09*. 174–196.

PRETTENHOFER, P. AND STEIN, B. 2010. Cross-Language Text Classification Using Structural Correspondence Learning. In *Proc. of ACL'10*. 1118–1127.

RADEV, D. R., QI, H., WU, H., AND FAN, W. 2002. Evaluating Web-based Question Answering Systems. In *Proc. of LREC'02*.

RAVICHANDRAN, D. AND HOVY, E. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proc. of ACL'02*. 41–47.

SHAH, C. AND CROFT, W. B. 2004. Evaluating High Accuracy Retrieval Techniques. In *Proc. of SIGIR'04*. 2–9.

SHINYAMA, Y. AND SEKINE, S. 2006. Preemptive Information Extraction Using Unrestricted Relation Discovery. In *Proc. of HLT-NAACL'06*. 304 – 311.

SIBSON, R. 1973. SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method. *Computer Journal 16,* 1, 30–34.

SUMITA, E. 2001. Example-Based Machine Translation Using DP-matching between Word Sequences. In *Proc. of DMMT'01*. ACL, 1–8.

TAKCI, H. AND SOGUKPINAR, I. 2004. Centroid-Based Language Identification Using Letter Feature Set. In *Proc. of CICLing'04*. 640–648.

TANAKA-ISHII, K. AND ISHII, Y. 2007. Multilingual Phrase-Based Concordance Generation in Real-Time. *Information Retrieval 10,* 3, 275–295.

TANAKA-ISHII, K. AND NAKAGAWA, H. 2005. A Multilingual Usage Consultation Tool Based on Internet Searching: More than a Search Engine, Less than QA. In *Proc. of WWW'05*. 363–371.

TURNEY, P. 2005. Measuring Semantic Similarity by Latent Relational Analysis. In *Proc. of IJCAI'05*. 1136–1141.

TURNEY, P. 2006. Similarity of Semantic Relations. *Computational Linguistics 32,* 3, 379–416.

TURNEY, P. 2008a. A Uniform Approach to Analogies, Synonyms, Antonyms, and Associations. In *Proc. of COLING'08*. 905–912.

TURNEY, P. D. 2008b. The Latent Relation Mapping Engine: Algorithm and Experiments. *Journal of Artificial Intelligence Research (JAIR) 33*, 615–655.

VEALE, T. 2003. The Analogical Thesaurus. In *Proc. of the IAAI'03*. AAAI Press, 137–142.

YILMAZ, E., KANOULAS, E., AND ASLAM, J. A. 2008. A Simple and Efficient Sampling Method for Estimating AP and NDCG. In *Proc. of SIGIR'08*. 603–610.

Received Aug. 2011; revised Nov. 2011, Jan. 2012; accepted Jan. 2012