SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Wikipedia Link Structure and Text Mining for Semantic Relation Extraction

Towards a Huge Scale Global Web Ontology

Kotaro Nakayama, Takahiro Hara and Shojiro Nishio

Dept. of Multimedia Eng., Graduate School of Information Science and Technology
Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
TEL: +81-6-6879-4513 FAX: +81-6-6879-4514
{nakayama.kotaro, hara, nishio}@ist.osaka-u.ac.jp

Abstract. Wikipedia, a collaborative Wiki-based encyclopedia, has become a huge phenomenon among Internet users. It covers huge number of concepts of various fields such as Arts, Geography, History, Science, Sports and Games. Since it is becoming a database storing all human knowledge, Wikipedia mining is a promising approach that bridges the Semantic Web and the Social Web (a. k. a. Web 2.0). In fact, in the previous researches on Wikipedia mining, it is strongly proved that Wikipedia has a remarkable capability as a corpus for knowledge extraction, especially for relatedness measurement among concepts. However, semantic relatedness is just a numerical strength of a relation but does not have an explicit relation type. To extract inferable semantic relations with explicit relation types, we need to analyze not only the link structure but also texts in Wikipedia. In this paper, we propose a consistent approach of semantic relation extraction from Wikipedia. The method consists of three sub-processes highly optimized for Wikipedia mining; 1) fast preprocessing, 2) POS (Part Of Speech) tag tree analysis, and 3) mainstay extraction. Furthermore, our detailed evaluation proved that link structure mining improves both the accuracy and the scalability of semantic relations extraction.

1 Introduction

Wikipedia, a collaborative Wiki-based encyclopedia, has become a huge phenomenon among Internet users. According to statistics of Nature, Wikipedia is about as accurate in covering scientific topics as the Encyclopedia Britannica[1]. It covers concepts of various fields such as Arts, Geography, History, Science, Sports, Games. It contains more than 2 million articles (Oct. 2007, English Wikipedia) and it is becoming larger day by day while the largest paper-based encyclopedia Britannica contains only 65,000 articles.

As a corpus for knowledge extraction, Wikipedia's impressive characteristics are not limited to the scale, but also include the dense link structure, sense disambiguation based on URL, brief link texts and well structured sentences. The fact that these characteristics are valuable to extract accurate knowledge from Wikipedia is strongly confirmed by a number of previous researches on

Wikipedia Mining[2–5]. These researches are mainly about semantic relatedness measurements among concepts. Besides, we proposed a scalable link structure mining method to extract a huge scale association thesaurus in a previous research [4]. In that research, we developed a huge scale association thesaurus dictionary extracting a list of related terms from any given term. Further, in a number of detailed experiments, we proved that the accuracy of our association thesaurus achieved notable results. However, association thesaurus construction is just the beginning of the next ambitious research *Wikipedia Ontology*, a huge scale Web ontology automatically constructed from Wikipedia.

Semantic Wikipedia [6] is an impressive solution for developing a huge scale ontology on Wikipedia. Semantic Wikipedia is an extension of Wikipedia which allows editors to add semantic relations manually. Another interesting approach is to use Wikipedia's category tree as an ontology [7–9]. Wikipedia's categories are promising resources for ontology construction, but categories can not be used as an ontology since the structure of Wikipedia category is just a taxonomy and do not provide explicit relation types among concepts.

In contrast to these approaches, we propose a full-automated consistent approach for semantic relation extraction by mining Wikipedia article texts. Since a Wikipedia article is a set of definitive sentences, the article text is yet another valuable resource for ontology construction. The method consists of three subprocesses highly optimized for Wikipedia mining; 1) fast preprocessing, 2) POS (Part Of Speech) tag tree analysis, and 3) mainstay extraction. Furthermore, we show the potential of important sentence analysis for improving both accuracy and scalability of semantic relations extraction.

The rest of this paper is organized as follows. In section 2, we explain a number of researches on Wikipedia Mining for knowledge extraction in order to make our stance clear. In section 3, we describe our proposed integration method based on NLP and link structure mining. We describe the results of our experiments in section 4. Finally, we draw a conclusion in section 5.

2 Related Works

2.1 Wikipedia Mining

As we mentioned before, Wikipedia is an invaluable Web corpus for knowledge extraction. Researches on semantic relatedness measurement are already well conducted[2–5]. WikiRelate [5] is one of the pioneers in this research area. The algorithm finds the shortest path between categories which the concepts belong to in a category graph. As a measurement method for two given concepts, it works well. However, it is impossible to extract all related terms for all concepts because we have to search all combinations of category pairs of all concept pairs (2 million $\times$ 2 million). Furthermore, using the inversed path length as semantic relatedness is a rough method because categories do not represent semantic relations in many cases. For instance, the concept "Rook (chess)" is placed in the category "Persian loanwords" together with "Pagoda," but the relation is not semantical, it is just a navigational relation. Therefore, in our previous research, we

proposed $pfibf$ (Path Frequency - Inversed Backward Link Frequency), a scalable association thesaurus construction method to measure relatedness among concepts in Wikipedia.

2.2 Wikipedia and Web Ontology

Semantic Wikipedia[6] is an impressive predecessor of this research area. It allows editors to put additional tags to define explicit relations between concepts. For example, assume that there is a sentence written in Wiki format like this;

```
'London' is the capital city of [[England]]
```

"[[...]]" is a hyperlink tag to another article (concept) and will be translated into a hyperlink when it is shown to readers, so the readers can understand that "London" is the capital of "England." However, obviously, machines can not understand the relation type if no NLP techniques are used because the relation is written in natural language. To solve this problem, Semantic Wikipedia allows users to add special annotations like this;

```
'London' is the capital city of [[capitalof::England]]
```

Semantic Wikipedia is a promising approach for a huge scale Web ontology construction but we wish an automated approach without any additional human-effort since a Wikipedia article already includes rich semantic relations.

3 Proposed method

To achieve full-automated Web ontology construction from Wikipedia, we propose a consistent approach for semantic relation extraction by mining Wikipedia article text. Basically, the proposed method extracts semantic relations by parsing texts and analyzing the structure tree generated by a POS parser. However, parsing all sentences in an article is not efficient since an article contains both valuable sentences and non-valuable sentences by mixture. Our assumption is that it is possible to improve accuracy and scalability by analyzing only important sentences for the topic.

In this section, we describe our proposed method for semantic relation extraction from Wikipedia. The whole flow of the proposed method is performed in the following three phases;

1. Preprocessing
   (Trimming, chunking and partial tagging)
2. Parsing and POS structure tree analysis
3. Mainstay extraction.

These phases are described in detail in the following subsections.

3.1 Preprocessing

Before we parse sentences, we need to trim, chunk and segment the sentences in order to make them processable for the parser. For this aim, we usually use statical NLP tools, however these tools cannot process the Wikipedia articles correctly since the articles are written in a special syntax composed of HTML tags and special Wiki command tags such as triple quotations, brackets for hyperlinks and tables. That is why we developed our own Preprocessor for this aim. Preprocessing is accomplished in three sub steps; 1) Trimming, 2) Chunking and 3) Partial tagging.

First, the preprocessor trims a Wikipedia article to remove unnecessary information such as HTML tags and special Wiki commands. We also remove table tags because table contents are usually not sentences. However, we do not remove link tags ("[[...]]") because links in Wikipedia are explicit relations to other pages and we use this link information in the following steps.

Second, the preprocessor separates the article into sentences. Basically, an article is separated into sentences by periods ("."). However, abbreviations etc. also use ".", so the preprocessor does not separate a sentence if the following character is a small letter. This simple strategy works very well in almost all cases (Over 99%) for Wikipedia articles. Furthermore, since it is based on neither semantic nor statistic methods, the process is much faster than ordinary chunkers. After separating an article into sentences, each sentence is separated into semantic chunks (phrases). Basically, terms are separated by white space (" "), but terms are bounded if these terms are placed in quotations or link tags.

Finally, phrases in quotations and link tags are tagged as nouns to help the following parsing phase. Bounding and partial tagging are helpful information for the parsing process because one of the most difficult technical issues in parsing natural language is chunking and bounding. Especially for domain specific terms or new terms, parsers often cannot parse the sentence structure correctly.

3.2 Parsing and Structure Tree Analysis

After the preprocessing, partially tagged and chunked sentences are given. In this phase, we parse each sentence to get a structure tree and analyze that structure tree to extract relations between concepts. To parse sentences, we adopted an unlexicalized PCFG (Probabilistic Context-Free Grammars) parsing method based on the factored product model. We used the Stanford NLP parser[10] for this purpose. It can parse a sentence accurately if the sentence is trimmed, chunked and tagged correctly, even if the sentence contains hyperlink tags ("[[...]]"). Figure 1 shows the detailed illustration of this phase.

"/NN" is a special POS tag for nouns, which is added in the partial tagging process. A list of main POS (Part Of Speech) tags used in this research is shown in Table 1.

The parser gets a partially tagged sentence and constructs a structure tree for the given sentence. For instance, assume that there is a semi-tagged sentence like this: "[[Madrid]]/NN is the [[capital]]/NN and largest city of [[Spain]]/NN .

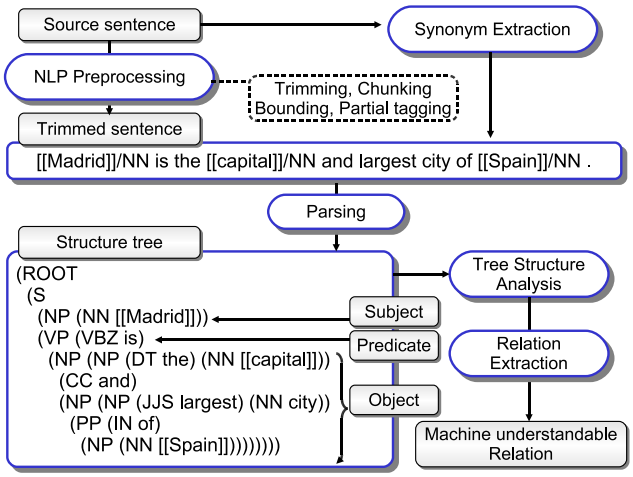

Fig. 1. Overview of the content mining process.

". The parser generates a structure tree like Figure 1. After that, the structure tree is analyzed in order to extract semantic relations in the following steps:

1. Extract "(NP ...) (VP (VBZ/VBD/VBP ...) (NP ...))" pattern from the parsed sentence.
2. Co-reference resolution
3. For both NP, split the NP into two NP parts if the NP contains CC. After that, perform step 1 again.
4. Finally, extract the 1st NP part as a subject, VB part as a predicate, the 2nd NP part as an object.

In the first step, we extract "(NP ...) (VP (VBZ/VBD/VBP ...) (NP ...))" and assume that the 1st NP part is the subject, the VB part is the predicate, the 2nd NP part is the object respectively. In the second step, the parser determines whether the subject is a co-reference of the topic of the article. To do that, we used two strategies mentioned in Nguyen's work[11]. The first strategy is to use the article title. If the all terms appeared in subject part are contained in the title of the article, the subject is determined as a co-reference to the topic. The second strategy is to use the most frequently used pronoun in the article. In the third step, NP will be separated if it contains CC such as "and" and "or". In the fourth step, if the 1st NP is a literal and a synonym of the concept representing the article, then the NP is replaced by the concept of the article. Finally, the first NP part is extracted as a subject, the VB part as a predicate, the 2nd NP part as an object.

The first step's POS tag pattern can be replaced by other alternatives. Currently, we prepared following three patterns for the first step.

1. (NP ...) (VP (VBZ/VBD/VBP ...) (NP ...))
   Normal pattern. E. g. "is-a"

Table 1. POS tags.

Tag	Description
NN	singular or mass noun
NNS	Plural noun
NNP	Singular proper noun
NNPS	plural proper noun
NP	Noun phrase
VB	Base form verb
VBD	Past tense
VBZ	3rd person singular
VBP	Non 3rd person singular present
VP	Verb phrase
JJ	Adjective
CC	Conjunction, coordinating
IN	Conjunction, subordinating

2. (NP ...) (VP (NP (NP ...) (PP (IN ...) ...))
   Subordinating pattern. E. g. "is-a-part-of"
3. (NP ...) (VP (VBZ ...) (VP (VPN ...) ...))
   Passive pattern. E. g. "was-born-in"

We can prepare further POS tag patterns to improve the coverage of semantic relation extraction. However, in this research, we applied only these three basic patterns to confirm the capability of this research direction. We also extract a relation even if the object part does not contain any hyperlinks to other pages. We call it *literal object*. For example, assume that there is a sentence with the following structure tree;

```
Brescia is a city.

(S (NP (NNP [[Brescia]]))
    (VP (VBZ is)
        (NP (DT a) (NN city))))
```

The object part is "a city" but it is not a hyperlink to an article about "city" but it is just a literal. Literal object is not machine understandable but the literal information is useful depending on the application even if the meaning of the term can not be specified uniquely.

3.3 Mainstay extraction for object

By performing the process described above, we distinguish subject part and object part. After that, we need to extract mainstays for both subject part and object part respectively. A mainstay is a semantic central term (or phrase) in the part. For instance, assume that there is a following sentence and structure tree. In this phase, for the 2nd NP (object part), replace the NP by the last NN/NNS in the NP if the NP parts consist of JJ and NN/NNS. So in the case

shown below, the parser obtains "[[astronomer]]" as the mainstay of the object part.

```
Lutz_D._Schmadel is [[Germany|German]] [[astronomer]].

(S (NP (NN Lutz_D._Schmadel)
    (VP (VBZ is)
        (NP (NN [[Germany|German]]) (NN [[astronomer]])
))))
```

The 2st NP consists of two NN and both of them have a hyperlink to other pages. The 1st NN has a link to a country "Germany" but it is used as a adjective, so it can not be a mainstay of the object part. So in this case, we have to obtain "[[astronomer]]" as the subject.

3.4 A Parsing Strategy: ISP

We conducted a small number of experiments using the above algorithms and realized that parsing all sentences is quite time consuming work and sometimes returns irrelevant results. More detailed preliminary investigation and experiment showed that it is possible to reduce the calculation and improve the accuracy of semantic relation extraction by filtering non important sentences in the article.

The Important Sentence Parsing (ISP) method parses sentences that seem important for an article (concept). We used $pfibf$ (Path Frequency - Inversed Backward link Frequency), an association thesaurus construction method we proposed in a previous research[4][1]. An association thesaurus is a set of terms and association relationships among them. The ISP method uses the association thesaurus to detect whether a sentence is important to an article or not. In this section, we describe the essentials of the method.

Basic Strategy of pfibf Wikipedia consist of a set of articles (concepts) and hyperlinks among them, thus they can be expressed by a graph $G = \{V, E\}$ (V: set of articles, E: set of hyperlinks). Let us consider how we can measure the relatedness between any pair of articles (v_i, v_j). The relatedness is assumed to be strongly affected by the following two factors:

- the number of paths from article v_i to v_j,
- the length of each path from article v_i to v_j.

The relatedness is strong if there are many paths (sharing of many intermediate articles) between two articles. In addition, the relatedness is affected by the path length. In other words, if the articles are placed closely together in the graph G and share hyperlinks to same articles, the relatedness is estimated to be higher than between farther ones. Therefore, if all paths from v_i to v_j are given

[1] The method name was $lfibf$ in the past and was changed to $pfibf$

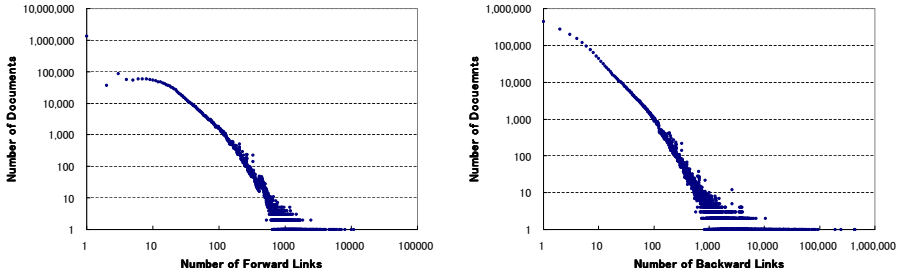

Fig. 2. Zipf distribution of the Wikipedia link structure.

as $T = \{t_1, t_2, ..., t_n\}$, we define the relatedness pf (path frequency) between them as follows:

$$pf(v_i, v_j) = \sum_{k=1}^{n} \frac{1}{d(|t_k|)}. \tag{1}$$

$d()$ denotes a function which increases the value according to the length of path t_k. A monotonous increasing function such as a the logarithm function can be used for $d()$.

In addition, the number of links between individual articles is also estimated as a factor of relatedness because the dense link structure is one of the most interesting characteristics of Wikipedia. *Dense* means that Wikipedia has a lot of *inner links*, links from pages in Wikipedia to other pages in Wikipedia. This means that articles are strongly connected by many hyperlinks. Let us show statistics of link structure analysis for Wikipedia that we investigated. Figure 2 shows the distribution of both backward links and forward links. Our statistics unveiled that both forward links and backward links have typical Zipf distribution, containing a few nodes that have a very high degree and many with low degree.

The statistics shows that we need to consider the characteristics to design algorithms for analyzing the Wikipedia link structure. For instance, assume that there is an article which is referred to from many other articles. This article would have a lot of short paths from many articles. This means that it has a strong relatedness to many articles if we used only pf. However, this kind of articles must be considered as a general concepts, and the importance of general concepts is not high in most cases. Therefore, we must consider the inversed backward link frequency ibf as follows in addition to the two factors above. We therefore define the algorithm $pfibf$ as follows:

$$ibf(v_j) = \log \frac{N}{bf(v_j)}, \tag{2}$$

$$pfibf(v_i, v_j) = pf(v_i, v_j) \cdot ibf(v_j). \tag{3}$$

N denotes the total number of articles and $bf(v_j)$ denotes the number of backward links of the page v_j. This means a page which shares forward/backward links with a specific page but not does not share with other pages, has a high $pfibf$.

Dual Binary Tree (DBT) The counting of all paths between all pairs of articles in a huge graph is a computational resource consuming work. Thus, making it efficient is a serious issue on Wikipedia mining. Using adjacency matrices and multiplication is not a clever idea because of the low scalability. Wikipedia has more than 2 million articles, thus we need several terabytes just for storing data. Further, we need unimaginably much time to calculate the multiplication because the order is $O(N^3)$. However, a large number of elements in the adjacency matrix of a Web site are zero, thus effective compression data structures and analysis methods are the key to achieve high scalability on Wikipedia mining. Therefore, we propose an efficient data structure named *Dual binary tree* (DBT) and a multiplication algorithm for the DBT.

Since the adjacency matrix of a Web site link structure is a sparse matrix (almost all elements are zero), the DBT stores only the non-zero elements for data compression. The DBT consists of two types of binary trees; i-tree and j-tree. Each element in the i-tree corresponds to a row in the adjacency matrix and each i-tree element stores a pointer to the root of a j-tree. This means that the DBT consists of totally $N+1$ (1 i-tree and N j-trees) binary trees. The point is that operations for both getting and storing data are very fast because the number of steps is in both cases $O(logN)$.
The function $j\text{-}Tree(i)$ extracts all elements in the ith row of the adjacency matrix A. $a_{j,k}$ denotes the element in the jth row and kth column of the matrix. The first loop will be executed N times, but the numbers of cycles of the second and third loop depend on the average link number M. Thus the total number of steps is $O(NlogN) \cdot O(M^2)$. Further, our statistics unveiled that M is constantly 20 to 40 in Wikipedia in spite of the evolvement of the matrix size N. Finally, the result is stored in another DBT R.

We conducted a benchmark test for the DBT and the multiplication algorithm compared with conventional methods. We used GNU Octave (with ATLAS library), one of the most effective numerical algebra implementations, as a base line method because a study[12] has proved that the performance for sparse matrix operations on Octave is better than that of Matlab, the most popular and well tuned numeric computation environment all over the world. In [12], it is described that "Octave implements a polymorphic solver for sparse matrices, where the exact solver used to factorize the matrix, depends on the properties of the sparse matrix itself." Table 2 shows the result of the performance comparison of $N \times N$ matrix multiplication with the density D.

N is the number of rows, equivalent to the number of columns in the adjacency matrix. Density D is the rate of non-zero elements in the matrix. It can be calculated by the following formula:

$$D = \frac{Number\ of\ non\ zero\ elements}{N^2}. \tag{4}$$

Table 2. Benchmark of multiplication.

Order (N)	Density (D)	Avg. link (M)	Octave	DBT
10,000	1.e-5	0.1	0.62	0.01
10,000	1.e-4	1	0.63	0.09
10,000	1.e-3	10	0.87	5.18
15,000	1.e-5	0.15	1.39	0.01
15,000	1.e-4	1.5	1.42	0.28
15,000	1.e-3	15	2.15	17.74
20,000	1.e-5	0.2	2.49	0.02
20,000	1.e-4	2	2.55	0.62
20,000	1.e-3	20	4.72	42.72
50,000	1.e-5	0.5	74.94	0.14
50,000	1.e-4	5	75.25	6.24

(Unit: sec.)

The result of the benchmark test proved that the DBT is very beneficial for multiplication on a matrix whose density is less than 1.e-4. Further, as the size of N increases, it also advantages the performance.

English Wikipedia has 3.8 million pages (Sept. 2006, including redirect pages), 73.3 million links and the density is about 5.e-6. This means that the adjacency matrix of Wikipedia is a typical sparse matrix with a huge number of rows and columns. Therefore, the DBT is more suitable for Wikipedia Mining than other numerical algebra implementations such as Octave. What we should consider, however, is that the DBT is suitable only while the matrix is sparse enough. Repeated multiplication makes the matrix dense, thus after each multiplication, all elements except top k ranked elements in each row should be removed to keep the sparsity of the matrix.

pfibf with DBT In this section, we describe the concrete flow of $pfibf$ calculation using a DBT. Since $pfibf$ analyzes both forward and backward links of the articles, first we calculate A' by adding A and the transpose matrix A^T as follows:

$$A' = A + A^T. \tag{5}$$

By calculating the power of A', we can extract the number of paths for any pair of articles in n-hop range. An element $a'^n_{i,j}$ in matrix A'^n denotes the number of paths from article v_i to article v_j whose length is n. However, before calculating A'^n, each element in A should be replaced by the following formula to approximate ibf (Formula (2)):

$$a'_{i,j} \leftarrow a'_{i,j} \cdot log\frac{N}{|B_{v_j}|}. \tag{6}$$

$|B_{v_j}|$ denotes the number of backward links of article v_j. Finally, we can extract the $pfibf$ for any pair by adding the matrices A'^1, A'^2, ... , A'^n as follows:

$$pfibf(i,j) = \sum_{l=1}^{n} \frac{1}{d(n)} \cdot a'^{l}_{i,j}. \tag{7}$$

$d()$ denotes a monotonically increasing function such as a logarithm function which increases the value according to the length of path n.

FB Weighting After a number of experiments to evaluate the accuracy of $pfibf$, we realized that the accuracy decreased in particular situations. Then, after conducting further experiments in order to detect the cause, we finally realized that the accuracy of general term analysis is worse than the accuracy of domain specific terms. General terms have the following characteristics:

- They have a lot of backward links,
- They are referred to from various topic-ranges,
- The content is trustful because it is usually edited by many authorities.

General terms, such as "United states," "Marriage" and "World War II," are referred to from various articles in various topic ranges. This means that the backward link analysis cannot be converged because the topic locality is weaker than in domain-specific terms such as "Microsoft" and "iPod." Although the backward link analysis is not convergent, the forward link analysis is effective because the contents are trustful and usually edited by many authorities.

In contrast to this, domain-specific terms have a much stronger topic locality. Although they have less links from other pages and the contents are sometimes not trustful, each link from other pages is topically related to the content. Therefore, we developed the *FB weighting* method which flexibly changes the weight of the forward link analysis and backward link analysis as follows:

$$W_b(|B_d|) = 0.5/(|B_d|^{\alpha}), \tag{8}$$

$$W_f(|B_d|) = 1 - W_b(|B_d|). \tag{9}$$

$|B_d|$ is the backward link number of document d. The constant α must be optimized according to the environment. After a number of experiments, an α value of about 0.05 was recognized to be suitable for the link structure of Wikipedia. The weight W_b is multiplied for each element on A and W_f for A^T as well. Thus formula (5) must be modified into the following formula (10):

$$A' = W_f \cdot A + W_b \cdot A^T. \tag{10}$$

Table 3 shows an example of an association thesaurus constructed by $pfibf$ with FB weighting. For example, when analyzing the article "Google," associated concepts such as "Search engine", "PageRank" and "Google search" are extracted from the association thesaurus.

We also conducted several experiments in the previous research[4] and the results proved that the FB Weighting method is significantly more effective

Table 3. Sample of queries and terms extracted by pfibf with FB weighting.

Query	Extracted association terms		
Sports	Basketball	Baseball	Volleyball
Microsoft	MS Windows	OS	MS Office
Apple Inc.	Macintosh	Mac OS X	iPod
iPod	Apple Inc.	iPod mini	iTunes
Book	Library	Diamond Sutra	Printing
Google	Search engine	PageRank	Google search
Horse	Rodeo	Cowboy	Horse-racing
Film	Actor	Television	United States
DNA	RNA	Protein	Genetics
Canada	Ontario	Quebec	Toronto

for association thesaurus construction than other traditional methods such as link co-occurrence analysis and TF-IDF. Especially for domain-specific terms, it achieved remarkable accuracy.

Important Sentence Detection By using $pfibf$, a set of important links for each article (concept) in Wikipedia can be extracted. ISP detects important sentences in a page from sentences containing important words/phrases for the page. It crawls all sentences in the article to extract sentences containing links to the associated concepts. The extracted sentences are then parsed as the important sentences in the article. For each links in a sentence, the parser calculates $pfibf$ and the max value denotes the importance of the sentence. The importance can be used for filtering unimportant sentences by specifying thresholds.

4 Experiments and discussion

First, we analyzed the whole Wikipedia link structure and gathered 65,391 articles (pages) that have more than 100 backward links, to filter noisy pages. After that, we randomly selected about 100 articles as a test set. We applied Preprocessing, parsing and structure tree analysis proposed in Section 3.

Table 4 shows some examples of explicit relations extracted by our method. We can see that it extracts various relations such as “borders,” “hosted” and “separates”. However, machines cannot understand the meaning “borders” without any instruction from humans. So, in order to make the predicate part machine understandable, we have to define the relation between predicates. For example, “is” and “was” have the same meaning but the tense is different. By giving this kind of knowledge, it will be inferable relations. We believe that the amount of relations among verbs are limited compared with relation between nouns.

Table 5 shows examples of literal relations extracted by our method. We realized that literal objects are often extracted when the object part is a too common word such as “city” or “town.” We believe that the reason for this lack

Table 4. Examples of extracted explicit relations.

Subject	Predicate	Object
Apple	is	Fruit
Bird	is	Homeothermic
Bird	is	Biped
Cat	is	Mammal
Computer	is	Machine
Isola_d'Asti	is	Comune
Jimmy_Snuka	is	Professional_wrestler
Karwasra	is	Gotra
Mineral_County,_Colorado	is	County
Nava_de_Francia	is	municipality
Sharon_Stone	is	Model
Sharon_Stone	is	Film_producer
Al Capone	was	gangster
Gaius Valerius Catullus	was founded by	Vladimir Lenin
Colorado	is one of	U.S. states
Quartz	is one of	mineral
Djibouti	is bordered by	Eritrea
Djibouti	is bordered by	Ethiopia
Djibouti	is bordered by	Somaliland

of links is just because of the difficulty for making links for a lot of common words. To make these literal relations machine understandable, we have to specify the meaning of these too common words.

Turning now to the accuracy of our proposed method. We realized that some irrelevant semantic relations have been extracted. For example, the semantic relation "[[Niagara Falls]] (carry) vehicles" is extracted from a sentence "it carries vehicles, trains, and pedestrians between Canada." However, the main subject of this sentence is "Whirlpool Rapids Bridge" that appeared in the previous sentence. This is due to the limitation of the co-reference resolution method based on frequent pronouns. Sometimes, "it" or "she/he" are most frequent pronouns but they are not used for the main topic of the article. To confirm the capability of ISP to filter irrelevant semantic relations, we evaluated the precision by specifying thresholds to filter unimportant sentences. Figure 3 shows the result of this evaluation. It is clear that the importance of sentence affects accuracy of semantic relation extraction. This means that our conviction that the sentence importance calculated by link structure analysis is helpful information to filter inaccurate semantic relations is strongly confirmed.

5 Conclusion

In this paper, we proved that Wikipedia is an invaluable corpus for semantic relation extraction by showing both detailed characteristics of Wikipedia and the effectiveness of our proposed method. Furthermore, the results showed that the

Table 5. Examples of extracted literal relations.

Subject	Predicate	Object
Taranto	is	Coastal city
The_Isley_Brothers	is	Black music group
Toronto_Islands	is	Chain
Mauritania	is	Country
Mauritania	is	Country
Ilirska_Bistrica	is	Town
Ilirska_Bistrica	is	Municipality
Brescia	is	City
Bolsheviks	were	Faction
Gaius Valerius Catullus	was	poet

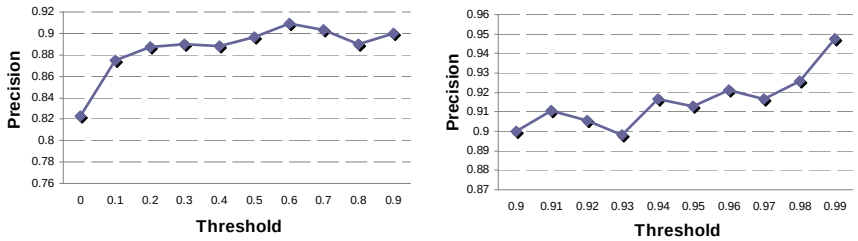

Fig. 3. Precision of ISP by filtering thresholds.

parsing strategies can improve the accuracy and scalability of semantic relation extraction.

More than anything else, the important thing this paper is trying to show is the possibility and capability of semantic relation extraction using Wikipedia knowledge. We believe that this direction will be an influential approach for Semantic Web in near future since it has great capability for constructing a global ontology. The extracted association thesaurus and semantic relations are available on our Web site.

- Wikipedia Lab.
  http://wikipedia-lab.org
- Wikipedia Thesaurus
  http://wikipedia-lab.org:8080/WikipediaThesaurusV2
- Wikipedia Ontology
  http://wikipedia-lab.org:8080/WikipediaOntology

The concrete results will be a strong evidence of the capability of this approach since other Wikipedia mining researches do not provide concrete results on the WWW in most cases. Our next step is to apply the extracted semantic relations to Semantic Web applications (Esp. Semantic Web search). To do that, we need further coverage of relations by enhancing the POS tag analysis patterns and mappings among relations.

6 Acknowledgment

This research was supported in part by Grant-in-Aid on Priority Areas (18049050), and by the Microsoft Research IJARC Core Project. We appreciate helpful comments and advices from Prof. Yutaka Matsuo, the University of Tokyo.

References

1. J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, pp. 900–901, 2005.
2. E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis.," in *Proc. of International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 1606–1611, 2007.
3. D. Milne, O. Medelyan, and I. H. Witten, "Mining domain-specific thesauri from wikipedia: A case study," in *Proc. of ACM International Conference on Web Intelligence (WI'06)*, pp. 442–448, 2006.
4. K. Nakayama, T. Hara, and S. Nishio, "Wikipedia mining for an association web thesaurus construction," in *Proc. of IEEE International Conference on Web Information Systems Engineering (WISE 2007)*, pp. 322–334, 2007.
5. M. Strube and S. Ponzetto, "WikiRelate! Computing semantic relatedness using Wikipedia," in *Proc. of National Conference on Artificial Intelligence (AAAI-06)*, pp. 1419–1424, July 2006.
6. M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *Proc. of International Conference on World Wide Web (WWW 2006)*, pp. 585–594, 2006.
7. S. Chernov, T. Iofciu, W. Nejdl, and X. Zhou, "Extracting semantics relationships between wikipedia categories," in *Proc. of Workshop on Semantic Wikis (SemWiki 2006)*, 2006.
8. D. N. Milne, O. Medelyan, and I. H. Witten, "Mining domain-specific thesauri from wikipedia: A case study," in *Web Intelligence*, pp. 442–448, 2006.
9. F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, (New York, NY, USA), pp. 697–706, ACM, 2007.
10. D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proc. of Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 423–430, 2003.
11. D. P. T. Nguyen, Y. Matsuo, and M. Ishizuka, "Relation extraction from wikipedia using subtree mining," in *Proc. of National Conference on Artificial Intelligence (AAAI-07)*, pp. 1414–1420, 2007.
12. D. Bateman and A. Adler, "Sparse matrix implementation in octave," 2006.