An Intelligent Web Agent to Mine Bilingual Parallel Pages via Automatic Discovery of URL Pairing Patterns

Chunyu Kit and Jessica Yee Ha Ng
Department of Chinese, Translation and Linguistics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong
{ctckit, ctjessng}@cityu.edu.hk

Abstract

This paper describes an intelligent agent to facilitate bitext mining from the Web via automatic discovery of URL pairing patterns (or keys) for retrieving parallel web pages. The linking power of a key, defined as the number of URL pairs that it can match, is used as the objective function for the search for the best set of keys that can find the greatest number of web page pairs within a bilingual website. Our experiments show that, with no prior knowledge such as ad hoc heuristics, no labelled data for training and no similarity analysis of Web page structure and content that are commonly involved in the existing approaches, a best-first search to approximate this optimization with an empirical threshold can recognize 98.1% true parallel web pages and discover many irregular pairing patterns that are unlikely to be discovered by other approaches.

1. Introduction

Parallel corpora (or bitexts) are a kind of valuable resource for many data-driven tasks in natural language processing (NLP), especially for machine translation (MT) and cross-language information retrieval (CLIR). Researchers have been exploring various means to exploit parallel corpora from the Web for years and a good number of Web mining systems, such as STRAND [6, 7, 8], BITS [5], PT-Miner [3], PTI [2] and, more recently, WPDE [11] and a DOM tree alignment model [9], have proved the feasibility of automatic acquisition of bitexts from parallel bilingual web pages. Most of these systems are language independent in principle and thus highly portable to new language pairs, except for the use of a language-specific bilingual lexicon as in BITS and PTI. A typical strategy of Web mining for bitexts from parallel web pages involves four fundamental steps: (1) locate bilingual websites, (2) crawl for URLs of possible parallel web pages, (3) match parallel pages, and (4) extract useful materials from the matched pages to serve different NLP tasks.

In this paper, we focus on (3), the core of the mining process. The previous approaches tackle this issue either by *ad hoc* URL string matching with predefined pairing patterns or via complicated similarity analysis of HTML structure and Web page content, and even sentence alignment. Our approach aims at developing a parallel web page pairing agent to overcome such disadvantages by a fully automatic way of inferring an optimal set of URL pairing patterns for the best match of all possible parallel pages within individual bilingual websites. With no prior knowledge and no labelled data for training, this approach has discovered a large number of URL pairing patterns for true parallel web pages, of which only a small portion could be found by the existing *ad hoc* methods in the field. Its basic assumptions and formulation are given in Section 3 and the details of its implementation in Section 3. Our experiments for evaluation are presented in Section 4. Section 5 concludes the paper with a discussion of our future work.

2. Assumptions and Objective Function

Our approach is based on a number of assumptions. The first one is that parallel web pages are maintained by some pairing patterns instead of random matching. These patterns realize the regular correspondence between the URLs of true parallel pages to ease manual access and maintenance, and may allow machines to recognize them automatically. Technically, a URL consists of four *fields*, namely, protocol, host, path and file name (or simply "file"),each of which can be decomposed into *tokens* by a special set of punctuations, including ":/.-_". Table 1 illustrates how a URL string is decomposed into fields and tokens.

A pairing pattern is a pair of substrings (mostly, URL tokens) in a true pair of URL strings that link them up into a

	Protocol	Host	Path	File
Field	http://	www.aud.gov.hk	/chi/home/	home_c.htm
Token	http	www, aud, gov, hk	chi, home	home, c, htm

Table 1. Fields and tokens in a URL string.

pair. One of the substrings can be null, as in "(null):chinese" for the URL pair "http://www.logisticshk.gov.hk/text.html" and "http://www.logisticshk.gov.hk/chinese/text.html". In fact, pairing patterns of this kind are rather popular in many bilingual websites and need to be handled properly.

The second assumption is that no cross-field linkage is allowed. This is based on our observation that a pairing pattern usually consists of two URL tokens from the same URL field. We are not sure if this would exclude a certain number of true URL pairs. But it does eliminate a great number of unlikely patterns and significantly narrow down the search space for true keys.

The third assumption is that a true pairing pattern is created for as many URL pairs as possible. A corollary that follows this is that the more URLs that a pattern candidate links up into pairs, the more likely it is to be a true pattern. Consequently, we have an objective function for the automatic inference of URL pairing patterns for parallel web pages within a given set of URLs from a bilingual website.

More specifically, we define the goodness, or the *linking power*, of a candidate key in terms of the number of URLs that it can match. Accordingly, the best key is the one with the greatest linking power within a website. We say a URL pair $<u, u'>$ is a *match* by the *key* $<k, k'>$ if $u - k = u' - k'$, with '$-$' to denote substring extraction. In the case of more than one possible way for substring extraction, we assume that any way to satisfy the equation will do. In this case, a key is also referred to as a linkage. A true match is a true pair of parallel web pages that are matched by a key. The only constraint here on URL matching is that a URL can have at most one other URL as its match within a bilingual website.

Our ultimate goal is to search for the best set of keys that cover as many true URL pairs as possible in a given website. Technically we have two options here to approach to this goal, i.e., either to deal with it as an optimization problem to infer the optimal set of keys with the greatest sum of linking power, or to opt for an empirical approach to approximate this optimization via best-first search. For the sake of simplicity and effectiveness, we opt for the latter, which allows various kinds of empirical search strategies e.g., using a threshold to filter out unlikely keys.

3. Implementation

Our implementation involves two individual modules, namely, a URL spider and a pattern recognizer. The spider retrieves URLs from a bilingual web host, and the recognizer discovers pairing patterns among the retrieved URLs and returns a set of URL pairs matched by the pairing patterns that it finds.

The spider crawls for as many URLs (including .htm, .html, .shtml, .asp, .jsp and .php files) as possible within a bilingual host starting from its entrance page. We take a shortcut to language identification, by examining the percentage of English characters (A-Z, a-z) in each web page. A percentage $> 50\%$ signifies an "English" page; otherwise "Chinese". The tokenization process yields two sets of tokens for each URL, one from its path and the other from its file name, henceforth referred to as set P and set F, respectively. The automatic inference of pairing patterns will be carried out within each set independently.

Given a set U of retrieved URL strings and a set T of all URL tokens from U, we define a score function as follows for a key $\kappa = < t, t' > \in T \times T$ to match a candidate URL pair $\pi = < u, u' > \in U \times U$.

$$f(\kappa, \pi) = \begin{cases} 1, \text{ if } u - \{t\} = u' - \{t'\}; \\ 0, \text{ otherwise.} \end{cases} \qquad (1)$$

Note that a URL is represented as a set of URL tokens after tokenization. Then, the linking power of a key κ within U is defined as

$$\mathrm{p}(\kappa) = \sum_{\pi \in U \times U} f(\kappa, \pi) \qquad (2)$$

The complexity of computing $\mathrm{p}(\kappa)$ by brute force enumeration for all $\kappa \in T \times T$ is $O(|T|^2|U|^2)$. A straightforward way to alleviate this complexity is to avoid searching through $T \times T$, because only a small portion of this set are true pairing patterns. Therefore, we opt for the following procedure to accumulate the linking power for each possible key in U.

> For every URL pair $\pi = < u, u' > \in U \times U$,
> If $u - \{t\} = u' - \{t'\}$ (i.e., sharing all but one token),
> Then, $\mathrm{p}(\kappa)$++ for the key candidate $\kappa = <t, t'>$.

On average, the time for this procedure is $O(L^2|U|^2)$, where L is the average number of tokens in the presentation of each $u \in U$. Once insignificant tokens such as htm and html are filtered out, this representation is of a very small number of tokens from T (be it P or F), mostly ≤ 3.

A particularly desirable side-effect of this procedure is that for a key candidate κ that matches at least a URL pair, the set of URL pairs that it matches, denoted as $\mathcal{P}(k)$, can be easily retained for later use. This procedure never bothers with any empty $\mathcal{P}(\kappa)$ for any κ with $\mathrm{p}(\kappa) = 0$.

Then, the final results of Web page pairing is simply the union of all $\mathcal{P}(\kappa)$ for $\mathrm{p}(\kappa) \geq \tau$, an empirical threshold. In order to comply with the constraint on URL pairing, however, we need to carry out the union operation in a way to allow keys to compete against each other. The conventional way has no means to prevent a URL from showing up twice. For example, if we have $< u, v > \in \mathcal{P}(k)$ and $< u, v' > \in \mathcal{P}(k')$, then u would match both v and v' in $\mathcal{P}(k) \cup \mathcal{P}(k')$, violating the constraint. To resolve this problem, a competition rule that a more powerful key has a greater power to match URLs is introduced and exercised by a special union operator $\sqcup$: if $\mathrm{p}(\kappa) > \mathrm{p}(\kappa')$, then $\mathcal{P}(\kappa) \sqcup \mathcal{P}(\kappa') = \mathcal{P}(\kappa)$ plus those in $\mathcal{P}(\kappa')$ sharing no URL within $\mathcal{P}(\kappa)$. Then, we have the final output as

$$\mathcal{P} = \bigsqcup_{\mathrm{p}(\kappa) > \tau} \mathcal{P}(\kappa). \tag{3}$$

4. Evaluation

Our experiments are carried out on 20 HK government websites arbitrarily selected from the Government Web Index at http://www.info.gov.hk/chorgdex.htm. By crawling through them, a total of 39,452 URLs are acquired. Among them, the smallest one contains 67 web pages and the largest 10,947. Our program for automatic URL pairing, following the algorithm given above, is applied to identify true parallel web pages via recognizing the pairing patterns within each of these websites. The experimental results are evaluated manually. Linkages between text-only pages and normal pages are also considered as true pairing as long as they are translation for each other. The evaluation results are presented in Table 2 in terms of the conventional precision, recall and F-measure.

Following the strategy set out above, all linkages with a linking power below a threshold are screened out. In theory, a larger threshold raises precision but lowers recall, and a smaller one does exactly the other way around. In the hope of achieving a nice balance between precision and recall for an optimal F-score for over performance, the threshold in all our experiments is empirically set to $1/10$ of the total number of URLs in each website. The number of true URL pairs is calculated as the sum of the retrieved true URL pairs and the number of manually found URL pairs in the remaining unpaired URLs. In total, 7 and 82 true keys are missed in Set P and Set F, respectively. The former loses 17 true URL pairs and the later 209. The relatively large number of the latter has to do with the tokenization for file names. For example, quite a number of URLs are matched by very weak keys such as “06e:06c”, “07e:07c” and “08e:08c”, instead of by a stronger one “e:c” that they share. These weak keys are usually filtered out by the threshold, and thus cannot contribute anything to the recall.

The precisions of pairing patterns in both path (Set P) and file name (Set F) are not so impressive, only of 66.67% and 69.44%, respectively, but the precisions of the URL pairs matched by these patterns show a remarkably sharp contrast, of 94.55% and 95.27%. This result indicates that although a certain number of false patterns, as presented in Table 2, are unexpectedly recognized as true, they only lead to a relatively small number of false URL pairs, confirming one of our assumptions that the true keys match most true URL pairs and the false ones match very few.

Category		Path (Set P)		File (Set F)		Path+File	
		Key	URL	Key	URL	Key	URL
True		51	7,524	107	4,201	158	11,725
Re-triev-ed	Total	66	7,940	36	4,190	102	12,130
	True	44	7,507	25	3,992	69	11,499
	False	22	433	11	198	33	631
Precision (%)		66.67	94.55	69.44	95.27	67.65	94.80
Recall (%)		86.27	99.77	23.36	95.02	43.67	98.07
F-score (%)		75.21	97.09	34.97	95.15	53.08	96.41

Table 2. Evaluation results.

The recall of pairing patterns in path, of 86.27%, is much higher than that in file name, of 23.36%, indicating a significantly stronger correspondence regularity in the former than in the latter for matching parallel web pages. This stronger regularity allows our program to demonstrate a better overall performance in recognizing pairing patterns in path than in file name, in terms of the F-scores in these two categories, namely, 75.21% vs. 34.97%. According to Table 2, about 2/3 of true URL pairs are paired up by path and only 1/3 by file name, revealing the web developers’ preference in linking up the parallel pages within these websites.

Since our approach is aimed at identifying parallel web pages, it is thus more significant to evaluate its performance in terms of its URL pairing outputs. Accordingly, recall is more important than precision, for it measures the success rate in retrieving true pairs. In general, the more true pairs successfully retrieved, the more bitexts we can acquired. Although a certain number of true pairing patterns are missed due to their weak linking power below the threshold, the URLs paired up by all other patterns recognized by our program grant very high recalls, of 99.77% and 95.02%, respectively, on matching the paths and file names of true parallel pages. Surprisingly, only 17 (among 7524) are missing for the former and only 209 (among 4201) for the latter, showing a very promising result.

Among the 20 websites involved in our experiments, all of them are found to have pairing patterns in path, but only 7 of them have pairing patterns in file name. In total, 29 and 19 true pairing patterns are found in the fields of path and file, respectively. This confirms again that parallel web pages are paired up more by means of path than file name. Table 3 shows the true pairing patterns (with usage frequencies) discovered by our program, demonstrating a vast variety of pairing patterns used by various web page developers. A few commonly used semantic patterns can also be found within these two sets of pairing keys, mostly by

Pairing patterns in path (Set P)		
en : tc (3122)	ehtml : zhtml (106)	en : textsc (26)
eng : chi (1270)	(null) : chinese (90)	en : texttc (26)
en : hk (750)	en : gb (68)	EN : TC (18)
english : tc_chi (395)	(null) : chi (45)	eng : ctext (17)
english : sc_chi (392)	ekey : ckey (31)	etext : chi (17)
english : chinese (366)	en : sc (28)	eng_text : big5_text (16)
en : ch (173)	texten : sc (28)	eng_text : gb_text (16)
eng : b5 (145)	texten : tc (28)	eng_text : gb (14)
eng : cn (144)	texten : texttc (28)	eng_text : big5 (14)
ehtml : chtml (106)	texten : textsc (28)	
Pairing patterns in file name (Set F)		
(null) : c (1303)	regione : regionc (26)	vismape : vismapc (13)
e : c (799)	12e : 12c (26)	08e : 08c (12)
(null) : tc (791)	09e : 09c (22)	01e : 01c (11)
t : tc (788)	06e : 06c (19)	05e : 05c (10)
1e : 1c (46)	02e : 02c (14)	hko : chko (10)
2e : 2c (46)	03e : 03c (13)	
3e : 3c (40)	07e : 07c (13)	

Table 3. Pairing patterns automatically found.

using various prefixes of a language name to represent that language, e.g., "e","en" and "eng" for English vs. "c", "ch" and "chi" for Chinese. Other variants are related to text coding, e.g.,"b5", "big5" and "tc" for traditional Chinese in Big5 and "gb" and "sc" for simplified Chinese in GB.

Interestingly, however, none of these patterns is shared by the two sets. A few are similar but not identical. This fact indicates that web page developers tend to create pairing keys for their own pages mostly at their own will instead of following any consensus or convention. This tendency creates a large variety of pairing patterns with much irregularity, e.g., "ehtml:zhtml", "en:hk" and "texten:sc/textsc/texttc/tc", that are unlikely to be properly dealt with by the existing *ad hoc* heuristics ever used in any previous approach. It is unlikely to have adequate heuristics to cover so many possible patterns that people can arbitrarily create at will. Only an automatic discovery approach, e.g., the one illustrated above, can cope with them satisfactorily.

5. Conclusion

In this paper, we have presented the basic ideas and technical details to implement an intelligent Web agent to automatically discover arbitrary URL pairing patterns for identifying parallel web pages in bilingual web sites. This approach is amazingly simple, elegant and effective, for it is based on automatic string pattern recognition, instead of prior knowledge such as *ad hoc* heuristics, labelled data for training, and complicated similarity analysis of Web page structure or content that are popularly involved in the previous approaches. It has illustrateed a remarkable success, retrieving 98.07% true parallel web pages through the 43.67% true URL pairing patterns correctly discovered. This outcome confirms our observation that most true URL pairs are matched by true patterns and very few by false ones.

Our future work will explore the following directions: (1) rescue weak keys by generalizing them into stronger ones, e.g., "e:c" at the bottom of Table 3, to avoid being filtered out by the threshold, aimed as enhancing the recall of web page pairing; (2) depress false keys via key competition or global optimization (also by the competition rule) to infer the optimal set of keys, aimed at improving the precision of web page pairing; (3) extract bitexts from the recognized web page pairs, via various kinds of text alignment techniques, to facilitate relevant NLP applications such as MT and cross-language IR; and (4) extend the current approach to mine parallel multi-lingual web pages to facilitate multi-lingual NLP tasks. Also, the fact in our experiments that URL pairing by keys in file name is less reliable than by those in path also points us to another possible improvement upon our current work.

References

[1] G. Cavaglia and A. Kilgarriff. Corpora from the Web. In *Proceeding of the 4th Annual CLUCK Colloquium*, pages 120–124, Sheffield, UK, January 2001.

[2] J. Chen, R. Chau, and C.-H. Yeh. Discovering parallel text from the World Wide Web. In *Proceedings of the 2nd Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 157–161, Dunedin, New Zealand, 2004.

[3] J. Chen and J.-Y. Nie. Parallel Web text mining for cross-language IR. In *RIAO-2000: Content-Based Multimedia Information Access*, pages 62–77, Paris, 12-14 April 2000.

[4] A. Kilgarriff and G. Grefenstette. Introduction to the special issue on the Web as corpus. *Computational Linguistics*, 29(3):333–348, 2003.

[5] X. Ma and M. Y. Liberman. Bits: A method for bilingual text search over the Web. In *MT Summit VII*, pages 13–17, Singapore, 1999.

[6] P. Resnik. Parallel strands: A preliminary investigation into mining the Web for bilingual text. In *AMTA-98*, pages 72–82, Langhorne, PA, 28-31 October 1998. Lecture Notes in Artificial Intelligence, Vol. 1529, Springer.

[7] P. Resnik. Mining the Web for bilingual text. In *ACL-99*, pages 527–534, Maryland, 20-26 June 1999.

[8] P. Resnik and N. Smith. The Web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, 2003.

[9] L. Shi, C. Niu, M. Zhou, and J. Gao. A DOM tree alignment model for mining parallel data from the Web. In *COLING/ACL-2006*, pages 489–496, Sydney, 17-21 July 2006.

[10] A. Way and N. Gough. wEBMT: Developing and validating an example-based machine translation system using the World Wide Web. *Computational Linguistics*, 29(3):421–457, 2003.

[11] Y. Zhang, K. Wu, J. Gao, and P. Vines. Automatic acquisition of Chinese-English parallel corpus from the Web. In *Proceedings of 28th European Conference on Information Retrieval*, pages 420–431. Lecture Notes in Computer Science, Vol. 3936, Springer, January 2006.