Statistical-Based Approaches for Non-Segmenting Languages

Virach Sornlertlamvanich
Thai Computational Linguistics Laboratory, NICT
112 Phahon Yothin Road, Klong Nueng, Kong Luang,
Pathumthani 12120, Thailand
Email: virach@tcllab.org

Abstract

We have been studying several approaches to cope with the exceptional features in non-segmenting languages. When there is no explicit information about the boundary of a word, parsing an input text is a formidable task. Not only the contemporary word list, but also the word usages have to be maintained to cover the use in current texts. The accuracy and efficiency in higher processing do heavily rely on the word identification process. In this paper, we introduce some statistical based approaches to tackle the problem due to the ambiguity in word segmentation. The word identification problem is then defined as a part of other for performing the unified language processing in total. To exhibit the ability in conducting the unified language processing, we selectively study the tasks of language identification, word extraction, dictionary-less search engine and term-based ontology alignment.

Key Words: non-segmenting language, unified language processing, statistical approach, probability, language identification, word extraction, search engine, ontology alignment

1. Introduction

Disambiguation is our major concern in natural language processing. Though the morphological and syntactic information play an important role in assisting the disambiguation process, degree of the allowing information can vary according to the type of the language. For instance, a space character between words reduces the task in identifying word boundary. Similarly, the grammatical markers, inflection and punctuation marks are quite meaningful information for identifying the role of each word and structural relations between words in a sentence. Consequently, it is natural to say that many approaches in language processing have been studied in terms of word units. A word unit is definitively a unit of language that native speakers can identify. Based on the classical approaches, without knowing the entity of word, it is not so efficient to develop the language model. For the language that does not allow to use a space character (or any special markers) to separate words in natural text, so-called a non-segmenting language, i.e. Thai, Chinese, Japanese, Korean, etc., a high performance word segmentation algorithm to identify word boundary is crucial. The performance of language processing totally relies on the efficiency and the accuracy of the word segmentation algorithm.

In our recent research, we proposed a language interpretation model to deal with an input text as a byte sequence rather than a sequence of words. It is an approach to unify the language processing model to cope with the ambiguities in word determination problem. The approach takes an input text in the earliest stage of language processing when the exhaustive recognition of total word identity is not necessary.

In this paper, we present the achievements in identifying language based on the study of 20 different languages, word candidates extraction based on the unified input byte sequence, indexing algorithm for full text retrieval applied in a search engine, and term based ontology alignment. Our experiments also show promising results for overcoming the drawbacks of the non-segmenting language.

2. Language Identification

Language identification is yet another challenging task when it is going to be conducted without any grammatical knowledge. Byte sequence is the only magic key in our approach to determine the language of the input text. We introduce string kernel for this language identification task. We conducted experiments using 2 kernelized versions of centroid-based method and support vector machine (SVM). The accuracies of identification are acceptable for both methods. The accuracies reach 95 percent with only 10 training sets (2 KB per set). It is also found that the simple centroid-based classifier is comparable to the SVM classifier based on the string kernel.

String kernel is introduced to compute the common subsequences of two strings. A kernel can be simply thought of as the inner product function between two vectors, $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$. With the recent advance in kernel methods, the kernel computation is not just limited to vectorial objects, but can be performed on sequences based on the so-called string kernels [2][13].

To describe how to compute the string kernel based on our proposed efficient method of using fast matching with suffix trees, we consider two strings of $\boldsymbol{u} = yzxxz$ and $\boldsymbol{v} = xyzxxxy$, where the range of considering substring length (r) is $1 \leq r \leq 2$ and the

set of characters (Σ) is Σ={x,y,z}. Figure 1 demonstrates the suffix tree for the string ***v*** = *xyzxxxy*.

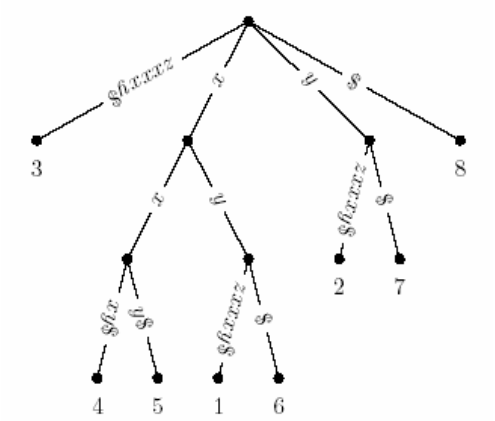

*Figure 1: Suffix tree for the string **v** = xyzxxxy$*

The following table lists all the matched between ***u*** and ***v***, where the number of all occurrences of substrings in common is given in parentheses.

u	v[i]	Score
y	y(2), yz(1)	$2\cdot\lambda^{2\cdot1} + 1\cdot\lambda^{2\cdot2} = 2\lambda^2 + \lambda^4$
z	z(1), zx(1)	$1\cdot\lambda^{2\cdot1} + 1\cdot\lambda^{2\cdot2} = \lambda^2 + \lambda^4$
x	x(4), xx(2)	$4\cdot\lambda^{2\cdot1} + 2\cdot\lambda^{2\cdot2} = 4\lambda^2 + 2\lambda^4$
x	x(4), xz(0)	$4\cdot\lambda^{2\cdot1} = 4\lambda^2$
z	z(1)	$1\cdot\lambda^{2\cdot1} = \lambda^2$

As a result, we can compute the string kernels with the computational complexity of O(c|***u***|+|***v***|).

We conducted experiments on 4 groups of languages i.e. *Multi4* (Thai, Chinese, Japanese, Korean), *Multi8-1* (English, French, Italian, Portuguese, Spanish, Swedish, German, Hungarian), *Multi8-2* (Czech, Polish, Croatian, Slovak, Slovenian, Bulgarian, Russian, Greek) and *Multi20* (all the 20 languages). The results are shown in Figure 1-5.

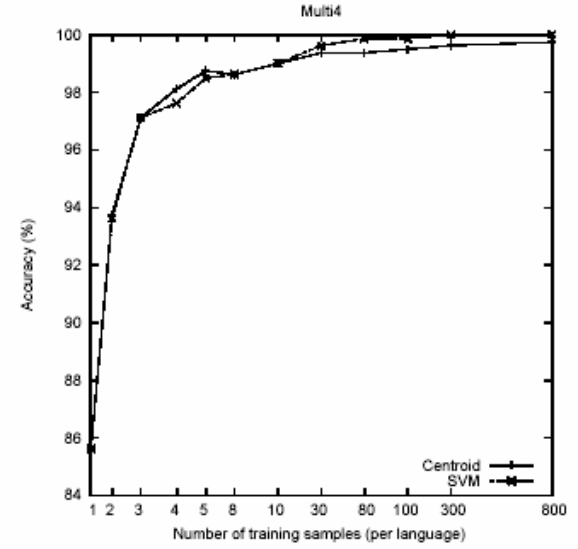

Figure 2: Classification accuracy on Multi4

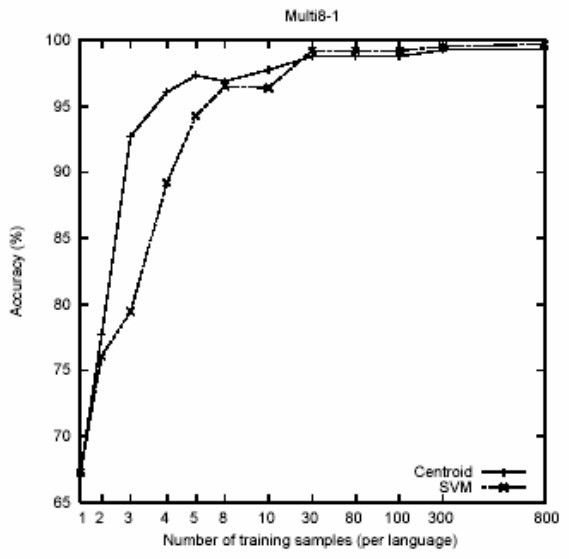

Figure 3: Classification accuracy on Multi8-1

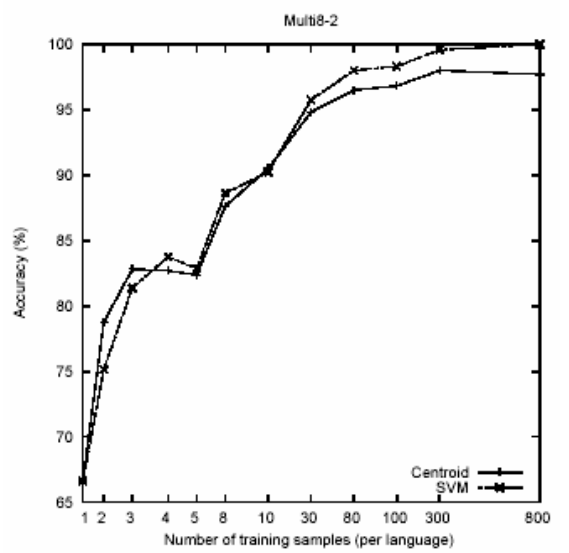

Figure 4: Classification accuracy on Multi8-2

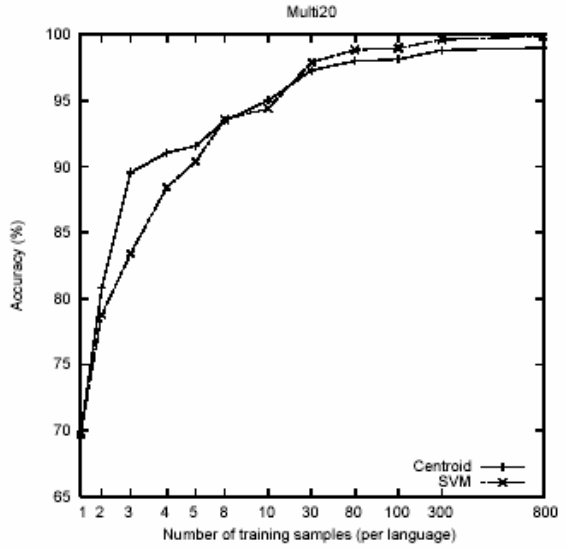

Figure 5: Classification accuracy on Multi20

In total, comparing to centroid-based method, SVM classifier yields slightly better performance when given more training samples, while the centroid-based classifier performs better for small numbers of training samples.

We hypothesize that, for the SVM classifier, only the critical support vectors are retained after the training process, whereas the centroid-based classifier can exploit the feature combination of the representative centroid that is the mean vector of all training samples. However, both had shown significant results in identifying languages of whatever groups of the languages.

3. Word Extraction

It is always arguable about what should be an appropriate word list for a non-segmenting language. There is no any explicit rule for explaining what a word looks like. Semantically defining that word is a unit of language that native speakers can identify is still vague. It mostly depends on individual's perception about word i.e. 'bookstore' vs. 'book store', 'open source' vs. 'opensource', 'newspaper' vs. 'news paper', etc..

A preliminary study of co-occurrence of substring has shown a promising result in extracting open compounds from text corpora [11]. The significant change in occurring frequency of a substring when expanded has invoked the possible observation of word boundary. We proposed another method for automatic word extraction from raw texts based on the algorithm that reflected the understanding of word being a string frequently used in most part of the texts [12]. We employed the C4.5 decision tree induction program [5] as the learning algorithm for word extraction. The induction algorithm proceeds by evaluating content of a series of attributes and iteratively building a tree from the attribute values with the leaves of the decision tree being the value of the goal attribute. At each step of learning procedure, the evolving tree is branched on the attribute that partitions the data items with the highest information gain. Branches will be added until all items in the training set are classified. To reduce the effect of overfitting, C4.5 prunes the constructed entire decision tree. It recursively examines each subtree to determine whether replacing it with a leaf or branch would reduce expected error rate. This pruning makes the decision tree better in dealing with the data different from the training data.

We treat the word extraction problem as the problem of word or non-word string disambiguation. The next step is to identify the attributes that are able to disambiguate word strings from non-word strings. The attributes used for the learning algorithm are as follows.

(i) Left Mutual Information and Right Mutual Information

The left mutual information (*Lm*), and right mutual information (*Rm*) of string xyz are defined as:

$$Lm(\boldsymbol{xyz}) = \frac{p(\boldsymbol{xyz})}{p(\boldsymbol{x})\,p(\boldsymbol{yz})}$$

$$Rm(\boldsymbol{xyz}) = \frac{p(\boldsymbol{xyz})}{p(\boldsymbol{xy})\,p(\boldsymbol{z})}$$

where

x is the leftmost character of ***xyz***

y is the middle substring of ***xyz***

z is the rightmost character of ***xyz***

p( ) is the probability function.

If *xyz* is a word, both *Lm(xyz)* and *Rm(xyz)* should be high. On the contrary, if *xyz* is a non-word string but consists of words and characters, either of its left or right mutual information or both must be low. For example, 'กปรากฏ' ('ก' (a Thai alphabet) + 'ปรากฏ' (The word means 'to appear' in Thai.)) must have low left mutual information.

(ii) Left Entropy and Right Entropy

Entropy [6] is the information measuring disorder of variables. The left and right entropy is exploited as another two attributes in our word extraction. Left entropy (Le), and right entropy (Re) of string y are defined as:

$$Le(\boldsymbol{y}) = -\sum_{\forall x \in \boldsymbol{A}} p(x\boldsymbol{y} \mid \boldsymbol{y}) \cdot log_2\, p(x\boldsymbol{y} \mid \boldsymbol{y})$$

$$Re(\boldsymbol{y}) = -\sum_{\forall z \in \boldsymbol{A}} p(\boldsymbol{y}z \mid \boldsymbol{y}) \cdot log_2\, p(\boldsymbol{y}z \mid \boldsymbol{y})$$

where

y is the considered string,

A is the set of all alphabets

x, z is any alphabets in ***A***.

If y is a word, the alphabets that come before and after y should have varieties or high entropy. If y is not a complete word, either of its left or right entropy, or both must be low. For example, 'ปราก' is not a word but a substring of word 'ปรากฏ' (appear). Thus the choices of the right adjacent alphabets to 'ปราก' must be few and the right entropy of 'ปราก', when the right adjacent alphabet is 'ฏ', must be low.

(iii) Frequency

It is obvious that the iterative occurrences of words must be higher than those of non-word strings. String frequency is also useful information for our task. Because the string frequency depends on the size of corpus, we normalize the count of occurrences by dividing by the size of corpus and multiplying by the average value of Thai word length:

$$F(\boldsymbol{s}) = \frac{N(\boldsymbol{s})}{Sc}.Avl$$

where

s is the considered string

*N(**s**)* is the number of the occurrences of ***s*** in corpus

Sc is the size of corpus

Avl is the average Thai word length.

We employed the frequency value as another attribute for the c4.5 learning algorithm.

(iv) Length

Short strings are more likely to happen by chance than long strings. Then, short and long strings should be treated differently in the disambiguation process. Therefore, string length is also used as an attribute for this task.

(v) Functional Words

Functional words such as 'จะ' (will) and 'ก็' (then) are frequently used in Thai texts. These functional words are used often enough to mislead the occurrences of string patterns. To filter out these noisy patterns from word extraction process, discrete attribute Func(s):

Func(s) = 1 if string s contains functional words,
= 0 if otherwise,

is applied.

(vi) First Two and Last Two Characters

A very useful process for our disambiguation is to check whether the considered string complies with Thai spelling rules or not. We employ the words in the Thai Royal Institute dictionary as spelling examples for the first and last two characters. Then we define attributes and for this task as follows.

$$Fc(\boldsymbol{s}) = \frac{N(s_1 s_2 *)}{ND}$$

$$Lc(\boldsymbol{s}) = \frac{N(* s_{n-1} s_n)}{ND}$$

where

$\boldsymbol{s}$ is the considered string and $\boldsymbol{s} = s_1 s_2 \ldots s_{n-1} s_n$

$N(s_1 s_2 *)$ is the number of words in the dictionary that begin with $s_1 s_2$

$N(* s_{n-1} s_n)$ is the number of words in the dictionary that end with $s_{n-1} s_n$

ND is the number of words in the dictionary.

We apply [14]'s algorithm to extract all strings from a plain and unlabelled 1-MB corpus which consists of 75 articles from various fields. For practical and reasonable purpose, we select only the 2 to 30 character strings that occur more than 2 times, have positive right and left entropy, and conform to simple Thai spelling rules. To this step, we get about 30,000 strings. These strings are manually tagged as words or non-word strings. The strings' statistics explained above are calculated for each string. Then the strings' attributes and tags are used as the training example for the learning algorithm. The decision tree is then constructed from the training data.

In order to test the decision tree, another plain 1-MB corpus (the test corpus), which consists of 72 articles from various fields, is employed. All strings in the test corpus are extracted and filtered out by the same process as used in the training set. After the filtering process, we get about 30,000 strings to be tested. These 30,000 strings are manually tagged in order that the precision and recall of the decision tree can be evaluated.

To measure the accuracy of the algorithm, we consider two statistical values: precision and recall. As shown in Table 1 and 2, the precision of our algorithm is 87.3% for the training set and 84.1% for the test set. The recall of extraction is 56% in both training and test sets. We compare the recall of our word extraction with the recall from using the Thai Royal Institute dictionary (RID). The recall from our approach and from using RID are comparable and our approach should outperform the existing dictionary for larger corpora. Both precision and recall from training and test sets are quite close. This indicates that the created decision tree is robust for unseen data. Table 3 also shows that more than 30% of the extracted words are not found in RID. These would be the new entries for the dictionary.

Table 1: The precision of word extraction

	No. of strings extracted by the decision tree	No. of words extracted	No. of non-word strings extracted
Training Set	1882 (100%)	1643 (87.3%)	239 (12.7%)
Test Set	1815 (100%)	1526 (84.1%)	289 (15.9%)

Table 2: The recall of word extraction

	No. of words that in 30,000 strings extracted	No. of words extracted by the decision tree	No. of words in corpus that are found RID
Training Set	2933 (100%)	1643 (56.0%)	1833 (62.5%)
Test Set	2720 (100%)	1526 (56.1%)	1580 (58.1%)

Table 3: Words extracted by the decision tree and RID

	No. of words extracted by the decision tree	No. of words extracted by the decision tree which is in RID	No. of words extracted by the decision tree which is not in RID
Training Set	1643 (100.0%)	1082 (65.9%)	561 (34.1%)
Test Set	1526 (100.1%)	1046 (68.5%)	480 (31.5%)

The attributes of such the character-based mutual information and entropy provide significant information to C4.5 algorithm for selecting appropriate candidates for words. The approach greatly supports the process of nominating word candidates for developing a dictionary, and later is extended to fulfill a dictionary-less search engine [10]. The search engine has introduced a word score as a heuristic value to determine the word likelihood of a string. The word score is a normalized value of a mutual information value. The minimum score of the left and right hand side of a string in question is assigned as the word score of the string. Based on the proposed approach, we successfully implemented a multi-lingual search engine with minimum modification.

4. Dictionary-less Search Engine

The performance of dictionary-based search engines is directly affected by the accuracy of word segmentation algorithms. Our previous work [10] discussed about two possible errors affected by the accuracy of dictionary-based word segmentation modules. Assuming that a dictionary contains 6 words: **a**, **b**, **c**, **ac**, **bc** and **cb**.

Case 1: Incorrect word segmentation

The content of the document ***A*** is **abcbcb**. By using a word segmentation module, the content is separated into **a**|**bc**|**bc**|**b**. Assuming that the correct segmentation is **a**|**b**|**cb**|**cb**. If the query is **cb**, it cannot be found in the document ***A*** or if the query is **bc**, the document ***A*** will be incorrectly returned.

Case 2: Unregistered word problem

The content of document ***A*** is **abcdac**. By using the word segmentation module, the content is separated into **a**|**bc**|**d**|**ac**. Assuming that the correct segmentation is **a**|**b**|**cd**|**ac** and **cd** is an unregistered word to the word segmentation. If the query is **bc**, the result from this document will be incorrect or if the query is **cd**, it cannot be found in the document ***A***.

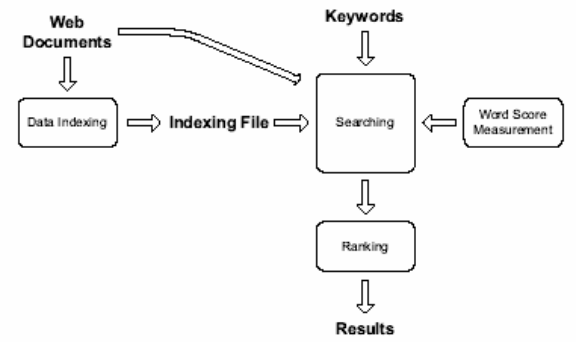

Figure 6: Dictionary-less Search Engine

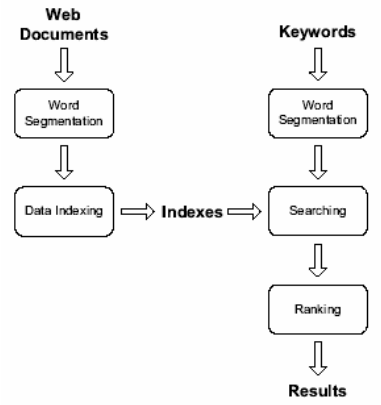

Figure 7: Dictionary-based Search Engine

The architecture of our dictionary-less search engine is illustrated in Figure 6 comparing to the typical dictionary-based search engine in Figure 7. It is composed of 3 major modules: (1) data indexing, (2) searching and (3) document ranking.

(1) Data Indexing

In typical search engines, web documents are separated into words to provide a word list for generating the indexes. In our approach, the data is considered to be the sequence of characters and indexed character by character. We adopt the enhanced suffix array [14] for indexing the data. All suffixes of the data string are indexed. Thus, the number of indexes is equal to the data size. The advantage of this indexing method is that it guarantees all search strings to be found, whereas the word indexing method depends on the word segmentation. This indexing method can also be applied to other languages since it does not require any dictionary and language-specific knowledge.

Based on the enhance suffix array, it requires *O(mlogN)* to access the string in the data, where *m* is a length of the search string and *N* is the number of indexes. The use of the suffix array guarantees that all search strings will be found. However, only the meaningful strings are preferred. If the found pattern is a part of other word, that pattern is inseparable. As a result, it is not valid as a meaningful word.

For example, assuming that the search query is short and likely to be a part of other strings such as "ยา" (drug), two strings are found i.e. (1) "กินยา" (take a drug) and (2) "พัทยา" (Pattaya, name of a district in Thailand). The first string can be separated into two words: (1) "กิน" (take, eat) and (2) "ยา" (drug). Thus, the word "ยา" in the first string is a meaningful word. For the second string, the first part "พัท" is a meaningless string and is strongly connected to the second part "ยา". Thus, the word "ยา" in the second string is meaningless since this string is inseparable.

From the example, the validity of a word can be decided from its surrounding context. If the word is strongly connected to other word and inseparable, it is likely to be a meaningless string. In contrast, the word is likely to be a meaningful string if it is loosely connected to other word and separable.

We use mutual information (MI) [1] to measure the degree of the co-occurrence of the query and its context. Let *xy* be a query, *ab* is the left context and cd is the right context of the string *xy*, the mutual information can be determined by the Equation 1-4.

$$MI_L(abxy) = \frac{p(abxy)}{p(ab)p(xy)} \quad (1)$$

$$MI_L(abxy) \approx \frac{Count(abxy)}{Count(ab)Count(xy)} \quad (2)$$

$$MI_R(xycd) = \frac{p(xycd)}{p(xy)p(cd)} \quad (3)$$

$$MI_R(xycd) \approx \frac{Count(xycd)}{Count(xy)Count(cd)} \quad (4)$$

If the MI value is high, *xy* is likely to be a part of the context. On the other hand, *xy* should be

independent from the context if the MI value is low. We define the inverse of MI as the word score. The word score is calculated by the Equation 5-6.

$$wscore_L(xy|ab) = 1 - norm(MI_L(abxy)) \quad (5)$$

$$wscore_R(xy|cd) = 1 - norm(MI_R(xycd)) \quad (6)$$

The $norm(\cdot)$ is the normalizing function which normalizes the argument from 0 to 1. At this point, the word score determines the probability of being a word of the string.

(3) Document Ranking

The word score from the previous step is not only used to determine the word boundary, it is also used to rank the document. That is, the document with higher word score will attain high rank.

We conduct an experiment in order to compare the dictionary-less search engine with the dictionary-based search engine. The experiment is based on the article [3]. We assign 20 queries to 5 volunteers. For each query, the volunteer is shown the top 10 results from each system. Then, each volunteer will choose the documents that are relevant to the query. Each result is considered to be relevant if at least 3 of the 5 volunteers assigned it as relevant for the query. Finally, the satisfaction score of both systems will be calculated for each query. We define the satisfaction score as the ratio of the relevant results to all available results.

The web documents used in the experiment can be divided into two groups. The first group is obtained from websites of newspapers, consisting of 5,853 documents (approximately 65 Mb). The second group contains general articles, not related to news. The second group consists of 7,710 documents (approximately 35 Mb).

The test queries are listed in Table 4. The queries are related to news. Thus, only results from the first group are preferred. The satisfaction score of two systems for each query is presented in Figure 8. From the figure, the mean of the dictionary-less search engine is higher than that of the dictionary-based approach. Furthermore, there are 10 queries that the dictionary-less approach is better than the dictionary-based approach, while the results of 5 queries are equal and the dictionary-based approach achieves higher results on other 5 queries.

We observe that the dictionary-based search engine faces difficult situation when the query is excessively segmented, and the segmented words are likely to be general terms. For example, one of the queries is "การส่งออก" (export). The word segmentation module separates this word as "การ | ส่ง | ออก". All three terms still have some meaning in Thai, but not directly relevant to the compound word. Moreover, these terms are general words and often parts of several words. Thus, the dictionary-based approach tends to return irrelevant documents, but have several locations of these general terms.

Another observation is that the incorrect segmentation does not always affect the search performance of the dictionary-based search engine. For example, a part of one query is "ผลกระทบ" (effect). It is incorrectly segmented into "ผลก | ระ | ทบ". All three terms are meaningless and not general terms. However, the dictionary-based search engine still effectively discovers these terms since some of this terms are quite unique. We also observe that the correctness of word segmentation is less important than the generality of segmented words. When the query is excessively segmented, the dictionary-based search engine still performs well if the segmented terms are not quite general. In contrast, the dictionary-based search engine tends to return irrelevant documents if the query is excessively segmented and the segmented words are general terms. This also explains why the dictionary-based search engine performs better on some queries. Although those queries are sometime incorrectly segmented, the dictionary-based search engine still finds related documents. The reason is that the segmented words are not general. Thus, these words are easily found.

Table 4: Queries

	Unsegmented queries	*Segmented queries*
1	บริจาค, สึนามิ	บริจาค, สึนามิ
2	เส้นตาย, ชัดดม	เส้นตาย, ชัดดม
3	มนัส, โชว์ตัว, จีน	มนัส, โชว์|ตัว, จีน
4	ผลกระทบ, ราคาน้ำมันแพง	ผลก|ระ|ทบ, ราคา|น้ำมัน|แพง
5	ไข้หวัดนก	ไข้หวัด|นก
6	ทุจริต, การเลือกตั้ง	ทุจริต, การ|เลือกตั้ง
7	จับกุม, ผู้ก่อการร้าย, ภาคใต้	จับกุม, ผู้ก่อการร้าย, ภาค|ใต้
8	นโยบาย, แก้ไข, ปัญหายาเสพติด	นโยบาย, แก้ไข, ปัญหา|ยา|เสพติด
9	ทดลอง, ลดค่าทางด่วน	ทดลอง, ลด|ค่า|ทางด่วน
10	ซื้อคืน, สัมปทาน, รถไฟฟ้า	ซื้อ|คืน, สัมปทาน, รถไฟฟ้า
11	ลงทุน, ในพม่า	ลงทุน, ใน|พม่า
12	ส่งเสริม, การท่องเที่ยว, ไทย	ส่งเสริม, การ|ท่องเที่ยว, ไทย
13	เลือกตั้ง, ประธานาธิบดี, ปาเลสไตน์	เลือกตั้ง, ประธานาธิบดี, ปา|เลสไตน์
14	เลือกตั้ง, ผู้ว่า, กทม.	เลือกตั้ง, ผู้|ว่า, กทม|.
15	สินค้าไทย, การส่งออก	สินค้า|ไทย, การ|ส่ง|ออก
16	แปรรูปรัฐวิสาหกิจ	แปรรูป|รัฐวิสาหกิจ
17	อุ้ม, นายสมชาย	อุ้ม, นาย|สม|ชาย
18	ฉลองปีใหม่	ฉลอง|ปี|ใหม่
19	พรทิพย์, ลาออก	พร|ทิพย์, ลา|ออก
20	สวนสนุก	สวน|สนุก

5. Term-Based Ontology Alignment

The shortage of language resources is potentially preventing our research in statistical based approach. A large enough corpus is necessary to capture the language model. It is obvious that the English language is majorly used in any forms, and almost all the other languages have some information related to the English language e.g. bi-lingual texts, bi-lingual dictionaries. The efforts to utilize the advantages of the English language are to increase the knowledge

about the target languages. We proposed a term-based ontology alignment [9][7] to increase our language and terminology resources via the English language resources. Words in the classes are used to create a vector model for each class. By computing the extended Jaccard similarity value, we are able to align the 2 concept hierarchies of Thai and English. We are extending our research to use English as a common language for aligning the concept hierarchies that do not have the direct link.

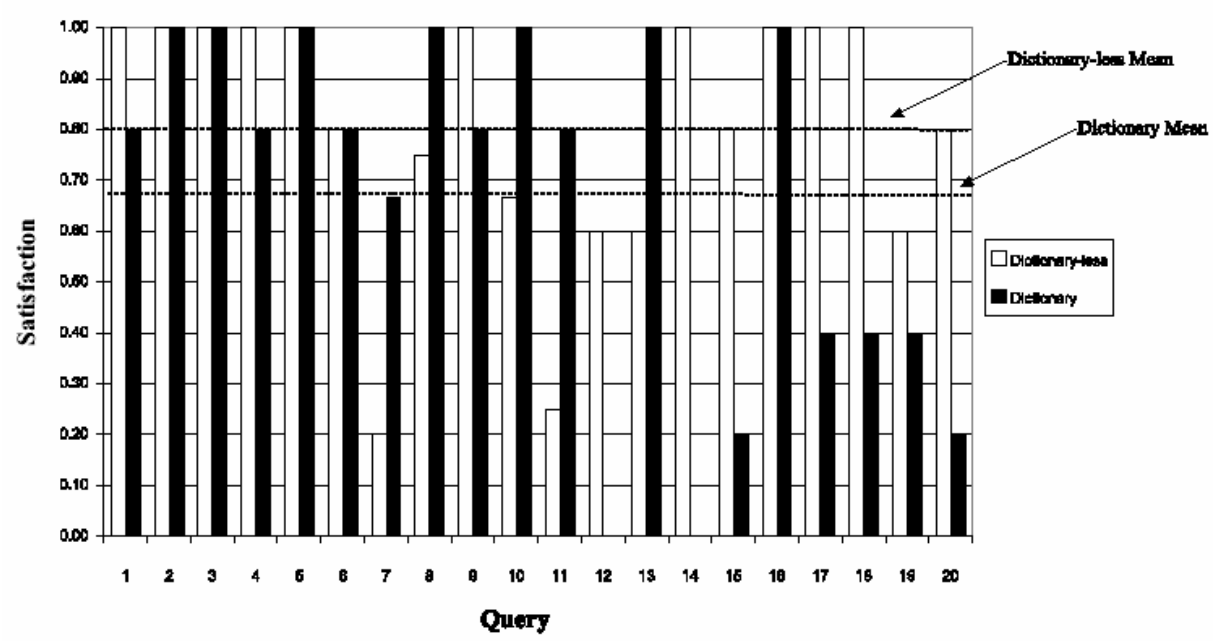

Figure 8: Satisfaction for the test queries

Given two ontologies called the source ontology T_s and the target ontology T_t, our objective is to align all the concepts (or semantic classes) between these two ontologies. Each ontology consists of concepts, denoted by $C_1, \ldots, C_k$. In general, the concepts and their corresponding relations of each ontology can be significantly different due to the theoretical background used in the construction process. However, for the lexical ontologies such as the MMT semantic hierarchy and the EDR concept dictionary, it is possible that the concepts may contain shared members in terms of English words. Thus, we can match the concepts between two ontologies using the similarity of the shared words.

In order to compute the similarity between two concepts, we must also consider their related child concepts. Given a root concept C_i, if we flatten the hierarchy starting from C_i, we obtain a nested cluster, whose largest cluster dominates all sub-clusters. As a result, we can represent the nested cluster with a feature vector $c_i = (w_1, \ldots, w_{|V|})^T$, where features are the set of unique English words V extracted from both ontologies, and w_j is the number of the word j occurring the nested cluster i. We note that a word can occur more than once, since it may be placed in several concepts on the lexical ontology according to its sense.

After concepts are represented with the feature vectors, the similarity between any two concepts can be easily computed. A variety of standard similarity measures exists, such as the *Dice coefficient*, the *Jaccard coefficient*, and the cosine similarity [4]. In our work, we require a similarity measure that can reflect the degree of the overlap between two concepts. Thus, the Jaccard coefficient is suitable for our task. Recently, Strehl and Ghosh [8] have proposed a version of the Jaccard coefficient called the *extended Jaccard similarity* that can work with continuous or discrete non-negative features. Let $\|\boldsymbol{x_i}\|$ be the L_2 norm of a given vector x_i. The extended Jaccard similarity can be calculated as follows:

$$JaccardSim(x_i, x_j) = \frac{x_i^T x_j}{\|x_i\|^2 + \|x_j\|^2 - x_i^T x_j}$$

We now describe an iterative algorithm for term-based ontology alignment. A_s mentioned earlier, we formulate that the ontology structure is in the form of the general tree. Our algorithm aligns the concepts on the source ontology T_s to the concepts on the target ontology T_t by performing search and comparison in the top-down manner.

Given a concept $C_i \in T_s$, the algorithm attempts to find the most appropriate concept $B^* \in T_t$, which is located on an arbitrary level of the hierarchy. The algorithm starts by constructing the feature vectors for the current root concept on the level l and its child concepts on the level $l + 1$. It then calculates the similarity scores between a given source concept and candidate target concepts. If the similarity scores of the child concepts are not greater than the root

concept, then the algorithm terminates. Otherwise, it selects a child concept having the maximum score to be the new root concept, and iterates the same searching procedure. Algorithms 1 and 2 outline our ontology alignment process.

Algorithm 1: OntologyAlignment
input : The source ontology T_s and the target ontology T_t.
output : The set of the aligned concepts A.
begin
  Set the starting level, $l \leftarrow 0$;
  while $T_s^{\langle l \rangle} \leq T_s^{\langle \max \rangle}$ **do**
    Find all child concepts on this level,
      $\{C_i\}_{i=1}^{k} \in T_s^{\langle l \rangle}$;
    Flatten $\{C_i\}_{i=1}^{k}$ and build their
      corresponding feature vectors, $\{c_i\}_{i=1}^{k}$;
    For each c_i, find the best matched concepts on T_t ,
      $B \leftarrow$ FindBestMatched(c_i);
      $A \leftarrow A \cup \{B, C_i\}$;
    Set $l \leftarrow l+1$;
  end
end

Algorithm 2: FindBestMatched(c_i)
begin
  Set the starting level, $l \leftarrow 0$;
  BestConcept $\leftarrow$ T_t (root concept);
  repeat
    $s_{tmp} \leftarrow$ *JaccardSim*(c_i,*BestConcept*);
    if $T_s^{\langle l \rangle} \leq T_s^{\langle \max \rangle}$ **then**
      return *BestConcept;*
    Find all child concepts on this level,
      $\{B_j\}_{j=1}^{h} \in T_t^{\langle l \rangle}$;
    Flatten $\{B_j\}_{j=1}^{h}$ and build corresponding
      feature vectors, $\{b_j\}_{j=1}^{h}$;
    $s_{j*} \leftarrow argmax_j$ *JaccardSim*(c_i, $\{b_j\}_{j=1}^{h}$);
    if $s_{j*} > s_{tmp}$ **then**
      BestConcept $\leftarrow B_{j*}$;
    Set $l \leftarrow l+1$;
  until *BestConcept does not change*;
  return *BestConcept;*
end

Figure 9 shows a simple example that describes how the algorithm works. It begins with finding the most appropriate concept on T_t for the root concept *1* $\in T_s$. By flattening the hierarchy starting from given concepts ('1' on T_s, and 'a', 'a-b', 'a-c' for T_t), we can represent them with the feature vectors and measure their similarities. On the first iteration, the child concept 'a-c' obtains the maximum score, so it becomes the new root concept. Since the algorithm cannot find improvement on any child concepts in the second iteration, it stops the loop and the target concept 'a-c' is aligned with the source concept '1'. The algorithm proceeds with the same steps by finding the most appropriate concepts on T_t for the concepts '1-1' and '1-2'. It finally obtains the resulting concepts 'a-c-f' and 'a-c-g', respectively.

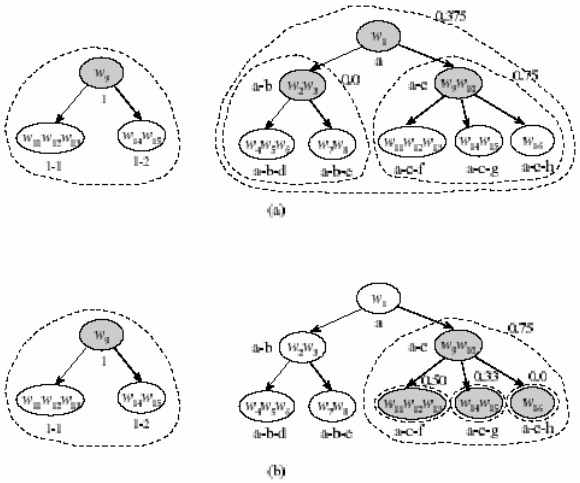

Figure 9: An example of finding the most appropriate concept on T_t for the root concept $1 \in T_s$

In our experiments, we used a portion of the MMT semantic hierarchy and the EDR concept dictionary as the source and the target ontologies, respectively. We considered the 'animal' concept as the root concepts and extracted its related concepts. In the EDR concept dictionary, however, the relations among concepts are very complex and organized in the form of the semantic network. Thus, we pruned some links to transform the network to a tree structure. Starting from the 'animal' concept, there are more than 200 sub-concepts (containing about 7,600 words) in the EDR concept dictionary, and 14 sub-concepts (containing about 400 words) in the MMT semantic hierarchy. It is important to note that these two ontologies are considerably different in terms of the number of concepts and words.

In our experiments, we used a portion of the MMT semantic hierarchy and the EDR concept dictionary as the source and the target ontologies, respectively. We considered the 'animal' concept as the root concepts and extracted its related concepts. In the EDR concept dictionary, however, the relations among concepts are very complex and organized in the form of the semantic network. Thus, we pruned some links to transform the network to a tree structure. Starting from the 'animal' concept, there are more than 200 sub-concepts (containing about 7,600 words) in the EDR concept dictionary, and 14 sub-concepts (containing about 400 words) in the

MMT semantic hierarchy. It is important to note that these two ontologies are considerably different in terms of the number of concepts and words.

The proposed algorithm is used to find appropriate EDR concepts for each one of 14 MMT concepts. The results are shown in Table 5. In the table, there are 6 relations (marked with the symbol '*') that are manually classified as exact mapping. This classification is done by inspecting the structures of both ontologies by hand. If the definition of a given MMT concept appears in the EDR concept and the algorithm seems to correctly match the most suitable EDR concept, this mapping will be classified as exact mapping. The remaining 8 MMT concepts, e.g. 'cold-blood' and 'amphibian', are mapped to closely related EDR concepts, although they are not considered to be exact mapping. The EDR concepts found by our algorithm for these 8 MMT concepts are considered to be only the subset of the source concepts. For example, the 'amphibian' concept of the MMT is mapped to the 'toad' concept of the EDR.

Table 5: Results of aligned concepts between MMT semantic hierarchy and EDR concept dictionary

MMT concept	*EDR concept dictionary*
--vertebrate	vertebrate
--warm-blood	mammal
--mammal	mammal
--bird	bird
--cold-blood	reptile
--fish	fish
--amphibian	toad
--reptile	Reptile
--snake	snake
--invertebrate	squid
--worm	leech
--insect	hornet
--shellfish	crab
--other sea creature	squid

By analyzing the results, we can classify the MMT words that cannot find any associated EDR words into 4 categories.

1. Incorrect spelling or wrong grammar: Some English words in the MMT semantic hierarchy are simply incorrect spelling, or they are written with wrong grammar. For example, one description of a tiger species is written as 'KIND A TIGER'. Actually, this instance should be 'KIND OF A TIGER'. The algorithm can be used to find words that possible have such a problem. Then, the words can be corrected by lexicographers.

2. Inconsistency: The English translation of Thai words in the MMT semantic hierarchy was performed by several lexicographers. When dealing with Thai words that do not have exact English words, lexicographers usually enter phrases as descriptions of these words. Since there is no standard of writing the descriptions, these are incompatibility between descriptions that explain the same concept. For example, the following phrases are used to describe fishes that their English names are not known.

- Species of fish
- A kind of fish
- Species of fresh water fish

3. Thai specific words: The words that we used in our experiments are animals. Several animals are region specific species. Therefore, they may not have any associated English words. In this case, some words are translated by using short phrases as English descriptions of these Thai words. Another way to translate these words is to use scientific names of species.

The problems mentioned earlier make it more difficult to match concepts by the algorithm. However, we can use the algorithm to identify where the problems occur. Then, we can use these results to improve the MMT ontology.

6. Summary and Future Work

The tasks of language identification, word extraction, dictionary-less search engine and term-based ontology alignment had been selected to study by means of the proposed model for unified language processing. The tasks had been evaluated and resulted in a significant performance. Therefore, a non-segmenting language can be efficiently processed without relying on the word boundary information at all. It shows that the footprint of byte sequence of a language is sufficient for processing an input text. It is a direct input with most reliable information. The advantages in these statistical based approaches are also a fundamental work for unifying multi-lingual tasks where language dependent parts can be lessened.

7. References

[1] Church, K. W., Robert, L., and Mark, L., "A status report on ACL/DCL," In Proceedings of the seventh Annual Conference of the UW Centre New OED and Text Research: Using Corpora, pp. 84–91, 1991.

[2] Haussler, D., "Convolution kernels on discrete structures," Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, 1999.

[3] Haveliwala, T., "Topic-sensitive pagerank," In Proceedings of the Eleventh International Conference on World Wide Web, pp. 517–526, 2002.

[4] Manning, C. D., and Sch¨utze, H., Foundations of statistical natural language processing, MIT Press, Cambridge, MA., 1999.

[5] Quinlan, J. R., "C4.5 Programs for Machine Learning," Morgan Publishers San Mated, California, 302p., 1993.

[6] Shannon, C. E., "A Mathematical Theory of Communication," Bell System Technical Journal 27, pp. 379-423, 1948.

[7] Shisanu Tongchim, Canasai Kruengkrai, Virach Sornlertlamvanich, Prapass Srichaivattana and Hitoshi

Isahara, “Analysis of an Iterative Algorithm for Term-Based Ontology Alignment,” to appear in Proceedings of The 2nd IJCNLP, Jeju Island, Korea, 2005.
[8] Strehl, A., Ghosh, J., and Mooney, R. J., “Impact of similarity measures on web-page clustering,” In Proceedings of AAAI Workshop on AI for Web Search pp. 58–64, 2000.
[9] Virach Sornlertlamvanich, Canasai Kruengkrai, Shisanu Tongchim, Prapass Srichaivattana and Hitoshi Isahara. “Term-Based Ontology Alignment,” Proceedings of the Second International Workshop on UNL, Other Interlinguals and their Applications, Mexico City, Mexico, 2005.
[10] Virach Sornlertlamvanich, Pongtai Tarsaku, Prapass Srichaivattana, Thatsanee Charoenporn and Hitoshi Isahara, “Dictionary-less Search Engine for the Collaborative Database,” Proceedings of the Third International Symposium on Communications and Information Technologies (ISCIT-2003), Songkhla, Thailand, 2003.
[11] Virach Sornlertlamvanich and Tanaka Hozumi, “The Automatic Extraction of Open Compounds from Text Corpora,” Proceedings of the 16th International Conference on Computational Linguistics (COLING-96), pp. 1143-1146, 1996.
[12] Virach Sornlertlamvanich, Tanapong Potipiti and Thatsanee Charoenporn, “Automatic Corpus-based Thai Word Extraction with the C4.5 Learning Algorithm,” Proceedings of the 18th International Conference on Computational Linguistics (COLING2000), Saarbrucken, Germany, July-August 2000, pp. 802-807, 2000.
[13] Watkins, C., “Dynamic alignment kernels,” Technical Report CSD-TR-98-11, Royal Holloway, University of London, 1999.
[14] Yamamoto, M. and Church, K. W., “Using Suffix Arrays to Compare Term Frequency and Document Frequency for All Substrings in Corpus,” Proceedings of the Sixth Workshop on Very Large Corpora, pp. 27-37, 1998.