A Statistical Corpus-Based Term Extractor

Patrick Pantel and Dekang Lin

Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1 Canada
{ppantel, lindek}@cs.ualberta.ca

Abstract. Term extraction is an important problem in natural language processing. In this paper, we propose a language independent statistical corpus-based term extraction algorithm. In previous approaches, evaluation has been subjective, at best relying on a lexicographer's judgement. We evaluate the quality of our term extractor by assessing its predictiveness on an unseen corpus using perplexity. Second, we evaluate the precision and recall of our extractor by comparing the Chinese words in a segmented corpus with the words extracted by our system.

1 Introduction

Term extraction is an important problem in natural language processing. The goal is to extract sets of words with exact meaning in a collection of text. Several linguists have argued that the base semantic unit of language are these terms. Applications of automatic term extraction include machine translation, automatic indexing, building lexical knowledge bases, and information retrieval.

In previous systems, evaluation has relied mostly on human assessments of the quality of extracted terms. This is problematic since experts often disagree on the correctness of a term list for a corpus. Consequently, it is difficult to replicate the evaluation procedure to compare different systems. Furthermore, experts normally evaluate only a few hundred terms. These tend to be the highest-ranking ones, those most easily recognizable by lexicographers. A term extraction tool that assists humans would be more useful if it were able to extract those terms less obvious to humans. This is difficult to evaluate.

In this paper, we present a language independent statistical corpus-based term extraction algorithm. First, we collect bigram frequencies from a corpus and extract two-word candidates. After collecting features for each two-word candidate, we use mutual information and log-likelihood ratios to extend them to multi-word terms. We experiment with both English and Chinese corpora. Using perplexity, we quantify the definition of a term and we obtain a comparative evaluation of term extraction algorithms. Furthermore, we evaluate the precision and recall of our term extractor by comparing the words in a segmented Chinese corpus with the words extracted by our system. Our evaluation methodology circumvents the problems encountered by previously used human evaluations. It also provides a basis for comparing term extraction systems.

2 Previous Work

There have been several approaches to automatic term extraction mostly for technical terminology and noun phrases. Many successful algorithms are statistical corpus-based approaches [2], [5], [10].

Several term extraction tools have been developed. Dagan and Church [4] proposed a tool that assists terminologists in identifying and translating technical terms. Smadja [19] developed a lexicographic tool, *Xtract*, which extracts collocations from English. Fung [11] later extended this model to extract words from Chinese corpora. The latter work was the first to attempt an automatic evaluation of term extraction. Previous methods used human experts to evaluate their extracted term lists. Fung first uses a tagger to retrieve Chinese words. Then, the extraction system is evaluated by counting the number of these words retrieved by the term extractor.

More recently, Eklund and Wille [8] describe an algorithm that utilizes discourse theory to extract terms from single-subject texts. Hybrid approaches combining statistical techniques with linguistic knowledge (syntax and morphology) have also emerged [14], [17], [18].

3 Term Extraction Algorithm

Our term extractor is a two-phase statistical corpus-based algorithm that extracts multi-word terms from corpora of any language (we experiment with English and Chinese corpora in this paper).

Our algorithm uses two metrics to measure the information between terms (or words): *mutual-information* (*mi*) and *log-likelihood* (*logL*) [7]. Mutual information is defined as:

$$mi(x,y) = \frac{P(x,y)}{P(x)P(y)} \tag{1}$$

where x and y are words or terms. Mutual information is highest when all occurrences of x and y are adjacent to each other and deteriorates with low frequency counts. To alleviate this problem, we use a second measure, log-likelihood, which is more robust to low frequency events. Let $C(x, y)$ be the frequency of two terms, x and y, occurring adjacent in some corpus (where the asterix (*) represents a wildcard). Then, the log-likelihood ratio of x and y is defined as:

$$\begin{aligned} logL(x,y) = {} & ll\left(\tfrac{k_1}{n_1}, k_1, n_1\right) + ll\left(\tfrac{k_2}{n_2}, k_2, n_2\right) \\ & - ll\left(\tfrac{k_1+k_2}{n_1+n_2}, k_1, n_1\right) - ll\left(\tfrac{k_1+k_2}{n_1+n_2}, k_2, n_2\right) \end{aligned} \tag{2}$$

where $k_1 = C(x, y)$, $n_1 = C(x, *)$, $k_2 = C(\neg x, y)$, $n_2 = C(\neg x, *)$, and:

$$ll(p,k,n) = k\log(p) + (n-k)\log(1-p) \tag{3}$$

```
Input:   A corpus $L$ in any language.
Step 1:  Collect bigram frequencies for $L$ in a proximity database $DB$.
Step 2:  For all 4-grams $w\ x\ y\ z$ in $L$, remove one count for $x\ y$ in $DB$ if
             - $mi(x, y) < mi(w, x) - k$ or
             - $mi(x, y) < mi(y, z) - k$.
Step 3:  For all entries $(x, y)$ in $DB$, add $(x, y)$ to a list $T$ if:
             - $C(x, y) > minCount$
             - $S(x, y) > minLogL$
Output:  The list $T$ of candidate multi-word terms.
```

Fig. 1. Candidate extraction algorithm.

The log-likelihood ratio is highest when all occurrences of *x* and *y* are adjacent to each other (as in mutual information). However, the ratio is also high for two frequent terms that are rarely adjacent. For example, the word pair (*the*, *the*) has a very high log-likelihood ratio in English even though it rarely occurs (mostly as a typographical error).

To overcome the shortcomings of mutual information and log-likelihood, we propose a hybrid metric. The score *S* for a pair (*x*, *y*) is defined as:

$$S(x,y) = \begin{cases} logL(x,y) & if\ \ mi(x,y) \geq minMutInfo \\ 0 & otherwise \end{cases} \quad (4)$$

We use the mutual information as an initial filter to eliminate term pairs such as (*the*, *the*). Below, we describe each phase of our term extraction algorithm.

3.1 Candidate Extraction

Figure 1 outlines the first phase of our term extraction algorithm. It extracts a list of two-word candidate terms from a corpus of any language. Optimally, this list contains all two-word terms as well as fragments of all multi-word terms.

In Step 1, we construct a proximity database consisting of the frequency counts for each adjacent pair of words in the corpus [16].

The purpose of Step 2 is to eliminate frequency counts for those adjacent words that are separated by a phrasal (word) boundary [1], [20]. A phrasal boundary separates words that are not part of a same term. Given a 4-gram (*w*, *x*, *y*, *z*), we assume there is a phrasal boundary between *x* and *y* if $mi(x, y) < mi(w, x) - k$ or $mi(x, y) < mi(y, z) - k$, for some fixed constant *k*.

Step 3 performs the selection of two-word candidates. An adjacent pair of words is selected if its frequency and score surpasses a fixed threshold. We experimentally set *minCount* to 3, *minLogL* to 5, and *minMutInfo* to 2.5.

```
Input:   A list $T$ of two-word candidates for a corpus $L$ in any language and a proximity da-
         tabase $DB$ consisting of bigram frequencies for $L$.
Step 1:  Accumulate features for candidate terms
         For each candidate $c$ in $T$
           For each $w_1\ w_2 \ldots c \ldots w_{2k-1}\ w_{2k}$ in $L$
             Add all possible substrings involving $c$ in $DB$.
Step 2:  Update the proximity database
         Remove each entry in $DB$ that has frequency < $minFreq$.
Step 3:  Extend two-word candidates into an initially empty list $E$
         For each candidate $c$ in $T$
           $extend(c, E, DB)$ – see Figure 3
           if most occurrences of $c$ in the corpus have not been extended then add $c$ to $E$.
Output:  The list $E$ of extracted multi-word terms.
```

Fig. 2. Multi-word term extraction algorithm.

3.2 Multi-Word Term Extraction

The input to the second phase of the term extraction algorithm is the proximity database and the list of two-word candidates extracted from phase one. The goal is to extend each candidate to multi-word terms (two words or more up to a fixed size). Below we describe the multi-word term extraction algorithm in two parts: the main extraction driver and the recursive term extension algorithm.

3.2.1 Extraction Driver

Figure 2 outlines the main driver for the multi-word term extraction algorithm.

The input proximity database consists of the bigram frequencies of a corpus. In the first step of the algorithm, we update this database with new features. Given a two-word candidate c, we consider all possible expansions e of c containing no more than k words on each side. We then count the frequency between e and all its adjacent words. Later, in our expansion algorithm, we will use these frequencies to determine whether e should expand to any of these adjacent words.

For example, suppose $k = 2$ and we have a candidate *drop-down* that occurred in a corpus in the following contexts:

- …from the drop-down list in…
- …Network Logon drop-down list when…

The features extracted for the first context are: (*drop down*, *list*), (*drop down list*, *in*), (*the*, *drop down*), (*the drop down*, *list*), (*the drop down list*, *in*), (*from*, *the drop down*), (*from the drop down*, *list*), and (*from the drop down list*, *in*).

In Step 2 of the algorithm, we remove those features that occurred only a fixed small number of times. In our experiments, we found that most extracted features occurred only once. This step significantly reduces the size of the proximity database and removes spurious features.

```
Input:  A multi-word term $c$ to extend into a list $E$ and a proximity database $DB$ consisting
        of bigram frequencies of a corpus $L$ and features extracted from Step 2 of Figure 2.
        Let $c_1$ and $c_2$ be the terms merged to create $c$.
Step 1: For each word $w$ adjacent to $c$ in $L$
          If $S(w, c) > S(c_1, c_2) - k$, add $w$ to a list $G$ sorted in decreasing order of $S(w, c)$.
Step 2: For each possible extension $g$ in $G$
          Let $p$ be the extended phrase ($c$ $g$) or ($g$ $c$)
          If $p$ is not a substring of a term in $E$
            If (not extend($p$) and filter($p$)) add $p$ to $E$
Step 3: If any $p$'s were extended or added to $E$ then return true, otherwise return false.
Output: The list of extracted multi-word terms is appended to $E$ and a boolean value indicat-
        ing whether or not at least one extension was made is returned.
```

Fig. 3. Recursive algorithm that extends a given multi-word term to larger terms.

Step 3 of Figure 2 uses the added features in the proximity database to extend candidate terms (see the next section for a description of the extension algorithm). We are then left only with deciding whether or not a two-word candidate *c* is a term. We verify this by obtaining the ratio of the frequencies of extended terms containing *c* as a substring to the frequency of *c*. If it is large, then this indicates that most occurrences of *c* in the corpus were also occurrences of an extended phrase of c. So, we only extract *c* if the ratio is small.

3.2.2 Term Extension Algorithm

Figure 3 describes the recursive term extension algorithm. The goal is to extend an input term *c* using the updated proximity database from Section 3.2.1.

In the first step of the algorithm, we build a sorted list *G* of all *good* extensions of *c* (the best extensions are first in the list). Let (c_1, c_2) be the two terms that compose *c*. For a two-word candidate, the first and second words are c_1 and c_2, respectively. A word *w* is a *good* extension of *c* if $S(w, c) > S(c_1, c_2) - k$, for some fixed threshold *k*. The frequency counts required to compute *S* are stored in the proximity database. The list *G* then contains all 1-word extensions from *c*. But, these might still be term fragments.

Step 2 is the recursive step of the algorithm. We loop through each good extension *g* in *G*. Let *p* be the extension of *c* with *g* (i.e. either *c g* or *g c*). Before processing *p*, we require that *p* is not a substring of an extracted term (i.e. *g* has not been previously extended). For example, suppose that *Jones Computer* and *Computer Publishing* are both two-word candidates and that the former is extended to *Jones Computer Publishing*. Now, suppose that we are attempting to extend *Computer Publishing* with *g* = *Jones*. Since *Jones Computer Publishing* has already been extracted, we do not want to extend *Computer Publishing* with *g* = *Jones*.

If *p* is not a substring of an extracted phrase, then we try to extend *p* recursively. If *p* is successfully extended to an even larger term, then we do not add *p* to *E*. However, if *p* is not extended then *p* is classified as a term and added to *E*.

Table 1. 10-fold cross-validation evaluation of the perplexity of our term extraction system.

Corpus	Unigram Perplexity	SP Perplexity	SP with mut-info
UNTS	647.01	523.93	547.94
NAG	706.54	605.59	654.94

So far, the algorithm is purely statistical. However, the final part of Step 2 provides the option of using linguistic knowledge to filter the extracted terms. For example, if the statistical processor treats punctuation marks as words, it is probable that some extended features will contain punctuation marks. This filter allows for easy removal of such erroneous terms.

The final step of the extension algorithm simply determines and returns whether or not *c* was extracted in whole or in part as a term.

4 Experimental Results

Below, we evaluate our term extractor using perplexity, precision and recall.

4.1 Perplexity

Perplexity measures how well a model predicts some data. In natural language, we often use perplexity to compare the predictiveness of different language models over a corpus. Let W be a random variable describing words with an unknown probability mass function $p(w)$ and let $C(w)$ be the frequency count of a word or term w in a language. Also, let $m(w) = C(w) / C(*)$ be an approximation of $p(w)$ (a unigram model), where * represents a wildcard. Since W is stationary and ergodic, the cross-entropy of W, $H(p, m)$, is defined as [3]:

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n}\log_2 m(w_1 w_2 w_3 \ldots w_n) \qquad (5)$$

Cross-entropy gives us an upper bound on the true entropy of W, $H(p, m) \geq H(W)$. Standard unigram models approximate the probability of a sequence of words by computing the product of the word probabilities (i.e. assume words are independent). We augment this model to also describe terms by computing the probability of a sequence of words as the joint probability of the terms and words in the sequence.

Using our term extractor, we extract a list of terms from a training corpus. To compute m, we require the frequency counts of all words and terms. We obtain these frequencies by counting all terms and words that are not terms in the training corpus. Using a testing corpus L with a finite number of words n, we approximate the cross-entropy $H(p, m)$ from formula (5) with:

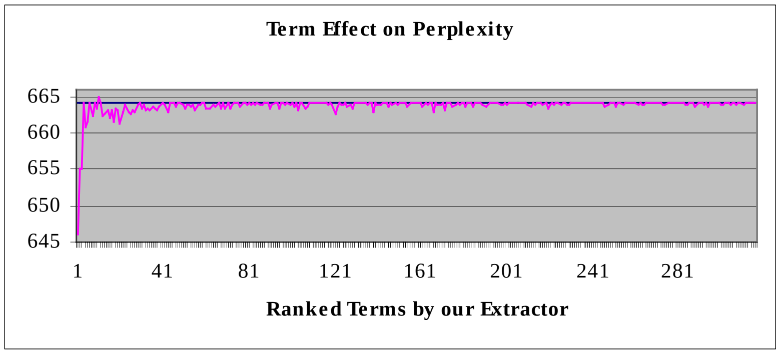

Fig. 4. The effect on perplexity when each ranked term is added individually. The left-most points represent the highest ranked terms.

$$H(p,m) = -\frac{1}{n}\sum_{t \in L} \log_2 m(t) \tag{6}$$

where t is a term or a word in L. The better the list of extracted terms, the more predictive the model will be of the testing corpus (i.e. the lower $H(p, m)$ will be). Hence, this model can serve to perform a comparative evaluation of different term extraction algorithms. A related measure, *perplexity*, is defined as:

$$perplexity(p,m) = 2^{H(p,m)} \tag{7}$$

4.1.1 Analysis

We used two corpora for evaluating the perplexity of our system: UNTS [12] consisting of 439,053 words and NAG [15] consisting of 117,582 words. We divided each corpus into 10 equal parts and performed ten-fold cross validation to test our system's perplexity. We used Witten-Bell discounting [21] to estimate the probability of unseen events. Table 1 presents a comparison between our system's perplexity (*SP*) and the perplexity of a unigram model (i.e. with an empty term list). On both corpora, the term list generated by our system significantly reduces perplexity.

We also experimented with a different expansion function for our term extraction algorithm. Instead of using the log-likelihood ratio as in Step 1 of the algorithm presented in Figure 3, we used the mutual information metric. The third column of Table 1 shows the result. This metric performs much worse on the smaller NAG corpus. This supports the claim from Section 3 that mutual information deteriorates with sparse data.

Finally, we divided the UNTS corpus in two equal parts (even vs. odd numbered chapters) and used one for training and the other for testing. We evaluated the effect that specific phrases had on the perplexity of our system. Figure 4 shows the variation in perplexity when each phrase extracted by our system is individually extracted. The horizontal line represents the perplexity of the standard unigram model on this test corpus, which is 664.1. The highest spike is caused by the term *boot record*. This is

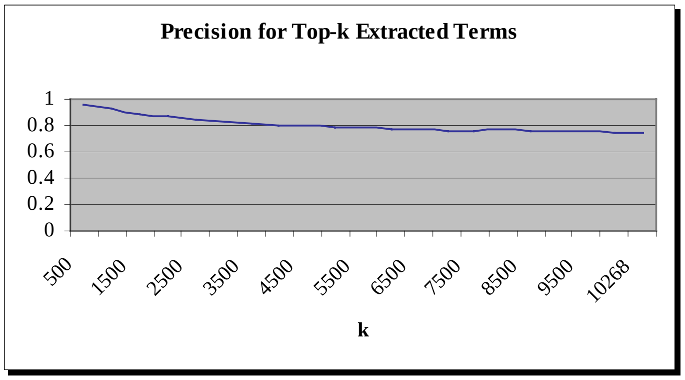

Fig. 5. The precision of the top-*k* words extracted by our term extractor.

because almost all of its occurrences were in chapters 9 and 29 (part of the training corpus). Finally, we manually created a list of 5 bad terms from a training corpus for UNTS. As expected the perplexity on the test set increased to 664.3.

4.2 Precision and Recall

Making use of a segmented Chinese corpus, we compute the precision and recall of our term extractor. Chinese text does not contain word boundaries and most Chinese words are one or two characters long. In fact, since Chinese characters carry a lot more information than English characters, the average length of a Chinese word contains 1.533 characters [12].

The task of identifying words in Chinese text is very similar to identifying phrasal words in English if one treats each Chinese character as a word. In fact, our term extractor can be applied straightforwardly to Chinese text.

Our extractor can be evaluated as retrieving multi-character words from a segmented Chinese corpus. The target words for the retrieval task are multi-character words in the segmented corpus with a frequency above a certain threshold. The percentage of words in the target set that are extracted by the term extractor is the recall. We measure the precision of the extractor by computing the percentage of the extracted words among the words in the segmented corpus (including those with frequency lower than the threshold).

4.2.1 Analysis

The test data is a cleaned up version of a segmented Chinese corpus [12]. It contains about 10MB of Chinese news text. We extracted 10,268 words from the corpus. Among them, 6,541 are words in the segmented corpus. A further 1,096 of our extracted words are found in HowNet, a Chinese lexical knowledge base [6]. This gives an overall precision of 74.4% for our extraction algorithm. This is a significant improvement over the precision of 59.3% given by Fung's extractor [11]. We also

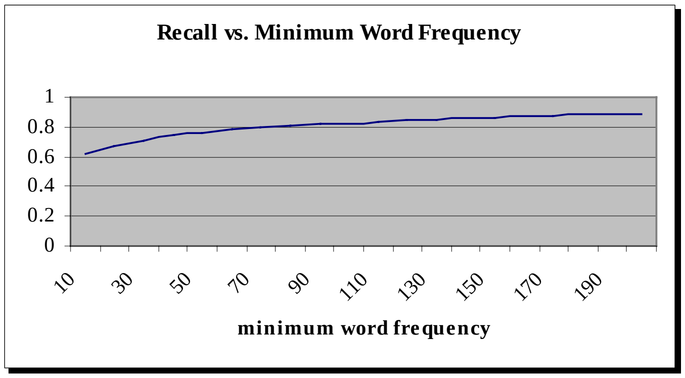

Fig. 6. The recall of our term extractor on words in the corpus that occurred a minimum number of times.

evaluated the precision of the top-*k* extracted words sorted by their log-likelihood ratio. Figure 5 shows the result.

Since the segmented corpus is generated automatically and is only lightly cleaned up, it contains many errors. The segmentor tends to break words aggressively. Upon inspection of the remaining 2,631 words not found in the corpus nor HowNet, we found many correct terms such as those shown in Table 2. Many of these words are names of persons and organizations, which are not readily found in lexicons such as HowNet.

There are 8,582 words in the segmented corpus that occurred at least 10 times. We extracted 5,349 of these, which gives an overall recall of 62.3%. Again, this is a significant improvement over the recall of 14% given by Fung's extractor [11]. Figure 6 shows the result of evaluating recall for sets with a fixed minimum frequency.

5 Conclusion and Future Work

In this paper, we presented a language independent statistical corpus-based term extractor. It improved the perplexity of a testing corpus by 16-23% and achieved 74.4% precision and 62.3% recall. Of the top 1000 terms retrieved by our system, we achieved 92.6% precision. Also, we recalled 89.1% of the terms that occurred at least 200 times in a corpus.

Our evaluation methodology provides a significant improvement over the current dependence on human evaluators for evaluating systems. It allows for easy comparison of extraction systems and it measures the precision and recall of a system at different word frequency levels.

A promising extension to our algorithm is to apply similar methods to non-linear structures such as dependency structures of sentences. The result would be collocational dependency structures.

Table 2. Some terms extracted by our system that are not found in the lexicon nor in the segmented Chinese corpus.

CHINESE TERM	ENGLISH TRANSLATION	CHINESE TERM	ENGLISH TRANSLATION
冀朝铸	person name	股票指数	stock index
经济特区	special economic zone	国际公约	international treaty
技术革新	technological innovation	国际足联	FIFA
集贸市场	farmer's market	国际奥委会	IOC
姜春云	person name	国际红十字会	International Red Cross
加勒比地区	Caribbean region	国家教委	State Education Commission
贾志杰	person name	国家科委	State Science Commission
军事演习	military exercise	美国国务卿贝克	US State Secretary Baker
单桂芳	person name	马克思主义哲学	Marxist philosophy
艰苦朴素	hard work and plain living (Chinese idiom)	联合国安理会	United Nation Security Council

References

1. Ando, R. K. and Lee, L. 2000. Mostly-unsupervised statistical segmentation of Japanese: Application to Kanji. In *Proceedings of NAACL-2000*. pp. 241-248. Seattle, WA.
2. Bourigault, D. 1992. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of COLING-92*. pp. 977-981. Nates, France.
3. Cover, T. M., Thomas, J. A. 1991. *Elements of Information Theory*. Wiley, New York.
4. Dagan, I. and Church, K. 1994. Termight: identifying and translating technical terminology. In *Proceedings of Applied Language Processing*. pp. 34-40. Stuttgart, Germany.
5. Daille, B., Gaussier, E., and Langé, J.M. 1994. Towards automatic extraction of monolingual and bilingual terminology. In *Proceedings of COLING-94*. pp. 515-521. Kyoto, Japan.
6. Dong, Z. 1999. Bigger context and better understanding - Expectation on future MT technology. In *Proceedings of ICMT&CLIP-99*. pp.17-25.
7. Dunning, T. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61-74.
8. Eklund, P. and Wille, R. 1998. A multimodal approach to term extraction using a rhetorical structure theory tagger and formal concept analysis. In *Proceedings of Conference on Cooperative Multimodal Communication: Theory and Applications*. pp. 171-175. Tilburg, Netherlands.
9. FDMC. 1986. *Xiandai Hanyu Pinlu Cidian*(Modern Chinese Frequency Dictionary). Beijing Language Institute Press.
10. Frantzi, K.T. and Ananiadou, S. 1999. The C-Value/NC-Value domain independent method for multi-word term extraction. *Natural Language Processing*, 6(3):145-179.
11. Fung, P. 1998. Extracting key terms from Chinese and Japanese texts. *The International Journal on Computer Processing of Oriental Language*. Special Issue on Information Retrieval on Oriental Languages, pp. 99-121.
12. Guo, J. 1998. Segmented Chinese corpus. ftp://ftp.cogsci.ed.ac.uk/pub/chinese/.
13. Jennings, R., Benage D. B., Crandall, S. and Gregory K. 1997. *Special Edition Using Windows NT Server 4: The Most Complete Reference*. Macmillan Computer Publishing.
14. Justeson, J.S. and Katz, S.L. 1996. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 3(2):259-289.
15. Kirch, O. 1995. *LINUX: Network Administrator's Guide*. O'Reilly.
16. Lin, D. 1998a. Extracting collocations from text corpora. In *Proceedings of COLING/ACL-98 Workshop on Computational Terminology*. Montreal, Canada.
17. Maynard, A. and Ananiadou, S. 2000. Identifying terms by their Family and friends. In *Proceedings of COLING-2000*. pp. 530-536. Saarbrucken, Germany.
18. Maynard, A. and Ananiadou, S. 1999. Identifying contextual information for multi-word term extraction. In *Proceedings of Terminology and Knowledge Engineering Conference-99*. pp. 212-221. Innsbruck, Austria.
19. Smadja, F. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143-177.
20. Sun, M., Shen, D. and Tsou B. K. 1998. Chinese word segmentation without using lexicon and hand-crafted training data. In *Proceedings of COLING-ACL-98*. pp. 1265-1271. Montreal, Canada.
21. Witten, I. H. and Bell, T. C. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory, 37(4).