Computational Linguistics & Chinese Language Processing

中文計算語言學期刊

A Publication of the Association for Computational Linguistics and Chinese Language Processing

This journal is included in THCI, Linguistics Abstracts, and ACL Anthology.

易繫辭曰上古結繩而
治後世聖人易之以書
契百官以治萬民以察
說文敘曰蓋文字者經
藝之本宣教明化之始
前人所以垂後後人所
以識古故曰本立而道
生知天下之至賾而不
可亂也教化既萌文心
雕龍則謂人之立言因
字而生句積句而成章
積章而成篇篇之彪炳

Vol.21 No.1 June 2016 ISSN: 1027-376X

International Journal of Computational Linguistics & Chinese Language Processing

International Journal of Computational Linguistics & Chinese Language Processing

Aims and Scope

International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP) is an international journal published by the Association for Computational Linguistics and Chinese Language Processing (ACLCLP). This journal was founded in August 1996 and is published four issues per year since 2005. This journal covers all aspects related to computational linguistics and speech/text processing of all natural languages. Possible topics for manuscript submitted to the journal include, but are not limited to:

- Computational Linguistics
- Natural Language Processing
- Machine Translation
- Language Generation
- Language Learning
- Speech Analysis/Synthesis
- Speech Recognition/Understanding
- Spoken Dialog Systems
- Information Retrieval and Extraction
- Web Information Extraction/Mining
- Corpus Linguistics
- Multilingual/Cross-lingual Language Processing

Membership & Subscriptions

If you are interested in joining ACLCLP, please see appendix for further information.

Copyright

Cover

Calligraphy by Professor Ching-Chun Hsieh, founding president of ACLCLP
Text excerpted and compiled from ancient Chinese classics, dating back to 700 B.C.
This calligraphy honors the interaction and influence between text and language

International Journal of
Computational Linguistics and Chinese Language Processing
vol. 21, no. 1, June 2016

Contents

Computational Linguistics and Chinese Language Processing
Vol. 21, No. 1, June 2016, pp. 1-28

A Segmentation Matrix Method for Chinese Segmentation Ambiguity Analysis

Yanping Chen[*,+], Qinghua Zheng[+], Feng Tian[+], and Deli Zheng[+]

Abstract

Chinese Segmentation Ambiguity (CSA) is a fundamental problem confronted when processing Chinese language, where a sentence can generate more than one segmentation paths. Two techniques are commonly used to identify CSA: Omni-segmentation and Bi-directional Maximum Matching (BiMM). Due to the high computational complexity, Omni-segmentation is difficult to be applied for big data. BiMM is easier to be implemented and has a higher speed. However, recall of BiMM is much lower. In this paper, a Segmentation Matrix (SM) method is presented, which encodes each sentence as a matrix, then maps string operation into set operations. To identify CSA, instead of scanning a whole sentence, only specific areas of the matrix are checked. SM has a computational complexity close to BiMM with recall the same as Omni-segmentation. In addition to CSA identification, SM also supports lexicon-based Chinese word segmentation. In our experiments, based on SM, several issues about CSA are explored. The result shows that SM is useful for CSA analysis.

Keywords: Segmentation Matrix, Segmentation Ambiguity.

1. Introduction

Chinese characters are originated from hieroglyphic and written next to each other without delimiter in between. The lack of orthographic words makes Chinese word segmentation difficult. It is often that a Chinese sentence can be parsed into several segmentation paths, which results in the Chinese Segmentation Ambiguity (CSA) problem. It can be roughly classified into two categories: Overlapping Ambiguity (OA) and Combinational Ambiguity (CA)[1] (Liang, 1984; Sun, 2001). For the OA problem, a sentence contains at least two

[*] Guizhou University
[+] Xi'an Jiaotong University
E-mail: ypench@gmail.com; qhzheng@mail.xjtu.edu.cn; ftian@sei.xjtu.edu.cn,; delizheng.2009@gmail.com

[1] Formal definitions of CA and OA are given in Section 2.

overlapped words. For example, “温柔和” contains two overlapped words: “温柔” (Gentle) and “柔和” (Soft). The character “柔” can be assembled with either “温” and “和”. Only one is suitable in a given context. In Chinese, every character can be either a morpheme or a word (Li, 2011). Then, given a word containing more than one characters, whether it is appropriate to segment it will lead to the CA problem. For example, “温柔” (Gentle) can be further segmented into “温/柔” (Warm/ Soft).

In Chinese, the OA is prevalent. For example, in the Penn Chinese Treebank corpus, there are 39% sentences are identified with this ambiguity. Therefore, the OA is widely studied in this field. When the CA is under consideration, the problem is more serious. For example, in the lexicon of our experiments, 75.25% words have the CA problem, even using a loose definition (See Definition 4 of Section 3). Furthermore, CA and OA are not independent. They often co-occur within a sentence, which worsens the performance of Chinese word segmentation (Chen *et al.*, 2012). The problem is deteriorated by the fact that Chinese has a large number of characters and words[2].

To identify CSA, two techniques are commonly used: Omni-segmentation and BiMM. Omni-segmentation tries to traverse every segmentation path in a sentence. All ambiguities can be identified. The problem of Omni-segmentation is that it has the highest computational and space complexities. For example, a sentence “江泽民在北京人民大会堂会见参加全国法院工作会议和全国法院系统打击经济犯罪先进集体表彰大会代表时要求大家要充分认识打击经济犯罪工作的艰巨性和长期性” (Meanings of this sentence can be ignored[3]) may generate 3,764,387,840 segmentation paths (Wang *et al.*, 2004). When a large-scale dataset is confronted, this method is difficult to be applied, unless additional information is available, *e.g.*, statistic information (Wang *et al.*, 2009). BiMM is frequently adopted for identifying CSA (Li *et al.*, 2003). It is easier to be implemented and has a higher speed. The disadvantage of BiMM is that overlapping ambiguity strings with even length (counted by characters) cannot be identified[4] (Sun *et al.*, 2001; Chang *et al.*, 2008). Furthermore, BiMM only identifies MOAS[5]. Without addition information, it cannot find individual Overlapping Ambiguity String (OAS) in a sentence. Therefore, many studies are mainly focused on MOAS

[2] Currently, more than 13000 characters and 69000 words are used by native Chinese people (*http://www.cp.com.cn/*).

[3] In Beijing's Great Hall, when meeting representatives attending the national court and the national court system against economic crime on behalf of advanced collective awards ceremony, Zemin Jiang asks everyone to fully understand that the work of combating economic crime is arduous and long-term.

[4] Section 2.2 gives an example of this combination pattern.

[5] A MOAS is an ambiguity string that no overlapping ambiguity is detected on both sides of the string. Formal definition can be seen in Definition 7.

(Sun *et al*., 1999; Li *et al*., 2008; Qiao *et al*., 2008; Li, 2011).

In this article, a Segmentation Matrix (SM) method is presented. It encodes lexical information of a sentence as a matrix. Then, set theory is developed to analyze CSA. With the complexity closing to BiMM, SM can identify every type of CSA the same as Omni-segmentation. In addition to CSA identification, SM is also available for Chinese word segmentation. Several lexicon-based methods are fully supported. Making use of the SM method, in our experiments, characteristics of CSA are studied, which show informative conclusions of CSA.

The contribution of this paper includes,

1. A SM approach is proposed, which encodes lexical information in a structured form. SM can make better use of sentence structure information for CSA analysis.
2. Formal definitions about CSA are defined under the framework of set theory, which maps string operations into set operations reducing the computational complexity.
3. Based on SM, characteristics of CSA are investigated. And several issues about CSA are studies.

The remainder of this paper is structured as follows: Section 2 reviews previous works. Section 3 gives formal definitions and notations about CSA. The SM method is discussed in Section 4. In Section 5, several issues about CSA are analyzed. The conclusion is given in Section 6.

2. Related Work

Given a sentence, CSA is identified when more than one segmentation paths are found. Therefore, CSA identification and Chinese word segmentation are two aspects of a problem. In this section, we first give a simple overview about Chinese word segmentation methods. Then CSA identification approaches are discussed.

2.1 Chinese Word Segmentation

Chinese word segmentation methods can be roughly classified into three categories: lexicon-based methods, statistical-based methods and hybrid methods.

Lexicon-based methods are easy to be implemented and has a high speed. They are still used for Chinese word segmentation in many applications. Maximum Matching (MM) is the most popular lexicon-based method. It is a greedy algorithm implemented by scanning a sentence from one side to another and greedily matching the longest lexicon entry until the end of a sentence is reached. There are two MM methods: Forward Maximum Matching (FMM) and Backward Maximum Matching (BMM). FMM scans from left to right, and BMM starts from the opposite direction.

In statistical-based methods, word segmentation is the way to find a segmentation path which has the maximum probability. They take advantages of mathematical models, such as Naive Bayesian (Li *et al*., 2003), Hidden Markov Model (Zhang *et al*., 2003), Conditional Random Fields (Peng *et al*., 2004; Tseng *et al*., 2005), Maximum Entropy (Xue, 2003), Graph-based Model (Zeng *et al*., 2013) and Compression-based algorithm (Teahan *et al*., 2000). There are research combine generative model with discriminative model, *e.g*., Wang *et al*. (2012). According the type of segmentation units, a sentence can be treated as a character sequence (character-based model) or a word sequence (word-based model). Studies were shown that the character-based approach are more successful (Xue, 2003; Zhao *et al*., 2010).

Hybrid methods integrate statistical-based and the lexicon-based methods (Gao *et al*., 2005) or utilize a joint model combining word segmentation with POS tagging or parsing (Wang *et al*., 2013; Sun, 2011; Li *et al*., 2011; Hatori *et al*., 2012). Hybrid methods try to use syntactic, semantic analyses or external knowledge to improve the performance (Wu *et al*., 1998; Huang *et al*., 2007; Zeng *et al*., 2011; Wu *et al*., 2011; Christiansen *et al*., 2011).

In statistical-based and hybrid methods, the tasks of word segmentation and CSA identification are combined into a unified framework. Therefore, the CSA problem is not obviously considered. As for the lexicon-based methods, greedy algorithms are used, which results in the CSA problem.

2.2 Chinese Segmentation Ambiguity Identification

In order to discuss related work, in this section, we use the same example provided by Wang *et al*. (2004): “当原子结合成分子时”[6]. The right segmentation path should be “当/原子/结合/成/分子/时” (when/ atoms/ combine/ molecules/ the time). The overlapping ambiguous string is “结合成分子时”. It can also be segmented into “结合/成分/子时” (combine/ ingredient/ midnight).

The central issue of CSA identification is that trying to find all possible segmentation paths. BiMM is the most popular method for CSA identification (Gao *et al*., 2011; Yao *et al*., 2012). It is implemented by running FMM and BMM respectively. The two outputs of FMM and BMM are compared. Different outputs imply the existence of segmentation ambiguity. The main disadvantage of BiMM is that overlapping ambiguity strings with even length cannot be identified. In this situation, both of FMM and BMM have the same output. As shown in Figure 1, two outputs of BiMM are the same.

[6] It can be translated into: “when atoms combine into molecules”.

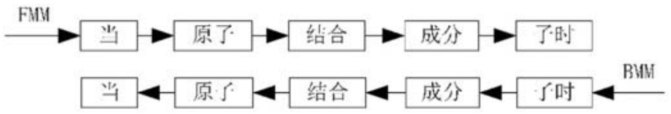


Figure 1. BiMM Method

Omni-segmentation tries to find every segmentation path in a sentence, which can be illustrated by a tree structure as shown in Figure 2. The root represents the start of a sentence. Nodes represent words of a sentence. Each branch implies a possible segmentation path.

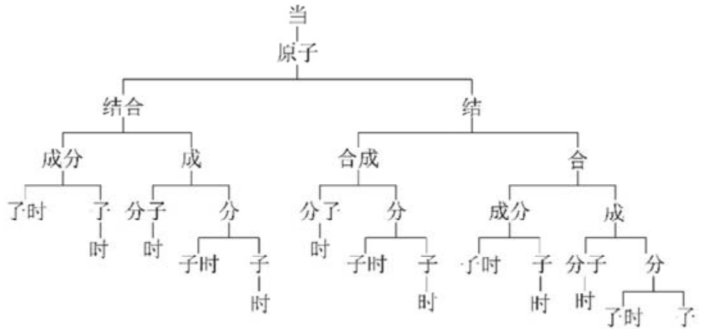


Figure 2. Omni-segmentation Method

The Directed Acyclic Graph (DAG) method was proposed by Zhang *et al*. (2002) as Figure 3 shows. Given a sentence represented as $c_0 \cdots c_{n-1}$ (c_i denotes Chinese characters), vertices $v_0, \cdots, v_n$ are used to separate it. Then, $v_0 c_0 v_1 c_1 \cdots v_{n-1} c_{n-1} v_n$ is generated. v_0 and v_n are the start and the end vertexes. If a substring $c_i \cdots c_j$ $(0 \le i, j \le n-1)$ matches a lexicon entry, then a directed edge (v_i, v_{j+1}) is added.

The DAG is used to collect possible segmentation paths. According to Zhang *et al*. (2002), if the 8-shortest paths are collected, this method can receive performance about 99.90% in recall to find correct segmentation paths.

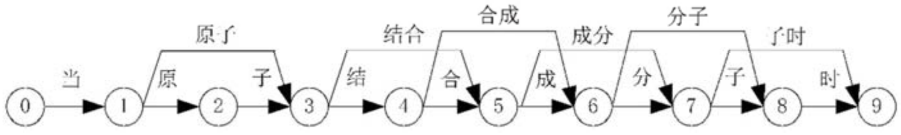


Figure 3. Directed Acyclic Graph

Wang *et al*. (2004) proposed a Maximum No-cover Ambiguity Graph (MNAG) as Figure 4 shows. Based on the principle of *Choosing the Longer Word*, MNAG can identify all overlapping ambiguities. This approach can reduce the number of segmentation paths. But identification of the combinational ambiguity is ignored.

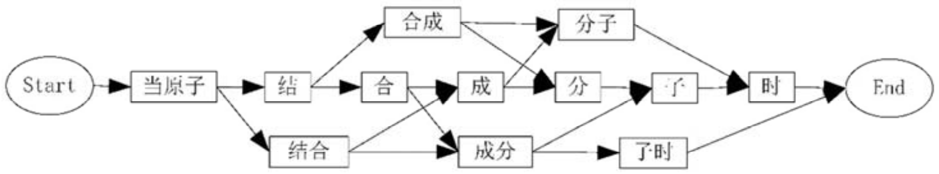


Figure 4. Maximum No-cover Ambiguity Graph

The word lattice method was proposed for Chinese word segmentation (Jiang *et al*., 2008). It is built by merging the output of outer segmenters. This method is mainly used as a re-ranking strategy. As shown in Figure 5, lattice nodes denote positions between characters. Edges covering subsequences of sentence denote words (Wang *et al*., 2013).

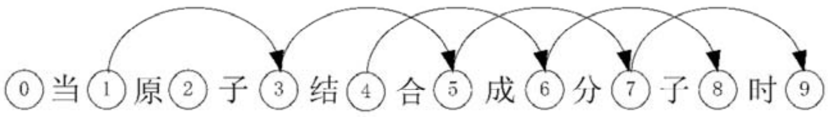


Figure 5. Word Lattice

There are other approaches proposed for CSA analyses, such as the *overlapping ambiguity elimination* model (Yao *et al*., 2012), the word-by-word scanning based maximum matching algorithm (Zhang *et al*., 2006; Sun *et al*., 2009), the method based on type theory (Gao *et al*., 2009) and the *coupling degree of double characters* method (Wand *et al*., 2007). These methods mainly focus on the overlapping ambiguity. Problems of combinational ambiguity and the difference between MOAS and OAS are rarely studied. In this paper, we propose a SM method. The detail of SM is discussed in Section 4. In the following, we first introduce definitions and notations used in this paper.

3. Definitions and Notations

Let $\mathbf{M}$ to be a segmentation matrix, $\mathbf{M}(i,j)$ is an element of $\mathbf{M}$ with coordinates Row i and Column j. A sentence is referred as S. The length of S is supposed to be n. We define two sets as:

$$\mathbf{P} = \{\ i \mid \text{an index of character position in } S; 0 \le i \le n-1; i \in N\}^{7}$$
$$\mathbf{W} = \{w \mid w \text{is a lexicon entry}\}$$

where N are the natural numbers. $\mathbf{P}$ is a partial order set. $\mathbf{W}$ denotes an employed lexicon. A closed interval $[i,j] = \{x | x \in \mathbf{P}, i \le x \le j\}$ denotes subset of $\mathbf{P}$. $S[i,j]$ $(i,j \in \mathbf{P}, i \le j)$ represents substring of S starting from the ith character to the jth character. By means of set operations, sentence operations are defined as follows

[7] Note that: all character positions are indexed from 0 to $n-1$.

$$(S[i,j] \cup S[i',j']) = S([i,j] \cup [i',j'])$$
$$(S[i,j] \cap S[i',j']) = S([i,j] \cap [i',j'])$$
$$S[i,j] \subseteq S[i',j'] \text{ iff } [i,j] \subseteq [i',j']$$
$$S[i,j] = S[i',j'] \text{ iff } [i,j] = [i',j']$$

Note that all sentence operations are implemented within indexes belonging to the same sentence S. Otherwise, these operations are nonsense.

Based on sentence operations, formal definitions about *segmentation path*, *combinational ambiguity*, *overlapping ambiguity*, etc. are defined as follows.

Definition 1: Let $\{[i,j_1],[j_1+1,j_2],\cdots,[j_m+1,j]\}$ to be a partition of $[i,j]$, then $\{S[i,j_1],S[j_1+1,j_2],\cdots,S[j_m+1,j]\}$ is a **Segmentation Path** of $S[i,j]$, referred to as $S(j_1,j_2,\cdots,j)$ or $S[i,j_1]/S[j_1+1,j_2]/\cdots/S[j_m+1,j]$.

In other words, a segmentation path is a partition of $\mathbf{P}$. For example, let S="温柔和善解人意" (gentle and understanding), because $\{[0,1],[2,2],[3,6]\}$ is a partition of $[0,6]$, then $S(1,2,6)$ is a segmentation path (or $S[0,1]/S[2,2]/S[3,6]$), which denotes "温柔/和/善解人意" (gentle/ and/ understanding). In this paper, $S[i,j]$ (square bracket) represents substring of S, and $S(i,j)$ (parentheses) represents a segmentation path.

Definition 2: Let $SP=\{[i,j_1],[j_1+1,j_2],\cdots,[j_m+1,j]\}$ to be a segmentation path, if $\forall w \in SP$ such that $w \in \mathbf{W}$. Then, the segmentation path SP is **in accord with** $\mathbf{W}$.

In this paper, we assume that all mentioned segmentation paths satisfy this constraint.

Definition 3: Let $S[i,j] \in \mathbf{W}$. If $SP=\{[i,j_1],[j_1+1,j_2],\cdots,[j_m+1,j]\}$ is a partition of $[i,j]$, then S has the **Combinational Ambiguity** (CA), and $S[i,j]$ is a Combinational Ambiguity String (CAS).

For example, let S="温柔和善解人意", $S[3,6]$="善解人意", $S[3,6] \in \mathbf{W}$ and $\{[3,3],[4,4],[5,6]\}$ is a partition of $[3,6]$. Because $S[3,3]$="善", $S[3,3] \in \mathbf{W}$, $S[4,4]$="解", $S[4,4] \in \mathbf{W}$, $S[5,6]$="人意", $S[5,6] \in \mathbf{W}$, therefore, S has combinational ambiguity. $S[3,6]$ is a CAS, referred to as $CAS[3,6]$.

In Chinese, almost every character can function either as a word or as a morpheme (Chen *et al.*, 1998). If Definition 3 is adopted, then words exceeding two characters will lead to the combinational ambiguity. Because disambiguation for combinational ambiguity is difficult (Luo *et al.*, 2002). Therefore, to reduce the combinational ambiguity problem, the following combinational ambiguity definition in a narrow sense is proposed.

Definition 4: Let $S[i,j] \in \mathbf{W}$, if $\exists j'(i \le j' < j)$ such that $S[i,j'] \in \mathbf{W} \wedge S[j'+1,j] \in \mathbf{W}$, then S has the **Narrow Sense Combinational Ambiguity**.

This definition is the same as that proposed by Liang (1984). In this paper, the narrow sense combinational ambiguity is used as our default definition, also referred as combinational ambiguity except where noted.

Definition 5: Let $S[i,j] \in \mathbf{W}$ and $S[i',j'] \in \mathbf{W}$, if $i < i' \le j$ and $[i,j] \cap [i',j'] \ne \varnothing$, then S has the **Overlapping Ambiguity** (OA). $S[i,j] \cup S[i',j']$ is an Overlapping Ambiguity String (OAS).

If $S[i,j]$ and $S[i',j']$ have overlapping ambiguity, then $S[i,j] \cap S[i',j']$ is an **Overlapping Chain String** (OCS) and $|S[i,j] \cap S[i',j']|$ is **Overlapping Chain Length** (OCL), where $|S[i,j] \cap S[i',j']|$ is the cardinality of $S[i,j] \cap S[i',j']$.

For example, let S=“温柔和善解人意”, $S[0,2]$=“温柔和”, $S[0,1] \in \mathbf{W}$ and $S[1,2] \in \mathbf{W}$. Because $[0,1] \cap [1,2] = [1,1] \ne \varnothing$, so that S has overlapping ambiguity. $S[0,1] \cup S[1,2] = S[0,2]$ is an OAS. $S[0,1] \cap S[1,2] = S[1,1]$ is an OCS, and OCL of $S[0,2]$ is 1.

In this paper, CAS and OAS are collectively referred as **Ambiguity String** (AS). Prefixes OA, CA and OC are used to indicate properties of $S[i,j]$ (Overlapping Ambiguity, Combinational Ambiguity and Overlapping Chain). For example, $OAS[i,j]$ means that $S[i,j]$ is an OAS.

Definition 6: Given $OAS[i,j]$ and $OAS[i',j']$, if $OAS[i,j] \cap OAS[i',j'] \ne \varnothing$, then $OAS[i,j]$ and $OAS[i',j']$ are **Addable**.

If $OAS[i,j]$ and $OAS[i',j']$ are addable, then the sum of $OAS[i,j]$ and $OAS[i',j']$ is $S([i,j] \cup [i',j'])$. It is also an OAS. If two OAS are addable, the overlapping chain string of $OAS([i,j] \cup [i',j'])$ can be calculated by

$$f(OAS[i,j] \cup OAS[i',j']) = f(OAS[i,j]) \cup (\overline{f(OAS[i,j])} \cap \overline{f(OAS[i',j'])}) \cup f(OAS[i',j']) \quad (1)$$

where $\overline{f(OAS[i,j])}$ is the relative complement of $f(OAS[i,j])$ in $OAS[i,j] \cup OAS[i',j']$.

In this field, the Maximum Overlapping Ambiguity String (MOAS) is widely used. It is defined as follows.

Definition 7: In a given sentence S, if an $OAS[i,j]$ is not addable with other OAS in S, then this $OAS[i,j]$ is a **Maximum Overlapping Ambiguity String** (MOAS).

For example, S_1="温柔和善解人意" has three OAS: $OAS[0,2]$, $OAS[1,3]$ and $OAS[2,6]$. All of them are addable. The result is $MOAS[0,6]$. By Eq. (1), an overlapping chain string of S_1 is $f(S[0,6]) = OCS[1,3]$. For another example, S_2="逐渐变成暗红色" has $OAS[0,3]$ and $OAS[4,6]$. $OAS[0,3]$ and $OAS[4,6]$ are not addable, then both of them are MOAS.

Given a set of OAS (referred as $\mathcal{A}_{OAS}$) in a sentence, the set of MOAS (referred to as $\mathcal{A}_{MOAS}$) is computed by merging every OAS that are *addable*. It is computed as follows.

$$\mathcal{A}_{MOAS} = \{a | (s,t \in (\mathcal{A}_{OAS} \cup \mathcal{A}_{MOAS})) \wedge (a = s \cup t) \wedge (s \cap t \neq \varnothing)\} \quad (2)$$

Because $\mathcal{A}_{MOAS}$ appears on both sides of this equation, it is an iterative process. It will be discussed in Section 4.2.3 that $\mathcal{A}_{MOAS}$ is easy to be implemented, because all elements in $\mathcal{A}_{OAS}$ and $\mathcal{A}_{MOAS}$ are ordered.

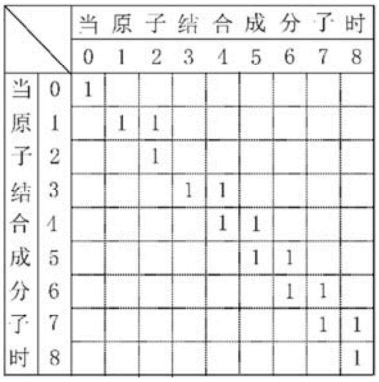
In ancient Chinese, there are no punctuations between sentences. Currently, symbols such as the period ("。"), question mark ("？"), exclamatory mark ("！"), semicolon ("；") or comma ("，") are widely used as sentence boundaries. The problem is that using of the comma is ambiguous. It may function as a sentence boundary or a separation of clauses. Therefore, disambiguation of sentence boundary is required (Xue *et al*., 2011). Because lots of language characteristics cannot exist while crossing punctuation, *e.g*., segmentation ambiguity, named entity, etc. (Chen *et al*., 2015b), **Sentence Fragment** is used to denote *substring of a sentence divided by punctuations.*

Definition 8: Sentence fragment is a substring of a sentence that contains no punctuation.

This notion is useful for Chinese NLP, *e.g*., Zhang *et al*. (2013), especially for unsupervised machine learning method, *e.g*., Zhang *et al*. (2003), Li *et al*. (2009).

4. Segmentation Matrix

Figure 6 gives an example of SM. Coordinates of SM represent characters of S. The element data type of SM is Boolean. $\mathbf{M}(i,j)=1$ means that word $S[i,j]\in\mathbf{W}$. To build SM of a sentence, by scanning a given sentence, when a lexicon entry is matched, the corresponding element is set to 1, otherwise to 0.



		当	原	子	结	合	成	分	了	时
		0	1	2	3	4	5	6	7	8
当	0	1								
原	1		1	1						
子	2			1						
结	3				1	1				
合	4					1	1			
成	5						1	1		
分	6							1	1	
了	7								1	1
时	8									1

Figure 6. Segmentation Matrix

4.1 Ambiguity Identification on SM

Following the definition of the *Combinational Ambiguity in a Narrow Sense* (See Definition 4), Solution 1 is proposed to identify combinational ambiguity strings.

Solution 1: For $\forall i \forall j(\mathbf{M}(i,j)=1)$, if $\exists j'(i\le j'<j\wedge\mathbf{M}(i,j')=1\wedge\mathbf{M}(j'+1,j)=1)$, then $S[i,j]$ is a CAS.

For example, in Figure 6, $\mathbf{M}(i,j)=\mathbf{M}(1,2)=1$, $\mathbf{M}(i,j')=\mathbf{M}(1,1)=1$, $i=1$, $j=2$, $j'=1$, $\mathbf{M}(j'+1,j)=\mathbf{M}(2,2)=1$, then $S[1,2]$ is a CAS. Based on Solution 1, Algorithm 1 gives the implementation of this solution[8].

[8] In this paper, the algorithms are given in C++ codes. Some changes are made for the sake of simplicity and convenience.

Algorithm 1

CA Identification on SM

Input:

1. SegMatrix[][], SM;

2. n, length of the given sentence;

3. l, length of the longest lexicon entry;

Output:

Combinational ambiguity strings are stored in the vector CAS_v;

Method:

```
(0)  for(int $i = 0$; $i < n$; $i + +$){
(1)      for(int $j = min(n-1, i+l)$; $j > i$; $j - -$){
(2)          if(SegMatrix[i][j]){                         \\$\mathbf{M}(i,j) = 1$
(3)              for(int $j' = j - 1$; $j' \geq i$; $j' - -$){
(4)                  if(SegMatrix[i][j'] && SegMatrix[$j' + 1$][j]){
(5)                      $CAS_v$.push_back(i, j', j);
(6)  }   }   }   }   }
```

As Algorithm 1 shows, three loops are used. Except the outer loop has an index n, another two nested loops have cycle indexes less than l. Therefore, the complexity of combinational ambiguity identification is $O(l^2n)$. The function min in Row (1) is adopted to decrease the search space. If a "break" clause is in the pace after Row (5), the algorithm will return when the combinational ambiguity is identified. Otherwise, as it shows, every combinational ambiguity in the sentence is collected.

Following Definition 3, Solution 2 is proposed to identify the overlapping ambiguity string.

Solution 2: For $\forall i \forall j (M(i,j) = 1)$, if $\exists i' \exists j' (\mathbf{M}(i', j') = 1 \wedge i' < i \wedge j' < j \wedge i \leq j')$, then $S[i', j]$ is an OAS, $S[j', i]$ is the overlapping chain string, and the overlapping chain length is $j' - i + 1$.

For example, in Figure 6, because $\mathbf{M}(i,j) = \mathbf{M}(4,5) = 1$, $\mathbf{M}(i', j') = \mathbf{M}(3,4) = 1$, $i = 4$, $j = 5$, $i' = 3$ and $j' = 4$, then $S[3,5]$ is an OAS, overlapping chain length equals 1 ($j' - i + 1 = 4 - 4 + 1 = 1$). Algorithm 2 implements the Solution 2.

Algorithm 2
Overlapping Ambiguity Identification on SM

Input:

1. SegMatrix[][], SM;
2. n, length of the given sentence;
3. l, length of the longest lexicon entry;

Output:

Overlapping ambiguity is stored in vector OAS_v;

Method:

```
(0)  for(int $i=0$; $i<n$; $i++$){
(1)      for(int $j=min(n-1,i+l)$; $j>i$ && $i<n$; $j--$){
(2)          if(SegMatrix[i][j]){                         \\$\mathbf{M}(i,j)=1$
(3)              for(int $i'=i-1$; $i'\geq 0$; $i'--$){
(4)                  for(int $j'=j-1$; $j'\geq i$; $j'--$){
(5)                      if(SegMatrix[i'][j'])\{          \\$\exists i'\exists j'(\mathbf{M}(i',j')=1)$
(6)                          $OAS_v$.push_back(i, j, i', j');
(7)  }   }   }   }   }   }
```

In Algorithm 2, four loops are used. The outer loop has index n. The other three nested loops have cycle index less than l. The complexity of overlapping ambiguity identification is $O(l^3n)$. Using Algorithm 2, instead of MOAS, each overlapping ambiguity string is recognized individually. After every OAS is identified, MOAS can be obtained by merging OAS that is addable (See Eq. (2)).

4.2 Segmentation on SM

In this section, we illustrate how Chinese word segmentation algorithms can be implemented on SM. Four lexicon based methods (FMM, BMM, BiMM and Omni-segmentation) are discussed. By mapping string operations into set operations, these processes only implement Boolean operations, which reduces the computing complexity.

4.2.1 FMM

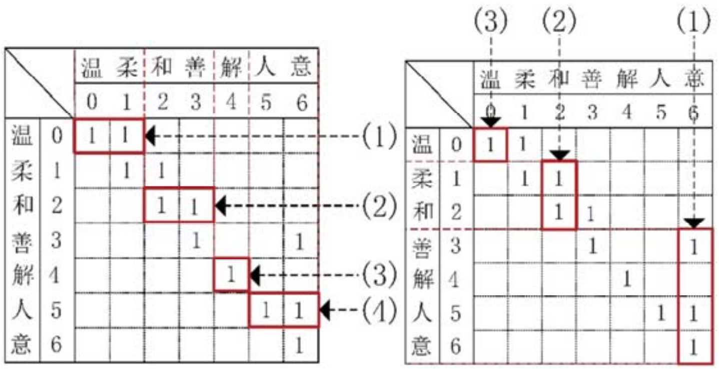
FMM is implemented by scanning each row of SM from right to left. If $\mathbf{M}(i,j)$ equals 1, then hold the coordinate j as j_0, and scan from the (j_0+1)th row again. Iterate this way until the end of the SM ($j=n-1$) is reached. Then, $S(j_0,j_1,\cdots,n-1)$ is a segmentation path of FMM. Step 1 to 3 give an example of this algorithm. Figure 7(a) shows the visualized process.

Step 1: Scan the 0th row from Column 6 to Column 0. Hit 1 at $\mathbf{M}(0,1)$, then set 1 as j_0.

Step 2: Scan the (j_0+1)th row in the same way, if $\mathbf{M}(i,j)$ equals 1, record every j.

Step 3: Iterate this way until the column-coordinate j equals 6.

As shown in Figure 7(a), the output is $S(1,3,4,6)$.



(a) FMM (b) BMM

Figure 7. Maximum Matching Segmentation

4.2.2 BMM

BMM is similar as FMM. The difference is that BMM processes the last column first and scan each column from top to bottom. If $\mathbf{M}(i,j)$ equals 1, hold *j* and restart from $(i-1)$th column again until the 0th column is reached.

For example, in Figure 7(b), first hit 1 at $\mathbf{M}(3,6)$, then hold 6 and scan from the 2th column again. In Column 2, $\mathbf{M}(1,2)$ equals 1, restart from the $(i-1)$th column until the column-coordination 0 is met. The output of BMM in this example is $S(0,2,6)$.

4.2.3 BiMM on SM

BiMM is implemented by running both FMM and BMM respectively. Two outputs are compared. Let $S_F(\cdots,i,m,\cdots,k,j,n-1)$ and $S_B(\cdots,i,s,\cdots,t,j,n-1)$ to be the output of FMM and BMM. Comparing S_F and S_B from right to left, if both coordinates[9] have the same value, hold this same value and decrease both by 1. Then, compare the new

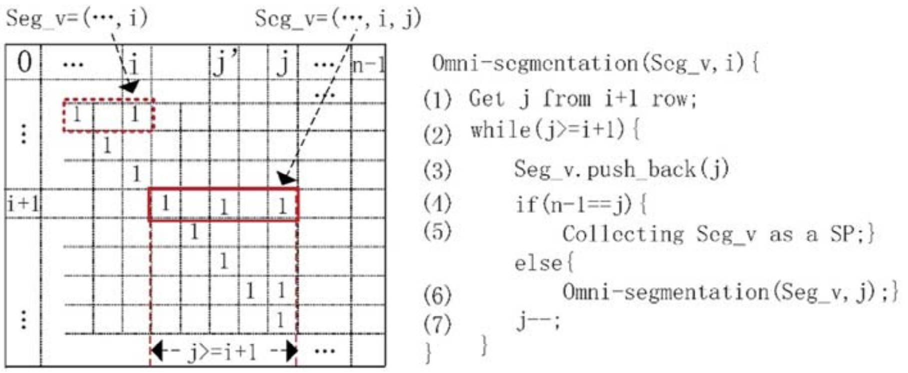
[9] In this place, segmentation path $S(j_1,j_2\cdots,j)$ are seen as a vector. The values of $S(j_1,j_2\cdots,j)$ are j_1, j_2, etc. The coordinates of $S(j_1,j_2\cdots,j)$ are the position index of this vector. For example, in $S(j_1,j_2\cdots,j)$, the value of coordinate 0 is j_1.

coordinates again and always update the held value if both are equal, until the unequal value is met for the first time. Now, the held value is the end of OAS (*e.g.* j in S_F and S_B). Subtract 1 to the coordinate with larger value. Continue this way, until the equal value is found again (*e.g.* i in S_F and S_B). At last, it is the start of the OAS. In this example, the OAS is $S[i+1,j]$. Iterate this way until both S_F and S_B are traversed.

4.2.4 Omni-segmentation on SM

Omni-segmentation tries to find every segmentation path in a given sentence. The number of segmentation paths may explode tremendously. It has the highest computational and space complexity. Based on SM, utilizing segmentation ambiguity information, we can apply Omni-segmentation method on substrings that have the segmentation ambiguity problem, then reducing generated segmentation paths.

For the reason that Omni-segmentation is useful in Chinese NLP, *e.g.*, Chen *et al.* (2014), Chen *et al.* (2015a), we give the algorithm that can be implemented on SM. As shown in Figure 8, this algorithm utilizes an iterative method, which can make better use of sentence structure information.



```
Omni-segmentation(Seg_v, i){
(1) Get j from i+1 row;
(2) while(j>=i+1){
(3)     Seg_v.push_back(j)
(4)     if(n-1==j){
(5)         Collecting Seg_v as a SP;}
        else{
(6)         Omni-segmentation(Seg_v, j);}
(7)     j--;
}   }
```

Figure 8. Omni-Segmentation on SM

Suppose that one output of the kth iteration is $Seg\_v = (\cdots, i)$. In the $(k+1)$th iteration, the $(i+1)$th Row is processed. In Step (1), the largest *j* with $\mathbf{M}(i+1,j)=1$ is obtained. In Step (2), add *j* into segmentation path Seg_v and judge whether or not j is the end of a sentence. If it equals $n-1$, then Seg_v is a segmentation path. If not, iterate this process. For all $\mathbf{M}(i+1,j)=1$ in Row $i+1$, recycle from Step (3) Step (7).

This algorithm may generate tremendous segmentation paths. The combinational ambiguity can be filtered for simplicity.

4.3 Complexity

In order to identify CSA, both Omni-segmentation and BiMM methods try to segment a sentence for finding possible segmentation paths. The lexicon is required to be accessed frequently, and generated segmentation paths must be held for comparison. These processes involve massive string manipulations and string storages, leading to a higher computational and space complexity. In SM, after SM was built, string operations are mapped into set operations. There is no need for SM to implement string operations and access the lexicon. Moreover, SM can make better use of sentence structure information, decreasing the computational complexity. This section discusses the complexity of SM.

Let m to be the number of lexicon entry, n is the length of a given sentence, l is the length of the longest lexicon entry. In a given lexicon, l is a constant. The computational and space complexity are given as follows.

4.3.1 Computational Complexity

Searching for an element in a lexicon has computational complexity $O(\log m)$. Finding every word in a sentence need access lexicon $O(l \times n)$ times in the worst case. Therefore, the construction of SM has computational complexity $O(l \times n \times \log m)$. This is the same as FMM or BMM. Because BiMM implements both FMM and BMM, so BiMM has complexity $O(2 \times l \times n \times \log m + c)$. Where c denotes the complexity to compare the output of FMM and BMM.

In the worst scenario, SM has $O(l \times n)$ elements equal 1. In order to identify each overlapping ambiguity, for each $\mathbf{M}(i, j) = 1$, $O(l \times l)$ elements of M should be scanned. So identification of the OAS has complexity $O(l \times n \times \log m + l^3 \times n)$. Because l^2 and $\log m$ have the similar order, identification of OAS on SM has computational complexity close to BiMM.

4.3.2 Space Complexity

BiMM generates two FMM and BMM output. The space complexity for BiMM is constant. When Omni-segmentation is employed, segmentation path can grow tremendously, therefore, leading to a higher space complexity. In BiMM and Omni-segmentation, generated segmentation paths need to be held for OAS or CAS identification.

In the SM method, a $n \times n$ matrix is required. It seems that the space complexity for SM is $O(n^2)$. But in practical application, we deal with a sentence or sentence fragment. Storages of matrix can be used repeatedly. OAS or CAS can be identified directly without saving any segmentation path. Therefore, space complexity of SM also closes to BiMM.

5. Experiments

Before conducting experiments, we give an example to show the comparison between SM and BiMM. The sentence is given as “江泽民在北京人民大会堂会见参加全国法院工作会议和全国法院系统打击经济犯罪先进集体表彰大会代表时要求大家要充分认识打击经济犯罪工作的艰巨性和长期性”. The outputs of FMM and BMM are listed as follows[10]:

S_F(0, 1, 2, 3, 5, 7, 9, 11, 12, 14, 16, 18, 20, 22, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 46, 48, 50, 51, 53, 55, 57, 59, 61, 62);

S_B(0, 1, 2, 3, 5, 7, 8, 10, 12, 14, 16, 18, 20, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 46, 48, 50, 51, 53, 55, 57, 59, 61, 62);

Making use of the BiMM method, 2 **MOAS** are detected: $MOAS[8,12]$, $MOAS[21,23]$. However, if SM method is employed, 5 **MOAS** and 10 **OAS** are found: $MOAS[8,12]$, $MOAS[15,18]$, $MOAS[21,23]$, $MOAS[24,27]$, $MOAS[38,41]$, $OAS[8,10]$, $OAS[9,11]$, $OAS[10,12]$, $OAS[15,17]$, $OAS[16,18]$, $OAS[21,23]$, $OAS[24,26]$, $OAS[25,27]$, $OAS[38,40]$, $OAS[39,41]$.

This example shows that BiMM is insufficient for OAS identification. To show more information, based on the Beijing University corpus (PKU corpus) of the Chinese word segmentation Bakeoff training data (Sproat *et al*., 2003), SM is compared with BiMM, DAG, and MNAG methods, as shown in Table 1.

Table 1. Comparison With Other Methods.

Model	MOAS		OAS		CAS	
	Type	Count	Type	Count	Type	Count
BiMM	8,409	19,090	×	×	×	×
DAG	7,369	12,337	18,888	51,895	38,200	515,151
MNAG	7,956	13,870	7,378	18,641	×	×
SM	19,269	52,072	26,580	101,514	39,310	555,574

Where, “×” means that this type of ambiguity cannot be identified by the corresponding method. It can be seen that SM shows better performance. Making use of the SM method, in the rest part of this section, several issues about CSA are discussed.

[10] The output may differ when different lexicon is adopted. In this place, we take the *Lexicon Common Words in Contemporary Chinese*.

In all experiments, we use the traditional $Precision$/$Recall$/$F\text{-}score$ (*P*/*R*/*F*) measurement to evaluate the performance. *Precision* is computed by $Correct\ Num / Extract\ Num$, and *Recall* is computed by $Correct\ Num / Real\ Num$. Where $Correct\ Num$ is the number of correctly recognized relation instances. $Extract\ Num$ is the number of instances that have been extracted. $Real\ Num$ refers to the number of annotated relation instances. F-score is computed by

$$\frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

In the PKU corpus, the training data and testing data are provided. In the Penn Chinese Treebank corpus, the 5-fold cross validation is adopted for training and testing. We average the results of five runs as the final performance. To implement the maximum entropy classifiers, we used the toolkit provided by Zhang (2004). We also run a CRF model for comparison[11].

5.1 Characteristics of Chinese Segmentation Ambiguity

Characteristics of CSA have been investigated by other research. Sun *et al*. (1999) analyzed MOAS in a corpus containing 100 million characters. Li *et al.* (2003) studied 730,000 MOAS extracted from 20 years the *People's Daily* corpus. Li *et al*. (2006) collected 14,906 high frequent MOAS from the *People's Daily* corpus. Qiao *et al*. (2008) investigated MOAS in several corpora, which has more than 1 billion characters.

In general, these research is mainly focused on analysing MOAS for a given corpus. Rare research was conducted to study the characteristics of CSA in a given lexicon. This section is devoted to this. The *Lexicon of Common Words in Contemporary Chinese* is employed, which contains 56,008 words and is published by the *Ministry of Education of the People's Republic of China* in 2008. Table 2 shows detected segmentation ambiguity information.

Table 2. Ambiguity about Lexicon.

CAS Inside Word	39,944	OAS in Overlapping Words	1,847,814
OAS Inside Word	939	OAS in Adjacent Words	1,757,756
		Total OAS	1,847,814

CAS (OAS) Inside Word refers to overlapping (or combinational) ambiguity strings in a single word. *OAS in Overlapping Words* refers to overlapping ambiguity strings that are generated by overlapping two possible words. For example, “文科” (Liberal Arts) and “科学” (Science) can be overlapped to generate an OAS “文科学”. *OAS in Adjacent Words* denotes ambiguity strings generated by two adjacent words (no overlapping). For example, “点” (Point)

[11] *http://crfpp.googlecode.com/svn/trunk/doc/index.html.*

and “射门” (Shot) can be combined into an OAS “点射门”. It can be segmented as “点/射门” (Point/ Shot) or “点射/门” (Fixed Fire/ Door). *Total OAS Types* is produced by merging results in *OAS in Overlapping Words* and *OAS in Adjacent Words*. It can be seen as the overlapping ambiguity space.

As shown in Table 1, combinational ambiguities inside words are pervasive, even the Definition 4 is adopted. Except 2,927 words containing only single character, 75.25% words have combinational ambiguity. The *Total OAS Types* has the same number as *OAS in Overlapping Word*, so that *OAS in Adjacent Words* can be seen as a subset of *OAS in Overlapping Words*.

In the following, we investigate CSA in a large-scale corpus. The corpus contains 52,961 texts involving various literary genres. Because CSA cannot exist across punctuation. Therefore, instead of whole sentences, we take sentence fragments under consideration. After erasing duplicated sentence fragments, there are 0.2 billion sentence fragments remained. The information is shown in Table 3.

Table 3. Information of Corpus.

Corpus Size	8.26 Gigabyte	Texts	52,961
Total Characters	2,703,512,684	Total Words	1,902,306,846
Token	69,087	Sentence Fragments	264,748,094

Figure 9 shows the distribution of sentence fragment length in our corpus.

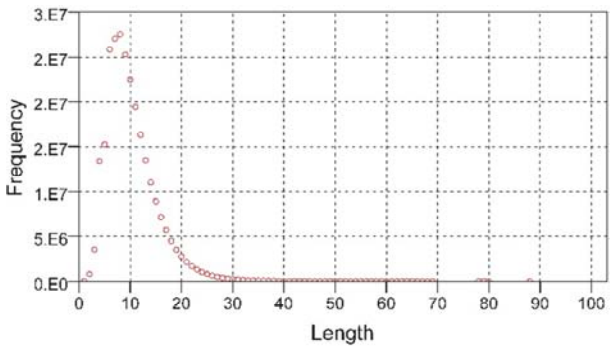


Figure 9. Distribution of Sentence Fragments Length

Almost 99% sentence fragments have length less than 26. Therefore, we set 128 as the size of SM. Longer sentence fragments are removed directly.

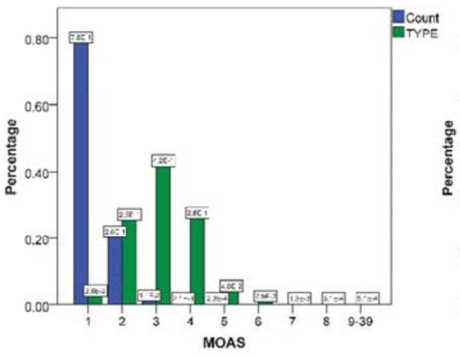
For each sentence fragment, Algorithm 1 and Algorithm 2 (See Section 4.1) are adopted to extract CAS and OAS. MOAS is obtained by merging OAS that are addable. These MOAS are referred to as *SM-MOAS*. In order to give a comparison, BiMM (See Section 4.2.3) is implemented to extract MOAS, referred as *BiMM-MOAS*. Table 4 gives information about CAS, OAS, *SM-MOAS* and *BiMM-MOAS*.

Table 4. Ambiguity about Corpus.

Ambiguity Type	CAS	OAS	SM-MOAS	BiMM-MOAS
Type	39,810	732,873	1,190,606	526,251
Count	579,238,862	40,921,520	32,641,105	23,424,525

Referring to Table 2, nearly all CAS types occurred in our corpus, but only 39.66% OAS types occurred. If the BiMM method is used, there are 91.52% sentence fragments having no overlapping ambiguity. If the SM method is employed, the rate reduces to 88.38%. It means that, by simple method, it is possible to get a massive sentence fragments without overlapping ambiguity for unsupervised methods (Li *et al*., 2003).

From Table 4, compared *SM-MOAS* and *BiMM-MOAS*, it can be seen that the number of *SM-MOAS* type is doubled. Therefore, only focusing on MOAS produced by BiMM is insufficient to study the CSA problem. In Figure 10, distributions of overlapping chain length about MOAS and OAS are compared.



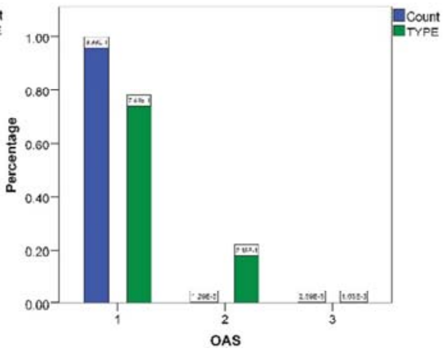
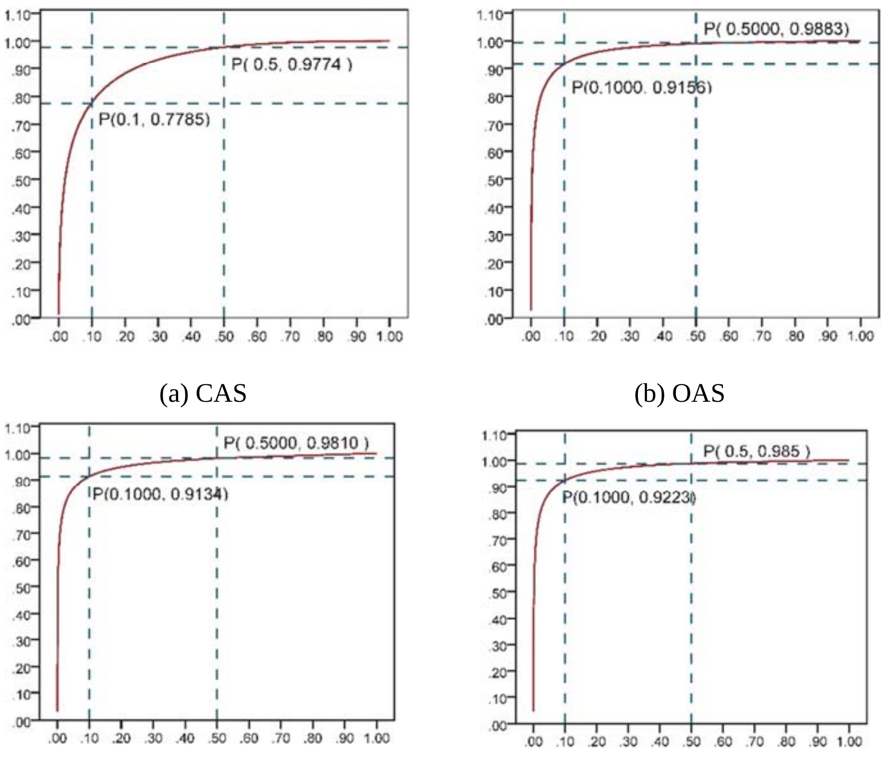


Figure 10. Distribution of Overlapping Chain Length

It can be seen that the distribution of overlapping chain length in MOAS is more complex, ranging from 1 to 39. However, it is simpler in OAS. There are 99.87% OAS has overlapping chain length equal to 1. This conclusion is useful for disambiguation. It can be modelled as a two-category classification problem.

Figure 11 shows the distributions of different CSA. X-axis represents the percentage of ambiguity string types in frequency-descending order. Y-axis is the percentage of occurrences. $P(x, y)$ represents x% of the highest frequency ambiguity string types covering y% occurrences. For each type of segmentation ambiguity, 10% high frequency ambiguity string types occupy 90% occurrences.



(a) CAS (b) OAS

(c) MOAS (d) BiMM MOAS

Figure 11. Distribution of Segmentation Ambiguity

5.2 Influence of Ambiguities on Word Segmentation

Based on FMM and BMM, Sun *et al.* (1995) analyzed the influence of CSA on Chinese word segmentation. Several conclusions were induced. These analyses mainly based on a corpus containing only 3,680 sentences. The influence of combinational ambiguity on Chinese word segmentation wasn't studied. In this experiment, the Penn Chinese Treebank corpus[12] is used to analyze the influence of CSA on Chinese word segmentation. This corpus is manually segmented, consisting of 2,448 text files, 71,232 sentences, 1,196,329 words and 1,931,381 Hanzi (Chinese character). The segmentation ambiguity information is given in Table 5.

Table 5. Ambiguity about Penn Chinese TreeBank.

Ambiguity Type	CAS	OAS	SM-MOAS	BiMM-MOAS
Type	11,672	24,069	17,868	13,5591
Count	61,615	81,694	47,417	18,8275

[12] *http://www.cis.upenn.edu/~chinese/ctb.html*

Characteristics of CSA about the Penn Chinese Treebank are the same as our corpus discussed in Section 5.1.

Before given the experiment in detail, we first explain the terms used in this part. The meaning of ambiguity free has two levels. The first is that, for a given lexicon, a sentence has no segmentation ambiguity. The other is that a sentence may contain segmentation ambiguity that cannot be identified by an employed method. SM can identify every segmentation ambiguity. Therefore, *SM Free* means that a sentence contains no segmentation ambiguity at all. But *BiMM Free* is not. It only means that no segmentation ambiguity can be identified by the BiMM method.

Because sentence boundaries are manually labelled in the corpus, therefore, instead of segmentation fragments, the sentences are directly used as segmenting units, which contains 71,232 sentences. Among them, 56,618 sentences are *BiMM free*, and 43,444 sentences are *SM free*. Then, FMM or BMM is employed to segment collected sentences[13]. Performances are given in Table 6. Column 2 in Table 6 lists the number of selected sentences.

Table 6. Influence of OAS.

Method	Sentence	Precision	Recall	F1
FMM	71,232	95.08%	92.14%	93.59%
BMM	71,232	95.05%	92.09%	93.58%
BiMM Free	56,618	96.67%	93.61%	95.12%
SM Free	43,444	96.74%	93.68%	95.19%

Using FMM (or BMM) method, the performance is already upto 93.59% (93.58%) in F-score. In Row 3, if both FMM and BMM have the same output (*BiMM Free*), the precision is 96.67%. In Row 4, *SM free* means that there is no overlapping ambiguity at all, but segmentation precision is only 96.74%. Therefore, combinational ambiguity can cause segmentation errors about 3.3%.

5.3 Influence of Lexicon size on Chinese Word Segmentation

Making use of syntax and semantic knowledge, machine learning methods are successful to process the CSA problem. But these methods also have the disadvantage that annotated data is required, which is time-consuming and costly in human labor, and migrating between different applications is difficult. Lexicon-based method is easier to be implemented and has reasonable performance. Because only a lexicon is required for segmentation, the requirement for annotated data and the process for training are avoided. Therefore, the lexicon-based method

[13] The employed lexicon is directly extracted from the same corpus.

still be used in this field. In this section, the influence of lexicon size on Chinese word segmentation is studied. We use the PKU testing data. The FMM is used as the default method. This issue was discussed by other researchers (*e.g.* Sun *et al.*, (2001)), but no quantitative analysis is given. The result is shown in Table 7.

Table 7. Influence of Lexicon Size.

No.	Lexicon	Entries	Precision	Recall	F1
1	Testing Words	13,148	98.95%	98.62%	98.78%
2	Training Words	55,303	84.34%	90.77%	87.44%
3	(2) + CWCC	85,486	85.32%	90.23%	87.71%
4	(3) + Medium Lexicon	312,065	83.42%	83.16%	83.29%
5	(4) + Maximum Lexicon	554,331	81.23%	80.51%	80.87%
6	(5) + Testing Words	555,475	89.16%	83.49%	86.23%
7	(2) + Testing Words	58,166	97.26%	95.93%	96.59%

In Table 7, five lexicons are employed. *Testing Words* are words collected from the testing data. Words in *Training Words* are collected from training data. Performances generated by both are used as the topline and baseline in the Chinese word segmentation Bakeoff competition (Emerson, 2005). *CWCC* denotes the *Lexicon of Common Words* in *Contemporary Chinese*. *Medium Lexicon* is collected from the Internet, which contains 298,032 words. *Maximum Lexicon* is generated by merging *Medium Lexicon* and a *Great Dictionary of Chinese* with 542,240 lexicon.

In Chinese word segmentation, OOV (out-of-vocabulary) is considered as the main obstacle to segment a sentence (Sproat *et al*., 2003; Huang *et al*., 2007). Comparing Row 7 to Row 2 and Row 6 to Row 5, after testing words was added, the performances increases 9.15% and 5.36% respectively. When the lexicon size increased, the influence of OOV is slacked down. By comparing the Row 6 to Row 7, the lexicon used by Row 7 is a subset of Row 6, but Row 6 is lower than Row 7 about 10.36%. It is caused by overlapping and combinational ambiguities. Row 1 and Row 6 also have the same problem. Without segmentation disambiguation, increasing lexicon size can result in worse performance in lexicon-based methods. In order to see the influences in more details, Table 8 lists the number of errors caused in the segmentation.

Table 8. Information about Error.

No.	Lexicon	Total	By OAS	By CAS	By OOV
1	Testing Words	712	355	357	0
2	Training Words	7,386	752	1,298	5,336
3	(2) + CWCC	7,130	900	1,894	4,336
4	(3) + Medium Lexicon	9,695	1,595	5,433	2,667
5	(4) + Maximum Lexicon	10,954	2,363	5,982	2,609
6	(5) + Testing Words	8,110	2,088	6,022	0
7	(2) + Testing Words	2,045	697	1,348	0

In Table 8, the strategy to count the number of errors is explained as follows. If a segmentation path "A/ BC", is falsely segmented into "AB/ C" (A, B and C are characters). Then this failure is counted as an OAS error. If a segmentation path "A/ B" is falsely segmented into a word "AB" (combinational ambiguity), it is counted as a CAS error. An OOV error is caused by a word (*e.g.* "AB") falsely segmented into small pieces ("A/ B"). Figure 12 compares errors caused by OAS, CAS and OOV.

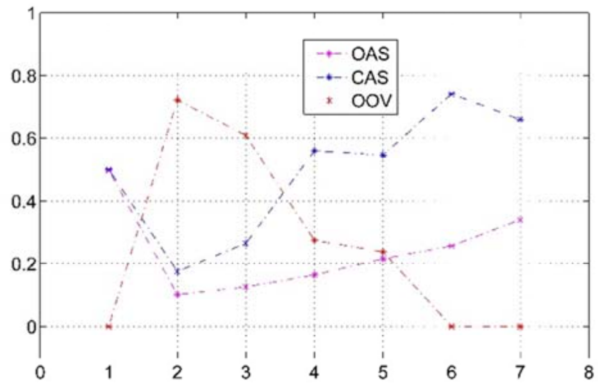


Figure 12. Influence of Lexicon Size on CSA and OOV

As shown in Figure 12, a larger dictionary decreases the OOV rate at the expense of increasing errors caused by OAS and CAS. When the size of the lexicon is large enough, without segmentation disambiguation, errors caused by CAS and OAS can exceed those caused by OOV. OAS is considered as a bottleneck of Chinese word segmentation. The result shows that, if the lexicon is large enough, the influence of CAS is the most critical. In practical applications, an encyclopedic dictionary with large number of lexicon entry is commonly adopted (Chien, 1997; Gao *et al.*, 2002). The result indicates that the influence of CAS is important. For the lexicon-based method, increasing lexicon entry does not always guarantee better performance.

5.4 SM Segmentation

For traditional statistical-based methods, word segmentation is the way to find the segmentation path which has a maximized probability. Conditional Random Fields (CRF) received the state-of-the-art performances (Peng *et al.*, 2004; Tseng *et al.*, 2005). The proposed SM method is effective to identify lexical ambiguities, but weak for segmentation disambiguation. For segmenting a sentence based on SM, instead of finding a maximized segmentation path, the process can be divided into three steps: *OOV detection*, *OAS disambiguation* and *CAS disambiguation*. In the OOV detection step, new words are detected by an employed model[14], which reduces errors caused by the OOV problem. After sentences were segmented (*e.g.*, by lexicon based method), the output can be further processed by OAS disambiguation and CAS disambiguation.

In this part, a preliminary experiment is given to demonstrate this process. The “closed test” is conducted based on the PKU corpus (Emerson, 2005). To make a comparison, a CRF model is implemented[15], which uses 3-Gram features and character features. The result is shown in Table 9, where Column *OAS* is the number of errors caused by OAS. Column *CAS* and Column *OOV* are the same.

Table 9. Performance On Segmentation.

Model	OAS	CAS	OOV	P	R	F1
FMM	752	1,298	5,336	84.34%	90.77%	87.44%
CRF	624	2,887	1,361	92.93%	91.32%	92.11%
SM+OOV	1,166	3,873	796	91.90%	88.84%	90.35%
SM+OOV+OAS	1,084	3,646	806	92.23%	89.37%	90.78%
SM+OOV+OAS+CAS	939	2,559	1,453	92.39%	91.14%	91.76%

In Table 9, the *SM+OOV* implements FMM, which uses words outputted by the CRF model (Row 2) and word extracted from training data. Comparing *SM+OOV* with FMM (Row 1), errors caused by OOV are reduced considerably. However, errors caused by OAS and CAS are increased. In the *SM+OOV+OAS*, another CRF model is trained only on OAS extracted from the training data, then implement the OAS disambiguation. In *SM+OOV+OAS+CAS*, a maximum entropy model is used to disambiguate CAS of *SM+OOV+OAS*. It also trained on

[14] In our experiment, we use CRF (trained on the training data) to segment the testing data, then collect generated new words.

[15] *http://crfpp.googlecode.com/svn/trunk/doc/index.html*

CAS extracted from training data. The result shows the performance is increased by OAS disambiguation and CAS disambiguation. Comparing with the CRF model, both show a similar performance.

Based on SM, the segmentation divides word segmentation into three steps. It provides an alternative way to process word segmentation. Making use of SM, each step can be optimized accordingly. However, the result also shows that the disambiguation of OAS and CAS are not independent. Decreasing one of them can influence the other. From Row 2 to Row 5, the CAS is also a challenging task for segmenting Chinese words.

6. Conclusions

In this paper, a SM method was provided, which represents lexicon information as a matrix. Under the framework of set theory, formal definitions about *segmentation path, combinational ambiguity, overlapping ambiguity,* etc. are given. By mapping string operations into set operation, SM is effective in CSA identification and also available for Chinese word segmentation. In our experiments, several issues about CSA were explored. For researchers who are interested in our work, the source code of our SM is available at *https://github.com/YPench/SMatrix/.*

7. Acknowledges

The research was supported in part by the National Science Foundation of China under Grant Nos. 91118005, 91218301, 91418205.

Reference

Chang, C. & Wei, J. (2008). Identification and disposal of ambiguity based on Omni-Segmentation arithmetic. *Computer Engineering and Applications*, 15.

Chen, K. & Bai, M. (1998).Unknown word detection for Chinese by a corpus-based learning method. *International Journal of Computational Linguistics and Chinese Language Processing, 3*(1), 27-44.

Chen, X., Li, L., & Liang, X. (2012). Eliminate Semantic Network Word Segmentation Ambiguity Method Research. *Microelectronics & Computer, 3,* 045.

Chen, Y., Zheng, Q., & Zhang, W. (2014). Omni-word Feature and Soft Constraint for Chinese Relation Extraction. *The Proceedings of ACL'14*, 572-581.

Chen, Y., Zheng, Q., & Chen, P. (2015a). Feature assembly method for extracting relations in Chinese. *Artificial Intelligence, 228,* 179-194.

Chen, Y., Zheng, Q., & Chen, P. (2015b). A Boundary Assembling Method for Chinese Entity-Mention Recognition. *Intelligent Systems, IEEE, 30*(6), 50-58.

Chien, L. (1997). PAT-tree-based keyword extraction for Chinese information retrieval. *ACM SIGIR Forum, 31*(SI), 50-58.

Emerson, T. (2005). The second international chinese word segmentation bakeoff. *The Proceedings of SIGHAN '05*, 133.

Gao, J., Goodman, J., Li, M., & Lee, K. (2002).Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, *1*(1), 3-33.

Gao, D., & Guo, J. (2009). Dealing with Chinese Overlapping Ambiguity Based on Type Functional Application. *The Proceedings of AICI '09*, *3*, 67-71.

Gao, Y., He, J., & Li, J. (2011). Research on Chinese phonetic string segmentation of sentential input. *The Proceedings of CECNet '11*, 4334-4337.

Gao, J., Li, M., Wu, A., & Huang, C. (2005). Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, *31*(4), 531-574.

Hatori, J., Matsuzaki, T., Miyao, Y., & Tsujii, J. (2012). Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. *The Proceedings of ACL '12*, *1*, 1045-1053.

Huang, C., & Zhao, H. (2007). Chinese Word Segmentation: A Decade Review. *Journal of Chinese Information Processing*, *21*(3), 8-19.

Jiang, W., Mi, H., & Liu, Q. (2008). Word lattice reranking for Chinese word segmentation and part-of-speech tagging. *The Proceedings of COLING '08*, *1*, 385-392.

Li, B. (2011). *Research on Chinese Word Segmentation and proposals for improvement*. (Master thesis). Roskilde University, Denmark.

Li, M., Gao, J., Huang, C., & Li, J. (2003). Unsupervised training for overlapping ambiguity resolution in Chinese word segmentation. *The Proceedings of SIGHAN '03*, *17*, 1-7.

Li, Z., & Sun, M. (2009). Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, *35*(4), 505-512.

Li, Z., Zhang, M., Che, W., Liu, T., *et al*. (2011). Joint models for Chinese POS tagging and dependency parsing. *The Proceedings of EMNLP '11*, 1180-1191.

Liang, N. (1984). Written Chinese word segmentation system-CDWS. *Journal of Beijing Institute of Aeronautics and Astronautics*, *4*, 97-104.

Luo, X., Sun, M., & Tsou, B. (2002).Covering ambiguity resolution in Chinese word segmentation based on contextual information. *The Proceedings of COLING '02*, *1*, 1-7.

Peng, F., Feng, F., & McCallum, A. (2004). Chinese segmentation and new word detection using conditional random fields. *The Proceedings of COLING '04*, 562.

Qiao, W., Sun, M., & Menzel, W. (2008). Statistical properties of overlapping ambiguities in Chinese word segmentation and a strategy for their disambiguation. *Text, Speech and Dialogue*, 177-186.

Sproat, R., & Emerson, T. (2003). The first international Chinese word segmentation bakeoff. *The Proceedings of SIGHAN '03*, *17*, 133-143.

Sun, W. (2011). A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. *The Proceedings of ACL '11, 1,* 1385-1394.

Sun, M., & Benjamin K (1995). Ambiguity Resolution in Chinese Word Segmentation. *The Proceedings of PACLIC '95*.

Sun, T., Liu, Y., Yang, L., *et al*. (2009). An ambiguity discovery algorithm on Chinese word segmentation based dictionary. *The Proceedings of WMWA '09,* 39-42.

Sun, M., & Tsou, B. (2001). A review and evaluation on automatic segmentation of Chinese. *Contemporary Linguistics, 3*(1), 22-32.

Sun, M., & Zou, J. (2001). A critical appraisal of the research on Chinese word segmentation. *Contemporary Linguistics, 1,* 002.

Sun, M., Zuo, Z., & Tsou, B. (1999). The role of high frequent maximal crossing ambiguities in Chinese word segmentation. *Journal of Chinese information processing, 13*(1), 27-37.

Teahan, W., Wen, Y., McNab, R., & Witten, I. (2000). A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, *26*(3), 375-393.

Tseng, H., Chang, P., Andrew, G., *et al*. (2005). A conditional random field word segmenter for sighan bakeoff 2005. *The Proceedings of SIGHAN '05*, 171.

Wand, S., & Wang, B. (2007). A Chinese Overlapping Ambiguity Resolution Method Based on Coupling Degree of Double Characters. *Journal of Chinese Information Processing, 5,* 004.

Wang, X., & Du, L. (2004). A Method of Sentence Segmentation That Check All Overlapping Ambiguity. *Acta Electronica Sinica, 32*(1), 50-54.

Wang, L., Song, S., Feng, J., & Chen, P. (2009). Chinese Segmentation System Combining Omni-Segmentation with Statistic. *Microelectronics & Computer, 5*, 019.

Wang, K., Zong, C., & Su, K. (2012). Integrating generative and discriminative character-based models for Chinese word segmentation. *ACM Transactions on Asian Language Information Processing, 11*(2), 7.

Wang, Z., Zong, C., & Xue, N. (2013). A Lattice-based Framework for Joint Chinese Word Segmentation, POS Tagging and Parsing. *The Proceedings of ACL '13,* 2013.

Wu, A., & Jiang, Z. (1998). Word segmentation in sentence analysis. *The Proceedings of ICCIP '98*, 169-180.

Wu, X., Zhang, M., & Lin, X. (2010). Parsing-based Chinese word segmentation integrating morphological and syntactic information. *The Proceedings of NLP-KE '11,* 114-121.

Xue, N. (2003). Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing, 8*(1), 29-48.

Xue, N., & Yang, Y. (2011). Chinese sentence segmentation as comma classification. *The Proceedings of ACL '11, 2*, 631-635.

Yao, H., Wang, Y., & Huang, J. (2012). An algorithm of solving Chinese segmentation overlapping ambiguous. *The Proceedings of CSAE '12*, *2*, 464-467.

Zeng, D., Wei, D., Chau, M., & Wang, F. (2011). Domain-specific Chinese word segmentation using suffix tree and mutual information. *Information Systems Frontiers*, *13*(1), 115-125.

Zeng, X., Wong, D., Chao, L., & Trancoso, I. (2013). Graph-based Semi-Supervised Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging. *The Proceeding of ACL '13*, 770-779.

Zhang, L. (2004). *Maximum entropy modeling toolkit for Python and C++*. Natural Language Processing Lab, Northeastern University, China.

Zhang, P., & Li, C. (2006). A new Context-Sensitive ambiguous phrase segmentation Algorithm. *Computer Systems & Applications*, *5*, 013.

Zhang, L., Li, L., He, Z., *et al.* (2013). Improving Chinese Word Segmentation on Micro-blog Using Rich Punctuations. *The Proceeding of ACL '13*.

Zhang, H., & Liu, Q. (2002). Model of Chinese Words Rough Segmentation Based on N-Shortest-Paths Method. *Journal of Chinese information processing*, *5*, 000.

Zhang, H., Yu, H., Xiong, D., & Liu, Q. (2003). HHMM-based Chinese lexical analyzer ICTCLAS. *The Proceedings of SIGHAN '03*, *17*, 184-187.

Zhao, H., Huang, C., Li, M., & Lu, B. (2010). A unified character-based tagging framework for Chinese word segmentation. *ACM Transactions on Asian Language Information Processing*, *9*(2), 5.

Computational Linguistics and Chinese Language Processing
Vol. 21, No. 1, June 2016, pp. 29-50

Linguistic Template Extraction for Recognizing Reader-Emotion

Yung-Chun Chang[*,+], Chun-Han Chu[*] Chien Chin Chen[+], and Wen-Lian Hsu[*]

Abstract

Previous studies on emotion classification mainly focus on the emotional state of the writer. By contrast, our research emphasizes emotion detection from the readers' perspective. The classification of documents into reader-emotion categories can be applied in several ways, and one of the applications is to retain only the documents that trigger desired emotions to enable users to retrieve documents that contain relevant contents and at the same time instill proper emotions. However, current information retrieval (IR) systems lack the ability to discern emotions within texts, and the detection of reader's emotion has yet to achieve a comparable performance. Moreover, previous machine learning-based approaches generally use statistical models that are not in a human-readable form. Thereby, it is difficult to pinpoint the reason for recognition failures and understand the types of emotions that the articles inspired on their readers. In this paper, we propose a flexible emotion template-based approach (TBA) for reader-emotion detection that simulates such process in a human perceptive manner. TBA is a highly automated process that incorporates various knowledge sources to learn an emotion template from raw text that characterize an emotion and are comprehensible for humans. Generated templates are adopted to predict reader's emotion through an alignment-based matching algorithm that allows an emotion template to be partially matched through a statistical scoring scheme. Experimental results demonstrate that our approach can effectively detect reader's emotions by exploiting the syntactic structures and semantic associations in the context, while

[*] Institute of Information Science, Academia Sinica, Taipei, Taiwan
E-mail: {changyc, johannchu, hsu}@iis.sinica.edu.tw
[+] Department of Information Management, National Taiwan University, Taipei, Taiwan
E-mail: patonchen@ntu.edu.tw

outperforming currently well-known statistical text classification methods and the stat-of-the-art reader-emotion detection method.

Keywords: Reader-Emotion Detection, Emotion Template, Template-based Approach, Text Classification, Sentiment Analysis.

1. Introduction

With the rapid growth of the Internet, the web has become a powerful medium for disseminating information. People can easily share information of daily experiences and their opinion anytime and anywhere on the social media, such as blogs, Twitter and Facebook. Therefore, sentiment analysis studies have gained increasing interest in recent years with academia and business corporations attempting to analyse and predict public trends by mining opinions that are subjective statements and reflect people's sentiments or perceptions about topics (Pang *et al.*, 2002). Moreover, human feelings can be quickly collected through emotion detection (Quan & Ren, 2009; Das & Bandyopadhyay, 2009). While previous researches on emotions mainly focused on detecting the emotions that the authors of the documents were expressing, it is worthy of noting that the reader emotions, in some aspects, differ from that of the authors and may be even more complex (Lin *et al.*, 2008; Tang & Chen, 2012). Regarding a news article for instance, while a journalist objectively reports up-going oil price and does not express his or her emotion in the text, a reader could yield angry or negative emotions. On the other hand, an infamous politician's blog entry describing his miserable day may not cause the opposing readers to feel the same way. While the author of an article may directly express his/her emotions through sentiment words within the text, a reader's emotion possesses a more complex nature, as even general words can evoke different types of reader's emotions depending on the reader's personal experience and knowledge (Lin *et al.*, 2007).

Instead of detecting writer's emotion, which has been already investigated extensively in previous studies (Zhang & Liu, 2010; Mukherjee & Liu, 2010; Si *et al.*, 2013), this paper aims to uncover the emotions of readers triggered by the document. Such research holds great potential for novel applications. For instance, an enterprise that possesses the business intelligence that is capable of identifying the emotional effect that a document inflicts on its readers can provide services to retain only the documents that evokes the desired emotions, enabling users to retrieve documents with relevant contents and meanwhile being instilled the proper emotions. As a result, users benefit by obtaining opportunities and advantages in the competitive market through a more efficient and quick manner. However, current information retrieval systems lack the ability to discern emotion within texts, and reader-emotion detection has yet to achieve comparable performance (Lin *et al.*, 2007). Machine learning-based approaches are widely used for sentiment analysis and emotion detection related researches. These approaches can usually generate accurate classifiers that assign a category label for each document with much lower labour cost. Nevertheless, the statistical models used by these classifiers are generally not in a human-readable form. Thus, it is difficult to pinpoint the reason for recognition failures and understand what exact emotion of the reader is triggered.

In light of this rationale, we proposed a flexible template-based approach (TBA) for reader-emotion detection that simulates such process in human perception. TBA is a highly automated process that integrates various types of knowledge to generate discriminative linguistic patterns for document representation. These patterns can be acknowledged as the essential knowledge for humans to understand different kinds of emotions. Furthermore, TBA recognizes reader's emotions of documents using an alignment-based algorithm that allows an emotion template to be partially matched through a statistical scoring scheme. Our experiments demonstrate that TBA can achieve a higher performance than other well-known text categorization methods and the state-of-the-art reader-emotion detection method.

The remainder of this paper is organized as follows. The next section contains a review of related works on reader-emotion detection approaches. We introduce the proposed emotion template-based approach for reader-emotion detection in Section 3, and its evaluation is described in Section 4. Finally, we provide some concluding remarks and potential future avenues of research in Section 5.

2. Related Work

Textual articles are one of the most common ways for persons to convey their feelings. Identifying essential factors that affect emotion transition is important for human language understanding. With the rapid growth of computer-mediated communication applications, such as social websites and micro-blogs, the research on emotion classification has been attracting more and more attention recently from enterprises toward business intelligence (Chen *et al*., 2010; Purver & Battersby, 2012). In general, a single text may possess two types of emotions: writer-emotion and reader- emotion. The research of writer-emotion investigates the emotion expressed by the writer when writing the text. Pang *et al*. (2002) designed an algorithm to classify movie reviews into positive and negative emotions. Mishne (2005), and Yang and Chen (2006) used emoticons as tags to train SVM (Cortes & Vapnik, 1995) classifiers at the document or sentence level, respectively. In their studies, emoticons were taken as mood or emotion tags, and textual keywords were considered as features. Wu *et al*. (2006) proposed a sentence level emotion recognition method using dialogs as their corpus, in which "Happy", "Unhappy", or "Neutral" was assigned to each sentence as its emotion category. Yang *et al*. (2006) adopted Thayer's model (1989) to classify music emotions. Each music segment can be classified into four classes of moods. As for sentiment analysis research, Read (2005) used emoticons in newsgroup articles to extract relevant instances for training polarity classifiers.

Nevertheless, the research of reader-emotion concerns the emotions expressed by a reader after reading the text. As the writer and readers may view the same text from different perspectives, they do not always share the same emotion. Since the recent increase in the popularity of Internet, certain news websites, such as Yahoo! Kimo News, incorporate the Web 2.0 technologies that allow readers to express their emotions toward news articles. Classifying emotions from the readers' point of view is a challenging task, and research on this topic is relatively sparse as compared to those considering the writers' perspective. While writer-emotion classification has been extensively studied, there are only a few studies on reader-emotion classification. Lin *et al*. (2007) first described the task of reader-emotion classification on news articles and classified Yahoo! News articles into 8 emotion classes (e.g. happy, angry, or depressing) from the readers' perspectives. They combined bigrams, words, metadata and word emotion categories to train a classifier for determining the reader-emotions toward news. Yang *et al*. (2009) automatically annotated reader-emotions on a writer-emotion corpus with a reader-emotion classifier, and studied the interactions between writers and readers with the writer-reader-emotion corpus.

Our approach differs from existing reader-emotion detection approaches in a number of aspects. First, we proposed an emotion template-based approach that mimics the perceptual behaviour of humans in understanding. Second, the generated emotion templates are human readable, and can be represented as the domain knowledge required for detecting reader-emotion. Therefore, it is helpful in elucidating how articles trigger certain types of emotions in their readers in a more comprehensive manner. Finally, in addition to syntactic features, TBA further considers the surrounding context and semantic associations to efficiently recognize reader-emotions.

3. Learning Reader-emotion Template from Raw Text

In this paper, we present a template-based approach (TBA) for detecting the reader-emotion of documents. We model reader-emotion detection as a classification problem, and define the reader-emotion detection task as the following. Let $W= \{w_1, w_2, \ldots,w_k\}$ be a set of words, $D= \{d_1, d_2, \ldots, d_m\}$ be a set of documents, and $E= \{e_1, e_2, \ldots , e_n\}$ be a set of reader-emotions. Each document d is a set of words such that $d \subseteq W$. The goal of this task is to decide the most appropriate reader-emotion e_i for a document d_j, although one or more emotions can be associated with a document. Our proposed method is different in that we take advantage of multiple knowledge sources, and implement an algorithm to automatically generate templates that represent discriminative patterns in documents. Our system, using the proposed method, mainly consists of three components: Crucial Element Labelling (CEL), Emotion Template Generation (ETG), and Emotion Template Matching (ETM), as shown in Figure 1. The CEL first uses prior knowledge to mark the semantic classes of words in the corpus. Then the ETG collects frequently co-occurring elements, and generates templates for each emotion. These templates are stored in the emotion-dependent knowledge base to provide domain-specific knowledge for our emotion detection. During the detection process, an article is first labelled by the CEL as well. Subsequently, the ETM applies an alignment-based algorithm that utilizes our knowledge base to calculate the similarity between each emotion and the article to determine the main emotion of this article. Details of these components will be disclosed in the following sections.

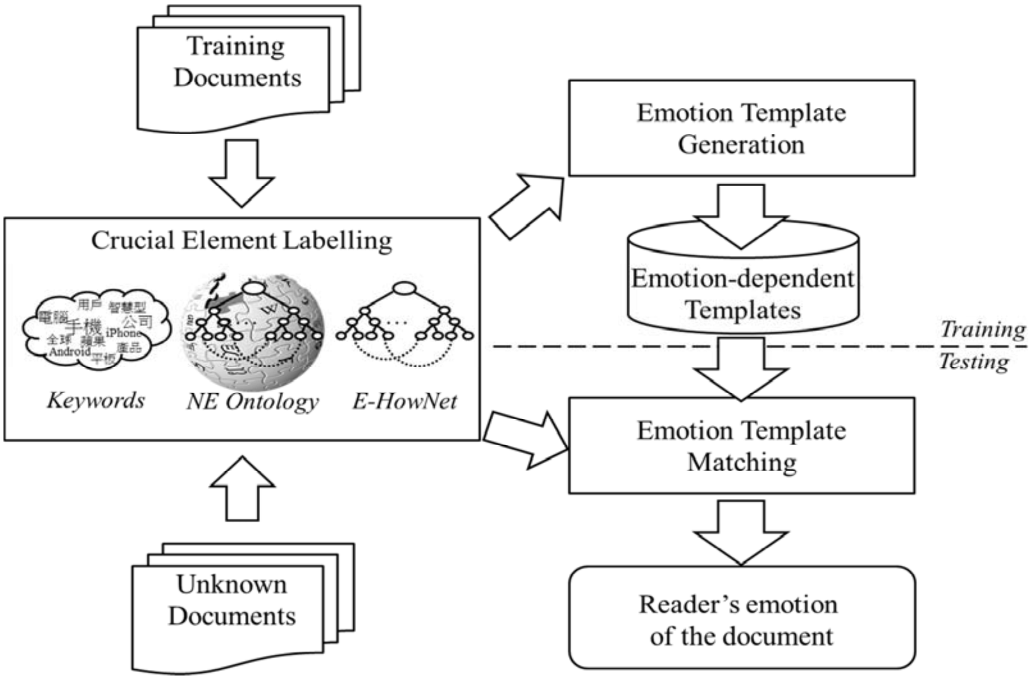

Training Documents
Emotion Template Generation
Crucial Element Labelling
Emotion-dependent Templates
Training
Testing
Keywords
NE Ontology
E-HowNet
Emotion Template Matching
Unknown Documents
Reader's emotion of the document

Figure 1. Architecture of our system.

3.1 Crucial Element Labelling (CEL)

TBA attempts to simulate the human perception of an emotion through the recognition of crucial elements. In this work, we capture crucial elements within documents by adopting a three-layer labelling approach that utilizes various knowledge sources, such as lexical dictionaries and Wikipedia, to induce template elements. First of all, since keywords within a reader-emotion are often considered as important information, we used the log likelihood ratio (LLR) (Manning & Schütze, 1999), an effective feature selection method, to learn a set of reader-emotion specific keywords. Given a training dataset, LLR employs Equation (1) to calculate the likelihood of the assumption that the occurrence of a word w in reader-emotion E is not random. In (1), e_i denotes the specific reader's emotion in the training dataset; $N(e_i)$ and $N(\neg e_i)$ are the numbers of on-emotion and off-emotion documents, respectively. $N(w$^$e_i)$, denoted as *k*, is the number of on-emotion documents containing *w*; the number of off-emotion documents containing *w*, $N(w^\wedge\neg e_i)$, is denoted as *l*. Altogether, the formula expresses the ratio of two likelihood functions. To simplify the formula, we also define *m* as the number of on-emotion documents with no word *w* (that is, $m = N(e_i) - k$), and n as that of off-emotion documents ($n = N(\neg e_i) - l$). The probabilities $p(w)$, $p(w|e_i)$, and $p(w|^\wedge e_i)$ are estimated using maximum likelihood estimation. A word with a large LLR value is closely associated with the reader-emotion. We rank the words in the training dataset based on their LLR values and select the top 200 to compile an emotion keyword list.

$$LLR(w,e_i) = 2\log\left[\frac{p(w|e_i)^k(1-p(w|e_i))^m p(w|\neg e_i)^l(1-p(w|\neg e_i))^n}{p(w)^{k+l}(1-p(w))^{m+n}}\right] \quad (1)$$

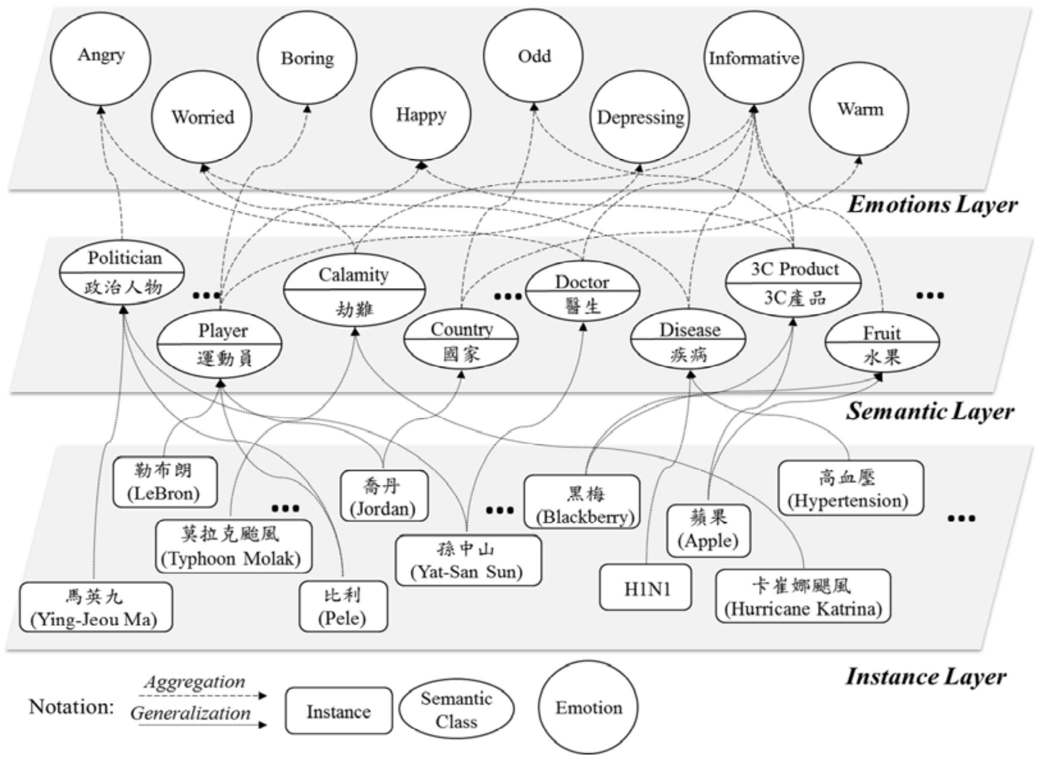


Figure 2. Architecture of named entity ontology.

Next, we aim to recognize named entities (NEs) from text to facilitate document comprehension and improve the performance of identifying topics (Bashaddadh & Mohd, 2011). Our labelling algorithm uses a string matching scheme to single out the keywords (if they exist in the sequences); therefore, segmentation is not required in the preprocess. However, exact-matching NEs may overlook many template elements since it omits semantic context. To remedy this problem, this paper adopts a novel structure to construct the NE ontology (NEO) for labelling crucial elements based on the levels of organization mentioned in (Lee *et al*. 2005) and (Wang *et al*. 2010). Figure 2 depicts the architecture of the NE ontology, which includes an emotion layer, a semantic layer, and an instance layer. There are eight types of emotions in the emotion layer, namely "Angry", "Worried", "Boring", "Happy", "Odd", "Depressing", "Informative" and "Warm". Moreover, each semantic class in the semantic layer denotes a general semantic meaning of named entities that can be aggregated from many emotions, including "政治人物 (Politician)", "疾病 (Disease)" and others. The instance layer represents 6323 named entities extracted from documents across eight emotions by the Stanford NER. In order to minimize the labour cost of instance generalization, we utilize Wikipedia to semi-automatically label NEs with their semantic classes as a way of generalization. Only NE labels for persons, places and organizations are taken into consideration, and Wikipedia's category tags are used to label NEs recognized by the Stanford NER[1]. We select the category tag to which the most *topic paths* are associated as the main semantic label for NEs in documents. A topic path is the classification hierarchy of a certain category; it can be considered as the traversal from general categories to more specific ones. A category name with more associated topic paths is considered to be more suitable to represent a NE for its appropriate scope of semantic coverage. For example, a query "歐巴馬 (Obama)" to the Wikipedia would return a page titled "巴拉克・歐巴馬 (Barack Obama)". Within this page, there are a number of category tags such as "民主黨 (Democratic Party)" and "美國總統 (Presidents of the United States)". These two category tags contain three and seven topic paths, respectively. Suppose "美國總統(Presidents of the United States)" is the one with more topic paths, our system will label "巴拉克・歐巴馬 (Barack Obama)" with the tag "[美國總統 (Presidents of the United States)]". Domain experts further annotated each named entity by their corresponding semantic classes for the purpose of generalization if the NE term not included in Wikipedia to its category tags. Each instance in the instance layer can connect to multiple semantic classes according to the generalized relations. For example, named entity "喬丹 (Jordan)" can be generalized to "國家 (country)" and "人名 (people)". In this manner, we can transform plain NEs to a more general class and increase the coverage of each label.

Finally, to incorporate even richer semantic context into our semantic template, we use the Extended HowNet (Chen *et al*., 2005), which is an extension of the HowNet (Dong *et al*. 2010). Extended HowNet contains a structured representation of knowledge and semantics. It connects approximately 90000 words in the CKIP Chinese Lexical Knowledge Base and HowNet, and includes additional highly frequent words that are specific in Traditional Chinese. It also contains a different formulation of each word to better fit its semantic

[1] http://nlp.stanford.edu/software/CRF-NER.shtml

representation, as well as the distinct definition of function and content words. A total of four basic semantic classes were applied, namely object, act, attribute, and value. Moreover, in comparison to HowNet, EHowNet possesses a layered definition scheme and complex relationship formulation, and uses simpler concepts to replace schemes as the basic element when defining a more complex concept or relationship. To illustrate the content of the EHowNet, let us take the definition of "血癌 (leukemia)" in EHowNet in Definition 1 as an example. In the most compact sense, leukemia is a type of cancer that originates from disorder of blood (cells); therefore, EHowNet presents the phenomenon and its occurring position for the term in the *Simple Definitio*n. To further detail its meaning, EHowNet goes an extra mile and explains both "cancer" and "blood" in the *Expanded Definition*. We can see that the EHowNet not only contains semantic representation of a word, but also its relations to other words or entities. This enables us to combine or dissect the meaning of words by using its semantic components. Following the method in (Shihet *al*. 2012), we extracted the main definition of each word as the semantic class label.

Definition 1:

Simple Definition:

{癌症|cancer:position={血液|blood}}

Expanded Definition:

{disease|疾病:position={BodyFluid|體液:telic={transport|送:patient={gas|氣:predication={respire|呼吸:patient={~}}},instrument={~}}},qualification={serious|嚴重}}

We exploit the taxonomies in EHowNet, which include lexical categories, synonyms, and semantic relations between different words or sets of words. With the resources stated, the CEL can transform words in the original documents into their corresponding semantic labels. Our research assigns clause as the unit for semantic labelling. To illustrate the process of CEL, consider the clause C_n = "希拉蕊順利代表民主黨贏得美國總統選舉 (Hillary Clinton represents the Democratic Party and won the Presidential election in the U.S.)", as shown in Figure 3. First, "希拉蕊(Hillary Clinton)" is found in the keyword dictionary and tagged. Then NEs like "民主黨 (Democratic Party)" and "總統選舉 (Presidential elections)" are recognized and tagged as "{政黨 (Party)}" and "{總統選舉 (Presidential elections)}". Subsequently, other terms such as "代表 (represent)" and "贏得 (won)" are labelled with their corresponding E-HowNet senses if they exist. Finally we can obtain the sequence: "[希拉蕊]:{代表}:{政黨}:{得到}:{國家}:{總統選舉} ([Hillary Clinton]:{represent}:{party}:{got}:{country}:{Presidential elections})", where domain keywords (represented by square-bracketed slots) are matched verbatim and semantic classes (represented by curly-bracketed slots) match all words that belong to their specific class. Since different annotation knowledge bases are processed through in a specific-to-general order, the proportion of overlapping annotations are relatively low. In the occasion when there is an overlap between the matched keywords, the longest keyword will be reserved. The labelling process not only effectively prevents errors caused by Chinese word segmentation, but also groups the synonyms altogether by using semantic labels. This enables us to generate distinctive and prominent semantic templates in the next stage.

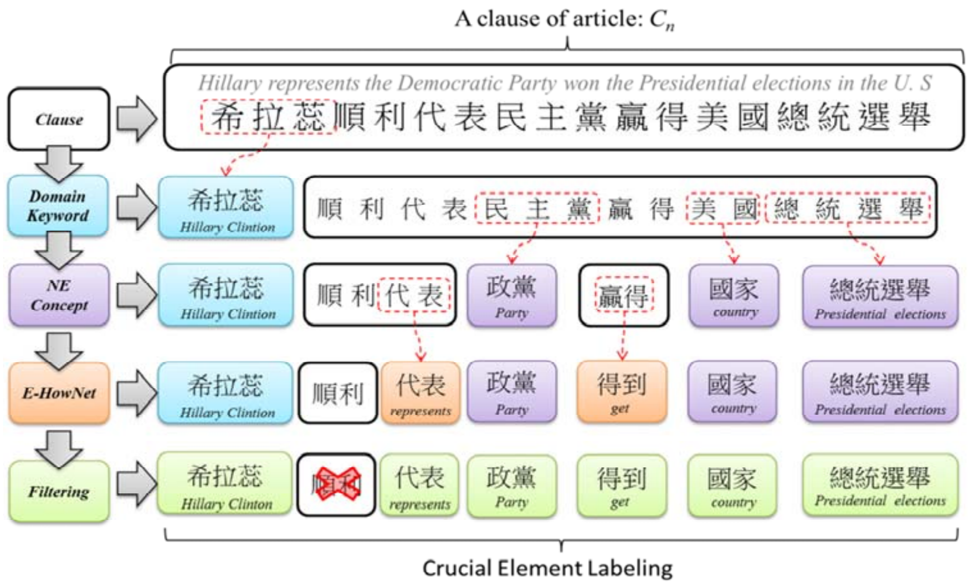

A clause of article: C_n
Hillary represents the Democratic Party won the Presidential elections in the U. S
希拉蕊順利代表民主黨贏得美國總統選舉
Clause
Domain Keyword
NE Concept
E-HowNet
Filtering
希拉蕊 Hillary Clintion
順 利 代 表 民 主 黨 贏 得 美 國 總 統 選 舉
順 利 代 表
政黨 Party
贏得
國家 country
總統選舉 Presidential elections
順利
代表 represents
得到 get
Crucial Element Labeling

Figure 3. Crucial element labelling process.

3.2 Emotion Template Generation

In our framework, a reader's emotion is represented by a set of semantic templates, which are sequences of crucial elements consisting of crucial elements and keywords. The emotion template generation (ETG) process aims at automatically generating N representative templates from sequences of crucial elements in the documents. These representative (or dominating) templates can be used as background knowledge for each reader's emotion when recognizing documents. More importantly the representative templates can be easily understood by humans. To illustrate, consider the emotion "Happy" and one of the automatically generated semantic templates, "{運動員 $_{\text{player}}$}:{得到 $_{\text{get}}$}:[冠軍 $_{\text{championship}}$]." We can think of various semantically similar sentences that are covered by this semantic template, e.g., "柯里帶領勇士贏得了 NBA 總冠軍賽 (Stephen Curry led the Warriors to win the NBA championship)" or "費德勒擊敗安迪·穆雷獲得溫普敦冠軍 (Roger Federer defeated Andy

Murray and won the Wimbledon championship)". Likewise, a similar template "{運動員 $_{player}$}:{得到 $_{get}$}:[分數 $_{score}$]" is capable of representing sentences like "布萊恩在最終賽贏得六十分 (Kobe Bryant won sixty points at his final game.)". This sort of human-interpretable knowledge cannot be easily obtained by ordinary machine learning models.

The ETG process is described as follows. We formulate reader-emotion template generation as a frequent pattern mining problem. Based on the co-occurrence of crucial elements, we can construct a *crucial element graph* (*CE graph*) to describe the strength of relations between them. Since crucial elements are of an ordered nature, the graph is directed and can be made with association rules. In order to avoid the generation of templates with insufficient length, we empirically set the minimum support of a crucial element as 100 and minimum confidence as 0.5 in our association rules. This is because we observed that the rank-frequency distribution of semantic classes followed Zipf's law (Manning & Schütze, 1999), so does the normalized frequency of semantic templates. Crucial elements of low frequency usually identify semantic that are irrelevant to the emotion. Hence, for each reader's emotion, we selected the first frequent crucial elements that accumulated frequencies reached 80% of the total crucial element frequency count in the reader-emotion documents. Thus, an association rule can be represented as (2):

$$\text{confidence}(CE_i \Rightarrow CE_j) = P(CE_j|CE_i) = \frac{\text{support}\,(CE_i \cup CE_j)}{\text{support}\,(CE_i)}, \tag{2}$$

where $support_{min}$=100, $confidence_{min}$=0.5.

Figure 4 is the illustration of a CE graph. In this graph, vertices (CE_x) represent crucial elements, and edges represent the co-occurrence of two elements, CE_i and CE_j, where CE_i precedes CE_j. The number on the edge denotes the confidence of two connecting vertices.

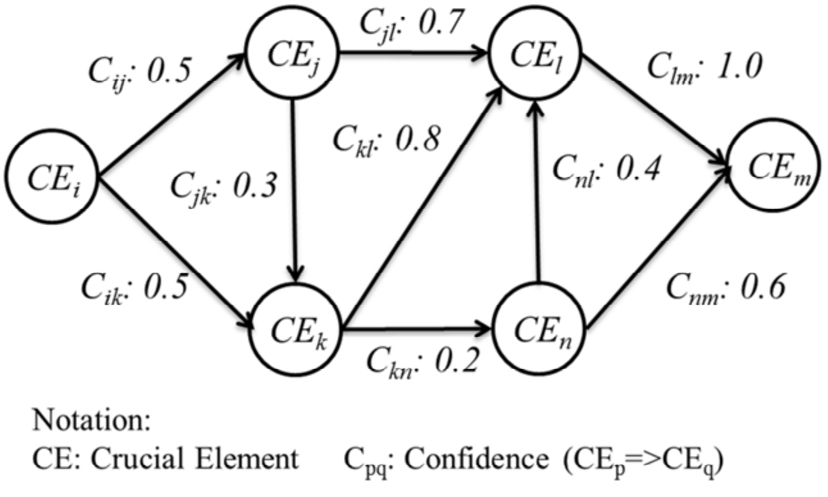


Figure 4. A CE graph for template generation.

After constructing all CE graphs, we then generate emotion templates by applying the random walk theory (Lovász, 1993) in search of high-frequency and representative elements for each reader's emotion. Let a CE graph G be defined as $G=(V,E)$ ($|V|=p$, $|E|=k$), a random walk process consists of a series of random selections on the graph. Every edge (CE_n, CE_m) has its own weight M_{nm}, which denotes the probability of a crucial element CE_n, followed by another element CE_m. For each element, the sum of weight to all neighboring elements $N(CE_n)$ is defined as (3), and the whole graph's probability matrix is defined as (4). As a result, a series of a random walk process becomes a Markov Chain. According to (Li *et al.*, 2010), the cover time of a random walk process on a normal graph is $\forall CE_n, C_{CE_n} \le 4k^2$. We can conclude that using random walk to find frequent patterns on CE graphs would help us capture even the low probability combinations and shorten the processing time.

$$\forall CE_n \sum_{m\in N(CE_n)} M_{nm} = 1 \tag{3}$$

$$Pr = \left[X_{t+1} = CE_m \middle| \begin{array}{l} X_t = CE_n \\ X_{t-1} = CE_k, \\ \cdots \\ X_0 = CE_i \end{array} \right] = Pr[X_{t+1} = CE_m \mid X_t = CE_n] = M_{nm} \tag{4}$$

Although the random walk process can help us generate templates from frequent patterns in CE graphs, it can also create some redundancy. Hence, a merging procedure is required to eliminate the redundant results by retaining only the templates with the longest length and highest coverage, and dispose of those that are completely covered by another template. For example, the template [歐巴馬]:{政黨}:{總統選舉} is completely covered by template [歐巴馬]:{代表}:{政黨}:{得到}:{國家}:{總統選舉}. Thus, the former template is removed. Moreover, the reduction of the crucial element space provided by template selection is critical. It allows the execution of more sophisticated text classification algorithms, which lead to improved results. Those algorithms cannot be executed on the original crucial element space because their execution time would be excessively high, making them impractical (Ricardo & Berthier, 2011). Therefore, selecting crucial elements closely associated with an emotion would improve the performance of reader-emotion detection. We use the log likelihood ratio (LLR) again to differentiate crucial elements for each emotion. Given a training dataset comprised of different reader's emotions, the LLR calculates the likelihood of the occurrence of a crucial element in the emotion. A crucial element with a large LLR value is thought to be closely associated with the emotion. Lastly, we rank the crucial elements in the training dataset based on a sum of crucial elements LLR values and retain the top 100 for this reader's emotion.

3.3 Emotion Template Matching

We believe the human perception of an emotion is obtained through recognizing important events or semantic contents to rapidly evoke their emotional response. For instance, when an article contains strongly correlated words such as “Japan (country)”, “Earthquake (disaster)” and “Tsunami (disaster)” simultaneously, it is natural to conclude that the article has a much higher chance of eliciting emotions like *depressed* and *worried* rather than *happy* and *warm*. TBA uses an alignment algorithm to measure the similarity of templates, since alignment enables a single template to match multiple semantically similar expressions with appropriate scores. During matching, a document is first labelled with crucial elements. Afterwards, an alignment-based algorithm (Needleman & Wunsch, 1970) is applied to determine to what degree a semantic template fits in a document. For each clause within a given document d_j, we first label crucial elements $cs = \{ce_1, \ldots, ce_n\}$, followed by the matching procedure that compares all sequences of crucial element in d_j to all emotion templates $ET = \{et_1, \ldots, et_j\}$ in each emotion, and calculates the sum of scores for each emotion. The emotion e_i with the highest sum of scores defined in (5) is considered as the winner.

$$Emotion = \arg\max_{e_i \in E} \sum_{et_n \in ET_{c_i}, cs_m \in CS_{d_j}} \Delta(et_n, cs_m) = \sum_k \sum_l \Delta(et_n \cdot st_k, cs_m \cdot ce_l) \tag{5}$$

st_k and ce_l represent the k^{th} slot of et_n and l^{th} crucial element of cs_m, respectively. Scoring of the matched and unmatched components in semantic templates is as follows: If $et_n \cdot st_k$ and $cs_m \cdot ce_l$ are identical, we add a matched score (*MS*) obtained from the LLR value of ce_l if it matches a keyword. Otherwise, the score is determined by multiplying the frequency of the crucial element in emotion e_i by a normalizing factor $\lambda = 100$ as in (6). On the contrary, if an element is not matched, the score of *insertion* or *deletion* is calculated. An insertion (*IS*), defined as a label that is present in the article but not in the template, can be accounted for by the inversed entropy of this crucial element (7), which can be thought of as the uniqueness or generality of this label. On the other hand, a deletion (*DS*), representing the label that exists only in the template and not in the article, is computed from the log frequency of this element as (8). Both types give negative scores to the sum. The matching algorithm is then applied to determine the reader-emotion of the article by comparing the sequence of crucial elements $C = \{c_1, c_2, \ldots, c_n\}$ in each clause of an article to every template $F = \{C_1, C_2, \ldots, C_m\}$ in each emotion. An illustration of the matching process of a sequence of semantic classes to an emotion template is shown in Figure 5. A match between the two sequences is given a positive score obtained from the LLR score of the semantic class in an emotion. Finally, the sum of scores of each emotion was computed, and the emotion with the highest score is considered as the winner. Through this method, each individual crucial element label is given a different weight according to its characteristics. Thus, the order of these labels is not the only determining factor in matching. The detailed algorithm is described in Algorithm 1.

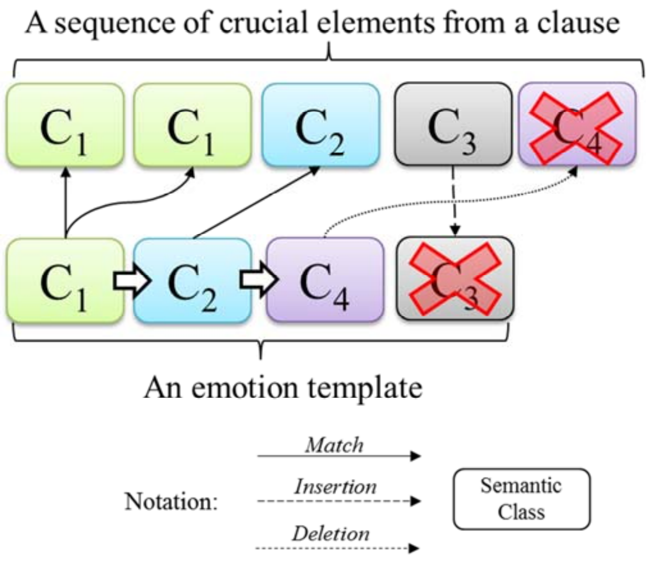


Figure 5. Illustration of an emotion template matching process.

Algorithm 1: Emotion Template Matching

Input: A emotion template *et* = {st_1, …,st_m}, *st*: slots; A sequence of crucial elements from a clause *cs* = {ce_1, …, ce_n}

Output: Matching score σ between *et* and *cs*

```
BEGIN
1: pos ← 0; σ ← 0;
2: FOR i = 1 to m DO
3: isMatched ← false;
4: pos ← current matched position in CE;
5:      IF found $ce_j$ EQUAL TO $st_i$ after pos THEN
6:           σ ← σ + MatchedScore($ce_j$);
7:           isMatched ← true;
8:      END IF
9:      IF isMatched != true THEN
10:          σ ← σ- DeletionScore ($st_i$);
11:          σ ← σ- InsertionScore ($ce_j$);
12:     END IF
13: END FOR
14: Output σ
END
```

$$MS(ce_l) = \begin{cases} LLR_{ce_l}, \text{if it matches a keyword} \\ \lambda \frac{f_{ce_l}}{\sum_{i=1}^{m} f_{ce_i}}, \text{ otherwise} \end{cases} \tag{6}$$

$$IS(ce_l) = \frac{1}{-\sum_{i=1}^{m} P(ce_{l_{e_i}}) \cdot \log_2(P(ce_{l_{e_i}}))} \tag{7}$$

$$DS(ce_l) = \log \frac{f_{ce_l}}{\sum_{i=1}^{m} f_{ce_i}} \tag{8}$$

4. Experiment

4.1 Experiment Setup

To the best of our knowledge, there is no publicly available corpus for reader-emotion detection. Therefore, we compiled a data corpus for performance evaluation, as shown in Table 1. The data corpus contains news articles spanning a period from 2012 to 2014 collected from Yahoo News[2]. It is an independent common resource for performance evaluation among reader-emotion research (e.g. Lin *et al.* (2007)), since it has a special feature which allows a reader of a news article to select from eight emotions the one that represent how the reader feels after reading a news article, i.e., “Angry”, “Worried”, “Boring”, “Happy”, “Odd”, “Depressing”, “Warm”, and “Informative”. To ensure the quality of the corpus, only articles with a clear statistical distinction between the highest vote of emotion and others determined by t-test with a 95% confidence level were retained. Finally, a total of 47,285 articles were retained from the original 68,026 articles, and they were divided into the training set consisting of 11,681 articles and the testing set consisting of 35,604 articles, respectively.

Table 1. The distribution of data corpus.

	Angry	*Worried*	*Boring*	*Happy*	*Odd*	*Depressing*	*Warm*	*Informative*
#Training	2,001	261	1,473	2,001	1,536	1,573	835	2,001
#Test	4,326	261	1,473	7,334	1,526	1,573	835	18,266
#Total	6,327	522	2,946	9,345	3,062	3,146	1,670	20,267

[2] https://tw.news.yahoo.com/

A comprehensive performance evaluation of the TBA with other methods is provided. The first is an emotion keyword-based model which is trained by SVM to demonstrate the effect of our keyword extraction approach (denoted as *KW-SVM*). Another is a probabilistic graphical model which uses the LDA model as document representation to train an SVM to classify the documents as either emotion relevant or irrelevant (Blei *et al*., 2003) (denoted as *LDA-SVM*). The last one is a state-of-the-art reader-emotion recognition method which combines various feature sets including bigrams, words, metadata, and emotion category words (Lin *et al*., 2007) (denoted as *CF-SVM*). To serve as a standard for comparison, we also included the results of Naive Bayes (McCallum & Nigam, 1998) as a baseline (denoted as *NB*). Details of the implementations of these methods are as follows. We employed CKIP[3] for Chinese word segmentation. The dictionary required by Naïve Bayes and LDA-SVM is constructed by removing stop words according to a Chinese stop word list provided by Zou *et al*. (2006), and retaining tokens that make up 90% of the accumulated frequency. In other words, the dictionary can cover up to 90% of the tokens in the corpus. As for unseen events, we use Laplace smoothing in Naïve Bayes, and an LDA toolkit[4] is used to perform the detection of LDA-SVM. Regarding the CF-SVM, the words outputted by the segmentation tool were used. The information related to news reporter, news category, location of the news event, time (hour of publication) and news agency were used as the metadata features. The extracted emotion keywords were used as the emotion category words, since the emotion categories of Yahoo! Kimo Blog was not provided. To evaluate the effectiveness of these systems, we adopted the accuracy measures used by Lin *et al*. (2007). We used macro-average and micro-average to compute the average performance. These measures are defined based on a contingency table of predictions for a target emotion E_k. The accuracy $A(E_k)$, macro-average A^M, and micro-average A^μ are defined as follows:

$$A(E_k) = \frac{TP(E_k) + TN(E_k)}{TP(E_k) + FP(E_k) + TN(E_k) + FN(E_k)} \tag{9}$$

$$A^M = \frac{1}{m} \sum_{k=1}^{m} A(E_k) \tag{10}$$

$$A^\mu = \frac{\sum_{k=1}^{m} TP(E_k) + TN(E_k)}{\sum_{k=1}^{m} (TP(E_k) + FP(E_k) + TN(E_k) + FN(E_k))} \tag{11}$$

where $TP(E_k)$ is the set of test documents correctly classified to the emotion E_k, $FP(E_k)$ is the set of test documents incorrectly classified to the emotion, $FN(E_k)$ is the set of test documents wrongly rejected, and $TN(E_k)$ is the set of test documents correctly rejected.

---

[3] http://ckipsvr.iis.sinica.edu.tw/

[4] http://nlp.stanford.edu/software/tmt/tmt-0.4/

4.2 Results and Discussion

The performances of emotion detection systems are listed in Table 2. As a baseline, the Naïve Bayes classifier is a keyword statistics-based system which can only accomplish a mediocre performance. Since it only considers surface word weightings, it is difficult to represent inter-word relations. The overall accuracy of the Naïve Bayes classifier is 36.84%, with the emotion "Warm" only achieving 15% accuracy.

On the contrary, the LDA-SVM included both keyword and long-distance relations, and greatly outperforms the Naïve Bayes with an overall accuracy of 76.1%. It even achieved the highest accuracy of 92.83% and 85.40% for the emotion "Worried" and "Odd", respectively, among all five methods. As we can see, the KW-SVM can bring about substantial proficiency in detecting the emotions with 77.70% overall accuracy. This indicates that reader's emotion can be recognized effectively by using only the LLR scores of keywords, since the likelihood of a word existing in a certain emotion is not random. Those with a larger LLR value are considered as closely associated with the emotion (Manning & Schütze, 1999). The TBA can further improve the basic keyword-based method with rich context and semantic information, thus achieving the best overall accuracy of 84.72%.

Table 2. Accuracy of emotion detection systems.

Topic	***Accuracy (%)***				
	NB	**LDA-SVM**	**KW-SVM**	**CF-SVM**	**TBA**
Angry	47.00	74.21	79.21	83.71	**83.92**
Worried	69.56	**92.83**	81.96	87.50	80.12
Boring	75.67	76.21	84.34	87.52	**87.88**
Depressing	73.76	81.43	85.00	87.70	**90.13**
Happy	37.90	67.59	80.97	86.27	**89.50**
Warm	15.09	87.09	79.59	85.83	**88.56**
Odd	73.90	**85.40**	77.05	84.25	85.32
Informative	20.60	44.02	74.74	**83.59**	82.10
A^M	51.69	76.10	80.36	85.80	**85.94**
A^μ	34.52	58.68	77.68	84.61	**84.72**

It is worth noting that the CF-SVM achieved a satisfactory accuracy around 80% among all emotions. This is because the combined lexicon feature sets (i.e. character bigrams, word dictionary, and emotion keywords) of CF-SVM improved the classification accuracy. In addition, the metadata of the articles are also associated with reader-emotion. For instance, we found that many sports related news articles evoke "Happy" emotion. In particular, 45% of all

"Happy" instances belong to the news category sports. It is also observed that an instance with the news category sports has a 31% chance of having the true class "Happy". Hence, the high accuracy of "Happy" emotion can be the result of people's general enthusiasm over sports rather than the result of a particular event. On top of that, the TBA can generate distinct semantic templates to capture alternations of similar combinations to achieve the optimal outcome. For instance, a semantic template generated by our proposed system, "{國家 $_{\text{country}}$}:[發生 $_{\text{occur}}$]:[地震 $_{\text{earthquake}}$]:{劫難 $_{\text{disaster}}$}", belongs to the emotion "Depressing". It is perceivable that this template is relaying information about disastrous earthquakes that occurred in a certain country, and such news often induce negative emotions among the readers. This example demonstrates that the automatically generated semantic templates are comprehensible for humans and can be utilized to effectively detect reader's emotion. Nevertheless, our system could not surpass the LDA-SVM in the emotion "Worried". It may be attributed to the fact that semantic templates generated in this emotion have inadequate quality. We examined some of the templates within this emotion and found that they mostly contain very general semantic classes, such as "{機構 $_{\text{institution}}$}:{組織 $_{\text{organization}}$}:{政黨 $_{\text{party}}$}:{實現 $_{\text{realize}}$}:{程度 $_{\text{degree}}$}:{念頭 $_{\text{thought}}$}", thereby reducing its accuracy. Despite the "Worried" emotion, we were able to identify distinctive semantic templates for the other emotions. For instance, the template "[婦女 $_{\text{women}}$]:{救助 $_{\text{help}}$}:[小孩 $_{\text{child}}$]:{當作 $_{\text{treat}}$}:{人 $_{\text{human}}$}:[認為 $_{\text{consider}}$]" was generated for the emotion "Warm", and it is reasonable that news about a woman helping a child would evoke a warm feeling in the readers' mind. The ability to generate such emotion-specific templates is considered as the main reason for TBA to outperform other systems.

As a last touch, we dig into the top keywords in each emotion category as an analysis of the news trend, listed in TABLE 3. As stated in the previous section, we observe that keywords related to "Happy" are mostly terms about sports, such as team names (e.g., "熱火 $_{\text{Miami Heat}}$" and "紅襪 $_{\text{Boston Red Sox}}$") or player names (e.g., "陳偉殷 $_{\text{Wei-Yin Chen}}$", a pitcher for the Baltimore Orioles). Similar findings had been made before as well (Chen *et al.* 2008). This is certainly a good indication for the performance of a sports team over a specified period of time. On the other hand, "Angry"-related keywords consist largely of political parties or public issues. For instance, the most noticeable word "美牛 United States beef" indicates the heated dispute on the import policy of beef from United States to Taiwan, which has been an issue in Taiwan-U.S. relations and leads to a domestic political unrest. Numerous political terms showed up in the top list too, such as "國民黨 $_{\text{Kuomintang}}$", "立法院 $_{\text{legislature}}$", and "立委 $_{\text{legislator}}$". This highlights the extracted emotion keywords are highly correlated with reader's emotion, thus tagging them in the emotion template helps our method discriminate reader-emotion.

As for the Depressing category, the keywords mostly relate to social events that involve severe weather or casualties. The phrase "大炳 Da Bing" refers to a Taiwanese performer who died in 2012 (around the time of our data retrieval) due to drug abuse. On the off chance that sports players suffered from low performance temporarily, they might show up in the category too due to readers' compassion. Theoretically if we have sufficient patterns containing negation terms, we will be able to generate representative templates that can give rise to the scores in negative emotions. However, in our current data set, the relatively low portion of negative cases affects little more than the score of positive emotion, hence affecting the system performance. We acknowledge that this is an important task and should be studied in our future work. Lastly, not surprisingly, the Warm category contains words associated with mostly social care or volunteer providing charity or assistance to social vulnerable groups, while economic news dominates the Informative emotion.

Table 3. A top 10 keyword list of each reader-emotion generated from TBA.

	Keywords discovered by the TBA
Angry	美牛 American beef，立委 Legislator，立法院 Legislative Yuan，國民黨 Kuomintang Party，政府 Government，證所稅 Stock exchange income tax，中油 CPC，總統 President，馬英九 Ma Ying-jeou，黨團 Political party
Worried	颱風 Typhoon，氣象局 Central Weather Bureau，病毒 Virus，感染 Infection，豪雨 Rain storm，地震 Earthquake，土石流 Mudslide，餘震 Aftershock，疾病 Disease，病例 Patient record
Boring	蘇貞昌 Su Jen Chang，陳水扁 Chen Shui-bian，名嘴 Critics，章子怡 Zhang Ziyi，陳文茜 Chen Wen-chien，宋正宇 Sung Jen-yu，節目 TV show，吳宗憲 Jacky Wu，藝人 Celebrity，女星 Female celebrity
Depressing	大炳 Da Bing，民進黨 Democratic Progressive Party，遺體 Remain，送醫 To hospital，溺水 Drown，急救 Emergency medical service，不治 Dead，失蹤 Missing，豪雨 Rain strom，曾雅妮 Tseng Yani
Happy	陳偉殷 Chen Wei-yin，安打 Strike，比賽 Game，熱火 Heat，紅襪 Red Sox，宏達電 HTC，殷仔 Yin，雷霆 Thunder，太空人 Astros，冠軍 Championship
Warm	孩子 Children，家扶 Fund for family，媽媽 Mother，父親 Father，志工 Volunteer，關懷 Care，基金會 Foundation，行善 Charity，幫助 Assistance，弱勢 Social vulnerable
Odd	男子 Man，警方 Police，女子 Woman，台灣 Taiwan，發現 discover，竟然 surprisingly，監視器 Security camera，網友 Internet user，立委 Legislator，美牛 American beef
Informative	工商時報 Commercial Times，市場 Market，報導 Report，營收 Revenue，成長 Growth，需求 Necessity，金融 Financial，產品 Product，自由時報 Liberty Times，投資 Investment

To summarize, the proposed TBA integrates the syntactic, semantic, and context information in text to identify reader-emotions and achieves the best performance among the compared methods. It also demonstrates the capabilities of our approach to integrate statistical and knowledge-based models. Notably, in contrast to models used by previous machine learning-based methods which are generally not human understandable, the generated templates can be acknowledged as the fundamental knowledge for each emotion and are comprehensible to the human mind.

5. Conclusion

With the rapid growth of computer mediated communication applications, the research on emotion classification has been attracting more and more attention recently from enterprises toward business intelligence. Recognizing reader-emotion concerns the emotion expressed by a reader after reading the text, and it holds the potential to be applied in fields that differ from writer-emotion detection applications. For instance, users are able to retrieve documents that contain relevant contents and at the same time produce desired feelings by integrating reader-emotion into information retrieval. In addition, reader-emotion detection can assist writers in foreseeing how their work will influence readers emotionally. In this research, we presented a flexible template-based approach (TBA) for detecting reader-emotion that simulates the process of human perception. By capturing the most prominent and representative pattern within an emotion, TBA allows us to effectively recognize the reader-emotion of text. Results of our experiments demonstrate that the TBA can achieve a higher performance than other well-known methods of reader-emotion detection. In the future, we plan to refine our TBA and employ it to other natural language processing applications. Also, further studies can be done on combining statistical models into different components in our system.

Acknowledgement

This research was supported by the Ministry of Science and Technology of Taiwan under grant MOST 103-3111-Y-001-027.

Reference

Bashaddadh, O.M.A. & Mohd, M. (2011). Topic detection and tracking interface with named entities approach. In *Proceedings of International Conference on Semantic Technology and Information Retrieval*, 215-219.

Blei, D.M., Ng, A.Y., & Jordan, M.I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993-1022.

Chen, K.J., Huang, S.L., Shih, Y.Y., & Chen, Y.J. (2005). Extended-HowNet: A representational framework for concepts. In *Proceedings of OntoLex – Ontologies and Lexical Resources IJCNLP-05 Workshop*, 1-6.

Chen, Y., Lee, S., Li, S., & Huang, C. (2010). Emotion cause detection with linguistic constructions. In *Proceedings of the 21th International Conference on Computational Linguistics*, 179-187.

Cortes, C. & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 1-25.

Das, D. & Bandyopadhyay, S. (2009). Word to Sentence Level Emotion Tagging for Bengali Blogs. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, 149-152.

Dong, Z.D., Dong, Q., & Hao, C.L. (2010). Hownet and its computation of meaning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, 53-56.

Garey, M.R. & Johnson, D.S. (1979). *Computers and intractability: A Guide to the Theory of NP- Completeness*. W. H. Freeman & Co. New York, NY, USA.

Johnson, D.S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, *9*(3), 256-278.

Read, J. (2005). Using Emotions to Reduce Dependency in Machine Learning Techniques for Sentiment Classification. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics Student Research Workshop*, 43-48.

Lovász, L. (2005). Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, *2*(1), 1-46.

Lee, C.S., Jian, Z.W., & Huang, L.K. (2005). A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *35*(5), 859-880.

Lin, H.Y., Yang, C.H., & Chen, H.H. (2007). What Emotions Do News Articles Trigger in Their Readers? In *Proceedings of 30th Annual International ACM SIGIR Conference*, 23-27.

Lin, K.H., Yang, C.H., & Chen, H.H. (2008). Emotion Classification of Online News Articles from the Reader's Perspective. In *Proceedings of International Conference on Web Intelligence*, 220- 226.

Manning, C.D. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

McCallum, A. & Nigam, K. (1998). A comparison of event models for Naïve Bayes text classification. In *Proceedings of AAAI/ICML-98 Workshop on Learning for Text Categorization*, 41-48.

Mishne, G. (2005). Experiments with mood classification in blog posts. In *Proceedings of the 1st Workshop on Stylistic Analysis of Text for Information Access*.

Mukherjee, A. & Liu, B. (2010). Improving Gender Classification of Blog Authors. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 207-217.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, 79-86.

Patwardhan, S. & Peterson, T. (2003). Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts.

Purver, M. & Battersby, S. (2012). Experimenting with distant supervision for emotion classification. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 482-491.

Quan, C. & Ren, F. (2009). Construction of a Blog Emotion Corpus for Chinese Emotional Expression Analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 1446-1454.

Ricardo, B.Y. & Berthier, R.N. (2011). *Modern Information Retrieval: The Concepts and Technology Behind Search*. New York: Addison Wesley.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*(5), 513-523.

Shih, C.W., Lee, C.W., Tsai, T.H., & Hsu, W.L., (2012). Validating Contradiction in Texts Using Online Co-Mention Pattern Checking. *ACM Transactions on Asian Language Information Processing*, *11*(4), 17. doi: 10.1145/2382593.2382599 http://doi.acm.org/10.1145/2382593.2382599

Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., & Deng, X. (2013). Exploiting Topic based Twitter Sentiment for Stock Prediction. In *Proceedings of The 51st Annual Meeting of the Association for Computational Linguistics*, 24-29.

Tang, Y.J. & Chen, H.H. (2012). Mining Sentiment Words from Microblogs for Predicting Writer-Reader-emotion Transition. In *Proceedings of 8th International Conference on Language Resources and Evaluation*, 1226-1229.

Thayer, R.E. (1989). *The Biopsychology of Mood and Arousal*, Oxford University Press.

Wang, M.H., Lee, C.S, Hsieh, K.L., Hsu, C.Y., Acampora, G., & Chang, C.C. (2010). Ontology-based multi-agents for intelligent healthcare applications. *Journal of Ambient Intelligence and Humanized Computing*, *1*(2), 111-131.

Wu, C.H., Chuang, Z.J., & Lin, Y.C. (2006). Emotion recognition from text using semantic labels and separable mixture models. *ACM Transactions on Asian Language Information Processing*, *5*(2), 165-183.

Yang, C.H., Lin, H.Y., & Chen, H.H. (2009). Writer Meets Reader: Emotion Analysis of Social Media from both the Writer's and Reader's Perspectives. In *Proceedings of International Conference on Web Intelligence*, 287-290.

Yang, C.H., Lin, K.H.Y., & Chen, H.H. (2007). Building emotion lexicon from Weblog corpora. In *Proceedings of 2007 IEEE/WIC/ACM International Conference on Web Intelligence*, 275-278.

Yang, Y.H., Liu, C. C., & Chen, H. H. (2006). Music emotion classification: A fuzzy approach. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, 81-84.

Zhang, L. & Liu, B. (2010). Extracting and Ranking Product Features in Opinion Documents. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 1462-1470.

Zou, F., Wang, F.L., Deng, X., Han, S., & Wang, L.S. (2006). Automatic construction of Chinese stop word list. In *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*, 1010-1015.

Computational Linguistics and Chinese Language Processing
Vol. 21, No. 1, June 2016, pp. 51-68

Enriching Cold Start Personalized Language Model Using Social Network Information[1]

Yu-Yang Huang[*], Rui Yan[+], Tsung-Ting Kuo[*], and Shou-De Lin[*]

Abstract

Personalized language models are useful in many applications, such as personalized search and personalized recommendation. Nevertheless, it is challenging to build a personalized language model for cold start users, in which the size of the training corpus of those users is too small to create a reasonably accurate and representative model. We introduce a generalized framework to enrich the personalized language models for cold start users. The cold start problem is solved with content written by friends on social network services. Our framework consists of a mixture language model, whose mixture weights are estimated with a factor graph. The factor graph is used to incorporate prior knowledge and heuristics to identify the most appropriate weights. The intrinsic and extrinsic experiments show significant improvement on cold start users.

Keywords: Language Model, Factor Graph, Social Network Analysis, Smoothing, Cold-Start Problem.

1. Introduction

Personalized language models on social network services are useful in many aspects (Xue *et al.*, 2009; Wen *et al.*, 2012; Clements, 2007). For instance, if the authorship of a document is in doubt, a language model may be used as a generative model to identify it. In this sense, a language model serves as a proxy of one's writing style. Furthermore, personalized language models can improve the quality of information retrieval and content-based recommendation

[1] A preliminary version of this paper appears as a short paper in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 611–617, Baltimore, Maryland, USA, June 23-25 2014.

[*] Graduate Institute of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.
E-mail: {r02922050, d97944007, sdlin}@csie.ntu.edu.tw

[+] Computer and Information Science Department, University of Pennsylvania, Philadelphia, U.S.A.
E-mail: ruiyan@seas.upenn.edu

systems, where documents or topics can be recommended based on the generative probabilities.

It is challenging, however, to build a personalized language model for users who just entered the system since the content posted by these users is insufficient to characterize them. These users are referred to as “cold start” users. Since the popularity of a system largely depends on whether new users will continue to stick to the system, it is even more critical to generate good recommendation results for new users. Therefore, this paper focuses on how to obtain a better personalized language model for cold start users.

To achieve the aforementioned purpose, the content written by friends on a social media site is exploited. It can be a reply or a post written by friends on sites like Facebook or Twitter. Here, the hypothesis is that friends, who usually share common interests, tend to discuss similar topics and use similar words than non-friends. In other words, we believe that a cold start user’s language model can be enriched and better personalized by incorporating content written by friends.

Intuitively, a linear combination of language models can be used to mix the content written by friends into a user’s language model. Nevertheless, it should be noticed that some documents are more relevant than others and should be weighted higher. To obtain better weights, some simple heuristics could be exploited. For example, we can measure the similarity or dissimilarity between a user language model and a document language model. In addition, documents being shared more frequently in a social network usually are considered to be more influential and should be expected to contribute more to the refinement of a user language model. More complex heuristics also can be derived. For instance, if we can categorize the documents and find that two documents are of the same category, then their weights should be more similar. The main challenge lies in how such heuristics can be utilized in a systematic manner to infer the weights of each document-level language model.

In this paper, we exploit the information on social network services in two ways. First, we impose the social dependency assumption via a finite mixture model. We model the true, albeit unknown, personalized language model as a combination of a biased user language model and a set of relevant document language models. Due to the noise inevitably contained in social media content, instead of using all available documents, we argue that, by properly specifying the set of relevant documents, a better personalized language model can be learnt. In other words, each user language model is enriched by a personalized collection of background documents.

Second, we propose a factor graph model to incorporate prior knowledge (*e.g.* the heuristics described above) into our model. Each mixture weight is represented by a random variable in the factor graph. An efficient algorithm is proposed to optimize the model and infer the marginal distribution of these variables. Useful information and heuristics are encoded into the model by a set of potential functions.

The main contributions of this work are summarized below.

- To solve the cold start problem encountered during the estimation of personalized language models, a generalized probabilistic framework is proposed. We incorporate social network information into user language models through the use of a factor graph model. An iterative optimization procedure utilizing perplexity is presented to learn the model parameters. To our knowledge, this is the first proposal to use a factor graph model to enrich language models.
- Perplexity is selected as an intrinsic evaluation, and experiment on authorship attribution is used as an extrinsic evaluation. The results show that our model yields significant improvements for cold start users.

2. Methodology

We describe how to construct and enrich a personalized language model in this section. In the first subsection, we propose a social-driven, personalized mixture language model. The original, poorly estimated user language model is enriched with a set of relevant document language models. In Section 2.2, a graphical model is presented to identify the mixture weights of each mixture component. The relative importance of each mixture component, *i.e.* document language model, is determined with the use of prior knowledge that comes from a social network. In Section 2.3, we describe how the model is optimized under a lack of labelled information.

2.1 Social-Driven Personalized Language Model

The language model of a collection of documents can be estimated by normalizing the counts of words in the entire collection (Zhai, 2008). To build a user language model, one naïve way is first to normalize word frequency $c(w,d)$ within each document, then average over all the documents in a user's document collection. The resulting unigram user language model is:

$$P_u(w) = \frac{1}{|\mathcal{D}_u|}\sum_{d\in\mathcal{D}_u}\frac{c(w,d)}{|d|} = \frac{1}{|\mathcal{D}_u|}\sum_{d\in\mathcal{D}_u}P_d(w) \quad (1)$$

where $P_d(w)$ is the language model of a particular document, $\mathcal{D}_u$ is the user's document collection, and $|\cdot|$ denotes the number of elements in a set. This formulation is basically an equal-weighted finite mixture model.

A simple yet effective way to smooth a language model is to linearly interpolate with a background language model (Chen & Goodman, 1996; Zhai & Lafferty, 2001). In the linear interpolation method, all background documents are treated equally. The entire document collection is added to the user language model $P_u(w)$ with the same interpolation coefficient. On social media, however, articles are often short and noisy. The user language models generated in this way are prone to overfitting. To obtain a better personalized user language model, we must take into consideration the complicated document-level correlations and dissimilarities, and this is where our idea was born.

Our main idea is to specify a set of relevant documents for the target user and enrich the user language model with these documents. Then, through the use of the information embedded in a social network, the relative importance of these documents is learnt. Suppose that the target user is u. Letting $\mathcal{D}_{rel}$ denote the content posted by people that are most relevant to u (*e.g.* friends on a social network), our idea can be concisely expressed as:

$$\widetilde{P_u}(w) = \lambda_u P_u(w) + \sum_{d \in \mathcal{D}_{rel}} \lambda_d P_d(w) \qquad (2)$$

where λ_d is the mixture weight of the language model of document d, and $\lambda_u + \sum \lambda_d = 1$. Documents posted by irrelevant users are ignored as we believe the user language model can be personalized better by exploiting the social relationship in a more structured way. In our experiments, we choose the documents posted by friends as $\mathcal{D}_{rel}$.

Also note that we have made no assumption about how the “base” user language model $P_u(w)$ is built. In practice, it need not be models following maximum likelihood estimation, but any language model can be integrated into our framework to achieve a more refined model. Furthermore, any smoothing method can be applied to the language model without degrading the effectiveness.

2.2 Factor Graph Model

Now, we discuss how the mixture weights can be estimated. We introduce a *factor graph model* to make use of the diverse information on a social network. Factor graph (Kschischang *et al.*, 2006) is a bipartite graph consisting of a set of *random variables* and a set of *factors* that signifies the relationships among the variables. It is best suited to situations where the data is clearly of a relational nature (Wang *et al.*, 2012). The joint distribution of the variables is factored according to the graph structure. Using a factor graph model, we can incorporate the knowledge into the potential function for optimization and perform joint inference over documents.

A factor graph model is presented in Figure 1. As can be seen from Equation 2, there are $|\mathcal{D}_{rel}|$ unknown mixture weights to be estimated. For each mixture weight λ_d, we put a Bernoulli random variable y_d in the factor graph. The value $y_d = 1$ means that the document d should be included in the enriched personalized language model $\widetilde{P_u}$ of the target user. In this sense, a larger value of $P(y_d = 1)$ implies a higher mixture weight of d. In particular, we set λ_d to be proportional to $P(y_d = 1)$ in the final estimation.

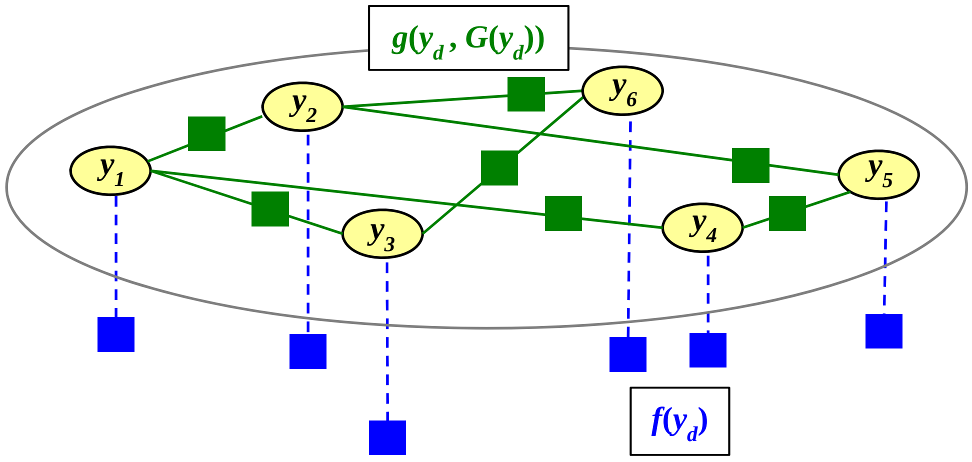


Figure 1. The proposed factor graph model.

Two kinds of potential functions are defined in the proposed factor graph model.

Local potential function *f(y_d)*. This potential function captures the local attributes of a random variable y_d. Suppose that the random variable y_d corresponds to a document d. The local potential function $f(y_d = 1)$ should take a larger value relative to $f(y_d = 0)$ if d is likely to contribute more significantly to the language model $\widetilde{P_u}$. The local potential function $f(y_d)$ is parameterized in a log-linear form:

$$f(y_d) = exp\{\alpha^T \mathbf{f}(y_d)\} \tag{3}$$

where $\mathbf{f} = \langle f_j \rangle^T$ is a vector of predefined feature functions and α is the parameter vector to be learnt. We assume that all feature functions $f_j(y_d)$ take a value of zero if $y_d = 0$. So, the larger the value of $\alpha^T \mathbf{f}(y_d = 1)$ is, the higher is the value $f(y_d = 1)$ relative to $f(y_d = 0)$. In other words, λ_d is (locally) believed to be higher.

In our experiment, we define the vector of feature functions as $\mathbf{f} = \langle f_{sim}, f_{oov}, f_{pop}, f_{cmf}, f_{af} \rangle^T$:

- *Similarity function f_{sim}*. The similarity between language models of the target user and a document should play an important role. We use cosine similarity between two unigram models in our experiments.
- *Document quality function f_{oov}*. The out-of-vocabulary (OOV) ratio is used to measure the quality of a document. It is defined as:

$$f_{oov} = 1 - \frac{|\{w : w \in d, w \notin V\}|}{|d|} \tag{4}$$

where V is the vocabulary set of the entire corpus, with stop words excluded.

- *Document popularity function f_{pop}.* To model the popularity of a document, this function is defined as the number of times the document d is shared.
- *Common friend function f_{cmf}.* It is defined as the number of common friends between the target user and the author of d.
- *Author friendship function f_{af}.* Assuming that documents posted by a user with more friends are more influential, this function is defined as the number of friends of the author of document d.

Pairwise potential function $g(y_d, G(y_d))$. For any two documents d_i and d_j, let the corresponding variables in the factor graph be y_i and y_j, respectively. The pairwise potential function defines the correlation of a variable y_i with another variable y_j. Similar to the local potential function, this function is parameterized as:

$$g(y_i, y_j) = exp\{\beta^T \mathbf{g}(y_i, y_j)\} \tag{5}$$

where **g** is a vector of feature functions indicating whether two variables are correlated. We assume that the two variables are not connected in the factor graph if $g(y_i, y_j) = 0$.

If we further denote the set of all variables linked to y_d as $G(y_d)$, then, for any variable y_d, we obtain the following result:

$$\begin{aligned} g(y_d, G(y_d)) &= \textstyle\prod_{y \in G(y_d)} g(y_d, y) \\ &= exp\{\textstyle\sum_{y \in G(y_d)} \beta^T \mathbf{g}(y_d, y)\} \end{aligned} \tag{6}$$

which is a function of y_d only. This expression will be used in the following equations.

We define the vector of feature functions $\mathbf{g} = \langle g_{rel}, g_{cat} \rangle^T$ as follows.

- *User relationship function g_{rel}.* We assume that two variables y_i and y_j are higher correlated if d_i and d_j are of the same author or the two authors are friends. The correlation should be even greater if the two documents are similar. Letting $a(d)$ denote the author of a document d and $\mathcal{N}[u]$ denote the closed neighborhood of a user u, we define

$$g_{rel} = \begin{cases} sim(d_i, d_j) & \text{if } a(d_j) \in \mathcal{N}[a(d_i)] \\ 0 & \text{if } a(d_j) \notin \mathcal{N}[a(d_i)] \end{cases}. \tag{7}$$

The similarity between documents $sim(d_i, d_j)$ is measured by the cosine similarity between two unigram language models.

- *Co-category function g_{cat}.* For any two variables y_i and y_j, it is intuitive that the two variables would have a higher correlation if d_i and d_j are of the same category. Letting $c(d)$ denote the category of document d, we define

$$g_{cat} = \begin{cases} sim(d_i, d_j) & \text{if } c(d_i) = c(d_j) \\ 0 & \text{if } c(d_i) \neq c(d_j) \end{cases}. \tag{8}$$

The flexibility of the proposed framework lies in the following aspects.

- The factor graph model is adaptable. The feature functions are not restricted to the ones we have used and can be freely added or redesigned in order to properly model different datasets.
- The set of relevant documents can be changed. In our experiment, we used the documents posted by friends to enrich the language model. Nevertheless, this is not a requirement. Whenever appropriate, documents posted by friends of friends, or any arbitrary set of documents can be adapted to tackle this problem.
- As we mentioned at the end of Section 2.1, the "base" user language model can be already smoothed by any technique. Furthermore, the language models need not be unigram models. If a higher order n-gram model is more suitable, it can be used in our framework. For our particular dataset, however, we find that it gives no advantage to use higher order n-gram models.

2.3 Model Inference and Optimization

Let Y be the set of all random variables. The joint distribution encoded by the factor graph model is given by multiplying all potential functions:

$$P(Y) = \frac{1}{Z}\prod_{d\in\mathcal{D}_{rel}} f(y_d)g\big(y_d, G(y_d)\big) \tag{9}$$

where Z is a normalization term to ensure that the probability sums to one.

The desired marginal distribution $P(y_d)$ can be obtained by marginalizing all other variables. Under most circumstances, however, the factor graph is densely connected. This makes the exact inference intractable, and approximate inference is required. After obtaining the marginal probabilities $P(y_d)$ with the approximate inference algorithm, the mixture weights λ_d in Eq. 2 are estimated by normalizing the corresponding marginal probabilities $P(y_d)$ to satisfy the constraint $\lambda_u + \sum \lambda_d = 1$. The normalization can be written as:

$$\lambda_d = (1-\lambda_u)\frac{P(y_d)}{\sum_{d\in\mathcal{D}_{rel}} P(y_d)}. \tag{10}$$

It can be verified that the above equation leads to a valid probability distribution for our mixture model.

The proposed factor graph model has $|\alpha| + |\beta|$ parameters, where $|\beta|$ means the dimensionality of the vector β. Combining Equation 2 and Equation 10, it can be observed that the total number of parameters in the mixture model is reduced from $1 + |\mathcal{D}_{rel}|$ to $1 + |\alpha| + |\beta|$, lowering the risk of overfitting.

A factor graph is often optimized by gradient-based methods. Unfortunately, since the ground truth values of the mixture weights λ_d are not available, we are prohibited from using these approaches. Here, we propose a two-step iterative procedure to optimize our model with respect to the model perplexity on held-out data.

At first, all of the model parameters (*i.e.* α, β, λ_u) are initialized randomly. Then, we infer the marginal probabilities of the random variables. Given these marginal probabilities, we can evaluate the perplexity of the user language model on a held-out dataset and search for better parameters. This procedure is repeated until convergence. We have also tried to train the model by optimizing the accuracy of the authorship attribution task. Nevertheless, we find that models trained by optimizing the perplexity give better performance.

3. Experiment

In this section, we evaluate the performance of language model enrichment with both intrinsic (perplexity) and extrinsic (authorship attribution) metrics. We compare the enriched language model with the original base language model and three intuitive enrichment methods.

3.1 Dataset and Experiment Setup

We performed experiments on the *Twitter* dataset collected by Galuba *et al.* (2010). Twitter data have been used to verify models with different purposes (Lin *et al.*, 2011; Tan *et al.*, 2011). To emphasize the cold start scenario, we randomly selected 15 users with about 35 tweets and 70 friends as candidates for an authorship attribution task. Our corpus consists of 4322 tweets. All words with less than 5 occurrences were removed. Stop words and URLs also were removed, and all words were stemmed. We identified the 100 most frequent terms as categories. The size of the vocabulary set is 1377.

We randomly partitioned the tweets of each user into training, validation, and testing sets. The reported result is the average of 20 random splits. In all experiments, we varied the size of training data from 1% to 15%, and held the same number of tweets from each user as validation and testing data. The statistics of our dataset, given 15% training data, are shown in Table 1.

Loopy belief propagation (Murphy *et al.*, 1999) was used to obtain the marginal probabilities. Parameters were searched with the pattern search algorithm (Audet & Dennis, 2002). To not lose generality, we used the default configuration in all experiments.

Table 1. Dataset statistics.

# of	Max.	Min.	Avg.
Tweets	70	19	35.4
Friends	139	24	68.9
Variables	467	97	252.7
Edges	9216	231	3427.1

3.2 Baseline Methods

We compared our framework with three baseline methods. The first ("*Cosine*") is a straightforward implementation that sets all mixture weights λ_d to the cosine similarity between the probability mass vectors of the document and user unigram language models. The second ("*PS*") uses the pattern search algorithm to perform constrained optimization over the mixture weights. Satisfying the constraint $\lambda_u + \sum \lambda_d = 1$, the algorithm iteratively searches for the optimal mixture weights λ_u and λ_d to lower the perplexity on the validation data. As mentioned in Section 2.3, the main difference between this method and ours ("*FGM*") is that we reduce the search space of the parameters by the factor graph model. Furthermore, social network information is exploited in our framework, while the PS method performs a direct search over mixture weights, discarding valuable knowledge. The third ("*LR*") models the probability $P(y_d)$ with the logistic function, where the local feature functions of each node are regarded as the independent variables, *i.e.* $P(y_d) = 1/(1 + e^{-\alpha^T \mathbf{f}(y_d)})$. By comparing our model to this baseline method, we want to show that the pairwise connections between nodes are useful.

Different from other smoothing methods, which usually are mutually exclusive, any other smoothing methods can be easily merged into our framework. In Equation 2, the *base language model* $P_u(w)$ can be already smoothed by any technique before being plugged into our framework. Our framework then enriches the user language model with social network information. We selected four popular smoothing methods to demonstrate such an effect, namely additive smoothing, absolute smoothing (Ney *et al.*, 1995), Jelinek-Mercer (JM) smoothing (Jelinek & Mercer, 1980), and Dirichlet smoothing (MacKay & Peto, 1994). Except for additive smoothing, the other three smoothing methods were all based on interpolation with the background corpus. The results of using only the base model (*i.e.* setting $\lambda_d = 0$ in Eq. 2) are denoted as "*Base*" in the following tables.

3.3 Perplexity

As an intrinsic evaluation, for each tweet in the testing set, we computed the perplexity of it under the author's own language model. The idea is that a better personalized language model should assign higher probability to a tweet if and only if it is written by the user himself.

The perplexity of a single sentence is defined as:

$$PPL(w_1 \cdots w_N) = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i)} \tag{11}$$

where $w_1 \cdots w_N$ is an unseen testing sentence. The overall perplexity on a collection of sentences is simply computed by concatenating them. A smaller value signifies better performance. The results are shown in Table 2, where the asterisks indicate a significant difference between the best score and the second best score by t-test at a significance level of

0.05.

Table 2. Testing set perplexity.

Train %	Additive					Absolute				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	903.1	711.6	701.3	843.9	**534.8***	864.7	695.3	696.7	814.7	**540.4***
5%	818.6	622.9	681.0	784.1	**504.3***	776.2	608.1	672.9	748.9	**509.6***
10%	749.9	559.8	658.1	726.4	**479.1***	703.2	543.7	638.9	681.3	**483.6***
15%	696.3	517.2	633.7	683.9	**471.7***	647.0	498.7	613.8	630.7	**469.4***
Train %	**Jelinek-Mercer**					**Dirichlet**				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	631.7	571.3	640.6	615.7	**533.1***	632.6	571.6	635.8	616.2	**532.3***
5%	588.9	524.5	602.2	574.9	**505.3***	589.8	525.0	602.1	575.6	**503.0***
10%	553.0	489.9	570.2	545.4	**478.7***	554.0	490.7	581.4	546.2	**477.4***
15%	529.0	469.3	561.4	523.5	**465.6***	529.8	469.8	567.1	524.5	**466.2**

Our method significantly outperforms all of the methods in almost all settings. Furthermore, all methods gradually improve as more data are used to train the model. As expected, the advantage of our method is more apparent when data is sparse. Also, our method works much better than the "*LR*" method, which demonstrates the usefulness of the pairwise connections between documents in the factor graph.

We observe that the "*PS*" method takes a long time to converge and is prone to overfitting, likely because it has to search a few hundred parameters on average. It can also be observed from the table that, if the base language model is already smoothed by JM or Dirichlet smoothing, the "*PS*" method will only worsen, instead of enrich, the language model.

In terms of testing set perplexity, the "*Cosine*" method is the second best method to enrich a user's language model. The gap between the "*Cosine*" method and our method becomes smaller as more training data is available. When the base model is already smoothed by JM or Dirichlet smoothing and the data is less sparse (for example, the "15%" row), the "*Cosine*" method performs almost as good as our method. Nevertheless, when the base user language model $P_u(w)$ is sparse (*i.e.* the "1%" or "5%" rows), the similarity scores are not reliable and the performance of this method is restricted due to the bias coming from the base model.

3.4 Authorship Attribution

The authorship attribution task is chosen as the extrinsic evaluation metric. Given a sentence of unknown authorship, the task is to identify its author from a finite set of candidate users. For a thorough survey of recent works on this topic, see (Stamatatos, 2009).

Here, the goal is not about comparing with the state-of-the-art approaches in authorship attribution, but showing that typical applications of language model techniques can benefit from our framework.

To apply a personalized language model on this task, a naïve Bayes classifier is implemented (Peng *et al.*, 2004). The most probable author of a document d is the one whose personalized language model yields the highest probability. It is determined by the following equation:

$$u^* = argmax_u\{\prod_{w \in d} \widetilde{P_u}(w)\} \tag{12}$$

where we assume uniformity in the candidate users, and $\widetilde{P_u}(w)$ is as defined in Eq. 2.

Notice that we have used the unigram probability in Eq. 12, as in all following experiments. In fact, we also have conducted experiments with bigram models. Although the bigram model achieves lower perplexity as expected, we have observed a 15 to 20 percent decrease of the accuracy on the authorship attribution task, compared to the unigram model. This signifies that higher order n-gram models may not be suitable for the sparse data scenario. Similar arguments also have been given by Peng (2004). The results are shown in Table 3, where the asterisks indicate a significant difference between the best score and the second best score, by t-test at a significance level of 0.05.

Comparing the four "*Base*" columns, we find that additive smoothing performs about as well as the other three smoothing methods. That is, blindly interpolating with the entire background corpus does not fix the sparse data problem.

Similar to the result in Section 3.3, the "*Cosine*" method gets better (even better than our method, if the JM or Dirichlet smoothing is applied) when more data is available. A possible explanation is that the cosine similarity between the base user language model $P_u(w)$ and a document language model $P_d(w)$ is not reliable when $P_u(w)$ is estimated from a small corpus. In these cases, using the cosine similarity alone is not enough and it is better to rely less on this information and include other types of features. Our factor graph model can alleviate this problem by bringing more information into the language model; hence, it performs better than this baseline method under such circumstances. Conclusions that can be drawn from the results of the "*PS*" and "*LR*" methods are similar to those in the previous section.

Table 3. Accuracy (%) of authorship attribution.

Train %	Additive					Absolute				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	0.5307	0.5853	0.6040	0.5700	**0.6587***	0.5120	0.5787	0.5947	0.5640	**0.6413***
5%	0.5967	0.6313	0.6267	0.6073	**0.6760***	0.5753	0.6233	0.6080	0.5947	**0.6687***
10%	0.6307	0.6640	0.6280	0.6433	**0.7053***	0.6153	0.6733	0.6220	0.6400	**0.6920**
15%	0.6513	0.6867	0.6400	0.6560	**0.7113***	0.6487	0.6967	0.6260	0.6560	**0.7200***
Train %	Jelinek-Mercer					Dirichlet				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	0.5400	0.6127	0.6107	0.5967	**0.6560***	0.5233	0.6127	0.6093	0.5860	**0.6527***
5%	0.6147	0.6607	0.6420	0.6173	**0.6787***	0.6020	0.6567	0.6360	0.6187	**0.6787***
10%	0.6373	**0.6967**	0.6633	0.6513	0.6860	0.6293	0.6927	0.6507	0.6433	**0.6947**
15%	0.6553	**0.7094**	0.6520	0.6593	0.6987	0.6533	**0.7100***	0.6467	0.6567	0.6907

3.5 Feature Effectiveness

In this section, we evaluate the effectiveness of the feature functions mentioned in Section 2.2. There are five local feature functions $(f_{sim}, f_{oov}, f_{pop}, f_{cmf}, f_{af})$ and two pairwise feature functions (g_{cat}, g_{rel}) in our model. Including λ_u, there are a total of eight parameters for each user language model $\widetilde{P_u}(w)$. Since all features are standardized before training, we can inspect the feature weights learnt from data to determine the relative importance of them. To be focused on the cold start setting, we choose to analyze the case where only 1% of the data is used during model training. We will refer to the feature weights as, say w_{sim}, for simplicity.

First, we inspect the mixture weight λ_u for the base user language model. The higher this value is, the better is the original base user language model $P_u(w)$. We plot this value for all 15 users (the x-axis) in Figure 2. The different colors of the bars represent the different smoothing methods that were applied to the base model $P_u(w)$.

As can be observed from Figure 2, the users can be roughly categorized into three types:

- λ_u is low for all smoothing methods, *e.g*. Users 1, 2, and 4.
- λ_u is high for all smoothing methods, *e.g*. Users 5, 7, 11, and 13.
- λ_u is higher for the JM and Dirichlet smoothing, but is lower for the additive and absolute smoothing, *e.g*. Users 3, 6, and 9.

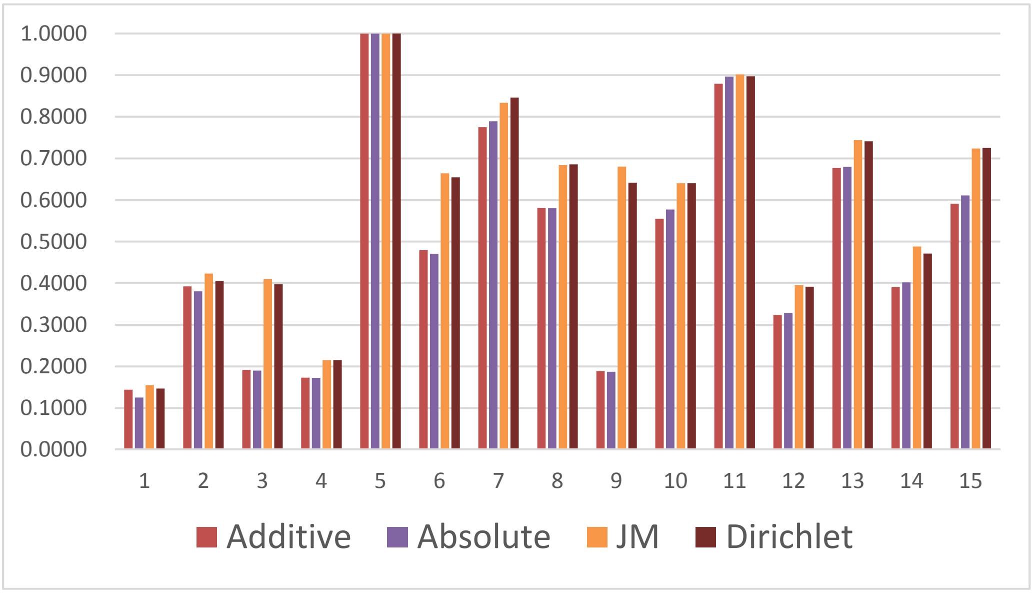


Figure 2. The mixture weight λ_u for each user.

We take Users 1, 5, and 6 as examples of the three types, respectively. Their feature weights are plotted in Figures 3, 4, and 5. It is worthwhile to mention that there is no user whose λ_u is lower for the JM and Dirichlet smoothing but is higher for the additive and absolute smoothing. This is consistent with the general idea that the JM and Dirichlet smoothing are better techniques than the other two.

Among all feature functions, f_{sim} generally has a higher feature weight w_{sim}. This is consistent with the experiments in Sections 3.3 and 3.4, where it has been shown that the cosine similarity may lead to an improved model performance. The weight w_{sim} has the highest value when λ_u is also high, as in Figure 4. This is reasonable because a higher λ_u signifies that the best model selected (with respect to the validation set perplexity, as described in Section 2.3) relies more on the base user language model $P_u(w)$. The similarity score computed from $P_u(w)$ should also be reliable.

By regarding each user as a sample, the correlation between λ_u and the feature weights can be computed. The results are shown in Table 4. There exists a relatively high positive correlation between λ_u and w_{sim}, as expected from the figures above. Also, for the pairwise features, the correlation coefficients are negative. This makes sense because, if the best model selected favors the base user LM more (*i.e.* a higher λ_u), then there should be no need to include the complicated pairwise features to enrich the language model. The negative correlation coefficients for w_{cmf} and w_{af} can be explained in the same way.

For some of the users whose λ_u is higher for JM and Dirichlet smoothing but lower for the other two, the feature weights exhibit a similar grouping. For User 6 (as shown in Figure 5), $w_{oov}, w_{pop}, w_{af}, w_{cat}$, and w_{rel} are lower if JM or Dirichlet smoothing is applied and higher if either of the other two is applied. This also explains the negative correlation between λ_u and w_{rel} (or w_{cat}).

Table 4. Correlation coefficient between λ_u and the feature weights.

Feature type	Local					Pairwise	
Feature weight	w_{sim}	w_{oov}	w_{pop}	w_{cmf}	w_{af}	w_{cat}	w_{rel}
ρ	0.3429	-0.0682	0.0112	-0.2316	-0.2166	-0.1583	-0.4920

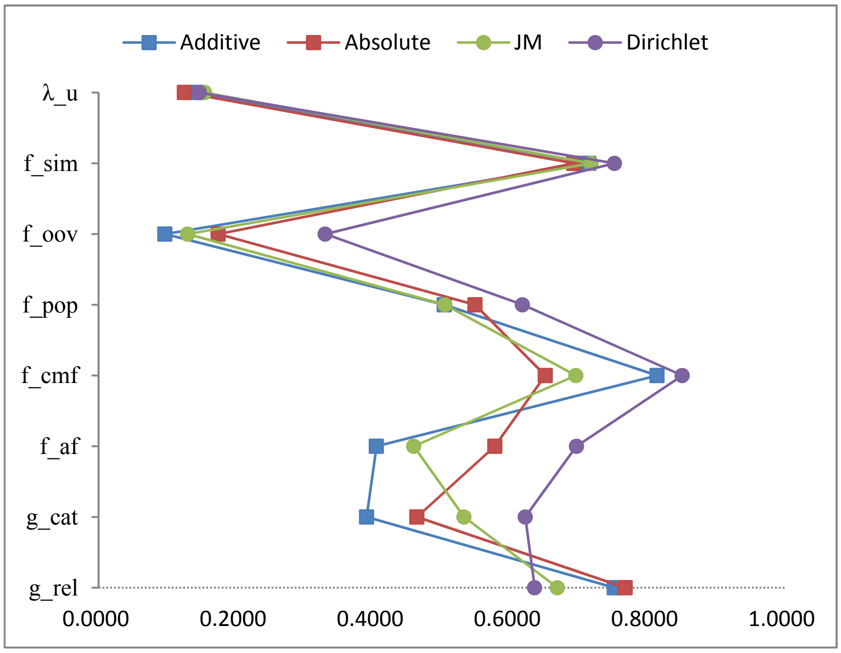


Figure 3. The feature weights for User 1. ($\lambda_u \approx 0.15$)

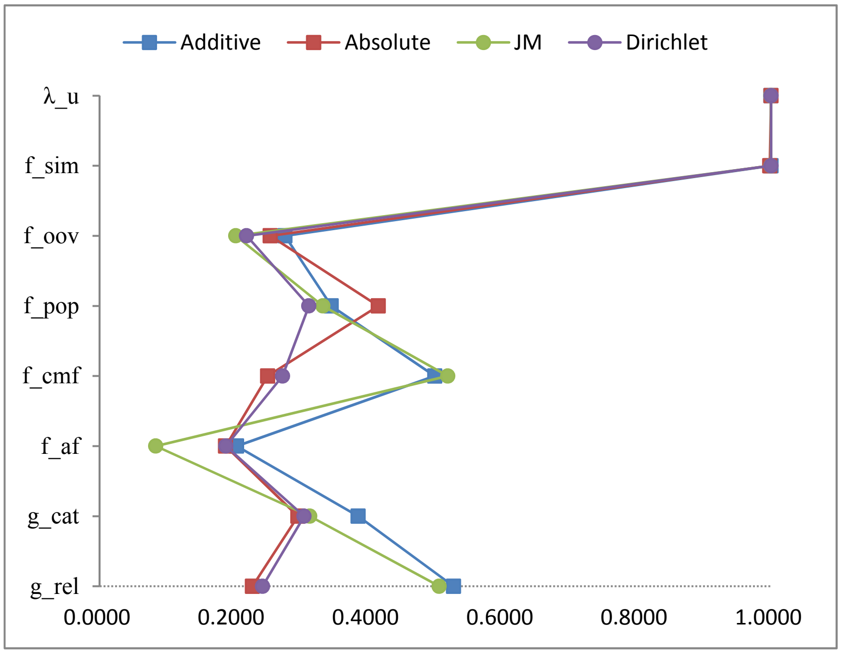


Figure 4. The feature weights for User 5. ($\lambda_u \approx 1$)

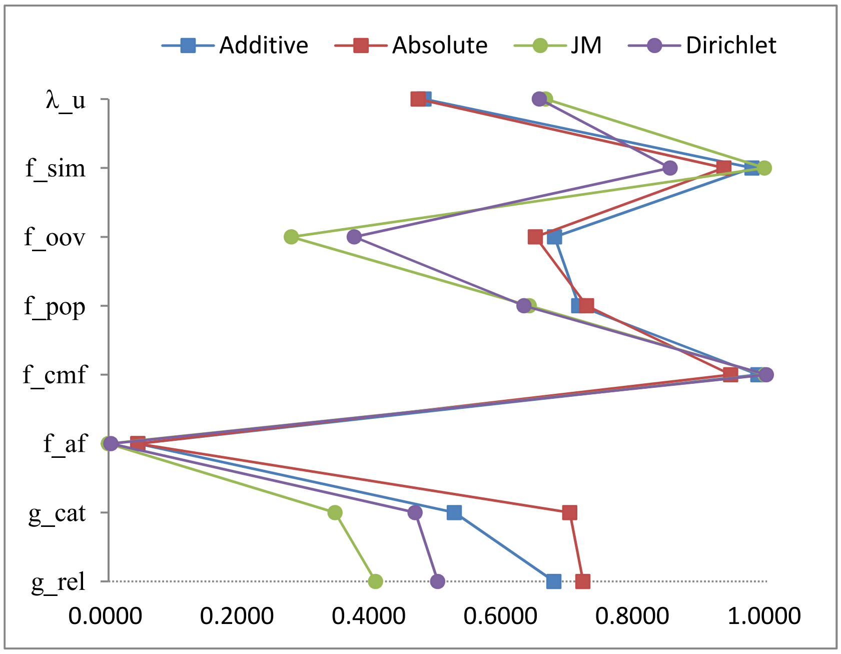


Figure 5. The feature weights for User 6. ($\lambda_u \approx 0.47$ for additive and absolute smoothing, and $\lambda_u \approx 0.65$ for JM and Dirichlet smoothing)

As has been mentioned, we want to demonstrate the usefulness of the pairwise feature functions (g_{rel} and g_{cat}) by comparing our model to the "*LR*" baseline. In fact, the usefulness of these feature functions can also be verified by observing that both w_{rel} and w_{cat} are greater than zero. By t-test at a significance level of 0.05, w_{cat} is significantly greater than zero for all users and all smoothing methods. Similarly, w_{rel} is significantly greater than zero for all except three of the ⟨user,smoothing method⟩ pairs (out of 60).

4. Related Work

Personalization has been studied for a long time in various textual related tasks. Personalized search is established by modeling user behavior when using search engines (Shen *et al.*, 2005; Xue *et al.*, 2009). A query language model could be also expanded based on personalized user modeling (Chirita *et al.*, 2007). Personalization has also been modeled in many NLP tasks, such as summarization (Yan *et al.*, 2011) and recommendation (Yan *et al.*, 2012). Different from our purpose, these models do not aim at exploiting social media content to enrich a language model. Wen *et al.* (2012, 2013) combines user-level language models from a social network, but instead of focusing on the cold start problem, they try to improve the speech recognition performance using a mass amount of texts on a social network. On the other hand, our work explicitly models the more sophisticated document-level relationships using a probabilistic graphical model.

5. Conclusion

The advantage of our model is threefold. First, prior knowledge and heuristics about the social network can be adapted in a structured way through the use of a factor graph model. Second, by exploiting a well-studied graphical model, mature inference techniques can be applied in the optimization procedure, making it much more effective and efficient. Finally, different from most smoothing methods, which are mutually exclusive, any other smoothing method can be incorporated into our framework to be further enhanced. Using only 1% of the training corpus, our model can improve the perplexity of base models by as much as 40% and the accuracy of authorship attribution by at most 13%.

Reference

Audet, C., & Dennis Jr, J. E. (2002). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, *13*(3), 889-903.

Chen, S. F., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics,* 310-318. Association for Computational Linguistics.

Chirita, P. A., Firan, C. S., & Nejdl, W. (2007). Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* 7-14. ACM.

Clements, M. (2007). Personalization of Social Media. In *Proceedings of the BCS IRSG Symposium: Future Directions in Information Access 2007*.

Galuba, W., Aberer, K., Chakraborty, D., Despotovic, Z., & Kellerer, W. (2010). Outtweeting the twitterers - predicting information cascades in microblogs. In *Proceedings of the 3rd conference on Online social networks,* 3-3. USENIX Association.

Jelinek, F. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 381-397.

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, *47*(2), 498-519.

Lin, J., Snow, R., & Morgan, W. (2011). Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* 422-429. ACM.

MacKay, D. J., & Peto, L. C. B. (1995). A hierarchical Dirichlet language model. *Natural language engineering*, *1*(03), 289-308.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 467-475. Morgan Kaufmann Publishers Inc..

Ney, H., Essen, U., & Kneser, R. (1995). On the estimation ofsmall'probabilities by leaving-one-out. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *17*(12), 1202-1212.

Peng, F., Schuurmans, D., & Wang, S. (2004). Augmenting naive bayes classifiers with statistical language models. *Information Retrieval*, *7*(3-4), 317-345.

Shen, X., Tan, B., & Zhai, C. (2005). Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management,* 824-831. ACM.

Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, *60*(3), 538-556.

Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., & Li, P. (2011). User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* 1397-1405. ACM.

Wang, Z., Li, J., Wang, Z., & Tang, J. (2012). Cross-lingual knowledge linking across wiki knowledge bases. In *Proceedings of the 21st international conference on World Wide Web,* 459-468. ACM.

Wen, T. H., Lee, H. Y., Chen, T. Y., & Lee, L. S. (2012). Personalized language modeling by crowd sourcing with social network data for voice access of cloud applications. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, 188-193.

Wen, T. H., Heidel, A., Lee, H. Y., Tsao, Y., & Lee, L. S. (2013). Recurrent neural network based language model personalization by social network crowdsourcing. In *INTERSPEECH*, 2703-2707.

Xue, G. R., Han, J., Yu, Y., & Yang, Q. (2009). User language model for collaborative personalized search. *ACM Transactions on Information Systems (TOIS)*, *27*(2), 11.

Yan, R., Nie, J. Y., & Li, X. (2011). Summarize what you are interested in: an optimization framework for interactive personalized summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1342-1351. Association for Computational Linguistics.

Yan, R., Lapata, M., & Li, X. (2012). Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers*-Volume 1, 516-525. Association for Computational Linguistics.

Zhai, C. (2008). Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, *1*(1), 1-141.

Zhai, C., & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 334-342. ACM.

Computational Linguistics and Chinese Language Processing
Vol. 21, No. 1, June 2016, pp. 69-90

Identifying the Names of Complex Search Tasks with Task-Related Entities

Ting-Xuan Wang*[*] and Wen-Hsiang Lu[*]

Abstract

Conventional search engines usually consider a search query corresponding only to a simple task. Nevertheless, due to the explosive growth of web usage in recent years, more and more queries are driven by complex tasks. A complex task may consist of multiple sub-tasks. To accomplish a complex task, users may need to obtain information of various task-related entities corresponding to the sub-tasks. Users usually have to issue a series of queries for each entity during searching a complex search task. For example, the complex task "travel to Beijing" may involve several task-related entities, such as "hotel room," "flight tickets," and "maps". Understanding complex tasks with task-related entities can allow a search engine to suggest integrated search results for each sub-task simultaneously. To understand and improve user behavior when searching a complex task, we propose an entity-driven complex task model (ECTM) based on exploiting microblogs and query logs. Experimental results show that our ECTM is effective in identifying the comprehensive task-related entities for a complex task and generates good quality complex task names based on the identified task-related entities.

Keywords: Complex Search Task, Task Name Identification, Task-related Entity.

1. Introduction

Conventional search engines usually consider single queries corresponding only to a simple search need. In reality, however, more and more queries are driven by complex search tasks (Guo & Agichtein, 2010; Jones & Klinkner, 2008). Generally, a real-life complex search task usually has more than one sub-task to be accomplished. Therefore, users usually cannot accomplish a complex search task by submitting only a single query. Some researchers have worked to try to identify sub-tasks in order to help users deal with complex search tasks (Tan

[*] Departmenr of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan
E-mail: playif@gmail.com; whlu@mail.ncku.edu.tw

et al., 2006; MacKay *et al*., 2008; Ji *et al*., 2011; Kotov *et al*., 2011; Yamamoto *et al*., 2012). Nevertheless, only identifying sub-tasks in a complex search task is not sufficient to help users who want to search the complex task name directly *e.g*., "北京旅遊 (travel to Beijing)". Understanding complex task names for complex search tasks can help search engines deal with complex-task-based queries. A complex search task can be represented by a task event and a task topic. For example, as shown in Figure 1, a complex task "travel to Beijing," which is composed of a task topic "Beijing" and a task event "travel," has at least three sub-tasks, including "book flight ticket," "reserve hotel room," and "survey maps". Users need to issue at least three queries for each sub-task including "Beijing flight ticket," "Beijing hotel," and "Beijing map". The queries targeting a sub-task usually focus on a task-related entity, such as "flight ticket," "hotel room," and "maps". Therefore, understanding task-related entities is very important for a complex task and can help search engines provide integrated search results containing a variety of information of distinct task-related entities.

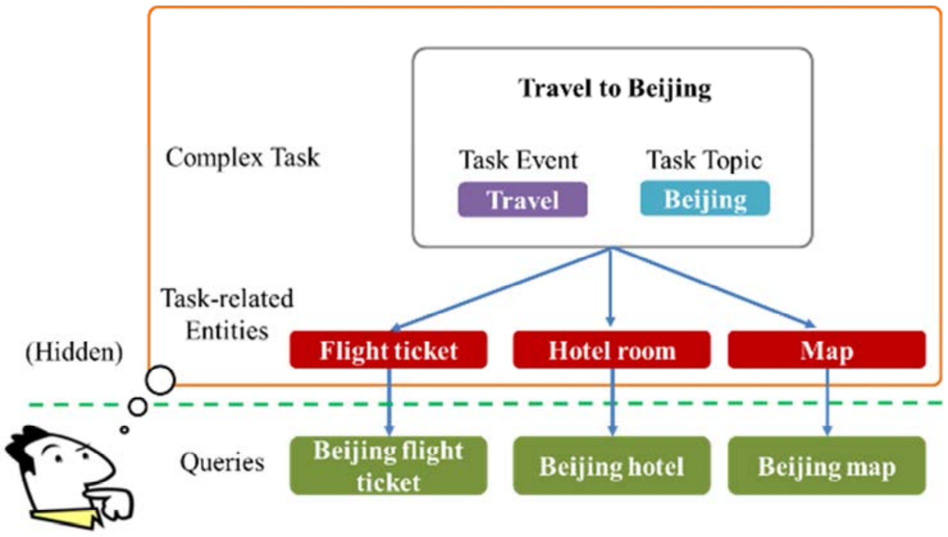


Figure 1. The structure of a complex task with task-related entities and search queries.

When users search for a complex task, we have found the users often have a task event that triggers the users to perform exploratory or comparative search behaviors, such as "prepare something," "buy something," or "travel somewhere". Furthermore, the search behaviors are usually around a certain task topic that is the subject of interest in the complex task. Users may describe the task event and task topic of their complex task with various task-related entities in microblogs, *e.g*., Twitter or Weibo[1]. Microblogs are a miniature version

[1] Weibo: http://weibo.com

of traditional weblogs. In recent years, many users have posted and shared their life details with others on microblogs every day. Due to the post length limitation (only 140 characters for Weibo), users tend to describe only key points of their life task. Table 1 shows an example of a microblog. We find the user, who has an ongoing complex task “北京旅遊 (travel to Beijing),” mentioned two task-related entities “機票 (flight ticket)” and “飯店 (hotel)”.

Table 1. A microblog post from Weibo mentioning an ongoing complex task “travel to Beijing”

Chinese	English Translation
今天已經訂好**機票**，只剩下找間**飯店**，就等著下禮拜去**北京旅遊**了~好期待!	I have already booked a **flight** today, and I only have to find a **hotel**. I’m about to **travel to Beijing** next week - good anticipation!

To understand and model a complex search task, some researchers have analyzed long and short-term user search behavior based on a single user’s search sessions (Agichtein *et al*., 2012; Liao *et al*., 2012; Mihalkova & Mooney, 2009; Tan *et al*., 2006). Nevertheless, a single user’s search session from query logs may not be sufficient in identifying complex search tasks since complex tasks may cross search sessions or be interleaved in a single search session. In this work, we address the problem of how to help users efficiently accomplish a complex task when submitting a single query or multiple queries. To complete the scenario illustrated in Figure 1, a complex task name with several task-related entities must be identified. Basically, the problem can be divided into the following three major sub-problems.

1. Collect queries related to the same complex search task.
2. Extract task-related entities from the collected queries.
3. Automatically identify the name of the complex search task.

The above three problems are very important but non-trivial to solve. We propose an entity-driven complex task model (ECTM) to automatically identify complex task names and task-related entities based on various web resources. To evaluate our proposed ECTM, we have conducted extensive experiments on a large dataset of real-world query logs. The experimental results show that our ECTM is able to identify a comprehensive task name for a complex task with related entities and generate good quality complex task names based on the identified task-related entities.

2. Related Work

Search session partition: Recent studies show that about 75% of search sessions are searching for complex tasks (Field & Allan, 2013). To help users deal with their search tasks, researchers have devoted effort to understand and identify tasks from search sessions. Boldi *et al.* (2008) proposed a graph-based approach to dividing a long-term search session into search tasks. Guo and Agichtein (2010) investigated the hierarchical structure of a search task with a series of search actions based on search sessions. Agichetin *et al*. (2012) conducted a comprehensive analysis of search tasks and classified them based on several aspects, such as intent, motivation, complexity, work-or-fun, time-sensitive, and continued-or-not. Beeferman and Berger (2000) proposed a graph-based iterative approach to clustering query logs. Wen *et al*. (2001) clustered similar queries based on query content and document clicks. Cui *et al*. (2011) grouped search queries from a click-through log with similar search tasks using the random walk method. Unfortunately, exploring only the query content and click-through information for query clustering may not obtain precise results since queries usually are short and ambiguous. Note that the above works only focus on single-goal task discovery, while our work focuses on identifying a complex task with relevant sub-tasks over a series of sessions.

Cross-session task prediction: Kotov *et al*. (2011) noticed that a complex task may require a user to issue a series of queries, spanning a long period of time and multiple search sessions. Thus, they addressed the problem of modeling and analyzing complex cross-session search tasks. Lucchese *et al*. (2011) tried to identify task-based sessions in query logs by semantic-based features extracted from Wiktionary and Wikipedia to overcome a lack of semantic information. Ji *et al*. (2011) proposed a graph-based regularization algorithm to predict popular search tasks and simultaneously classify queries and web pages by building two content-based classifiers. White *et al*. (2013) improved the traditional personalization methods for search-result re-ranking by exploiting similar tasks from other users to re-rank search results. Wang *et al*. (2011) addressed the problem of extracting cross-session tasks and proposed a task partition algorithm based on several pairwise similarity features. Raman *et al*. (2013) investigated intrinsic diversity (ID) for a search task and proposed a re-ranking algorithm according to the ID tasks.

Complex task search: To discriminate between simple and complex tasks, we define simple tasks as triggering only one sub-task. Complex tasks may trigger more than one sub-task. A complex task consists of several sub-tasks, and each sub-task goal may be composed of a sequence of search queries. Therefore, modeling the sub-tasks is necessary for identifying a complex task. Jones and Klinkner (2008) proposed a classification-based method to divide a single search session into tasks and sub-tasks based on the four types of features, including time, word, query log sequence, and web search. Lin *et al.* (2012) defined a search goal as an action-entity pair and utilized a web trigram to identify fine-grained search goals. Jones, Yamamoto, *et al*. (2012) proposed an approach to mining sub-tasks for a task using query clustering based on bid phrases provided by advertisers. The most important difference between our work and previous works is that we further try to identify task names with related task-intrinsic entities. To the best of our knowledge, there is no existing approach to utilizing microblogs in dealing with task identification and identifying human-interpretable names. In this work, we propose an entity-driven complex task model (ECTM) to deal with the problems mentioned above. To the best of our knowledge, there is no existing approach to utilizing multiple resources in dealing with task identification and identifying human-interpretable names for complex search tasks with various task-related entities.

3. Entity-driven Complex Task Model

3.1 System Architecture

To improve current search engines without support for complex task searches, we proposed an entity-driven complex task model (ECTM), which can automatically identify the name of a complex task and discover task-related entities behind the complex task. Figure 2 shows our ECTM architecture. The ECTM utilizes web resources, including query logs and microblogs. Given a search query that is driven by a latent complex task, there are three major stages in the ECTM to suggest integrated search results.

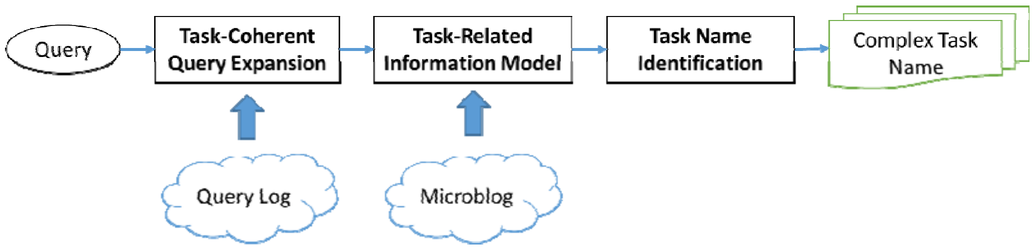


Figure 2. Architecture of our proposed entity-driven complex task model.

1. **Task-Coherent Query Expansion:** Integrate an expanded task-coherent query set to help identify the latent complex search task. In this stage, we utilize query logs and user search sessions to collect task-coherent queries.

2. **Task-Related Information Model:** Extract multiple task-related entities from a task-coherent query set, then retrieve microblog posts based on extracted task-related entities.

3. **Task Name Identification:** Identify the complex task name consisting of a task topic and a task event extracted from the retrieved microblog posts.

In the following, we describe the details of each major stage in the ECTM.

3.2 Task-Coherent Query Expansion

In fact, only using a single query is insufficient for finding the latent complex task. We thus try to extract other relevant task-coherent queries from search sessions. Although users may persistently search for the same complex task over a period of time, they may also simultaneously search for multiple interleaved tasks (Liu & Beklin, 2010; MacKay & Watters, 2008). Therefore, identifying task-coherent queries from search sessions is an important and non-trivial issue. The process for extracting task-coherent queries is described below.

Given an input query q and query logs Q, we first separate queries in the query logs into search sessions by the time gap of 24 hours. We extract search sessions containing the input query q and obtain a set of sessions S_q. To extract task-coherent queries Q_t from the session set S_q, we employ a log-linear model (LLM) with the following three useful features.

Average Query Frequency: Generally, the frequency of a query reflects its popularity. To avoid a long session resulting in high query frequency, we calculate the normalized query frequency as:

$$f_{AQFrequency}(q_t) = \frac{1}{|\boldsymbol{S}_{q_t}|} \times \sum_{s \in \boldsymbol{S}_{q_t}} \frac{freq(q_t, s)}{|s|} \tag{1}$$

where $freq(q_t,s)$ is the frequency of the query q_t in the session s, $\boldsymbol{S}_{q_t}$ is the sessions containing q_t, $|s|$ is the number of queries in the session s, and $|\boldsymbol{S}_{q_t}|$ is the number of sessions containing query q_t in the set $\boldsymbol{S}_{q_t}$.

Session Coverage: The queries occurring in several sessions are candidates in terms of task-coherence. To collect queries occurring in many sessions, we use average session frequency, which can be calculated as follows:

$$f_{ASFrequency}(q_t) = \exp\left(\frac{|\boldsymbol{S}_{q_t}|}{|\boldsymbol{S}_q|}\right) \tag{2}$$

where $|\boldsymbol{S}_q|$ is the number of sessions containing the input query q in the set $\boldsymbol{S}_q$, $|\boldsymbol{S}_{q_t}|$ is the number of sessions containing query q_t in the set $\boldsymbol{S}_{q_t}$, and $\exp(\cdot)$ is the exponential function.

Average Query Distance: Since queries that close to the input query in a search session may have high task-coherence degree for the latent complex task. We thus use normal distribution to estimate the task-coherence for each query:

$$f_{AQDistance}(q_t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma^2}} \tag{3}$$

where σ is standard deviation (which is set to 6.07, according to our training dataset, see Section 4.1.2), d is the average number of queries between q_t, and input query q in sessions.

We employ a log-linear model to calculate the probability of each candidate task-coherent query based on the features described above:

$$P(q_t; \boldsymbol{W}) = \frac{\exp(\sum_{i=1}^{|F|} w_i f_i(q_t))}{Z(\boldsymbol{Q}_t)} \tag{4}$$

where $\boldsymbol{Q}_t$ is the set of all candidate queries in the session set $\boldsymbol{S}_q$, $|F|$ is the number of used feature functions $f_i(q_t)$, $\boldsymbol{W}$ is the set of weighting parameters w_i of feature functions, and $Z(\boldsymbol{Q}_t)$ is a normalizing factor set to the value $Z(\boldsymbol{Q}_t) = \sum_{q_t \in \boldsymbol{Q}_t} \exp(\sum_{i=1}^{|F|} w_i f_i(q_t))$.

3.3 Task-Related Information Model

Sometimes, complex search task names will not occur in the expanded query set $\boldsymbol{Q}_t$; therefore, we cannot directly identify a complex task name from the query set $\boldsymbol{Q}_t$. In this stage, we expand the content of $\boldsymbol{Q}_t$ by extracting task-related entities and use the entities to retrieve microblog posts from a microblog search service, such as Weibo[2].

3.3.1 Task-Related Entity Extraction

We first try to extract task-related entities from expanded task-coherent query set $\boldsymbol{Q}_t$. In fact, the queries in the query set $\boldsymbol{Q}_t$ usually consist of a task topic and a task-related entity. For example, a query "北京機票 (Beijing flight ticket)" contains a topic "北京 (Beijing)" and an entity "機票 (flight ticket)". To realize the Part-Of-Speech (POS) of the task-related entity in the query set $\boldsymbol{Q}_t$, we generated statistics on 2000 queries randomly selected from a query log. The entities were labeled with a POS tag by a Chinese segmentation and tagging tool. Table 2 shows the results of the POS tag distribution of queries. We find that most entities are common nouns (87.5%), such as cellphone or flight ticket. For the POS of the task topic, we find that 78.9% of task topics are proper nouns and 19.8% are common nouns. Therefore, we extract task-related entities from the query set $\boldsymbol{Q}_t$ by extracting all common nouns in each set of queries and select the top-frequency proper noun as a candidate task topic. We thus can obtain a candidate task topic and a list of task-related entities $\boldsymbol{E}_t$ ordered by the occurrence frequency.

Table 2. The POS tag distribution of task entities and task topics

POS tag	Entity	Topic
Common Noun	87.5%	19.8%
Proper Noun	7.3%	78.9%
Others	5.2%	1.3%

3.3.2 Task-Related Microblog Retrieval

To identify a complex task name that does not occur in queries, the basic idea is to collect microblog posts from microblog search engines based on the given task-related entities. According to our observations, a microblog post containing most of the task-related entities may also contain the task name (see the example in Table 1). Unfortunately, a microblog post may contain only a portion of the entities required for a complex task. To overcome the above problem, we identify pseudo queries based on all subsets containing two or three entities from

[2] http://s.weibo.com/weibo/%E5%8C%97%E4%BA%AC%E6%97%85%E9%81%8A

top-*n* entities of $\boldsymbol{E}_t$. To make sure that each pseudo query is relevant to the candidate topic *t*, we combine the candidate topic *t* with each pseudo query and retrieve a set of microblog posts via microblog search engines.

3.4 Task Name Identification

Based on the identified task-related entities and microblog posts in the previous stage, we aim to identify a complex task name. To identify a suitable task name, we utilize conditional random field (CRF) (Lafferty *et al*., 2001) to automatically label each term in a microblog post with a task-semantic tag.

3.4.1 Automatically Labeling of Task Name

To realize the structure of a complex task name, we annotated 244 distinct complex task names from 513 search sessions (the details of the annotation process will be described in Section 4.1.2). Table 3 shows the statistics of the structure distribution. We found most task names consist of a task topic and a task event, such as “購買三星手機 (buy Samsung cellphone),” where “三星 (Samsung)” is the task topic, and “購買手機 (buy cellphone)” is the task event. We also find that the task event usually is composed of a transitive verb (*i.e.*, buy) and an event object (*i.e.*, cellphone). Nevertheless, some events are intransitive verbs needing no event object, such as “英語學習 (English learning),” where ”學習 (learning)” is an intransitive verb in Chinese. Therefore, we define the two types of events as Event 1 and Event 2, where Event 1 consist of a transitive verb ($\mathrm{E_{1V}}$) and an object ($\mathrm{E_{1O}}$), and Event 2 is only an intransitive verb. We aim to automatically label each term in a microblog post with one of the five task-semantic tags, *T* (topic), E_{1V} (Event 1), E_{1O} (Event object), E_2 (Event 2), and *O* (Others).

Table 3. The statistics of complex task name structure distribution

Task Name Pattern	Percentage	Example
Topic + Event 1	54.92%	購買三星手機 (Buy Samsung cellphone)
Topic + Event 2	40.16%	英語學習 (English learning)
Others	4.92%	--

To automatically label each term with a task-semantic tag, we employ a supervised probabilistic graphical model, conditional random field (CRF), which is suitable to predict the latent structure of sentences [10]. CRF can predict sequences of labels for sequences of terms in a sentence. We use a popular CRF implementation "CRF++,"[3] which can adopt multiple features for each term. In the following, we describe the two types of features for complex task name identification, including term-based and post-based features.

3.4.2 Features for Complex Task Name Identification

(1) Term-based features

There are five term-based features proposed in this work.

Stop word: A stop word usually is unimportant and not a task name. We consider stop word as a binary feature to indicate if the term is a task name.

Candidate topic: In the previous stage (see Section 3.3.1), we extracted a candidate topic from task-coherent query set $\boldsymbol{Q}_t$. Therefore, we can utilize the candidate topic as a binary feature for indicating if a term is a task topic.

Term frequency: Generally, a term in a post with high frequency may indicate the term is more important than other terms in the post. The term frequency is normalized by dividing the largest frequency of terms in the post. The normalized term frequency is divided into three ranges, including high [1, 0.8), middle (0.2, 0.8], and low [0, 0.2].

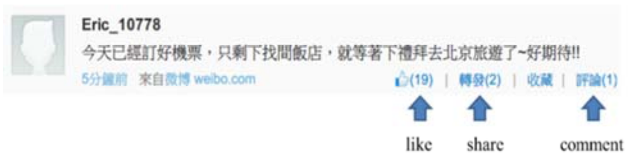
Document frequency: A term occurring in several search-result posts may indicate the term is a task topic or a task event. The reason is that the search-result posts usually are related to a certain complex task. We normalize the document frequency by dividing the post number of search results. The normalized document frequency is divided into three intervals, including high [1, 0.8), middle (0.2, 0.8], and low [0, 0.2].

POS tag: According to our observation, the POS tag of a task topic usually is a proper noun (N_p) or a common noun (N_c), and the POS tag of a task event usually is a transitive verb (V_t) + common noun (N_c) or an intransitive verb (V_i). To enhance the accuracy of the CRF model, we only use four types of POS tag "V_i," "V_t," "N_c," and "N_p" and others are labeled as "Others".

(2) Post-based Features

According to our observation, a post describing a complex task usually is more important or popular for an author or the author's friends. For example, when users write posts talking about their wedding, they will receive more attention than other ordinary posts. Therefore, we try to calculate a post importance score based on four post features, including descriptive, interactive, attractive, and influential degrees, according to the metadata of microblog posts. Figure 3 shows a real example of a microblog post collected from Weibo. We can see that there is some metadata on the post, such as the click times of "like," "share," and "comment".

[3] CRF++: http://crfpp.googlecode.com/svn/trunk/doc/index.html


Eric_10778

今天已經訂好機票，只剩下找間飯店，就等著下禮拜去北京旅遊了~好期待!!

5分鐘前 來自微博 weibo.com

Figure 3. A real example post containing various metadata, such as the times of "like," "share," and "comment"

Descriptive degree: Generally, a complex task needs more words to describe a variety of subtasks. Thus, we assume that a microblog post p with longer context can provide more content about the complex task. Nevertheless, some spam posts may contain long text with repeated terms. Therefore, we calculate the entropy to represent post descriptive degree:

$$F_{Descriptive}(p) = -\sum_{w \in Term(p)} P(w) \log_2 P(w) \tag{5}$$

where *Term*(p) is a set of terms in post p and $P(w)$ is the occurrence probability of term w.

Interactive degree: If a post has many "comments," we assume the post is more interactive. An interactive post has higher probability of mentioning a complex task. We formulate the interactive degree of a post as:

$$F_{Interactive}(p) = \frac{CommentCount(p)}{\max_{p_i \in P} CommentCount(p_i)} \tag{6}$$

where *CommentCount*(p) is the "comment" number of a post p and where $\max(\cdot)$ function returns the max "comment" number of a post p_i in the post set P.

Attractive degree: If a post receives many "likes," we assume the post is more attractive. An attractive post has higher probability to mention a complex task. We formulate the attraction of a post as:

$$F_{Attractive}(p) = \frac{LikeCount(p)}{\max_{p_i \in P} LikeCount(p_i)} \tag{7}$$

where *LikeCount*(p) is the "like" number of a post p and where $\max(\cdot)$ function returns the max "like" number of a post p_i in the post set P.

Influential degree: If a post was shared many times, we assume the post is more influential. An influential post has higher probability to mention a complex task. We formulate the influential degree of a post as:

$$F_{Influential}(p) = \frac{ShareCount(p)}{\max_{p_i \in P} ShareCount(p_i)} \tag{8}$$

where *ShareCount*(p) is the "share" number of a post p and where $\max(\cdot)$ function returns the max "share" number of a post p_i in the post set P.

Since the CRF model only accept features for terms (not for posts), we need to transform the post importance score to term importance score. The basic idea is that, if a term occurs often in more important posts, we can assume the term is important. Therefore, we calculate average term importance based on post importance as follows:

$$f_{TermImportance}(w) = \frac{\sum_{p_i \in P_w} PostImportance(p_i)}{|P_w|} \quad (9)$$

where *PostImportance*(p_i) can be replaced by one of the above four feature functions, P_w is a set of posts containing the term *w*, and $|P_w|$ is the number of posts in the set P_w. We further normalize $f_{TermImportance}(w)$ as follows:

$$f_{NormalizedTermImportance}(w) = \frac{f_{TermImportance}(w)}{\max_{w_j \in W} f_{TermImportance}(w_j)} \quad (10)$$

where ***W*** is the set of all terms. Finally, we divide the normalized term importance score into three ranges, high [1, 0.8), middle (0.2, 0.8], and low [0, 0.2].

3.4.3 Complex Task Name Composition

Based on the task-semantic tagging results of the CRF model, we can calculate the highest frequency task-semantic tagging terms, including e_{1V}, e_{1O}, e_2, and *t*, for each type of task-semantic tag E_{1V}, E_{1O}, E_2, and *T*, respectively. To compose a semantically suitable complex task name *c*, we use a rule-based algorithm that considers the frequency and POS of each task-semantic tagging term. Figure 4 shows the algorithm of complex task name composition (CTNC), which combines a task topic and a task event into a complex task name, where *Freq*(*e*) is the frequency of an event *e* and *POS*(*t*) is the POS tag of a topic *t*. We first compare the term frequency of a transitive event e_{1V} and an intransitive event e_2. If the frequency of e_2 is greater than e_{1V}, CTNC simply returns a complex task name composed of <*t*+e_2>, *e.g.*, "北京<*t*>旅遊<*e*2> (Beijing<*t*> travel<e_2>)". Otherwise, if the topic *t* is a common noun, CTNC returns a complex task name composed of <e_{1V}+*t*>, *e.g.*, "學習<e_{1V}>英語<*t*> (learn<e_{1V}> English<*t*>)". Otherwise, if the topic *t* is a proper noun, CTNC returns a complex task name composed of e_{1V}+*t*+e_{1O}, *e.g.*, "購買<e_{1V}>三星< *t* >手機<e_{1O}> (buy<e_{1V}> Samsung< *t* > cellphone<e_{1O}>)".

```
Algorithm: complex task name composition
Input: task-semantic tagging terms $e_{1V}$, $e_{1O}$, $e_2$, $t$
Output: A complex task name $c$
If $Freq(e_{1V}) < Freq(e_2)$ Then return $t + e_2$
Else If $POS(t)$ is "common noun" Then return $e_{1V} + t$
Else return $e_{1V} + t + e_{1O}$
```

Figure 4. The algorithm of complex task name composition

4. Experiments

4.1 Experimental Setup

4.1.1 Dataset

We use a month of query logs from the Sogou search engine as our dataset. The query logs contain 21,422,773 records with 3,163,170 distinct queries. We group these query records into search sessions according to user ID. Since a complex task may span a period of time, we used 24 hours as the time gap to segment search sessions, which resulted in 264,360 search sessions.

4.1.2 Data Labeling

In this work, we employed three annotators to label each query with a task-related entity and a latent complex task name. Since the query logs are diverse and often ambiguous, heuristically labeling the task-related entities and task names for each query may lead to inconsistent results. To identify reasonable and consistent training/testing data for evaluating our ECTM, a formal annotation method procedure should be provided. In the following, we describe the guidelines for annotators on how to label a task-related entity and a complex task name for each search query.

In general, a search query should give one entity that users focus on. Annotators thus only focus on queries containing exactly one entity and discard other queries.

To better interpret task-related entities and task names for queries, annotators are encouraged to exploit external resources, *e.g.*, clicked pages, search results for queries, or query context (*i.e.*, other queries in the same search session).

Since a complex task should be determined based on the whole search session, annotators should complete the labelling of all task-related entities in a search session before they begin to label task names.

From the 264,360 obtained sessions, we randomly labeled 5,142 sessions with task names and entities. For each task, we further examined the labeled results, and unified the similar task names annotated by different annotators. For instance, “北京旅遊 (travel to Beijing)” and “北京旅行 (trip to Beijing)” would be unified to “北京旅遊 (travel to Beijing)”. Table 4 shows an example search session of our labeled results. In fact, it is not easy to find a search session containing a good complex search task search intent. Each query belonging to a complex search task was labeled with a task-related entity. After excluding the tasks containing less than three entities and some controversial tasks, we obtained only 523 complex tasks from 513 sessions. We found that, although there were many interleaving simple tasks in one session, few complex tasks occurred in the same session. In other words, we found users seldom deal with two complex tasks simultaneously within the same period of time. The statistics of the labeled results are shown in Table 5. On average, there are 5.68 (2972 / 523) entities per task in our labeled dataset.

Table 4. An example search session with annotated task-related entities for each query. These search sessions obviously searched for the complex search task "結婚準備 (prepare for wedding)".

Chinese		English Translation	
Query	**Task-related Entity**	**Query**	**Task-related Entity**
結婚選購首飾	首飾	Wedding jewelry purchase	Jewelry
結婚首飾	首飾	Wedding jewelry	Jewelry
結婚禮服	禮服	Order wedding dress	Dress
結婚戒指展示	戒指	Wedding ring gallery	Ring
黃金戒指	戒指	Gold ring	Ring
購買黃金戒指	戒指	Buy gold ring	Ring

Table 5. The statistics of our labeled results for complex tasks

Data Type	Total Count	Distinct Count
Complex Task	523	244
Query	3,741	1,715

Training data: We sampled 100 complex tasks as the training dataset, which contained 724 queries and 424 distinct entities. For the log-linear model (LLM) used in task-coherent query expansion, we identified pairwise training data according to our labeled data set. Each query pair indicates if two queries belong to the same complex task. Therefore, we could obtain 4950 (*i.e.*, the combination number C_2^{100}) query pairs to train LLM. For the CRF model used in task name identification, we manually labeled 30 microblog posts retrieved for each complex task; thus, there were 3000 microblog posts for training the CRF model.

Testing data: We used the remaining 423 complex tasks as the testing dataset, which contained 3017 queries and 1426 distinct entities. For each testing complex task, we randomly selected a query from the testing task as the input query.

4.1.3 Task Name Quality Labeling

Since the automatically identified task name may be semantically the same as our annotated task name but represented in different lexicons, we cannot simply judge each identified task name by an automatic keyword matching approach. To overcome the above problem, we employed three judges to give scores independently for identified task names of all compared methods (the details of the compared methods will be described in Section 4.1.5). To ensure the fairness for all compared methods, we designed a labeling procedure.

1. Merge all identified task names from different methods into a large task name list.
2. Remove the duplicated task names in the task name list.
3. Shuffle the task names in order to hide the information of the original rank of different methods.

In order to give a relevance score, the judges could look at our pre-labeled task names and survey the information of the complex tasks. The score for each task name should be 0, 1, or 2. We define the criterion of giving a relevance score as follows.

Bad (score of 0): A bad complex task name is irrelevant to the complex task or is semantically unsuitable.

Fair (score of 1): A fair task name is semantically suitable but the judges cannot determine whether the task name can represent the complex task.

Good (score of 2): A good task name is semantically suitable and semantically the same as the pre-labeled task name.

Since there were three judges (thus, we had three relevance scores for each task), we needed to decide the final relevance scores for evaluating performance of the compared methods. We used the majority decision to decide the final relevance score for each task. For instance, when a name was labeled with relevance scores 1, 0, and 0, the final relevance score would be 0. If a task name was labeled with three distinct relevance scores 0, 1, and 2, the final relevance score would be 1. We only considered a task name with a relevance score of 2 as a correct task name.

4.1.4 Evaluation Metrics

Inclusion rate: The inclusion rate is to evaluate the fraction of the top *n* identified complex task names that include at least one correct complex task name. The equation of inclusion rate is given as follows:

$$InclusionRate = \frac{|inclusion(T)|}{|T|} \tag{11}$$

where $|T|$ is the number of testing tasks (*i.e.*, 423) and where $|inclusion(T)|$ is the number of testing tasks that can find at least one correct task name in top *n* identified complex task names.

Mean reciprocal rank (MRR): Since we only need one correct task name for each testing data, we use MRR, which only considers the rank of the first returned correct task name for each testing data task. To calculate the MRR, the equation is as follows:

$$MRR = \frac{1}{|T|}\sum_{i=1}^{|T|}\frac{1}{rank(i)} \tag{12}$$

where $|T|$ is the number of testing tasks and $rank(i)$ is the rank of the first identified correct task name of the i_{th} testing task.

Normalized discounted cumulative gain (NDCG): To realize the overall quality of the top *n* identified task name, we also use *NDCG* to evaluate the performance for different methods. According to scores of identified task name, we first have to calculate the *DCG* as follows:

$$DCG = score_1 + \sum_{i=2}^{n}\frac{score_i}{\log_2(i)} \tag{13}$$

where *n* is the number of identified task names and $score_i \in \{0,1,2\}$ is the relevance score of the top-i task name. In order to normalize the *DCG* value from 0 to 1, the DCG divided by *IDCG*, called *NDCG*, is defined as follows:

$$NDCG = \frac{DCG_n}{IDCG_n} \tag{14}$$

where *IDCG* can be calculated the same as *DCG* with an ideal rank, which was ranked by labeled scores.

4.1.5 Task Name Identification Method Comparison

LRM_SERP (linear regression model with search engine result snippet): This method was proposed by Zeng *et al*. (2004) to identify salient phrases from search-result snippets. Salient phrases are identified using several regression models with five proposed features, including TFIDF, phrase length, intra-cluster similarity, cluster entropy, and phrase independence. We used the linear regression model (LRM) with the five features proposed in their work to extract salient phrases as task names from search-result snippets. Since using only the testing input query to collect search-result snippets may not be fair for identifying complex task names, we used our produced pseudo queries with task-related information model to collect search-result snippets from a web search engine (see Section 3.3). The weight for each feature was set as in Zeng *et al*.'s work.

LRM_MB (linear regression model with microblog): We also tried to adopt a linear regression model with microblog posts in order to compare the resources of SERP and microblog based on the five features proposed in Zeng *et al*.

LRM_MB+ (linear regression model with microblog plus): This was used to compare the performance of the linear regression model (LRM) and with our proposed microblog features with quantified values in LRM_MB.

ECTM (entity-driven complex task model): This is our proposed entity-driven complex task model, which utilizes microblog as extending data for a task-coherent query set. We used all features proposed in this work for training a CRF model to identify the name of complex task. The only difference between LRM_MB+ and our method is that the former first extracts bigrams and trigrams from posts as candidate phrases before using LRM to determine their correctness, and our method uses CRF to directly label each term in the posts with a task-semantic tag (*i.e.*, topic or event).

4.1.6 Parameter Selection

For each testing task, we need to determine the number of the top *n* selected task-related entities to produce pseudo queries for retrieving microblog posts. Since we use all subsets containing two or three entities of the top *n* entity set to produce pseudo queries, the entity number is critical to the number of produced pseudo queries (*i.e.*, the total number of pseudo queries is C_2^n plus C_3^n, where C_k^n is the *k*-combination of the entity set containing *n* entities), also making it critical to computational time. To realize how many entities should be selected for each testing task to achieve the best performance of identifying a complex task name, we randomly selected 15 complex tasks from the testing dataset to make the preliminary experiment. For each pseudo query, we retrieved the top 20 microblog posts and search-result snippets via a microblog search engine and a web search engine, respectively. We calculated the average top 3 inclusion rate for each testing task of LRM_SERP and LRM_MB. Figure 5 shows the different number of entities for producing pseudo queries along with the average top 3 inclusion rate. We can see both LRM_SERP and LRM_MB achieved better inclusion rates when using five entities to compose 20 (*i.e.*, $C_2^5 + C_3^5$) pseudo queries. Generally, a small number of entities achieved a worse inclusion rate since the retrieved microblog posts or search-result snippets for a complex task were insufficient. When the number of selected entities was greater than five, the number of unrelated entities also increased, resulting in a worse inclusion rate. To understand the correct number of microblog posts that should be used in our training data, we also conducted a preliminary experiment. Figure 6 shows that, when we used 30 posts, our ECTM could achieve the best precision. Therefore, in the following experiments, we used the top five entities for all methods when identifying pseudo queries.

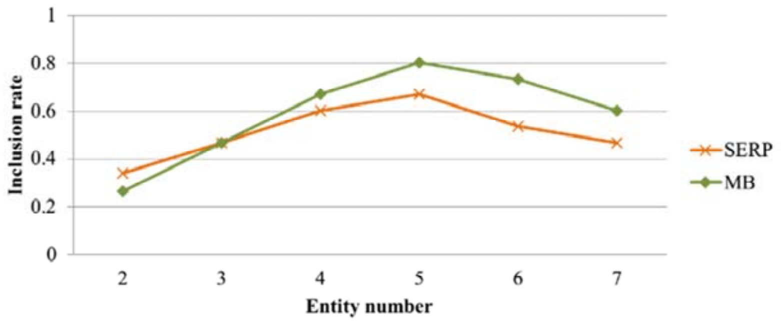


Figure 5. The top 3 task name inclusion rate of different number of entities for producing pseudo queries

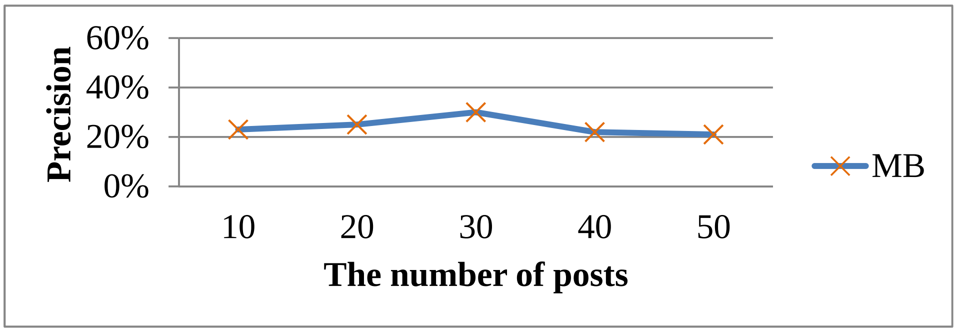


Figure 6. The precision for different number of microblog posts (MB) in complex task name identification.

4.2 Results of Task Name Identification

Table 6 shows the overall results of task name identification using different methods. Generally, the four methods achieved adequate top 5 inclusion rate (0.68, 0.72, 0.79, and 0.83 for LRM_SERP, LRM_MB, LRM_MB+, and ECTM, respectively). We found LRM_MB using a microblog is better than LRM_SERP using an SERP in identifying the complex task name. According to our analysis, microblog posts contain more task names, even in tail posts. On the contrary, only a few top-ranked search result snippets contain complex task names. The reason is that microblog posts are identified by users; thus, they have more likelihood to talk about a real-life complex task. Therefore, LRM_MB can achieve better performance than LRM_SERP. We also compared the performance between using and not using our proposed features in a microblog post (*i.e.*, LRM_MB and LRM_MB+). Using microblog features can slightly improve the overall performance since important posts usually mention a complex task. Our proposed ECTM using CRF to automatically label task-semantic tags for each term can improve the performance significantly, and it achieved Top-1 MRR of 0.57.

To realize whether the identified complex task names are semantically suitable, some example of top-1 identified task names are shown in Table 7. For the query “北京機票 (Beijing flight ticket),” only our ECTM identified correct task name “北京旅遊 (Beijing travel)”. LRM_SERP identified incorrect task name “旅行社旅行 (travel agency travel),” which is not semantically suitable. The reason is that search-result snippets sometimes are not represented as complete natural language sentences. Therefore, when we try to extract a bigram or trigram, there are some combinations that are not semantically suitable. LRM_MB and LRM_MB+ identified incorrect task names “訂購自由行 (reserve independent travel)” and “國外旅遊網訂 (online ordering to travel in foreign countries),” which are semantically suitable but not very related with Beijing travel. The reason is that the above two methods do not consider the task topic when identifying complex task names. According to our observation, the task topic of a complex task usually occurs in almost every search query. For the testing task query “深圳會計待遇 (Shenzhen accounting salary),” only LRM_MB+ identified the correct task name “申請會計工作 (apply for accounting work),” and our ECTM identified an incorrect task name “申請深圳工作 (apply for Shenzhen work)”. According to our analysis, ECTM identified an incorrect task topic “深圳 (Shenzhen),” which is a location name in China and occurs many times in a task-coherent query set (see Section 3.2). Nevertheless, the correct task topic should be “會計 (accounting)”, which specifies the career searched by the user. As a result, we find our ECTM may identify incorrect task names when the task topic is not the most frequent term in the task-coherent query set.

Table 6. The overall complex task identification performance comparison of four methods

Method	**MRR**			**NDCG**			**IR**		
	Top 1	**Top 3**	**Top 5**	**Top 1**	**Top 3**	**Top 5**	**Top 1**	**Top 3**	**Top 5**
LRM_SERP	0.37	0.41	0.47	0.22	0.35	0.31	0.37	0.45	0.68
LRM_MB	0.42	0.52	0.54	0.27	0.46	0.42	0.42	0.65	0.72
LRM_MB+	0.48	0.54	0.59	0.39	0.54	0.53	0.48	0.68	0.79
ECTM	**0.57**	**0.63**	**0.66**	**0.45**	**0.61**	**0.58**	**0.57**	**0.71**	**0.83**

Table 7. The example of top-1 complex task names generated by four compared methods, where * sign indicates the generated complex task name is incorrect

Testing task query	**LRM_SERP**	**LRM_MB**	**LRM_MB+**	**ECTM**
北京機票 (Beijing flight ticket)	*旅行社旅行 (Travel agency travel)	*訂購自由行 (Reserve independent travel)	*國外旅遊網訂 (Online ordering to travel in foreign countries	**北京旅遊 (Beijing travel)**
手機最新報價 (This newest prices of cellphones)	*中華門號續約 (Renewal cellphone number of Chunghwa)	*組裝電腦 (DIY computer)	*購買配件 (Buy accessories)	**購買手機 (Buy cellphones)**
深圳會計待遇 (Shenzhen accounting salary)	*考到深圳車牌 (Get Shenzhen license plated)	*深圳會計兼職 (Shenzhen part-time job of account)	**申請會計工作 (Apply for Shenzhen work)**	*申請深圳工作 (Apply for accounting work)

5. Discussion

For our proposed ECTM, we discuss two issues of producing pseudo queries with a candidate topic and the limitations of task name composition in the following.

(1) Producing pseudo queries with a candidate topic: In general, pseudo queries containing a candidate topic can achieve better precision when retrieving microblog posts. If the pseudo queries do not contain a topic, it is hard to find the correct topic for the task name. For example, two tasks "北京旅遊 (travel to Beijing)" and "青島旅遊 (travel to Tsingtao)" may have many of the same task-related entities, such as "機票 (flight tickets)," "飯店 (hotel)," and "地圖 (maps)". Therefore, all of the microblog posts containing these entities may be retrieved and result in our ECTM identify an incorrect task name. Nevertheless, when the identified candidate topic is incorrect, our task name identification model usually is unable to identify a correct task name. How to improve the precision of identifying a task topic is still an important issue.

(2) The limitations of task name composition: Our proposed ECTM can identify task names composed of correct task-semantic tags effectively. Nevertheless, some identified task names that contain an event object may have semantic flaws. For instance, we identified a complex task name "治療北京打鼾 (treat Beijing snore)," which is composed of an event "治療 (treat)," a topic "北京 (Beijing)," and an event object "打鼾 (snore)," based on our definition of task name composition strategy, which considers only POS patterns. Actually, the more semantically suitable task name is "北京治療打鼾 (Beijing treat snore)" that is composed of a topic “北京 (Beijing),” an event “治療 (treat),” and an event object “打鼾 (snore)”. In this work, we still cannot provide a perfect solution for composing task names.

6. Conclusion and Future Work

In this work, we proposed an entity-driven complex task model (ECTM), which addressed the problem of improving the user experience when searching for a complex task. We exploited various web resources, including query logs, microblogs, and search-result snippets, to enhance the performance of our ECTM. To identify a human-interpretable complex task name from short-content queries, we utilized microblog posts and investigated several useful features to train the CRF model to automatically identify complex task names. Experimental results show that ECTM efficiently identifies complex task names with various task-related entities. Nevertheless, there are still some problems that need to be solved. In the future, we will try to investigate other useful features to improve the task name identification when dealing with real-life complex task queries.

References

Agichtein, E., White, R. W., Dumais, S. T., & Bennett, P. N. (2012). Search, Interrupted: Understanding and Predicting Search Task Continuation. In *Proc. of SIGIR 2012*.

Beeferman, D. & Berger, A. (2000). Agglomerative Clustering of a Search Engine Query log. In *Proc. of KDD ‘00*, 407-416.

Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., & Vigna, S. (2008). The Query-Flow Graph: Model and Applications. In *Proc. of CIKM 2008.*

Cui, J., Liu, H., Yan, J., Ji L., Jin R., He, J., Gu, Y., Chen, Z., & Du, X. (2011). Multi-view Random Walk Framework for Search Task Discovery from Click-through Log. In *Proc. of CIKM 2011*.

Feild, H. & Allan, J. (2013). Task-Aware Query Recommendation. In *Proc. of SIGIR 2013*.

Guo, Q. & Agichtein, E. (2010). Ready to Buy or Just Browsing? Detecting Web Searcher Goals from Interaction Data. In *Proc. of SIGIR 2010*.

Ji, M., Yan, J., Gu, S., Han, J., He, X., Zhang, W. V., & Chen, Z. (2011). Learning Search Tasks in Queries and Web Pages via Graph Regularization. In *Proc. of SIGIR 2011*.

Jones, R., & Klinkner, K. (2008). Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs. In *Proc. of CIKM ‘08*, 699-708.

Kotov, A., Bennett, P. N., White, R. W., Dumais, S. T., & Teevan, J. (2011). Modeling and Analysis of Cross-Session Search Tasks. In *Proc. of SIGIR 2011*.

Lafferty, J., Mccallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML*, 282-289.

Liao, Z., Song, Y., He, L.-W., & Huang, Y. (2012). Evaluating the Effectiveness of Search Task Trails. In *Proc. of WWW 2012*.

Lin, T., Pantel, P., Gamon, M., Kannan, A., & Fuxman, A. (2012). Active Objects: Actions for Entity-Centric Search. In *Proc. of WWW 2012*.

Liu, J & Belkin, N. J. (2010). Personalizing Information Retrieval for Multi-Session Tasks: The Roles of Task Stage and Task Type. In *Proc. of SIGIR 2010*.

Lucchese, C., Orlando, S., Perego, R., Silvestri, F., & Tolomei, G. (2011). Identifying Task-based Sessions in Search Engine Query Logs. In *Proc. of WSDM 2011*.

MacKay, B. & Watters, C. (2008). Exploring Multi-Session Web Tasks. In *Proc. of CHI '08*, 1187-1196.

Mihalkova, L. & Mooney, R. (2009). Learning to Disambiguate Search Queries form Short Sessions. In *Proc. of ECML '09*, 111-127.

Raman, K., Bennett, P. N., & Collins-Thompson, K. (2013). Toward Whole-Session Relevance: Exploring Intrinsic Diversity in Web Search. In *Proc. of SIGIR 2013*.

Tan, B., Shen, X., & Zhai, C. (2006). Mining Long-Term Search History to Improve Search Accuracy. In *Proc. of KDD '06*, 718-723.

Wang, T.-X., & Lu, W.-S. (2011). Identifying Popular Search Goals behind Search Queries to Improve Web Search Ranking. In *Proc. of AIRS 2011*.

Wen, J.-R., Nie, J.-Y., & Zhang, H.-J. (2001). Clustering User Queries of Search Engine. In *Proc. of WWW 2001*.

White, R. W., Chu, W., Hassan, A., He, X., Song, Y., & Wang, H. (2013). Enhancing Personalized Search by Mining and Modeling Task Behavior. In *Proc. of WWW 2013*.

Yamamoto, T., Sakai, T., Iwata, M., Yu, C., Wen, J.-R., & Tanaka, K. (2012). The Wisdom of Advertisers: Mining Subgoals via Query Clustering. In *Proc. of CIKM 2012*.

Zeng, H.-J., He, Q.-C., Chen, Z., Ma, W.-Y., & Ma, J. (2004). Learning to Cluster Web Search Results. In *Proc. of SIGIR 2004*.

The Association for Computational Linguistics and Chinese Language Processing

(new members are welcomed)

Aims：

1. To conduct research in computational linguistics.
2. To promote the utilization and development of computational linguistics.
3. To encourage research in and development of the field of Chinese computational linguistics both domestically and internationally.
4. To maintain contact with international groups who have similar goals and to cultivate academic exchange.

Activities：

1. Holding the Republic of China Computational Linguistics Conference (ROCLING) annually.
2. Facilitating and promoting academic research, seminars, training, discussions, comparative evaluations and other activities related to computational linguistics.
3. Collecting information and materials on recent developments in the field of computational linguistics, domestically and internationally.
4. Publishing pertinent journals, proceedings and newsletters.
5. Setting of the Chinese-language technical terminology and symbols related to computational linguistics.
6. Maintaining contact with international computational linguistics academic organizations.
7. Dealing with various other matters related to the development of computational linguistics.

To Register：

Please send application to:

The Association for Computational Linguistics and Chinese Language Processing
Institute of Information Science, Academia Sinica
128, Sec. 2, Academy Rd., Nankang, Taipei 11529, Taiwan, R.O.C.

payment： Credit cards(please fill in the order form), cheque, or money orders.

Annual Fees：

regular/overseas member：	NT$ 1,000 (US$50.-)
group membership：	NT$20,000 (US$1,000.-)

life member：ten times the annual fee for regular/ group/ overseas members

Contact：

Address： The Association for Computational Linguistics and Chinese Language Processing
Institute of Information Science, Academia Sinica
128, Sec. 2, Academy Rd., Nankang, Taipei 11529, Taiwan, R.O.C.

Tel.：886-2-2788-3799 ext. 1502 Fax：886-2-2788-1638

E-mail: aclclp@hp.iis.sinica.edu.tw Web Site: http://www.aclclp.org.tw

Please address all correspondence to Miss Qi Huang, or Miss Abby Ho

The Association for Computational Linguistics and Chinese Language Processing

Membership Application Form

Member ID#：________________

Name：__________________________________ Date of Birth：_______________

Country of Residence：____________________ Province/State：____________________

Passport No.：___________________________ Sex: ________________________

Education(highest degree obtained)：____

Work Experience：____

____

Present Occupation：____

Address：____

____

Email Add：____

Tel. No：____________________________ Fax No：__________________________

Membership Category：☐ Regular Member ☐ Life Member

Date： _____/____/____（Y-M-D）

Applicant's Signature：

Remarks： Please indicated clearly in which membership category you wish to register, according to the following scale of annual membership dues：

Regular Member	：	US$ 50.- （NT$ 1,000）
Life Member	：	US$500.-（NT$10,000）

Please feel free to make copies of this application for others to use.

Committee Assessment：

中華民國計算語言學學會

宗旨：

（一） 從事計算語言學之研究
（二） 推行計算語言學之應用與發展
（三） 促進國內外中文計算語言學之研究與發展
（四） 聯繫國際有關組織並推動學術交流

活動項目：

（一）定期舉辦中華民國計算語言學學術會議（Rocling）

（二）舉行有關計算語言學之學術研究講習、訓練、討論、觀摩等活動項目

（三）收集國內外有關計算語言學知識之圖書及最新發展之資料

（四）發行有關之學術刊物，論文集及通訊

（五）研定有關計算語言學專用名稱術語及符號

（六）與國際計算語言學學術機構聯繫交流

（七）其他有關計算語言發展事項

報名方式：

1. 入會申請書：請至本會網頁下載入會申請表，填妥後郵寄或E-mail至本會

2. 繳交會費：劃撥：帳號：19166251，戶名：中華民國計算語言學學會
信用卡：請至本會網頁下載信用卡付款單

年費：

終身會員：	10,000.-	（US$ 500.-）
個人會員：	1,000.-	（US$ 50.-）
學生會員：	500.-	（限國內學生）
團體會員：	20,000.-	（US$ 1,000.-）

連絡處：

地址：台北市115南港區研究院路二段128號 中研院資訊所(轉)
電話：(02) 2788-3799 ext.1502 傳真：(02) 2788-1638
E-mail：aclclp@hp.iis.sinica.edu.tw 網址: http://www.aclclp.org.tw
連絡人：黃琪 小姐、何婉如 小姐

中 華 民 國 計 算 語 言 學 學 會
個 人 會 員 入 會 申 請 書

會員類別	☐終身 ☐個人 ☐學生	會員編號	（由本會填寫）		
姓　　名		性別		出生日期	年　月　日
				身分證號碼	
現　　職		學　　歷			
通訊地址	☐☐☐				
戶籍地址	☐☐☐				
電　　話		E-Mail			
申請人:　　　　　　（簽章） 中　華　民　國　　年　　月　　日					

審查結果:

1. 年費：

   終身會員：　10,000.-
   個人會員：　1,000.-
   學生會員：　500.-（限國內學生）
   團體會員：　20,000.-

2. 連絡處：

   地址：台北市南港區研究院路二段128號 中研院資訊所(轉)
   電話：(02) 2788-3799　ext.1502 傳真：(02) 2788-1638
   E-mail：aclclp@hp.iis.sinica.edu.tw　　網址: http://www.aclclp.org.tw
   連絡人：黃琪 小姐、何婉如 小姐

3. 本表可自行影印

The Association for Computational Linguistics and Chinese Language Processing (ACLCLP) PAYMENT FORM

Name: ___________________________________(Please print) Date: _______________

Please debit my credit card as follows: US$ _____________

❑ VISA CARD ❑ MASTER CARD ❑ JCB CARD Issue Bank:__________________

Card No.: ________ -_________-________ -_________ Exp. Date:____________(M/Y)

3-digit code: _________ (on the back card, inside the signature area, the last three digits)

CARD HOLDER SIGNATURE: ___________________________________

Phone No.: _________________________E-mail: ____________________________________

Address: ____

PAYMENT FOR

US$ _____ ❑ Computational Linguistics & Chinese Languages Processing (IJCLCLP)

Quantity Wanted: ______

US$ _____ ❑ Journal of Information Science and Engineering (JISE)

Quantity Wanted: ______

US$ _____ ❑ Publications:________________________________

US$ _____ ❑ Text Corpora: ______________________________

US$ _____ ❑ Speech Corpora:_____________________________

US$ _____ ❑ Others: __________________________________

US$ _________ ❑ Membership Fees ❑ Life Membership ❑ New Membership ❑Renew

US$ _________ = Total

Fax 886-2-2788-1638 or Mail this form to:

ACLCLP
℅ IIS, Academia Sinica
Rm502, No.128, Sec.2, Academia Rd., Nankang, Taipei 115, Taiwan

E-mail: aclclp@hp.iis.sinica.edu.tw
Website: http://www.aclclp.org.tw

中 華 民 國 計 算 語 言 學 學 會
信用卡付款單

姓名: ____________________________(請以正楷書寫)　　日期:：____________________

卡別：❑ VISA CARD　　❑ MASTER CARD ❑ JCB CARD　　發卡銀行：______________

信用卡號：__________-__________-__________-__________　　有效日期：__________(m/y)

卡片後三碼：______________（卡片背面簽名欄上數字後三碼）

持卡人簽名：_______________________________________(簽名方式請與信用卡背面相同)

通訊地址：____

聯絡電話：____________________________E-mail：_________________________________

備註：為順利取得信用卡授權，請提供與發卡銀行相同之聯絡資料。

付款內容及金額：

NT$__________ ❑ 中文計算語言學期刊(IJCLCLP) _______________

NT$__________ ❑ Journal of Information Science and Engineering (JISE)

NT$__________ ❑ 中研院詞庫小組技術報告_____________________

NT$__________ ❑ 文字語料庫 ____________________________________

NT$__________ ❑ 語音資料庫 ____________________________________

NT$__________ ❑ 光華雜誌語料庫1976~2010

NT$__________ ❑ 中文資訊檢索標竿測試集/文件集

NT$__________ ❑ 會員年費：❑續會　　　❑新會員　　　❑終身會員

NT$__________ ❑ 其他: _____________________________________

NT$_______________ = 合計

填妥後請傳真至 02-27881638 或郵寄至:
11529台北市南港區研究院路2段128號中研院資訊所(轉)中華民國計算語言學學會 收
E-mail: aclclp@hp.iis.sinica.edu.tw
Website: http://www.aclclp.org.tw

Publications of the Association for Computational Linguistics and Chinese Language Processing

		Surface	AIR (US&EURP)	AIR (ASIA)	VOLUME	AMOUNT
1.	no.92-01, no. 92-04(合訂本) ICG 中的論旨角色與 A Conceptual Structure for Parsing Mandarin -- Its Frame and General Applications--	US$ 9	US$ 19	US$15	_____	_____
2.	no.92-02 V-N 複合名詞討論篇 & 92-03 V-R 複合動詞討論篇	12	21	17	_____	_____
3.	no.93-01 新聞語料庫字頻統計表	8	13	11	_____	_____
4.	no.93-02 新聞語料庫詞頻統計表	18	30	24	_____	_____
5.	no.93-03 新聞常用動詞詞頻與分類	10	15	13	_____	_____
6.	no.93-05 中文詞類分析	10	15	13	_____	_____
7.	no.93-06 現代漢語中的法相詞	5	10	8	_____	_____
8.	no.94-01 中文書面語頻率詞典(新聞語料詞頻統計)	18	30	24	_____	_____
9.	no.94-02 古漢語字頻表	11	16	14	_____	_____
10.	no.95-01 注音檢索現代漢語字頻表	8	13	10	_____	_____
11.	no.95-02/98-04 中央研究院平衡語料庫的內容與說明	3	8	6	_____	_____
12.	no.95-03 訊息為本的格位語法與其剖析方法	3	8	6	_____	_____
13.	no.96-01 「搜」文解字－中文詞界研究與資訊用分詞標準	8	13	11	_____	_____
14.	no.97-01 古漢語詞頻表(甲)	19	31	25	_____	_____
15.	no.97-02 論語詞頻表	9	14	12	_____	_____
16.	no.98-01 詞頻詞典	18	30	26	_____	_____
17.	no.98-02 Accumulated Word Frequency in CKIP Corpus	15	25	21	_____	_____
18.	no.98-03 自然語言處理及計算語言學相關術語中英對譯表	4	9	7	_____	_____
19.	no.02-01 現代漢語口語對話語料庫標註系統說明	8	13	11	_____	_____
20.	Computational Linguistics & Chinese Languages Processing (One year) (Back issues of *IJCLCLP*: US$ 20 per copy)	---	100	100	_____	_____
21.	Readings in Chinese Language Processing	25	25	21	_____	_____
				TOTAL	_____	_____

10% member discount: ___________Total Due:__________

- **OVERSEAS USE ONLY**
- PAYMENT： ☐ Credit Card (Preferred)
  ☐ Money Order or Check payable to "The Association for Computation Linguistics and Chinese Language Processing " or “中華民國計算語言學學會”
- E-mail：aclclp@hp.iis.sinica.edu.tw

Name (please print): ____________________________ Signature: ____________________________

Fax: ____ E-mail: ______________________________

Address：____

中華民國計算語言學學會
相關出版品價格表及訂購單

編號	書目	會　員	非會員	冊數	金額
1.	no.92-01, no. 92-04 (合訂本)　ICG 中的論旨角色 與 A conceptual Structure for Parsing Mandarin--its Frame and General Applications--	NT$ 80	NT$ 100	_____	_____
2.	no.92-02, no. 92-03 (合訂本) V-N 複合名詞討論篇 與V-R 複合動詞討論篇	120	150	_____	_____
3.	no.93-01　新聞語料庫字頻統計表	120	130	_____	_____
4.	no.93-02　新聞語料庫詞頻統計表	360	400	_____	_____
5.	no.93-03　新聞常用動詞詞頻與分類	180	200	_____	_____
6.	no.93-05　中文詞類分析	185	205	_____	_____
7.	no.93-06　現代漢語中的法相詞	40	50	_____	_____
8.	no.94-01　中文書面語頻率詞典（新聞語料詞頻統計）	380	450	_____	_____
9.	no.94-02　古漢語字頻表	180	200	_____	_____
10.	no.95-01　注音檢索現代漢語字頻表	75	85	_____	_____
11.	no.95-02/98-04　中央研究院平衡語料庫的內容與說明	75	85	_____	_____
12.	no.95-03　訊息為本的格位語法與其剖析方法	75	80	_____	_____
13.	no.96-01「搜」文解字－中文詞界研究與資訊用分詞標準	110	120	_____	_____
14.	no.97-01　古漢語詞頻表（甲）	400	450	_____	_____
15.	no.97-02　論語詞頻表	90	100	_____	_____
16	no.98-01　詞頻詞典	395	440	_____	_____
17.	no.98-02　Accumulated Word Frequency in CKIP Corpus	340	380	_____	_____
18.	no.98-03 自然語言處理及計算語言學相關術語中英對譯表	90	100	_____	_____
19.	no.02-01 現代漢語口語對話語料庫標註系統說明	75	85	_____	_____
20	論文集 COLING 2002 紙本	100	200	_____	_____
21.	論文集 COLING 2002 光碟片	300	400	_____	_____
22.	論文集 COLING 2002 Workshop 光碟片	300	400	_____	_____
23.	論文集 ISCSLP 2002 光碟片	300	400	_____	_____
24.	交談系統暨語境分析研討會講義（中華民國計算語言學學會1997第四季學術活動）	130	150	_____	_____
25.	中文計算語言學期刊（一年四期）　年份：______（過期期刊每本售價500元）	---	2,500	_____	_____
26.	Readings of Chinese Language Processing	675	675	_____	_____
27.	剖析策略與機器翻譯 1990	150	165	_____	_____
			合　計	_____	_____

※　此價格表僅限國內 (台灣地區) 使用

劃撥帳戶：中華民國計算語言學學會　　劃撥帳號：19166251

聯絡電話：(02) 2788-3799 轉1502

聯絡人： 黃琪 小姐、何婉如 小姐　　E-mail:aclclp@hp.iis.sinica.edu.tw

訂購者：＿＿＿＿＿＿＿＿＿＿＿　收據抬頭：＿＿＿＿＿＿＿＿＿＿＿＿＿

地　　址：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

電　　話：＿＿＿＿＿＿＿＿＿＿＿＿　E-mail: ＿＿＿＿＿＿＿＿＿＿＿＿

Information for Authors

International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP) invites submission of original research papers in the area of computational linguistics and speech/text processing of natural language. All papers must be written in English or Chinese. Manuscripts submitted must be previously unpublished and cannot be under consideration elsewhere. Submissions should report significant new research results in computational linguistics, speech and language processing or new system implementation involving significant theoretical and/or technological innovation. The submitted papers are divided into the categories of regular papers, short paper, and survey papers. Regular papers are expected to explore a research topic in full details. Short papers can focus on a smaller research issue. And survey papers should cover emerging research trends and have a tutorial or review nature of sufficiently large interest to the Journal audience. There is no strict length limitation on the regular and survey papers. But it is suggested that the manuscript should not exceed 40 double-spaced A4 pages. In contrast, short papers are restricted to no more than 20 double-spaced A4 pages. All contributions will be anonymously reviewed by at least two reviewers.

Copyright：It is the author's responsibility to obtain written permission from both author and publisher to reproduce material which has appeared in another publication. Copies of this permission must also be enclosed with the manuscript. It is the policy of the CLCLP society to own the copyright to all its publications in order to facilitate the appropriate reuse and sharing of their academic content. A signed copy of the IJCLCLP copyright form, which transfers copyright from the authors (or their employers, if they hold the copyright) to the CLCLP society, will be required before the manuscript can be accepted for publication. The papers published by IJCLCLP will be also accessed online via the IJCLCLP official website and the contracted electronic database services.

Style for Manuscripts: The paper should conform to the following instructions.
1. Typescript: Manuscript should be typed double-spaced on standard A4 (or letter-size) white paper using size of 11 points or larger.
2. Title and Author: The first page of the manuscript should consist of the title, the authors' names and institutional affiliations, the abstract, and the corresponding author's address, telephone and fax numbers, and e-mail address. The title of the paper should use normal capitalization. Capitalize only the first words and such other words as the orthography of the language requires beginning with a capital letter. The author's name should appear below the title.
3. Abstracts and keywords: An informative abstract of not more than 250 words, together with 4 to 6 keywords is required. The abstract should not only indicate the scope of the paper but should also summarize the author's conclusions.
4. Headings: Headings for sections should be numbered in Arabic numerals (i.e. 1.,2....) and start form the left-hand margin. Headings for subsections should also be numbered in Arabic numerals (i.e. 1.1. 1.2...).
5. Footnotes: The footnote reference number should be kept to a minimum and indicated in the text with superscript numbers. Footnotes may appear at the end of manuscript
6. Equations and Mathematical Formulas: All equations and mathematical formulas should be typewritten or written clearly in ink. Equations should be numbered serially on the right-hand side by Arabic numerals in parentheses.
7. References: All the citations and references should follow the APA format. The basic form for a reference looks like

Authora, A. A., Authorb, B. B., & Authorc, C. C. (Year). Title of article. *Title of Periodical, volume number*(issue number), pages.

Here shows an example.

Scruton, R. (1996). The eclipse of listening. *The New Criterion, 15*(30), 5-13.

The basic form for a citation looks like (Authora, Authorb, and Authorc, Year). Here shows an example. (Scruton, 1996).
Please visit the following websites for details.
(1) APA Formatting and Style Guide (http://owl.english.purdue.edu/owl/resource/560/01/)
(2) APA Stytle (http://www.apastyle.org/)

No page charges are levied on authors or their institutions.

Final Manuscripts Submission: If a manuscript is accepted for publication, the author will be asked to supply final manuscript in MS Word or PDF files to clp@hp.iis.sinica.edu.tw

Online Submission: http://www.aclclp.org.tw/journal/submit.php

Please visit the IJCLCLP Web page at http://www.aclclp.org.tw/journal/index.php

International Journal of
Computational Linguistics & Chinese Language Processing
vol.21 no.1 June 2016

Contents

章無疵也章之明靡句無玷也句之清英字不妄也文賦曰選義按部考辭就班就所傳達者觀之禮記曰發志為言發言為名傳曰言以足志文以足言易曰書不盡言言不盡意詩序曰在心為志發言為詩情動於中而形於言蓋情志發而語言成語言工而文字傳也

ISSN: 1027-376X

The Association for Computational Linguistics and Chinese Language Processing