文字列照合マシンを利用した複合語キーワードの効率的抽出法

林 淑隆† 中野英雄†
獅々堀正幹† 青江順一†

本論文では，日本語文書から複合語キーワード（以後，キーワードと呼ぶ）を効率的に抽出する手法を提案する．本手法では，抽出条件に集合表現を導入した規則の記述を定義する．そして，規則集合の照合マシン（抽出マシンと呼ぶ）と候補語に対する部分文字列処理マシン（候補マシンと呼ぶ）を提案する．抽出マシンでは，包含関係に基づく照合アルゴリズムを提案し，候補マシンでは，候補語に対して部分文字列の関係が定義できる構成アルゴリズムを提案する．以上の抽出の高速化に加えて，提案した抽出エンジンは，分野別あるいは目的別に抽出規則が独立に構築でき，より精度の高いキーワード抽出が可能となる．形態素が10,129個である文書に対して29個の規則を定義した結果，形態素解析を除いた抽出時間は0.658秒となり，34種類の文書に対するキーワード抽出実験により，再現率が8.34%，適合率が2.67%向上することが分かった．

An Efficient Method for Extracting Keywords of Compound Words Using Pattern Matching Machines

YOSHITAKA HAYASHI,† HIDEO NAKANO,† MASAMI SHISHIBORI† and JUN-ICHI AOE†

Extracting keywords efficiently is an important task in text retrieval systems. In Japanese texts, there are many compound words consisting some kinds of characters (Katakana, Kanji, etc.) and the text has no delimiter among words. Therefore, extracting keywords from such a text takes a lot of time. This paper presents a technique of detecting keywords from compound keywords by introducing a set of rules, which represents multi-attribute conditions for keyword construction. A multi-attribute pattern matching machine for a finite number of rules is presented and storing keyword candidates together with information about both long term and short term words. The approach is estimated by theoretical analysis. By the simulation results for 34 Japanese text files, it is shown that the algorithm presented has performed 19.4 ms/KB and that the recall/precision ratio of extracting expected keywords increases from 65.89/26.56% to 74.23/29.23%.

1. まえがき

文書からのキーワード抽出は，文書処理や管理の分野で重要な技術であるが，日本語文書では，自由に造語される多くの複合語がキーワード候補になるので，複合語の取り扱いが重要な問題となる[1)~5),8)~11)]．

キーワード候補となる複合語を辞書に登録することは非現実的であり，短単位語のみを辞書に登録し，複合語の構成情報を抽出条件として別に管理する手法[3)]が一般的である．たとえば，単語ごとにキーワードの評価を行い，その連続部分を複合語として抽出する手法[4),5)]や，キーワードパターンに照合する単語の連続部分を複合語として抽出する手法[3)]などがある．しかし，これらの多彩な抽出条件を統合化して記述する抽出規則の形式化の議論がないので，目的別あるいは分野別に規則を構築し，抽出精度をあげることは困難であった．また，頻繁に生じる抽出条件や文字列照合を効率化した汎用的な抽出エンジンも開発されていない．

以上の問題点に対して，本論文では，抽出条件に集合表現を導入し，汎用的な抽出条件が記述できる抽出規則を定義する．そして，規則集合の照合マシン（抽出マシン）と，包含関係に基づく照合アルゴリズムを提案する．次に，抽出マシンから得られた候補語に対する部分文字列処理マシン（候補マシン）を提案する．候補マシンでは，候補語に対して部分文字列の関係が定義できる構成アルゴリズムと，文字列の構成単位による高速な文字列処理を提案する．

† 徳島大学工学部
Faculty of Engineering, Tokushima University

以下，2章では抽出規則を定義し，3章では規則の抽象化を提案し，4章では抽出マシンを提案し，5章では候補マシンを提案し，6章で本手法の評価を行う．7章では本論文をまとめ，今後の課題について触れる．

2. キーワード抽出の諸条件

2.1 複合語キーワードの構造

2.1.1 複合語規則

日本語文書では，多くの複合語がキーワード候補になるので，複合語を抽出するために，複合語の構造を定義する文法規則（**複合語規則**と呼ぶ）が必要である．宮崎[7),12)]による係り受け規則を基本に定義した複合語規則の例を**表1**に示す．ただし，この規則は品詞間の接続関係を定義するので，＜接頭辞－名詞＞と＜名詞－名詞＞が連鎖する複合語“超/音波/処理”も抽出される（'/'は形態素の区切りを示す）．しかし，この規則の定義は分野や目的により異なる．たとえば，“超/小型”は抽出せず，“超/小型/メモリ”を抽出したい場合[2)]，＜接頭辞－名詞＞の代わりに，＜接頭辞－名詞－名詞＞を定義する．この複合語規則を解析表にコンパイルし，その解析表を駆動する汎用的照合エンジンが必要である．

2.1.2 複合語を構成する形態素の属性制約

たとえば，複合語規則＜固有名詞1－名詞2＞で，＜固有名詞1＞を製品名に，＜名詞2＞の字種をカタカナに限定するように，より詳細な条件を複合語規則に記述したい場合がある．したがって，形態素に対する意味や字種属性を規則に制約でき，かつその規則が処理できる照合エンジンが必要となる．

2.2 キーワード評価規則

複合語規則は，抽出される複合語の構造を記述したものであるが，次に定義する抽出評価規則および削除評価規則は，抽出された複合語のキーワード性を評価する条件である．

2.2.1 抽出評価規則

抽出評価規則は，キーワードの抽出条件となる形態素制約であり，例を**表2**に示す．属性HEAD（前置抽出属性）は直前の複合語に，TAIL（後置抽出属性）は直後の複合語に，BOTH（前後置抽出属性）は前後の複合語に得点を加点することを意味する．また，“に対して”などの定型表現も品詞＜格助詞相当句＞を定義し，形態素として取り扱う．これらの抽出条件を汎用的な抽出規則として記述するためには，複合語規則と同様に，形態素“格助詞”に対して，表記“は”や属性“HEAD”および得点“20”を抽出規則に制約できる照合エンジンが必要となる．

表1 複合語規則の例
Table 1 An example of rules for compound words.

品詞列	例
＜名詞－名詞＞	音波/処理
＜接頭辞－名詞＞	超/音波
＜名詞－接尾辞＞	近代/化
＜接尾辞－名詞＞	“効率的抽出法”における“的/抽出”の部分
＜名詞－接尾辞－名詞＞	全国/的/ネットワーク

表2 抽出評価規則の例
Table 2 An example of extracting rules.

品詞と条件	例	属性	得点
格助詞の前	は，が	HEAD	20
格助詞相当句の前	に対して，に対する	HEAD	20
指示連体詞の後	この，これらの	TAIL	20
提題表現の前	についていえば，としては	HEAD	30
提題表現の前後	における	BOTH	30

表3 削除評価規則の例
Table 3 An example of deleting rules.

品詞と条件	例	属性	得点
接頭辞	各，同，諸，当該，本，両	DEL	0
接尾辞	やすい，全体，下，中，以外	DEL	0
並列表現の前後	と，や，あるいは	BOTH	−20
不要語	例えば，今回，場合，そのとき	DEL	0

2.2.2 削除評価規則

削除評価規則は，キーワードの削除と抑制条件であり[3),8)]，接辞辞書を基本とする例を**表3**に示す．属性DEL（delete：形態素削除属性）は，対応する形態素をキーワード候補としないことを意味する．属性BOTHは抽出評価規則のBOTHと同一であるが，負の得点を定義することで，前後のキーワード候補の抑制条件を表現する☆．この規則についても2.2.1項と同様な照合エンジンが必要となる．

2.3 複合語キーワードの構成要素

キーワードは，長単位語（複数形態素で構成）と短単位語（単一自立語の形態素）が考えられる[2),9)]．長単位語優先処理では，キーワードとならない短単位語を含むことができるが，文字列の完全一致による検索では，長単位語に含まれる短単位語の検索漏れにより，検索精度が落ちる．これに対して，短単位語優先処理ではこの検索漏れは生じないが，複合語の構成情報が崩されるので，キーワード自体の精度に問題が残る．この抽出条件に対して，従来は抽出した候補に対して，

☆ キーワード評価規則は，抽出評価規則と削除評価規則に分類されるが，属性DELを除いて，得点の正負により，抽出評価規則（正数）と削除評価規則（負数）とに分類される．

部分文字列の併合や分離処理を行っていたので，キーワード数が多くなると処理時間が長くなる．したがって，この処理の高速化が必要となる．

2.4 頻度情報

最も基本的な抽出情報である頻度情報[3),10)]は，以上の抽出条件における評価点処理，および短単位語・長単位語の処理と効率的に融合する必要がある．

3. 抽出規則の定義と照合マシン

3.1 抽出規則の抽象化

前章で定義した複合語規則や評価規則などを，すべて記述することは不可能であり，これらの規則を抽象化して定義することが重要となる．そこで形態素構造を，表記と品詞情報以外に属性制約を定義できる多属性制約（集合表現）で記述する方式を提案する．

R を（属性名，属性値）なる組を要素とする集合とし，規則構造と呼ぶ．次に抽出規則を定義する．

$R_1 R_2 \cdots R_n \, (1 \leq n)$

なお，次の属性名と属性値を導入する．

(1) 形態素の必須要素である表記と品詞の属性名をそれぞれSTR（string）とCAT（category）で表す．簡単のために活用情報は品詞情報に含める．

(2) 汎用的な抽出条件を記述するために，自由な属性名と属性値を定義できるものとする．

なお，構造 R に対する属性値は，次の関数 v で得られるものとする．

$v(R, \text{属性名}) = \text{属性値}$

（例1） 指示連体詞と名詞を表す次の規則を考える．

$R_1 = \{(\text{CAT}, \text{指示連体詞})\}; R_2 = \{(\text{CAT}, \text{名詞})\}$

R_1 は要素 (CAT, 指示連体詞) にだけ制約を受けるので，“この”“これらの”等の表記を抽象化できる．R_2 も同様にすべての名詞として抽象化できる．また，R_1 の指示連体詞を接頭辞に変更し，接頭辞“各”を照合しない属性を (DEL,TRUE) で与えるとき，次の R_1 で接頭語の削除規則が定義できる．

$R_1 = \{(\text{CAT}, \text{接頭辞}), (\text{DEL}, \text{TRUE})\}$

このように構造 R を集合とし，規則を抽象化することで，規則数が抑制できる．

規則構造 R の照合対象である形態素解析結果も規則構造 R と同じ集合表現で定義するが，形態素構造 M として規則構造 R と区別する．形態素解析エンジンの汎用性を維持するために，各属性に対してその値を得る辞書を形態素辞書とは独立に構築する必要がある．したがって，形態素構造 M に対して属性名ATTRの属性値を決定する関数DEFINEを次に定義する．

$\text{DEFINE}(v(M, \text{STR}), v(M, \text{CAT}), \text{ATTR})$

たとえば，$M = \{(\text{STR}, \text{“各”}), (\text{CAT}, \text{接頭辞})\}$ に対して，属性名DELの値は，次より得られる．

$\text{DEFINE}(\text{“各”}, \text{接頭辞}, \text{DEL})$

3.2 照合マシンの提案

前節より抽出規則の記述能力は高くなるが，逆に照合アルゴリズムは難しくなる．規則構造を単に文字としてとらえた場合の照合マシンとしては，AhoとChorasick[13)]による複数キーワードの文字列照合マシン（マシンAC）がよく知られている．しかし，本論文で提案する照合マシンでは，次の点で新規性がある．

(1) マシンACの照合対象は文字列であり，文字の完全一致の照合法である．しかしながら，3.1節で提案した抽象化された規則では，集合の包含関係による照合を行わなければならない．したがって，包含関係による照合法を用いてキーワード候補を決定する**抽出マシン**を新しく提案する．

(2) 抽出マシンを用いることで，複合語規則とキーワード評価規則を1つのマシン上に構成することができ，キーワード候補の抽出と評価を同時に処理できる．

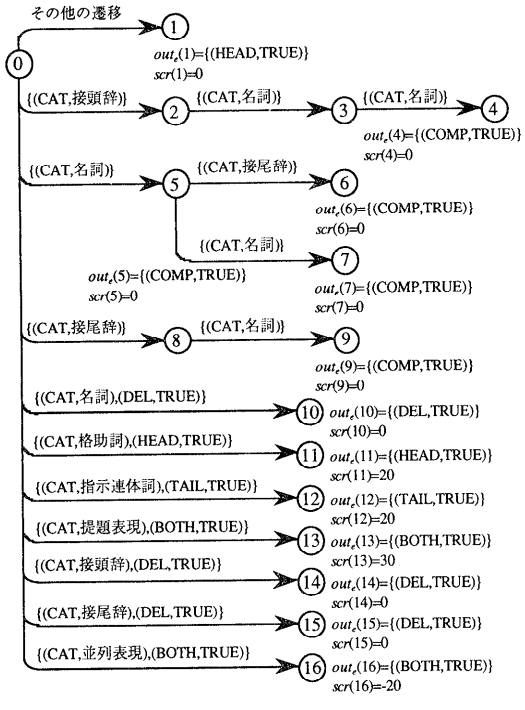
(3) キーワード抽出では，2.3節で述べた長単位語と短単位語の併合・分離処理が必要である．マシンACでは，長単位語に含まれる短単位語は容易に検出できるが，その逆は複雑な処理を必要とする☆．本手法では，長単位語と短単位語の併合・分離処理を効率的に行う**候補マシン**を新しく提案する．候補マシンは，抽出マシンで決定されたキーワード候補を格納し，目的とするキーワードを高速に決定し，頻度情報も処理できる．

4. 抽出マシン

4.1 抽出マシンの定義

マシンACは，照合対象が文字であるので，集合表現構造を照合対象とするために，抽出マシンを定義する．キーワード候補への評価点を定義する関数 *scr*（score）を新しく準備し，抽出マシンの状態集合を S_e，規則構造 R の集合を J，整数の集合を I とするとき，抽出マシンは次の4つの関数で定義される．ただし，*output* 関数 out_e は規則に対応する制約属性の集合とする．

☆ *goto* 関数および *failure* 関数を逆に遷移する必要があり，計算コストが増大する．



(a) Functions g_e, out_e and scr.

$f_e(3)=5$ $f_e(4)=7$ $f_e(6)=8$ $f_e(7)=5$ $f_e(9)=5$ 以外は $f_e(s)=0$

(b) A failure function.

図1 抽出マシン
Fig. 1 The machine of extracting keywords.

goto 関数 g_e : $S_e \times J \rightarrow S_e \cup \{ fail \}$
failure 関数 f_e : $S_e \rightarrow S_e$
output 関数 out_e : $S_e \rightarrow \{\alpha \mid \alpha =$ (属性名, 属性値)$\}$
score 関数 scr : $S_e \rightarrow I$

構造 R_1 と R_2 の要素数が等しく，かつすべての要素内容（属性と属性値）が等しいとき，$R_1 = R_2$ が成立することとする．この等価条件下では，5 章で議論するマシン AC の構成法と同様に抽出マシンは構成できる．

2 章の規則に対応する抽出マシンの例を図 **1** に示す．

状態 2 から 9 は，複合語規則（表 1）を示し，状態 11，12，13 は抽出評価規則（表 2）を，状態 10，14，15，16 は削除評価規則（表 3）を示す．状態 4，5，6，7 の out_e 関数の要素 (COMP,TRUE) は複合語（compound words）規則に照合したことを意味する属性である．また，初期状態 0 に対するセルフループを，状態 1 への遷移として定義する．

4.2 包含関係による状態遷移

抽出マシンの照合対象は集合構造であるため，包含関係による状態遷移を行う．したがって，入力構造 M に対して状態 s からの遷移が可能か否かを判定する関数 CHECK (s, M) を以下に定義する．

[関数 CHECK (s, M)]
手順（C-1）: {入力構造への任意属性の付加}

状態 s から出る遷移ラベル（規則構造 R）に含まれる任意の属性名 ATTR のすべてに対して，VALUE $\leftarrow$ DEFINE $(v(M, \text{STR}), v(M, \text{CAT}), \text{ATTR})$ を決定し，(ATTR,VALUE) を構造 M の要素に加える．
手順（C-2）: {状態遷移の決定}

$R \subseteq M$ なる関係を満足する $g_e(s, R) = t$ が存在すれば t を返し，そうでなければ 0 を返す．ただし，$R \subseteq M$ なる R が複数存在する場合は，要素数の多い R を優先する．さらに，要素数が同数の R が存在する場合は，あらかじめ優先順位の付けられた属性名を要素に含む規則構造を選択する．

(**例 2**) 図 1 の状態 $g_e(0, R) = 14$ なる規則構造 $R =$ {(CAT, 接頭辞),(DEL,TRUE)} に対して，次の入力構造 M_1 と M_2 を考える．

$M_1 =$ {(STR, “超”), (CAT, 接頭辞)}
$M_2 =$ {(STR, “各”), (CAT, 接頭辞)}

R は任意属性 (DEL,TRUE) を含むので手順（C-1）の

FALSE = DEFINE (“超”, 接頭辞, DEL)
TRUE = DEFINE (“各”, 接頭辞, DEL)

により，次が準備される．

M_1 = {(STR,“超”),(CAT, 接頭辞),(DEL,FALSE)}
M_2 = {(STR,“各”),(CAT, 接頭辞),(DEL,TRUE)}

手順（C-2）において，$R \subseteq M_2$ は成立するが，$R \subseteq M_1$ は成立しないので，CHECK(0, M_2) は状態 14 を返す．しかし，CHECK(0, M_1) は $g_e(0, R') = 2$ なる規則構造 R' ={(CAT, 接頭辞)} に対して，$R' \subseteq M_1$ が成立するので，状態 2 を返す．

4.3 抽出アルゴリズム

抽出アルゴリズムの入力は，形態素解析結果である入力構造列 α と抽出マシンであり，出力は抽出されたキーワード候補 KEY とその評価点 SCR (SCORE) の組 (KEY, SCR) を要素とする候補集合 K_SET である．

[抽出アルゴリズム]
手順（EX-1）: {各種変数の初期化}

現在処理中の状態 s を 1，入力構造の位置 i を 1，入力構造の先頭 h を 1，出力となる K_SET を ϕ（空集合）に初期化する．キーワード候補 KEY を NULL（空文字列）に，その評価点 SCR を 0 に初期化する．
手順（EX-2）: {入力構造の移動}

i を 1 から n まで変化させて，以下の手順（EX-3）

から手順（EX-5）を繰り返す．

手順（EX-3）:｛遷移の確認｝

入力構造 M_i に対して CHECK $(s, M_i) = t$ なる t が 0 ならば手順（EX-4）へ，0 でなければ手順（EX-5）へ進む．

手順（EX-4）:｛*failure* 関数による遷移｝

s に $f_e(s)$ をセットする．$f_e(s) = 0$ のとき，複合語規則の連鎖が切れることを意味するため，h に i を代入する．手順（EX-3）に戻る．

手順（EX-5）:｛形態素の追加｝

$out_e(t) \neq \phi$ のとき，キーワード抽出関数 GENE $(M_h \cdots M_i, t)$ を実行し，h に i を代入して手順（EX-2）に戻る．

［関数 GENE $(M_i \cdots M_j, t)$ ］

関数 GENE は入力構造 $M_i \cdots M_j$ からキーワードを抽出し，評価点を加算する．以下に，その手順を示す．

手順（G-1）:｛規則による処理の分岐｝

SCR に $scr(t)$ を加算する．また α = (COMP, TRUE) ならば，変数 x に i を代入し j まで変化させて，KEY に入力構造 M_i から M_j までの表記列をキーワード候補として連鎖する．α = (HEAD,TRUE) のとき，手順（G-2）へ進み，α = (BOTH,TRUE) のとき，手順（G-3）へ進む．α がこれら以外のときは，関数を終了する．

手順（G-2）:｛前置抽出属性 HEAD の処理｝

$KEY \neq$ NULL かつ $SCR > 0$ ならば，(KEY, SCR) を K_SET に加え，KEY を NULL に，SCR を 0 に初期化する．

手順（G-3）:｛前後置抽出属性 BOTH の処理｝

$KEY \neq$ NULL かつ $SCR > 0$ ならば，(KEY, SCR) を K_SET に加え，KEY を NULL に初期化する．ただし，後置抽出のために SCR に $scr(t)$ を代入し，評価点を保持する．

（**例 3**）入力文“これらの/各/超/小型/メモリ/における/問題/点/や/手法/も/…”を考える．**表 4** にアルゴリズムの動作を示す．

M_1 = {(STR,“これらの”),(CAT, 指示連体詞)} は，R = {(CAT, 指示連体詞), (TAIL,TRUE)}, $g_e(0, R) = 12$ より，DEFINE で表 4 のように拡張される．ここで，$M_1 \subset R$ であるので，CHECK $(0, M_1) = 12$ となり，状態 12 に遷移し，$out_e(12)$ = {(TAIL,TRUE)} より，GENE(M_1,12) において $scr(12) = 20$ を加算し，(KEY, SCR) は (NULL,20) となる．また，照合の先頭位置 $h = 2$ となる．

M_2 は CHECK $(12, M_2) = 0$, $f_e(12) = 0$ により，状態 0 に遷移する．CHECK $(0, M_2) = 14$, $out_e(14)$ = {(DEL,TRUE)} であるので，GENE (M_2,14) において，$scr(14) = 0$ であり，(KEY, SCR) に変化はない．M_3, M_4 の処理後，M_5 では $out_e(4)$ = {(COMP,TRUE)}, GENE (M_3 M_4 M_5, 4) より，KEY に M_3 から M_5 までの表記“超小型メモリ”を追加する．したがって (KEY, SCR) = (“超小型メモリ”,20) となる．

M_6 では $out_e(13)$ = {(BOTH,TRUE)}, GENE (M_6,13) より，$scr(13) = 30$ を加算し，(“超小型メモリ”,50) を K_SET に加える．(KEY, SCR) = (NULL,30) に初期化する．M_7, M_8 の処理後，(KEY, SCR) = (“問題点”,30) となる．

M_9 では K_SET に (“問題点”,10) を加え，SCR は -20 となる．M_{10} の処理後，M_{11} では $scr(1) = 0$ を加算するが，(KEY, SCR) = (“手法”,-20) により，この組は K_SET に追加されない．ここで，状態 1 が定義されてない場合，後置抽出の際に状態 0 に対して抽出処理をしなければならないが，状態 0 は遷移の起点であり，関数 f_e の遷移先となる状態が多いので，状態 1 を定義した方が処理が簡潔になる．

このように抽出マシンは，1 度の解析でキーワード候補の抽出と評価を同時に行うことができる．

5. 候補マシン

5.1 アルゴリズムの概要

抽出マシンの出力 K_SET が，候補マシンの入力となる．まず，K_SET = {(“自然”,40),(“自然言語”,50),(“自然環境”,50),(“言語”,80)} に対する候補マシン（**図 2**）で概要を説明する．

短単位語処理では，“自然言語”と“自然環境”の共通接頭辞“自然”と，“自然言語”の接尾辞“言語”を候補として優先し，逆に長単位語処理では，“自然”と“言語”の複合語“自然言語”と，“自然”を接頭辞に持つ“自然環境”を優先する．

この部分文字列処理を効率的に行うために，候補マシンではマシン AC に新しい関数 *rel*（relation）を導入する．ただし，*output* 関数はキーワード KEY と評価点 SCR の組 (KEY, SCR) を要素とする集合とし，out_c で記述する．

状態 s に対する $rel(s)$ は，次を満足する要素 t を持つ．$out_c(s) \ni (x, h)$ なるキーワード x は，$out_c(t) \ni (y, k)$ なるキーワード y の部分文字列☆と

☆ $t = s$ により，$x = y$ となる場合も含むものとする．

表4 抽出アルゴリズムの動作
Table 4 Actions of an extracting algorithm.

M	入力構造の要素	規則構造の要素	状態 s の変化	$out_e(s)$ と GENE	h	(KEY, SCR)
M_1	(STR,“これらの”) (CAT, 指示連体詞) (TAIL,TRUE)	(CAT, 指示連体詞) (TAIL,TRUE)	$g_e(0,R)=12$	(TAIL,TRUE) GENE(M_1,12)	1 2	(NULL,20)
M_2	(STR,“各”) (CAT, 接頭辞) (DEL,TRUE)	(CAT, 接頭辞) (DEL,TRUE)	$f_e(12)=0$ $g_e(0,R)=14$	(DEL,TRUE) GENE(M_2,14)	3	
M_3	(STR,“超”) (CAT, 接頭辞)	(CAT, 接頭辞)	$f_e(14)=0$ $g_e(0,R)=2$			
M_4	(STR,“小型”) (CAT, 名詞)	(CAT, 名詞)	$g_e(2,R)=3$			
M_5	(STR,“メモリ”) (CAT, 名詞)	(CAT, 名詞)	$g_e(3,R)=4$	(COMP,TRUE) GENE($M_3M_4M_5$,4)	6	(“超小型メモリ”,20)
M_6	(STR,“における”) (CAT, 提題表現) (BOTH,TRUE)	(CAT, 提題表現) (BOTH,TRUE)	$f_e(4)=7$ $f_e(7)=5$ $f_e(5)=0$ $g_e(0,R)=13$	(BOTH,TRUE) GENE(M_6,13)	7	(“超小型メモリ”,50) $\in K\_SET$ (NULL,30)
M_7	(STR,“問題”) (CAT, 名詞)	(CAT, 名詞)	$f_e(13)=0$ $g_e(0,R)=5$			
M_8	(STR,“点”) (CAT, 名詞)	(CAT, 名詞)	$g_e(5,R)=7$	(COMP,TRUE) GENE(M_7M_8,7)	9	(“問題点”,30)
M_9	(STR,“や”) (CAT, 並列表現) (BOTH,TRUE)	(CAT, 並列表現) (BOTH,TRUE)	$f_e(7)=5$ $f_e(5)=0$ $g_e(0,R)=16$	(BOTH,TRUE) GENE(M_9,16)	10	(“問題点”,10) $\in K\_SET$ (NULL,−20)
M_{10}	(STR,“手法”) (CAT, 名詞)	(CAT, 名詞)	$f_e(16)=0$ $g_e(0,R)=5$	(COMP,TRUE) GENE(M_{10},5)	11	(“手法”,−20)
M_{11}	(STR,“も”) (CAT, 格助詞)	(CAT, 格助詞)	$f_e(5)=0$ $g_e(0,R)=1$	(HEAD,TRUE) GENE(M_{11},1)	12	(“手法”,−20),(NULL,0)

なる．たとえば，$rel(8) \ni 4$ より，$out_c(8)$ の“言語”は，$out_c(4)$ の“自然言語”の部分文字列であることが分かる．

次に，短単位語あるいは長単位語処理に対する SCR への加点を，$out_c(s) \neq \phi$ なる状態 s に対してそれぞれ関数 $short(s)$ と $long(s)$ で定義し，図2にその例を示す．この加点条件をキーワード候補の部分文字列の重複数 $OVERLAP$ で説明する．$short(s)$ は $rel(s)$ の要素数 c に対して，重複数 $OVERLAP = c-1$ と加点重み p の積により，$short(s) = (c-1)p$ で定義できる．なお，後で提案する rel 関数の構成法を簡単にするために，$rel(s)$ は s を含むので，$c-1$ の -1 が必要となる．たとえば，図2の $rel(2) = \{2,4,6\}$ より，“自然”に対する加点は $short(2) = (OVERLAP-1)p = (3-1)p = 2p$ となる．すなわち，状態2の“自然”は，状態4の“自然言語”と状態6の“自然環境”で重複している．

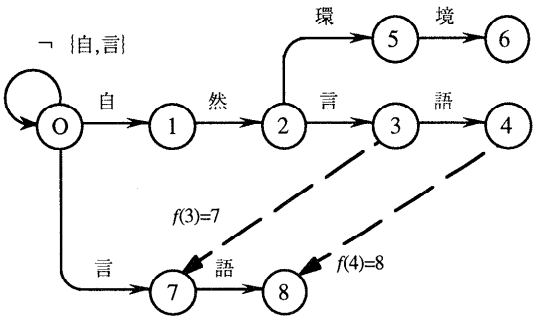
$long(s)$ は，$s \in rel(t)$，$t \neq s$ なる状態 t の数に等しい $OVERLAP$ に対して，$long(s) = OVERLAP \times p$ と定義できる．図2の $long(4)$ の $OVERLAP$ は，$rel(2) = \{2,4,6\}$，$rel(8) = \{8,4\}$ の2つに状態4が含まれるので，$long(4) = OVERLAP \times p = 2p$ となる．すなわち，状態4の“自然言語”は，状態2の“自然”と状態8の“言語”で重複している．

加点重み $p = 50$ の場合の短単位語と長単位語処理の得点変化を**表5**に示す．短単位語処理では，入力 K_SET の4位候補“自然”が，$SCR + 2p = 40 + 100 = 140$ で1位になる．また，長単位語処理では入力 K_SET の1位の短単位語候補“言語”が3位になり，上位に長単位語候補が優先される．

次節に，この新しい関数 rel を導入した候補マシンの構成アルゴリズムを与える．

5.2 候補マシンの構成

$goto, out_c, rel$ 関数の構成アルゴリズム1（**図3**）と，



(a) Functions *goto* and *failure*

$out_c(2)$= {(“自然”,40)} $rel(2)$= {2,4,6}
$out_c(4)$= {(“自然言語”,50)} $rel(4)$= {4}
$out_c(6)$= {(“自然環境”,50)} $rel(6)$= {6}
$out_c(8)$= {(“言語”,80)} $rel(8)$= {8,4}

(b) Functions out_c and *rel*.

short(2)=2p *long*(2)=0
short(4)=0 *long*(4)=2p
short(6)=0 *long*(6)=p
short(8)=p *long*(8)=0

(c) Functions *short* and *long*.

図 2 候補マシンの例
Fig. 2 An example of a candidate machine.

表 5 キーワード候補の例
Table 5 An example of keyword candidates.

入力 *K_SET*	短単位語 *K_SET*	長単位語 *K_SET*
(“言語 ,80)	(“自然 ,140)	(“自然言語,150)
(“自然言語,50)	(“言語 ,130)	(“自然環境,100)
(“自然環境,50)	(“自然言語,50)	(“言語 ,80)
(“自然 ,40)	(“自然環境,50)	(“自然 ,40)

```
Algorithm 1: Construction of the goto,
             out_c and rel functions.
Input: K_SET={(y1,s1),(y2,s2),...,(yk,sk)}.
Output: Functions g, out_c and rel.
Method.
  begin
    newstate ← 0;
    for i ← 1 until k do enter(yi,si)
    for all a such that g(0,a)=fail do
      g(0,a) ← 0
    rel(0) ← φ;
  end

  procedure enter(a1a2 ... am,scr):
  begin
    state ← 0; j ← 1;
1>  queue ← empty;
2>  while g(state,aj) ≠ fail do
      begin
        state ← g(state,aj); j ← j+1;
3>      queue ← queue ∪ {state};
      end
4>  for p ← j until m do
      begin
        newstate ← newstate+1;
        g(state,ap) ← newstate;
        state ← newstate;
        rel(state) ← φ;
5>      queue ← queue ∪ {state};
      end
    out_c(state) ← {(a1a2 ... am,scr)};
6>  while queue ≠ φ do
      begin
        let r be the next state in queue
        queue ← queue−{r};
        rel(r) ← rel(r) ∪ {state};
      end
  end
```

図 3 候補マシン構成アルゴリズム（*goto* 関数）
Fig. 3 An algorithm of constructing a goto function.

$failure, out_c, rel$ 関数の更新アルゴリズム 2（図 **4**）を示す．なお，行頭に“行番号>”を付加して説明する．

アルゴリズム 1 は，*K_SET* の要素 (y_i, s_i) を手続き enter に送り，行番号 2 の **while** 文で定義済みの遷移を可能な限りたどり，行番号 4 の **for** 文で新しい遷移を定義する．行番号 1 の *queue* は遷移後の状態を行番号 3 と 5 で記憶する．キーワードの追加終了後，行番号 6 の **while** 文で *rel* 関数を作成する．ここで定義される *rel* 関数は，共通接頭辞に関するものである．なお，$rel(s)$ は処理の簡単化のために，$out_c(s) \neq \phi$ でない s だけでなく，すべての状態 s に対して定義される．

図 2 の候補マシンで説明する．まず，候補 (“自然”,40) に対して，手続き enter では $g(0,$“自”$) = 1$ と $g(1,$“然”$) = 2$ を作成し，$queue = \{1,2\}$ より，$rel(1) = rel(2) = \{2\}$ となる．次の (“自然言語”,50) では，状態 1，2 への遷移後，$g(2,$“言”$) = 3$ と $g(1,$“語”$) = 4$ を作成し，$queue = \{1,2,3,4\}$ より，$rel(1) = rel(2) = \{2,4\}$，$rel(3) = rel(4) = \{4\}$ となる．(“自然環境”,50) では，$queue = \{1,2,5,6\}$ より，$rel(1) = rel(2) = \{2,4,6\}$，$rel(5) = rel(6) = \{6\}$ となる．(“言語”,80) では，$g(0,$“言”$) = 7$ と $g(7,$“語”$) = 8$ の作成より，$rel(7) = rel(8) = \{8\}$ となる．

アルゴリズム 2 では *failure* 関数の作成時に，共通接頭辞以外の部分文字列に関する *rel* 関数を定義するが，この定義はマシン AC の次の性質に従っている[13]．

[性質] $f(s) \neq 0$ なる状態 s は，初期状態から状態

```
Algorithm 2: Construction of the failure
             and rel functions.
Input: Function g, out_c and rel from Algorithm 1.
Output: Function f, out_c and rel.
Method.
  begin
    queue ← φ;
1>  for each a such that g(0,a)=s≠0 do
      begin
        queue ← queue ∪ {s}; f(s) ← 0;
      end
2>  while queue ≠ φ do
      begin
        let r be the next state in queue
        queue ← queue−{r};
3>      for each a such that g(r,a)=s≠ 0 do
          begin
            queue ← queue ∪ {s}; state ← f(r);
            while g(state,a)=fail do
              state ← f(state);
            f(s) ← g(state,a);
            out_c(s) ← out_c(s) ∪ out_c(f(s));
4>          if s ∉ rel(f(s)) then
              begin
5>              t ← s;
6>              while f(t)≠0 do
                  begin
7>                  rel(f(t)) ← rel(f(t)) ∪ rel(s);
                    t ← f(t);
8>                end
              end
          end
      end
  end
```

図4 候補マシン構成アルゴリズム（*failure* 関数）
Fig. 4 An algorithm of constructing failure and rel functions.

$f(s)$ までの遷移ラベルを連鎖した文字列 x と，初期状態から状態 s までの遷移ラベルの連鎖文字列 y において，x は y の部分文字列となる．

したがって，*rel* 関数の更新では $rel(f(s))$ に $rel(s)$ の要素を追加する．図2の $f(4)=8$（“自然言語”と“言語”）では，$rel(f(4))$ に $rel(4)$ の要素を追加し，この処理を行番号5から8で行う．行番号6の **while** 文で，部分文字列がなくなるまでこの処理を行う．また行番号4の **if** 文は，重複した要素追加を防ぐ．

アルゴリズム2は，状態遷移図において幅優先探索で *failure*，out_c 関数を定義する．図2では，行番号1の **for** 文で *queue* に深さ1の状態1，7を格納後，$f(1)=f(7)=0$ が定義される．行番号2の **while** 文で深さ1の状態を取り出し，行番号3の **for** 文で深さ2の状態2，8を *queue* に格納し，深さ2の *failure* 関数 $f(2)=g(0,\text{“然”})=0$ と $f(8)=g(0,\text{“語”})=0$ を定義する．深さ3の状態3，5では，$f(3)=g(0,\text{“言”})=7$ が定義され，このとき行番号7で $rel(f(3))=\{8\}$ と $rel(3)=\{4\}$ の和集合により，$rel(7)=\{8,4\}$ に更新する．以降，同様にして，$f(5)=g(0,\text{“環”})=0$，$f(4)=g(f(3),\text{“語”})=8$ が定義され，$rel(8)=\{8,4\}$ に更新する．

6. 評　価

6.1 理論的評価

実時間処理を考慮して，抽出マシンの照合時間，候補マシンの構成時間，短単位語と長単位語処理に対する理論的解析を行う．

6.1.1 抽出マシンの理論的評価

入力となる形態素構造の長さを m とするとき，*goto* 関数の判定コストを $O(1)$ とすれば，抽出マシンの照合時間は，マシンACと同様に，規則数に関係なく $O(m)$ となる．ただし，*goto* 関数の判定コストは $O(1)$ でなく，関数CHECK (s,M) に依存する．構造の属性名に順序を定義するとき，構造の集合は（属性名，属性値，ポインタ）なるノードを属性名の順番で並べたソートリストで表現できる．また，*goto* 関数もリスト構造で表現する[13]．属性名の種類を A とするとき，関数CHECKの手順（C-2）の包含関係を判定する計算量は，規則構造と形態素構造の最大長 A のソートリストをマージソートするコスト $A+A=2A$ と同じであり，$O(A)$ となる．この判定時に，形態素構造 M に未定義の属性名が検出できるので，関数DEFINEの計算量を一定とすると，手順（C-1）は（C-2）の計算量に含めることができる．また，状態 s に定義される最大遷移数を B とすれば，関数CHECKの最大時間計算量は $O(AB)$ となる．この計算量はマシンACの計算量 $O(B)$ より多くなるが，規則の記述能力向上とのトレードオフである．

6.1.2 候補マシンの理論的評価

キーワード長の総和を n とするとき，マシンACの構成時間計算量は $O(n)$ である．アルゴリズム1では遷移したすべての状態を *queue* に格納するので，この計算量が $O(n)$ となる．また，キーワード数を k とするとき，アルゴリズム2は，行番号6の **while** 文で最悪 $n-k$ 回の遷移を行うが，$n \gg k$ であるので，この計算量も $O(n)$ となる．したがって，候補マシンの構成に対する時間計算量はマシンACと同じで，$O(n)$ となる．

短単位語処理の関数 $short(s)$ は，$out_c(s) \neq \phi$ なる s に対する $rel(s)$ の要素数より決定できるが，その要素数は，*rel* 関数の決定時に蓄積カウンタを記憶

する $count(s)$ を用意することで行え，しかもそのコストはアルゴリズムの計算量に影響を与えない．したがって，短単位処理の計算量は，$short(s)$ を $out_c(s)$ の SCR に加点する計算量 $O(k)$ となる．

$long(s)$ は，s を含む $rel(t)$ を決定するコストに比例し，$rel(t)$ は状態番号をビット位置に対応させたビットベクトルで表現できる．したがって，ベクトル中の任意位置のビット値を確認するコストを一定とすれば，関数 $long(s)$ の計算量は定義されている rel 関数の数 k に比例し，$O(k)$ となる．この評価は，5 で提案した処理に限定されたものであるが，SCR, rel 関数以外で利用できる頻度等の情報は，$out_c(s) \neq \phi$ なる s に対して定義でき，総合評価点を決定する関数（本論文での $short$, $long$ 関数）が複雑にならなければ，計算量 $O(k)$ は増加しない．

また，マシン AC から $short(s)$ や $long(s)$ を構成する場合，$goto$ 関数を逆遷移するための時間計算量 $O(n)$，および $failure$ 関数を逆遷移するための時間計算量 $O(n)$ が，最悪の場合必要となる．したがって，マシン AC の構成時間計算量 $O(n)$ の他に，最悪で時間計算量 $O(2n)$ が必要となる．さらに，このような逆遷移を行うために，$goto$ 関数および $failure$ 関数の逆関数を必要とするので，逆関数のための記憶領域も余分に必要となる．このように，マシン AC の構成時間 $O(n)$ より多くの計算コストを必要とすることが分かる．候補マシンは関数 rel を使用して，構成時間計算量 $O(n)$ にこの逆関数の計算コストを吸収し，$short(s)$ や $long(s)$ の決定時間を $O(k)$ に抑えている．したがって，候補マシンにおける長単位・短単位処理は，マシン AC に比べて，高速であることが分かる．

6.2 実験による評価

提案手法は C 言語で開発し，計算機は DELL OptiPlex GXMT 5133 を使用した．実験には，作成者がキーワードをあらかじめ設定している一般文書（手紙，雑誌など）8，論文の概要（情報処理関連）13，特許の紹介文（自然言語処理関連）13 の合計 34 文書を利用した．まず，提案マシンの性能を評価するために，34 文書を 1 つのファイルにまとめて実験を行い，抽出規則数も最大数（複合語規則数 18；抽出評価規則数 7；削除規則数 4；不要語数 83）を定義した．文書サイズは約 34 KB，総形態素数は 10,129 であり，形態素解析時間は，形態素辞書などすべてディスクキャッシュにある環境で 19.92 秒であった．この規則に対する抽出マシン構成時間は 0.543 秒であった．規則の平均長は 2.5，規則構造の属性名の総種類 5 個，規則構造に定義された平均属性名 1.9 個，抽出マシンの最大遷移数は 23 個となった．

この実験データに対して，候補マシンへのキーワード候補数は 2,188；抽出キーワードは長単位語処理で 796 個；短単位処理で 623 個；形態素解析時間を除く抽出時間は 0.658 秒；抽出マシンの照合時間は 0.634 秒；候補マシンの構築時間は 0.024 秒であった．この高速性は，規則構造に定義された平均属性名 1.9 個で分かるように，自由に定義できる属性名を利用して規則構造が効率的に抽象化して記述でき，6.1 節の理論的評価 $O(AB)$ の A の値が大きくならなかったことと，候補マシンによる部分文字列処理の効率化によるものと思われる．特に候補マシンの構成後，短単位語と長単位語の抽出処理時間は，それぞれ 0.066，0.072 秒と非常に高速になる．したがって，候補マシンを構築すれば，多彩なキーワードが $short$, $long$ 等の関数を切り替えるだけで即座に得られるので，本手法の実用性は高いといえる．また，キーワード候補 2,188 個から，文字列比較により短単位語と長単位語の処理を行うと，約 250 倍の処理時間が必要となり，本手法の有効性が分かる．また，同じキーワード候補を入力としたマシン AC の構成時間は 0.022 秒であり，候補マシンの構成時間は約 10%しか低下しないことが分かった．

6.3 キーワード抽出の評価

本論文のもう 1 つの特徴は，分野別あるいは目的別に抽出条件を設定し，その条件を切り替えることで多彩なキーワードを抽出することである．その条件は，抽出マシンに反映される抽出条件と，候補マシンで行われる短単位語，長単位語処理のような出力条件である．実験では一般文書用と論文・特許文書用の条件を定義した．

一般文書用の複合語規則（11 種類）は，名詞，固有名詞，接頭辞，接尾辞から定義され，評価規則（10 種類）は特に主格を表す格助詞について評価点を高くし，指示連体詞に関する規則を削除した．これは一般文書では指示連体詞の後にある語がキーワード性に欠けるものが多いからである．

論文・特許文書用の複合語規則（18 種類）は，一般用の文法規則に数詞関連の規則を追加した．ただし＜名詞－数詞＞および＜接尾辞－数詞＞は，キーワード性に欠けるので追加しない．評価規則（11 種類）は，提題表現に関する規則と指示連体詞の規則を，他の評価規則に比べて優先的に評価した．

この実験では 6.2 節で利用した抽出規則を用いた場合に比べて，論文の概要 13 種類では再現率が

66.89%から73.30%，適合率が18.22%から19.49%に，特許の紹介文13種類では，再現率が64.89%から75.15%，適合率が34.90%から38.97%の向上が見られた．なお，再現率は正解キーワードのうち，どれだけが実際に抽出されたかを示し，適合率は，抽出キーワードのうちどれだけが正解であったかを示す[2]．

なお，頻度情報 $FREC$ (frequency) は，候補マシンの構成時に得られ，out_c 関数の要素 $(KEY, SCR, FREC)$ に組み込むことが容易である．このように，KEY に関する抽出情報を out_c 関数に定義し，提案した $long, short$ 関数の出力条件を自由に設計することで，実際の応用では抽出エンジンを変更することなしに，多彩なキーワードを抽出できる．したがって，提案手法は分野や目的に応じたキーワード抽出方法として高い汎用性と有用性を持つと思われる．

7. 結　論

本論文では，複合語キーワードの抽出に対して，次の新しい手法を提案した．

(1) 抽象化された抽出規則を用いて，集合の包含関係による状態遷移を行う抽出マシンを提案した．
(2) 1度の解析で，キーワード候補の抽出と評価を同時に処理することを可能にした．
(3) 複合語の構成単位による高速な部分文字列処理を行う候補マシンを提案した．

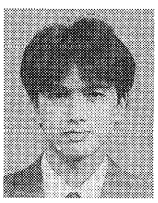
さらに提案手法は理論的かつ具体的に評価され，その有効性が示された．以上より，本論文の提案手法は，キーワード抽出の効率化，汎用性，抽出精度の向上に対して有効であるといえる．

今後，典型的なキーワードが定義されている文書から，抽出条件を学習する研究，また係り受け解析や構文解析を導入した規則の定義と照合アルゴリズムの研究が期待される．

参考文献

1) 中園　薫，白井　諭：日本語牽引自動生成システム，情処自然言語技術シンポジウム (Nov. 1984).
2) 小川泰嗣，望主雅子，別所礼子：複合語キーワードの自動抽出法，情処自然言語処理研究会，Vol.97, No.15, pp.103–110 (1993).
3) 木本晴夫：日本語新聞記事からのキーワード自動抽出と重要度評価，信学論 (D-I)，Vol.J74-D-I, No.8, pp.556–566 (1991).
4) 会森　清，依田　透，嵩原　哲：日本語キーワード抽出システムの開発および今後の課題，ドキュメンテーション・シンポジウム予稿集，pp.15–19 (1988).
5) 山口義一，杉山時之：自然言語による索引語自動抽出システムの概要とその索引語の分析，科学技術文献サービス，No.85, pp.31–40 (1988).
6) 津田和彦，入口浩一，青江順一：ストリングパターンマッチングマシンの動的構成法，信学論 (D-I)，Vol.J77-D-I, No.4, pp.282–289 (1994).
7) 宮崎正弘：係り受け解析を用いた複合語の自動分割法，情報処理学会論文誌，Vol.25, No.6, pp.970–979 (1984).
8) 吉村賢治，日高　達，吉田　将：日本語科学技術文における専門用語の自動抽出システム，情報処理学会論文誌，Vol.27, No.1, pp.33–40 (1986).
9) 伊藤　哲，丹羽寿男，萱島一弘，丸野　進，〆木泰治：利用目的に応じて最適化可能なキーワード抽出手法，信学技報，Vol.NLC93-53, pp.41–46 (1993).
10) 細野公男，後藤智範，諸橋正幸：パターン・マッチングによる重要語の自動抽出，情処自然言語処理研究会，Vol.39, No.1, pp.1–8 (1983).
11) 梅田茂樹，諸橋正幸，細野公男，原田隆史，後藤智範：漢字クラスターによる日本語文献の重要語抽出，情処自然言語処理研究会，Vol.58, No.5 (1986).
12) 宮崎正弘，池原　悟，横尾昭男：単語結合型辞書引きを用いた日英機械翻訳用辞書の構成，情処自然言語研究会，Vol.84, No.12, pp.87–94 (1991).
13) Aho, A.V. and Corasick, M.J.: Efficient String Matching: An Aid to Bibliographic Search *Comm. ACM*, Vol.18, No.6, pp.333–340 (1975).
14) Aoe, J.: An Efficient Implementation of Static String Pattern Matching Machines, *IEEE Trans. Softw. Eng.*, Vol.SE-15, No.8, pp.1010–1016 (1989).
15) Aoe, J.: An Efficient Digital Search Algorithm by Using a Double-Array Structure, *IEEE Trans. Softw. Eng.*, Vol.SE-15, No.9, pp.1066–1077 (1989).

(平成8年6月12日受付)
(平成9年2月5日採録)

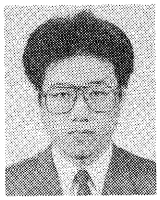


林　淑隆（学生会員）

昭和45年生．平成5年徳島大学工学部知能情報工学科卒業．平成7年同大大学院博士前期課程修了．現在，同大大学院システム工学専攻博士後期課程在学中．自然言語処理，文書処理の研究に従事．情報処理学会会員．

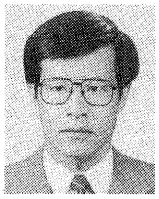
中野 英雄

昭和47年生．平成7年徳島大学工学部知能情報工学科卒業．平成9年同大大学院博士前期課程修了．自然言語処理の研究に従事．



獅々堀正幹（正会員）

昭和40年生．平成3年徳島大学工学部情報工学科卒業．平成5年同大大学院博士前期課程修了．平成7年同大学工学部知能情報工学科助手．情報検索，文書処理，自然言語処理の研究に従事．情報処理学会第45回全国大会奨励賞受賞．電子情報通信学会会員．



青江 順一（正会員）

昭和26年生．昭和49年徳島大学工学部電子工学科卒業．昭和51年同大大学院修士課程修了．同年同大学工学部情報工学科助手．現在同大学工学部知能情報工学科教授．この間コンパイラ生成系，パターンマッチングアルゴリズムの能率化の研究に従事．最近，自然言語処理，特に理解システムの開発に興味を持つ．著書「Computer Algorithms—Key Search Strategies」，「Computer Algorithms—String Matching Strategies」IEEE CS press．平成4年度情報処理学会「Best Author賞」受賞．工学博士．電子情報通信学会，人工知能学会，日本認知科学会，日本機械翻訳協会，IEEE，ACM，AAAI，ACL各会員．