2006－NL－172 (5)
2006／3／27

選好依存文法 (PDG) におけるスコアリング方式と評価実験について

平川 秀樹
(株) 東芝 研究開発センター 知識メディアラボラトリ
email: hideki.hirakawa@toshiba.co.jp

選好依存文法 (PDG) は，自然言語の形態素，構文，意味に渡る複数の圧縮共有データ構造を有し，各レベルの選好知識を活用して文解析を統合的に行う枠組みである．従来は，ノード (単語) 頻度やアーク (依存関係) 頻度といった選好知識を元に最適解探索を行う単項モデルであったが，本稿では，これをノード bigram やアーク共起頻度などを扱える２項モデルに拡張し，２項モデルにおける知識の統合評価方式ならびにその試験的な評価実験について述べる．

Preference Score Model of the Preference Dependency Grammar and its Evaluation

Hideki Hirakawa
Knowledge Media Laboratory
TOSHIBA Corporate Research & Development Center
1, Komukai-Toshiba-cho, Saiwaiku, Kawasaki, 212-8582, Japan

Preference Dependency Grammar (PDG) is a new framework for the morphological, syntactic and semantic analysis using multiple kinds of paked shared data structures to utilize multi-level preference knowledge. This paper proposes a new scoring model, called the binary preference model, which is an extension of the unary preference model adopted in PDG. This paper proposes an preference score integration method and reports the preliminary experiment.

1 はじめに

選好依存文法 (PDG) は，自然言語の形態素，構文，意味にわたる複数の圧縮共有データ構造を有し，各レベルの選好知識を活用して文解析を統合的に行う枠組みである．PDG では，語品詞トレリス (形態素系列)[*1]，ヘッド付き統語森 (句構造木)，(機能) 依存森 (構文的依存木)，意味依存森 (意味依存木) の４つの圧縮共有構造ならびに解釈構造を想定しており，依存森と呼ばれる共有データ構造を生成する．依存森は入力文に対する全依存木集合に対応するデータ構造であり，依存グラフと共起マトリックスより構成されている．図１に依存森 (機能依存森) の例を示す．共起マトリックスは，依存グラフのアークを行・列とし，アークの共起制約を表している．この共起制約を満足する木を整依存木と呼び，整依存木の集合が入力文に対する解釈の集合となる [11]．依存グラフのアークには，選好スコアが付与されており，このスコアの和が解釈 (依存木) の選好度を表す．依存森から最大のスコアをもつ整依存木を探索することで最適な解釈を得る．依存森中の最適な整依存木は，分枝限定法に基づくグラフ分枝アルゴリズムにより求めることができる [5]．

アークの選好スコアは，各種レベルの選好知識によるスコアを統合して付与される．例えば，グラフのノードに対応する語品詞のコーパス頻度や，ノード間の依存関係のコーパス頻度などを統合してアークスコアとすることができる．この従来モデルでは，ノードとアークのスコアは、それぞれ、各々が独立であり，その意味で単項選好モデルとなっている．本稿では，単項選好モデルの自然な拡張である PDG の２項選好モデルを提案し，２項モデルに基づいた選好スコアの統合手法を提案し，予備的な実験結果について報告する．

2 PDG の２項選好モデルへの拡張

本章では，２つのアークにより決定される選好スコアを扱う２項選好モデル (２項モデルとも呼ぶ) の

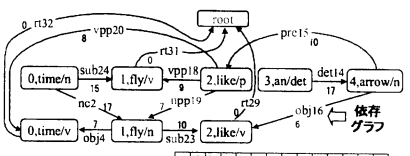
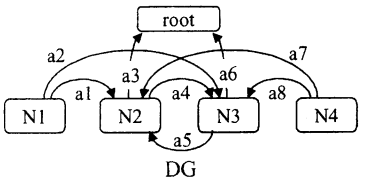


図１: “Time flies like an arrow” に対するスコア付き依存森

[*1] 語品詞は単語と品詞のペアである．例えば，“time” は “time/n”，“time/v” を語品詞として有する．



CM

		N1		N2		N3		N4	
		1	2	3	4	5	6	7	8
N1	1	−		○		○		○	
	2		−		○		○		○
N2	3	○		−		○		○	
	4		○		−		○		○
N3	5	○		○		−		○	
	6		○		○		−		○
N4	7	○		○		○		−	
	8		○		○		○		−

PM

		N1		N2		N3		N4	
		1	2	3	4	5	6	7	8
N1	1	10						10	5
	2		15		5			5	20
N2	3								
	4		5		15				
N3	5					20		-5	
	6								
N4	7	10	5			-5		10	
	8	5	20						10

図 2: 2項選好モデルの依存森の例

PDG について述べる.

2.1 依存森の拡張

本章では，依存森を2項モデルに拡張し，最適依存木の定義を与える.

(1) 選好マトリックス

2項のアークのスコアは，選好マトリックス (PM:Preference Matrix) と呼ばれる新規のマトリックスにより表現される.

図2に2項選好モデルの依存森を示す．アーク i とアーク j の間の選好スコアは PM(i,j) のスコアで表現される．PM(i,i),PM$(i,j)(i\neq j)$ は，それぞれ単項アークスコア，2項アークスコアを示す．単項アークスコアは単項モデルのスコア (アークに付与されたスコア) と同等である．選好スコアは負の値を取ることができる．空セルのスコアは 0 点であり，選好も非選好もしないニュートラルな値である．選好マトリックスの値がどんなに小さくても，他に可能な解釈がなければその解釈は解として選択される．制約マトリックスの場合は，解釈が完全に棄却 (枝刈り) される．この点が選好と制約の違いである.[*2]

図2に2項モデルの依存森 <DG,CM,PM> の形式的な例を示す．2項モデルの共起マトリックスは，単項モデルと同じであるが，選好マトリックスとの対応上，制約マトリックス (CM: Constraint Matrix) と呼ぶ．選好マトリックスの対角要素のスコアが単項アークスコア，それ以外が2項アークスコアである．この依存森は，$\{1,3,5,7\}$,$\{2,4,6,8\}$ の2つの整依存木を持っている.

(2) 2項モデルの最適依存木

単項モデルの依存木スコアは，依存木中のアークのスコアの総和と定義されている．2項モデルの依存木のスコアは次のように定義する.

$$\mathrm{score}(DT) = \sum_{a_i,a_j \in \mathrm{DT}, i \leq j} \mathrm{PM}(a_i, a_j) \qquad (1)$$

このスコアは，次のように依存木 DT 中のアークスコアの和として表現できる.

$$\mathrm{score}(DT) = \sum_{a_i \in \mathrm{DT}} \mathrm{arc\_score}(a_i, \mathrm{DT}) \qquad (2)$$

$$\mathrm{arc\_score}(a_i, \mathrm{DT}) = \mathrm{PM}(a_i, a_i) + \frac{1}{2} \sum_{a_j \in \mathrm{DT}, a_j \neq a_i} \mathrm{PM}(a_i, a_j) \qquad (3)$$

2項モデルのスコア定義は単項モデルの定義の一般化になっている.

2.2 グラフ分枝アルゴリズムの拡張

本節では，単項選好モデルの最適解探索アルゴリズム（グラフ分枝アルゴリズム）の2項モデルへの拡張について述べる．単項モデルに関するアルゴリズムの詳細については [5] を参照されたい.

2項モデルのグラフ分枝アルゴリズムは，単項モデルと同様に分枝限定法 [9] に基づいている．本節では，文献 [5] のグラフ分枝アルゴリズムの各コンポーネントに対する拡張部分について説明する.

(1) 部分問題

部分問題には，新たに選好マトリックス PM が追加される．PM は全ての部分問題で共有される.

(2) 暫定解と下界値を求めるアルゴリズム

暫定解と下界値を求めるアルゴリズムは基本的に単項モデルと同様である．違いは，アークのスコア計算である．単項モデルは単純に暫定解のアークの単項スコアの和であるが，2項モデルでは上記3式により計算される値となる.

(3) 上界値を求めるアルゴリズム

長さ n の入力文に対する依存森 <DG,CM,PM> が得られた時，部分問題 P は，DG の部分集合である DG' を依存グラフとして持つ．P に対する上界値 G は，依存森 <DG',CM,PM> に関して次のように定義される.

$$\mathrm{G} = \sum_{i=0}^{n-1} \max_{A \in \mathrm{arcs\_at}(i, \mathrm{DG}')} \mathrm{ubs\_arc}(A) \qquad (4)$$

$$\mathrm{ubs\_arc}(A) = \sum_{j=0}^{n-1} \mathrm{ub\_arc\_score}(\mathrm{A}, \mathrm{j}) \qquad (5)$$

ここで，ub_arc_score(A,j) は、position(A)=j の時 PM(A,A)，position$(A)\neq j$ の時，次となる。

$$\max_{X \in \mathrm{arcs\_at}(j, \mathrm{DG}'), \mathrm{CM}(A,X)=○} \frac{\mathrm{PM}(A,X)}{2}$$

式4は，上界値 G は各入力位置における ubs_arc の最大値の和であることを意味している．ubs_arc(A) は，A の単項アークスコアの和，すなわち PM(A,A) と，各位置のアークと A の間で ub_arc_score により定義される最大の2項アークスコアの和である．式4において選択されるアークの集合は，最尤整被覆アーク集合 (the maxinum well-covered binary arc set) と呼ぶ．このアーク集合は一般に木ではなく，また，ub_arc_score で選択されたアークと整合している必要もない.

(4) 分枝操作

分枝操作は，基本的に単項モデルと同様の処理である．不整合アークペア (arc_i, arc_j)(CM(i,j)$\neq$ ○) が

[*2] 但し，選好スコアの値域を特定の範囲に限定するなどすれば，マトリックス自体を1本化することは可能である.

グラフ分枝のために最尤整被覆アーク集合より取り出される．不整合アークペアが存在しない場合は，最尤整被覆アーク集合はその部分問題に対する最適解の1つである．全最適解を探索する場合は，下記 (6) に示すように全部分問題が最適解を持たないと保障されるまで分枝操作は継続する．

(5) 活性問題集合からの部分問題の選択
このプロセスは単項モデルと同じ最良上界探索戦略を採用する．

(6) 全解の探索
部分問題に対する新規の最適解が得られた時，その最適解は暫定解リストに追加され，さらに分枝操作が継続される．部分問題の依存グラフから除去されるアーク，すなわち分枝操作の対象アークは，得られた最適解中のアークのライバルアーク (位置と上界値スコアが同等以上のアーク) を取り出す．全ての部分問題の上界値が暫定値より小さくなると探索プロセスは終了する．

3 スコアリング

PDG では各種レベルでの各種選好知識を統合して扱う．スコアリングプロセスは，こうした選好知識を統合して依存森の選好マトリックスを構築するプロセスである．本稿では，依存森と最適解探索アルゴリズムに関するスコアリングのフレームワークを示し，いくつかの種類の多層の選好知識を統合するスコアリングの第一ステップについて述べる．

3.1 選好知識とスコア統合

3.1.1 スコア統合の方針

PDG のスコアリング方式に関して，次の点を考慮する．

(a) 選好知識としてはコーパスから得られるデータを利用する．
(b) 異なったコーパスから得られる異なった種類の選好知識を利用する (知識ソースに対する頑健性)
(c) スコアリングにおけるパラメータを学習により最適化する．

選好知識を人手開発することは非常に困難であるため，(a) は，大規模な選好知識を獲得する上で最も良い手法であると思われる．人間の直感とコンピュータの大規模処理能力との協調が大規模知識開発では有効であり，これは，規則ベースと統計ベースの手法の統合 [8, 7] を必要とすると考えている．(b) は大規模知識開発に対する要求項目である．種々の選好知識が種々のコーパスから得られている場合，それを旨く統合利用できることは非常に有用な性質であると考えている．(c) は最適な言語解析システムを構築するために必要であると考えている．本稿では，実際の学習手法については今後の課題とするが，スコアリングの式には各種のパラメータを導入している．

3.1.2 スコア統合のベース

選好知識を統合するためには，それらはなんらかの数値 (選好スコア) に変換する必要がある．選好知識に関する記述能力は，記述スキーマと最適解探索手法により規定される．例えば，語品詞 bigram の選好スコアは語品詞トレリスのアークのスコアで表現され，ビタビアルゴリズムにより最適な語品詞系列が計算できる．しかしながら，語品詞 trigram の選好スコアは，この方法では直接扱うことはできない．PDG は依存森を選好スコア統合のベースとして採用している．このため，語品詞トレリス，ヘッド付き統語森，機能依存森，意味依存森の各レベルの選好知識から得られる選好スコアは，依存森に統合される必要がある．現在開発中の PDG の実験システムは，機能依存森までを実装しており，入力文に対する機能依存木を計算する．以下では，機能依存森を対象に議論をする．意味依存森への対応は今後の課題とする．

既に述べた様に依存森の選好マトリックス PM は，単項アークスコア (対角要素) と2項アークスコアを表現している．アークは，依存ノードと支配ノードを含んでいるため，ノードに関するスコアは，アークのスコアにより表現することができる．すなわち，単項ノードスコアは，単項アークスコアに，2項ノードスコアは2項アークスコアに統合することができる．PM により表現可能な4つのスコア (単項ノードスコア，単項アークスコア，2項ノードスコア，2項アークスコア) が全ての選好知識のベースとなる．PM の要素の値は，unary_score 関数と binary_score 関数で定義される．

$$\mathrm{PM}(a_i, a_j) = \begin{cases} \tau \cdot \mathrm{unary\_score}(a_i) & (i = j) \\ (1-\tau) \cdot \mathrm{binary\_score}(a_i, a_j) & (i \neq j) \end{cases}$$

τ $(0 \leq \tau \leq 1)$ は，単項/二項分配率 (UB 率) と呼ばれ，単項スコアと2項スコアの調整に利用される．単項スコア，2項スコアは次節で述べる．3項以上の要素にからむ選好知識[*3]は，現状の PDG のスコアリング処理の対象外である．

3.2 スコア化関数とスケーリング係数

コーパスから得られる大半の選好知識は，単語，語品詞，語品詞系列，句構造，依存関係など，言語的要素や関係の頻度として表現される．こうした頻度情報は，統合処理のベースとなる選好スコアに変換する必要があり，スコア化関数と呼ぶヒューリスティック関数により行われる．次の要素 E に対する "logave" 関数がスコア化関数の基本形である．

$$\mathrm{basic\_score}(E) = \mathrm{logave}(X, AddX, AveX) = \mathrm{BaseScore} \cdot \frac{\log((X+1) + AddX)}{\log(AveX + 1)}$$

ここで，X は要素 E の頻度，$AddX$ は頻度補整項，$AveX$ は E が属しているデータタイプの頻度である．$BaseScore$ は，平均頻度に対して割りあてられる標準スコアであり，1000 に設定されている．例えば，単語"theorem" のコーパス中の頻度が 99 であり，平均の単語頻度が 9 である場合，"theorem" の基本スコアは，頻度補整が 0 である場合には，次に示すように 2000 となる．

$$\mathrm{basic\_score}(theorem) = \mathrm{logave}(99, 0, 9) = 1000 \cdot \frac{\log(99+1)}{\log(9+1)} = 2000$$

[*3] 例えば，語品詞 trigram，4つ以上の構成素を持つ句構造規則

logave の頻度は 1 でバイアスされており，頻度 0 の要素のスコアは 0 となる．X が $AveX$ と等しい場合 (E が平均頻度を持つ) はスコアは $BaseScore$ となる．これは頻度の標準化のために導入されている．"log" は，頻度を平準化するために導入されている．理論的な裏づけは特にないが，予備実験の結果，この平準化なしでは結果が悪くなるため導入されている．頻度補整項は，例えば，合成語に対する単語頻度の補整 (合成語は頻度が低いが，選好度は高い) に利用される．

PDG は，スケーリング係数と呼ぶ別のタイプの関数を導入している．スケーリング関数は，要素 E に対するスケーリング係数を生成する．スケーリング関数'f' が要素 E に対して定義されていると，基本的に要素 E に対するスコアは，スコア化関数とスケーリング関数の積となる．

$$\mathrm{score}(E) = \mathrm{f}(E) \cdot \mathrm{basic\_score}(E)$$

スケーリング関数は，要素 E の分布や重要度を表すヒューリスティック関数である．

3.3 単項スコア式

単項スコアは，基本的に単項ノードスコアと単項アークスコアの和である．

$$\mathrm{UnaryScore} = \frac{\alpha \cdot \mathrm{UnaryNodeScore} + (1-\alpha) \cdot \mathrm{UnaryArcScore}}{2}$$

ここで，$\alpha(0 \leq \alpha \leq 1)$ は，単項ノード/ アーク分配率 (UNA 率) である．

3.3.1 単項ノードスコア式

現状の実装では，単項ノードスコアとしては，コーパス中の語品詞頻度を採用している．PDG 実験システムの辞書は，複合語を含んでいるため，複合語に対しては頻度補整を行っている．

語品詞ノード N に対する単項ノードスコアは次の式により計算される．

$$\begin{aligned}&\mathrm{unary\_node\_score}(N)\\ &= \mathrm{logave}(\mathrm{freq}(N), \mathrm{un\_comp}(N), \mathrm{AveWPPF})\\ &= BaseScore \cdot \frac{\log(\mathrm{freq}(N) + 1 + \mathrm{un\_comp}(N))}{\log(\mathrm{AveWPPF} + 1)}\end{aligned}$$

ここで，$AveWPPF$ は，語品詞頻度平均である．un_comp(N) (unary node compensation) は，複合語の補整項であり，N が複合語以外では 0，複合語の時に次となる．

$$\mathrm{element\_freq}(N) + \mathrm{AveWPPF} \cdot CWC \cdot 2^{\mathrm{wrdlen}(N)-1}$$

ここで，element_freq(N) は，N 中の各単語の頻度の合計，CWC (Compound Word Coefficient) は，合成語と非合成語の選好を調整するパラメータ，wrdlen(N) は，N の語数である．

一般に複合語は，その要素である語に比べて頻度がかなり低いが，より高い選好度を持つべきであると考えられる．最初の項 element_freq(N) は，複合語が構成要素の語の総和より高い選好度を持つことを保障し，第 2 項が合成語の単語長に応じた補整 (長い合成語はより優先される) を行う．現状では CWC は 3 にセットされている．

3.3.2 単項アークスコア

単項アークスコアは，コーパス中の依存関係 (依存ノード，支配ノード，その間の依存関係の 3 次組) の頻度を基本的なリソースとしており，次の式により求められる．

$$\begin{aligned}&\mathrm{unary\_arc\_score(A)}\\ &= \mathrm{basic\_arc\_score(A)} \times \mathrm{distance\_ratio(A)} \times \mathrm{POS\_ratio(A)}\end{aligned}$$

basic_arc_score は，A に対する基本スコアを与える．distance_ratio と POS_ratio は，それぞれ，依存ノードと支配ノードの距離とアークのタイプに基づく補整を行うスケーリング係数である．

(1) basic_arc_score(A)

basic_arc_score は，アーク A のコーパス頻度 (ASIS アーク頻度と呼ぶ) から基本アークスコアを計算する．この標準的な頻度に加えて，3 つの付加的なアーク頻度が選好スコアのリソースとして用いられる．

(a) ASIS アーク頻度 (ASIS_AF)：アークのコーパス頻度
(b) 汎化アーク頻度 (GEND_AF)：一般化アークのコーパス頻度
(c) ASIS PP 頻度 (ASIS_PF)：前置詞付加のコーパス頻度
(d) 汎化 PP 頻度 (GEND_PF)：一般化された前置詞付加のコーパス頻度

アークは，依存ノード，支配ノード，依存関係の 3 要素よりなる複雑なデータであり，データスパースネスの問題を考慮する必要がある．(b) は，バックオフによりこの問題に対応するために導入している．汎化アークは，依存ノードと支配ノードの品詞を，大分類の品詞に汎化することにより得られる．

アーク：[time-nx] $\xrightarrow{\text{subj}}$ [fly-vt]
汎化アーク：[time-n] $\xrightarrow{\text{subj}}$ [fly-v]

(a) と (b) は，前置詞の付加に関する優先度を旨く記述できないため，これを補整するために，(c) と (d) が導入されている．次は，"see girl with telescope" に対する PP 付加の例を示している．

(e1) [see-vt] $\xleftarrow{\text{vpp}}$ [with-pre] $\xleftarrow{\text{pre}}$ [telescope-n]
(e2) [girl-n] $\xleftarrow{\text{npp}}$ [with-pre] $\xleftarrow{\text{pre}}$ [telescope-n]

(e1) と (e2) 中のアークの頻度は，2 つの前置詞付加の競合を旨く表現することができていない．この理由は，"[see-vt] $\xleftarrow{\text{vpp}}$ [with-pre]" と "[girl-n] $\xleftarrow{\text{npp}}$ [with-pre]" が "[with-pre] $\xleftarrow{\text{pre}}$ [telescope-n]" と独立であることから生じている．この問題は，次のように前置詞ノードをアークの関係に縮退させると解決する．

(e3) [see-vt] $\xleftarrow{\text{vpp\_with}}$ [telescope-n]
(e4) [girl-n] $\xleftarrow{\text{npp\_with}}$ [telescope-n]

現状の PDG は，このような構造をとっていない．この問題を解決するために (c) と (d) を導入している．(c) の ASIS PP 頻度は，上記 (e3) または (e4) の関係の頻度に相当し，汎化 PP 頻度はデータスパースネス

の補整のために導入されている．汎化された関係は，次に示すような (e3) に対応する関係を表している．

(e5) [see-v] $\xleftarrow{\text{with}}$ [telescope-n]

基本的なアークスコアは次のように定義されている．

$$\text{basic\_arc\_score}(A) = \mu \cdot \text{asis\_arc\_score}(A) + (1-\mu) \cdot \text{generalized\_arc\_score}(A)$$

asis_arc_score と generalized_arc_score は，それぞれ ASIS アーク頻度と汎化アーク頻度より計算されるスコアである．μ は，ASIS/汎化アーク分配率と呼ばれる分配率である．以下，これらを順に示す．

$$\text{asis\_arc\_score}(A) = \text{logave}(\text{asis\_max\_freq}(A), 0, \text{AveASIS\_AF}) = \text{BaseScore} \cdot \frac{\log(\text{asis\_max\_freq}(A)+1)}{\log(\text{AveASIS\_AF}+1)}$$

ここで、asis_max_freq(A) は、A の依存ノード DN と支配ノード GN のいずれかまたは両者が複合語の場合は，

$$\max\{\text{freq}(\text{Rel}, \text{I}, \text{J}) | \text{I} \subseteq \text{DN}, \text{J} \subseteq \text{GN}\} + \text{PPC} \cdot \text{asis\_pf}(A)$$

であり、さもなければ、

$$\text{asis\_af}(A) + \text{PPC} \cdot \text{asis\_pf}(A)$$

である．Rel,DN,GN は，アーク A の関係，依存ノード，支配ノードを示し，freq(Rel,DN,GN) は A の頻度である．asis_af と asis_pf は，それぞれ，(a) と (c) の頻度である．PPC は，前置詞付加の効果を調整する係数であり，現在は 5 に設定されている．AveASIS_AF は，コーパス中の ASIS アーク頻度の平均である．同様に，アーク A の汎化アークスコアが定義される．

$$\text{generalized\_arc\_score}(A) = \text{logave}(\text{gend\_max\_freq}(A), 0, \text{AveGEND\_AF}) = \text{BaseScore} \cdot \frac{\log(\text{gend\_max\_freq}(A)+1)}{\log(\text{AveGEND\_AF}+1)}$$

ここで，gend_max_freq(A) は，A の汎化された依存ノード GD_DN と汎化された支配ノード GD_GN のいずれかまたは両者が複合語の場合は，

$$\max\{\text{freq}(\text{GD\_Rel}, \text{I}, \text{J}) | \text{I} \subseteq \text{GD\_DN}, \text{J} \subseteq \text{GD\_GN}\} + \text{PPC} \cdot \text{gend\_pf}(A)$$

であり、さもなければ、

$$\text{gend\_af}(A) + \text{PPC} \cdot \text{gend\_pf}(A)$$

である．ここで，GD_Rel,GD_DN および GD_GN は，アーク A の汎化関係，依存ノード，支配ノードである．freq(GD_Rel,GD_DN,GD_GN) は，汎化アークの頻度，gend_af および gend_pf は，それぞれ上記の (b),(d) の頻度である．AveGD_AF は，コーパス中の汎化アークの頻度の平均である．

現在の実装では，ASIS/汎化アーク分配率は，ASIS 頻度と汎化頻度の全体的影響が等しくなるように設定されている．

$$\mu = \frac{\text{ASIS\_KIND\_NUM}}{\text{ASIS\_KIND\_NUM} + \text{GEND\_KIND\_NUM}}$$

ここで，ASIS_KIND_NUM,GEND_KIND_NUM は，それぞれ，コーパス中の ASIS アークの種類の数，汎化アークの種類の数である．

(2) スケーリング係数 distance_ratio(A)

距離率 distance_ratio は，依存ノードと支配ノードの距離によりアークのスコアを補整する．文献 [3] は，単語の 95.6%,99.0%が，それぞれ，単語距離 5,10 の間にその支配語を有していると報告している．また，距離パラメータは，各種解析システムで利用されている．[4, 2, 6].

今，アーク X,Y が同じ関係，依存ノード，支配ノードを持つ時，same(X,Y) と記述する．また，distance(A) を支配ノードと依存ノードの間の距離とすると，距離率は次のように定義する．

$$\text{distance\_ratio}(A) = 1 + \text{K} \cdot \log(\text{distance\_degree}(A))$$

ここで，K は，距離率の度合いを調整するパラメータであり，現在の設定は次のようである．

$$\text{K} = \frac{0.5}{\log(10)}$$ *4

基本的に，distance_degree は，アーク A に関して，距離 D をもつアークの頻度のアーク A 全体の平均に対する割合である．

$$\text{distance\_degree}(A) = \text{logave}(\text{df}(A), \text{dfc}(A), \text{average\_df}(A))$$

ここで，$df(A)$ は，$same(X,A)$, $distance(X) = distance(A)$ であるようなアーク X の頻度である．dfc は，次で定義される距離頻度補正を行う．

$$\text{dfc}(A) = \begin{cases} 1 & (\text{df}(A) = 0) \\ 0 & (\text{それ以外}) \end{cases}$$

$average\_df(A)$ は，次に示すように定義される．

$$\text{average\_df}(A) = \frac{1}{|\text{DS}|} \sum_{D \in \text{DS}} \text{freq}(A, D)$$

ここで，DS $= \{D_i | D_i = \text{distance}(A_i), \text{same}(A_i, A), A_i \in \text{CorpusArcs}\}$ である．$freq(A,D)$ は，$same(X,A)$, $distance(X) = D$ であるようなアーク X の頻度である．

(3) スケーリング係数: POS_ratio(A)

アークタイプ率 POS_ratio(A) は，アークのタイプに応じてアークスコアの補正を行う．アークタイプは主にアークの依存ノードのタイプにより決定される．

$$\text{POS\_ratio}(A) = \begin{cases} 0.01 & (A \text{の依存ノードが冠詞}) \\ 0.2 & (\text{依存ノードが } be, \text{関係が } subj) \\ 1 & (\text{その他}) \end{cases}$$

最初の補正項は，冠詞の影響を軽減するために導入している．冠詞を含むアークの頻度は大きいが，冠詞の依存先は全体構造を決める上で重要度は低い．2番目の補正項は，be 動詞の2つの解釈 (copula と現在進行形) の補正のために導入されている．これらの2つの補正項は，解析結果を目視した結果得られた知見を元に組み込まれている．こうした補正は，将来的には学習により自動的に調整されるべきものである．

3.4 2項スコア式

2項スコアは，基本的に2項ノードスコアと2項アークスコアの和である．

$$\text{BinaryScore} = \frac{\beta \cdot \text{BinaryNodeScore} + (1-\beta) \cdot \text{BinaryArcScore}}{2}$$

*4 もし distance_degree(A) が 10 なら,distance_ratio は 1.5 となり，アークスコアは 1.5 倍される．

ここで，$\beta(0 \leq \beta \leq 1)$ は，二項ノード/アーク分配率である．

3.4.1 2項ノードスコア式

現状の実装では，2項ノードスコアとしては，コーパス中の語品詞の bigram 頻度を採用している．汎化語品詞の bigram 頻度もデータスパースネス対応のため組合せ利用している．2項ノードスコアは，ノード N に対する次の bigram 頻度を基に計算される．

(a) GWPP_BGM 頻度 (WPP_BF)：汎化 WPP bigram の頻度
ex. [time-nx] [fly-vt] (WPP bigram) → [time-n] [fly-v] (GWPP bigram)

(b) POS_BGM 頻度 (POS_BF)：POS bigram の頻度
ex. [time-nx] [fly-vt] (WPP bigram) → [nx] [vt] (POS bigram)

2項ノードスコアは次のように定義されている．

$$\begin{aligned}&\text{binary\_node\_score}(N1, N2)\\&= \chi \cdot \text{GWPP\_BGM}(N1, N2)\\&\quad +(1-\chi) \cdot \text{POS\_BGM}(N1, N2)\end{aligned}$$

GWPP_BGM と POS_BGM は，それぞれ GWPP bigram 頻度と POS bigram 頻度から計算されるノードスコアを表している．χ は,BIGRAM 分配率と呼ばれ，後ほど定義を与えることとし，まず，GWPP_BGM，POS_BGM について述べる．

GWPP_BGM($N1$,$N2$) は，次で定義される．

$$\begin{aligned}&\text{GWPP\_BGM}(N1, N2)\\&= \text{logave}(\text{GWPPBGM\_freq}(N1, N2),\\&\qquad \text{GWPPBGM\_comp}(N1, N2), \text{AveGWPP\_BF})\\&= \text{BaseScore}^{\frac{\log(\text{GWPPBGM\_freq}(N1,N2)+1+\text{GWPPBGM\_comp}(N1,N2))}{\log(\text{AveGWPP\_BF}+1)}}\end{aligned}$$

AveGWPP_BF は，汎化語品詞の bigram 頻度の平均である．GWPPBGM_comp は，次で定義される複合語に対する補正項である．

$$\begin{aligned}&\text{GWPPBGM\_comp}(N1, N2) =\\&\text{CBC} \cdot \text{AveGWPP\_BF} \cdot (\text{wrdnum}(N1) + \text{wrdnum}(N2) - 2)\end{aligned}$$

ここで，CBC は合成語の頻度補正の度合いを調整する係数であり，現在は 3 に設定されている．

POS_BGM($N1$,$N2$) は，次で定義される．

$$\begin{aligned}&\text{POS\_BGM}(N1, N2)\\&= \text{logave}(\text{POSPBGM\_freq}(N1, N2),\\&\qquad \text{POSBGM\_comp}(N1, N2), \text{AvePOS\_BF})\\&= \text{BaseScore}^{\frac{\log(\text{POSBGM\_freq}(N1,N2)+1+\text{POSPBGM\_comp}(N1,N2))}{\log(\text{AvePOS\_BF}+1)}}\end{aligned}$$

AvePOS_BF は，コーパスにおける POS bigram の頻度の平均である．POSBGM_freq は，次で定義される頻度補正項である．

$$\begin{aligned}&\text{POSBGM\_comp}(N1, N2)\\&= \text{CBC} \cdot \text{AvePOS\_BF} \cdot (\text{wrdnum}(N1) + \text{wrdnum}(N2) - 2)\end{aligned}$$

現状の実装では，bigram 分配率は，汎化 bigram 頻度と POS bigram 頻度の影響が全体的に等しくなるよう，次のように設定されている．

$$\chi = \frac{\text{GWPP\_BGM\_NUM}}{\text{GWPP\_BGM\_NUM} + \text{POS\_BGM\_NUM}}$$

ここで，GWPP_BGM_NUM,POS_BGM_NUM はそれぞれ，汎化語品詞 bigram の種類の数と POS bigram の種類の数である．

3.4.2 2項アークスコア式

2項アークスコアは，単項アークスコアと競合する，より広いコンテキストに基づく選好知識を表現できる．例えば，“eat gasoline” は，“gasoline” は通常 “eat” できないことから低い単項アークスコアを持つ可能性があるが，“This car eats gasoline” という文ではこの限りでなく，“eat gasoline” の選好スコアは “eat” の主格により変化すると言える．この種の選好知識は2項アークの共起のスコアにより表現される．

現実装では，2項アークスコアとしては，コーパス中の文に対する依存木中の連接アーク（親子関係，兄弟関係にあるアーク）に対する頻度を採用している．データスパースネス対応のため，連接アーク頻度として，汎化語品詞を用いたものと，さらに品詞情報は don't care にしたもの（単語の同一性のみで一致を取ったもの）の2種類を用いている．

(a) 汎化語品詞 CAC 頻度 (GWPPCACF)：汎化 WPP をノードとした連接アークの頻度

(b) 単語 CAC 頻度 (WRDCACF)：単語をノードとした連接アークの頻度

アーク $A1$,$A2$ に対する2項ノードスコアは次のように定義されている．

$$\begin{aligned}&\text{binary\_arc\_score}(A1, A2) =\\&\psi \cdot \text{GWPP\_CAC}(A1, A2) + (1-\psi) \cdot \text{WRD\_CAC}(A1, A2)\end{aligned}$$

GWPP_CAC と WRD_CAC は，それぞれ GWPPCACF と WRDCACF から計算されるノードスコアを表している．ψ は，連接アーク分配率と呼ばれ，後ほど定義を与えることとし，まず，GWPP_CAC と WRD_CAC について述べる．

GWPP_CAC($A1$,$A2$) は，次で定義される．

$$\begin{aligned}&\text{GWPP\_CAC}(A1, A2)\\&= \text{scr}_1\text{ogave}(\text{GWPPCAC\_freq}(A1, A2),\\&\qquad \text{GWPPCAC\_comp}(A1, A2), \text{AveGWPPCAC\_FR})\\&= \text{BaseScore}^{\frac{\log(\text{GWPPCAC\_freq}(A1,A2)+1+\text{GWPPCAC\_comp}(A1,A2))}{\log(\text{AveGWPP\_CAC}+1)}}\end{aligned}$$

AveGWPPCAC_FR は，汎化語品詞に関する連接アーク頻度の平均である．GWPPCAC_comp は，複合語に対する補正項であり，現状では合成語の超過単語数と平均頻度の 3 倍をアサインしている．

WRD_CAC($A1$,$A2$) は，次で定義される．

$$\begin{aligned}&\text{WRD\_CAC}(A1, A2)\\&= \text{logave}(\text{WRDCAC\_freq}(A1, A2),\\&\qquad \text{WRDCAC\_comp}(A1, A2), \text{AveWRDCAC\_FR})\\&= \text{BaseScore}^{\frac{\log(\text{WRDCAC\_freq}(A1,A2)+1+\text{WRDCAC\_comp}(A1,A2))}{\log(\text{AveWRDCAC\_FR}+1)}}\end{aligned}$$

AveWRDCAC_FR は，コーパスにおける単語に関す

る連接アーク頻度の平均である．WRDCAC_freq は，複合語に対する補正項であり，現状では合成語の超過単語数と平均頻度の 3 倍をアサインしている．

現状の実装では，連接アーク分配率は，汎化語品詞 CAC 頻度と単語 CAC 頻度の影響が全体的に等しくなるよう，次のように設定されている．

$$\psi = \frac{\text{GWPP\_CAC\_NUM}}{\text{GWPP\_CAC\_NUM} + \text{WRD\_CAC\_NUM}}$$

ここで，GWPP_CAC_NUM と WRD_CAC_NUM は，それぞれ，汎化語品詞連接アークの種類の数と単語連接アークの種類の数である．

4 評価実験

4.1 実験環境

実験対象コーパス，正解データ，PDG の文法ならびに選好知識を用意した．正解データは，文に対する正解依存木である．選好知識としては，3 章で述べた，単項/２項ノードスコア，単項/２項アークスコアの各種を採用している．実験対象の素材は，マニュアルなどからなる英語の技術文書約 62 万文 (463 万語[*5]) である．正解データならびに語品詞情報を大規模に用意するために，PDG とは別の既存の文解析システム [1](これをオラクルシステムと呼ぶ) の出力結果を正解データや選好知識源として用いることとした．オラクルシステムは，長年蓄積改良されてきた規則ベースの大規模文法を有し，技術文書，Web 文書，メール文書などの翻訳に利用されている．以下，正解データならびに語品詞の頻度情報の準備について説明する．

元コーパスには，文抽出の誤りに起因する非文や技術文書に特有な表やインデックスなどの文要素以外の表現が多数含まれている．また，オラクルシステムで構文解析に失敗する文は，正解の生成ができない．このため，次の文は，実験対象 62 万文から除外した．

(a) オラクルシステムで構文解析失敗の文 (7.1 万文)
(b) (a) 以外でピリオドで終わらない文 (20.4 万文)
(c) (b) 以外で大文字で始まらない文 (22 万文)

この結果，実験対象データとして 125320 文 (1844758 語,14.7 語/文) が得られた．これらの文に対しては，オラクルシステムにより，正解語品詞，正解依存木が付与されている．

15 文に 1 文の割合でオープンデータを抽出し，実験対象データをオープンデータ (8605 文,126684 語,14.7 語/文) とクローズデータ (116715 文,1718074 語,14.7 語/文) に分離した．オープンデータは評価実験に利用し，クローズデータは，語品詞頻度の算出に用い，その結果を選好知識のベースとして用いた．抽出語品詞数は，186.9 万件 (異なり数 44470) となった．語品詞数は形態素解析を行なった結果の語数 (形態素数) であるため，wc でカウントしている単語数とは一般に一致しない．オラクルシステムの実際の精度の概要をつかむため，対象文 (PDG で構文解析が成功する文) よりオープンデータの語数分布に大まかにあわせて文長 4〜22 の文をランダムに 136 文選択し，人手による正解との比較を実施した．この結果，オラクルシステムのアーク正解率は 97.2%であり，対象データに対しては人手正解との大幅な乖離はなかった．

文法に関しては，基本的な構文を解析する小規模な基本文文法を用意した．基本文法は，名詞/動詞/形容詞/副詞句/前置詞句など，単文/重文/複文など，命令/平叙/疑問文，関係節/分詞節など，5 文型/THERE 構文などをカバーするが，挿入，省略，倒置，慣用表現的構文 (ex. not only 〜 but also) などは基本的に含んでいない．任意構成要素を展開した文法規則数は，907 規則である．この文法は，オラクルシステムと同じ体系の依存構造を出力する．また，形態素解析は，オラクルシステムと実験システムとで同一のデータを使用している．

4.2 実験結果

Prolog で実装した PDA のインタプリタを用い，上記オープンデータ 8605 文に対して，基本文文法を用いて評価実験を行なった．処理系リソースの制約から単語数 23 以上の文は対象外とし，対象文数は 6882 文となった．このうち構文解析に成功した文は 4334 文であり，解析成功率は 63%である．解析失敗文については，通常部分解析を合成するなどの処理が行なわれるが，本稿では解析成功文のみを分析対象としている．また，スコアリングのパラメータの設定としては，単項/二項分配率=0.5，単項ノード/アーク分配率=0.5，二項ノード/ アーク分配率=0.5 とした．

4.2.1 選好知識別アーク正解率

図 3 にアーク正解率 [10] の入力文長さに対する値を各種選好知識の組合せについて示す．選好スコアのタイプには，単項ノード (und:unary node)，２項ノード (bnd: binary node)，単項アーク (uac:unary arc)，２項アーク (bac:binary arc) の計４種類が存在し，これらの組合せは，選好知識を使わないベースラインの値を入れて全部で１６通り存在する．このうち，ベースライン (base)[*6] ，各種類単体 (und,bnd,uac,bac)，単項ノード+単項アーク (und+uac)，単項ノード+２項ノード (und+bnd)，２項ノード+単項アーク (bnd+uac) の組合せを選

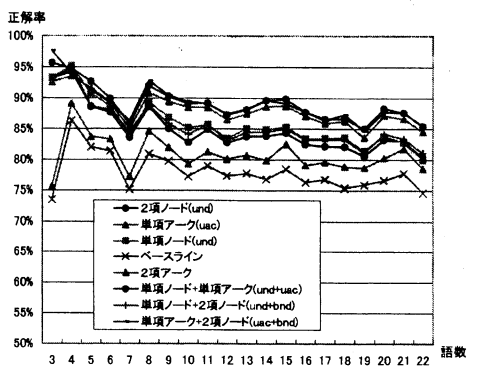


図 3: 選好知識の違いによるアーク正解率の差 (全体文対象)

[*5] unix の wc コマンドで算出した値

[*6] 選好知識の利用がない場合には全解が最適解となる．今回の実験では最適解の最大数を 100 に限定して実験した．

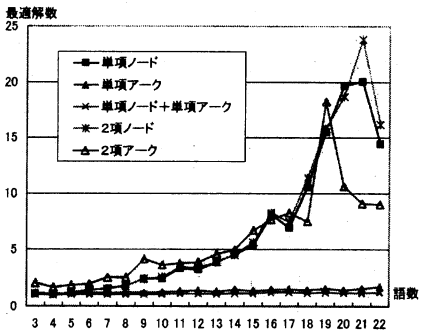


図 4: 選好知識の違いによる平均最適解個数の差

択して図示している．各組合せに対する精度の平均は，base/77.4%,und/84.6%,uac/87.4%,bnd/83.6%,bac/80.4%,und+uac/88.3%,und+bnd/84.3%,und+bac/85.1%,uac+bnd/88.3%,uac+bac/87.0%,bnd+bac/85.0%,und+uac+bnd/83.6%,und+uac+bac/80.4%,und+uac+bnd+bac80.4%である．各種類の選好知識の適用により精度の向上が図れるが，単項アークスコアが最も精度向上に寄与し，次が単項ノードスコアの順になっている．最大の平均精度は，単項ノードと単項アークの2つの知識を組み合わせた場合の 88.3%であり，最も精度向上が低かったのは2項アークの知識を用いた場合 (bac,und+uac+bac,und+uac+bnd+bac) の 80.4%であった．

今回の実験の結果からは，単項ノードと単項アークの2つを組み合わせた選好スコアを用いるのが最も優れた結果を生むことが判明した．単項モデルは，計算量的にも2項モデルに比べて優れている．今回の実験では，全般に単項知識を利用した場合は，2項知識を利用した場合に比べて精度が高い結果となっている．この原因としては，データスパースネスの問題により，2項モデルでは十分な曖昧性解消が行えなかったということも考えられる．この点に関してより大規模なデータを利用した実験が必要である．

4.2.2 選好知識別による最適解数

選好知識は，文解析の精度だけでなく，最適解の数を規定する．図 4 に入力文長さに対する最適解の平均個数を各種選好知識とその組合せに対して示す．各組合せに対する最適解数の全体平均は，und/5.1,uac/1.3,bnd/5.3,bac/5.1,und+uac/1.1 であり，uac が最適解の数の限定に最も寄与していると言える．最適解数を 100 に限定した場合において，base の最適解数平均は 13.9 であった．前節の議論と同様，2項選好スコアに関しては，データスパースネスの問題が存在することが考えられるため，より大規模な実験による検証が必要であると考えている．

5 おわりに

本稿では，選好依存文法における選好知識記述を従来の単項モデルから2項モデルに拡張し，2項モデルに対して最適解探索を行うグラフ分枝アルゴリズムについて説明した．また，2項モデルにおける知識統合評価方式を提案し試験的な評価実験を行った．この結果，単項ノードスコアと単項アークスコアの2つを統合して用いたスコアリング方式が最も性能的に優れた結果を出した．単項モデルは，最適解探索の計算量的にも優れているため，今回の実験からは単項モデルを基本とすることが適切であると結論付けられる．但し，2項モデルについては，データスパースネスの問題による性能ネックの存在も考えられ，更なる検討も必要である．また，スコアリングに導入された各種パラメータを，学習手法により最適化することによる性能向上も今後の検討課題である．

参考文献

[1] S. Amano, H. Hirakawa, H. Nogami, and A. Kumano. The toshiba machine translation system. *Future Computing Systems*, 2(3), 1989.

[2] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[3] M. J. Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the ACL (ACL-96)*, pages 184–191, 1996.

[4] J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING'96*, pages 340–345, 1996.

[5] H. Hirakawa. Graph branch algorithm: An optimum tree search method for scored dependency graph with arc co-occurrence constraints. In 情報処理学会, 自然言語処理研究会 *NL-169-16*, pages 101–108, 2005.

[6] R. McDonald, K. Crammer, and F. Pereira. Spanning tree methods for discriminative training of dependency parsers. Technical report, UPenn CIS, 2005.

[7] S. Riezler, H. King, T., M. Kaplan, R., R. Crouch, T. Maxwell III, J., and M. Johnson. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the ACL (ACL-02)*, pages 271–278, 2002.

[8] Y. Su, K., H. Chiang, T., and S. Chang, J. An overview of corpus-based statistics-oriented (cbso) techniques for natural language processing. *International Journal of Computational Linguistics & Chinese Language Processing (CLCLP)*, 1(1):101–157, 1996.

[9] 茨木俊秀. 組み合せ最適化. 産業図書出版, 1983.

[10] 平川秀樹. 選好依存文法 (pdg) における文解析能力の評価方式について. 情報処理学会論文誌, 46(11):2744–2752, 2005.

[11] 平川秀樹. 統語森に対応する圧縮共有型依存構造「依存森」について. In 情報処理学会 自然言語処理研究会 *NL-167-9,IPSJ*, pages 53–62, 2005.