依存構造に基づくコロケーションの自動抽出

高山 宏規 †　　加藤 芳秀 ††　　大野 誠寛 †††　　松原 茂樹 †　　石川 佳治 †

† 名古屋大学大学院情報科学研究科
†† 名古屋大学情報連携統括本部
††† 名古屋大学情報基盤センター
takayama@db.itc.nagoya-u.ac.jp

1　はじめに

コロケーションとは，言語的，あるいは慣用的な結びつきをもつ単語の系列である．コロケーションを大量に収集し，辞書として整理すれば，自然言語処理における言語資源として，あるいは外国語を学習する際の教材として有用である．

これまでに，コーパスからコロケーションを抽出する手法は数多く提案されているが，その多くは，2 単語間の結合度を評価し，コロケーションを抽出する．そのため，単語バイグラムのコロケーションしか抽出できない．

本稿では，依存構造が付与されたコーパスからコロケーションを自動抽出する手法を提案する．本手法の特徴は，以下の点である．

1. 3 つ以上の単語から構成されるコロケーションを抽出できる．
2. 文中の離れた場所に単語が出現するコロケーションを抽出できる．

本手法では，これまでに提案された結合度の評価尺度を依存構造に適用できるように拡張する．これにより，上述の特徴を持つコロケーション抽出が実現できる．

2　関連研究

本節では，依存構造に基づきコロケーションを抽出する従来の手法について概観する．

Lin は依存関係にある単語間のコロケーションの結合度を評価する手法を提案している [1]．Lin の手法では，依存関係にある 2 単語間の結合度を自己相互情報量 (pointwise mutual information, PMI) に基づき評価している．しかし，この手法では 2 単語で構成された依存構造の結合度しか評価できない．

葛原らは，依存構造を利用し，英語論文執筆に役立つ英語表現を自動で抽出する手法を提案している [2]．葛原らの手法では，依存関係で連結された単語列をコーパスから抽出し，依存構造において，その単語列に特定の単語が隣接するかどうかをエントロピーに基づき評価する．単語列に特定の単語が隣接する場合，その単語列は英語表現ではなく，英語表現の一部である可能性が高いため，これを英語表現の候補から除外する．この手法では，単語列を構成する要素間の結びつきは評価していない．

Martens らは，依存構造の付与されたコーパスから最小記述長に基づき依存構造パターンを抽出する手法を提案している [3]．この手法では，依存構造コーパスに対する符号化法を定め，依存構造パターンを用いてコーパスを圧縮したときに，その符号長がどれだけ短くなるかによりパターンを評価する．評価尺度が符号化法に依存するといった問題や，要素間の結びつきの度合いがその評価尺度にどのように反映されているか明らかでないといった問題がある．

3　依存構造に基づくコロケーションの抽出

本節では，依存構造が付与されたコーパスからコロケーションを抽出する手法を提案する．本手法では，3 つ以上の単語を含む依存構造の結合度を定義する．依存構造をベースにすることにより，3 つ以上の単語から構成されるコロケーションや，文中の離れた場所に単語が出現するコロケーションを抽出できる．

まず，提案手法の概略について述べる．前提として，コーパスには依存構造が付与されているものとする．この依存構造は文ごとに一つの木構造を構成する．本手法では，まず，この木構造に頻出する木構造パターンを抽出する．この木構造パターンは，依存関係で連結された単語列と対応している．次に，得られた木構造パターンを，自己相互情報量をベースにした結合度により評価し，コロケーションを抽出する．

3.1　依存構造に基づく木構造の構築

一般に，英文に対する依存構造は一つの木構造を構成するので，まず，その点から説明する．

英文中の各単語をノードとし，依存される単語を親ノード，依存する単語を子ノードと定めると，文に対して一つの木構造が与えられる．単語が右から依存するか左から依存するかを区別する場合は，エッジに依存関係の方向に関するラベルを与えればよい（以下で

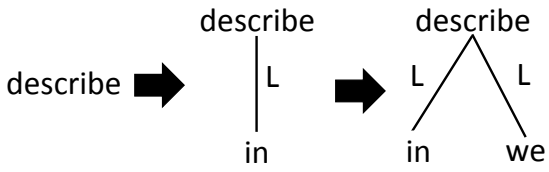

図 1: 拡張の例

は，右から依存する場合のラベルを R，左から依存する場合のラベルを L とする）．木構造に含まれる木構造パターンは，依存関係で連結された単語列に対応する．葛原ら [2] や Martens ら [3] の手法では，依存構造を木構造とみなし，木構造パターンを抽出するが，本稿で提案する手法も，この点ではこれらの手法と同様である．

3.2 木構造パターンの抽出

依存関係で連結されたコロケーションを得るため，本手法では，コーパス中の文を前節で述べた木構造で記述し，この木構造から木構造パターンを抽出する．木構造集合に含まれるすべての木構造パターンを抽出するのは，データサイズの面から見て現実的ではない．そこで本手法では，従来の手法と同様に，木構造パターンの抽出において閾値を設定し，出現頻度が閾値以下のパターンを抽出しない．これにより効率的にパターンを抽出する．

木構造パターンの抽出は，木構造マイニングアルゴリズム FREQT[4] をベースとしている．FREQT は，木構造集合から，ある閾値以上出現する木構造パターンを抽出する手法である．FREQT では，始めに，木構造集合から，サイズが 1，つまり単一のノードからなるパターンを列挙する．これらの中から，出現頻度があらかじめ定められた閾値以上のもののみを残す．次に，サイズが 1 の頻出パターンに新たにノードを一つ付加することにより，サイズが 2 の頻出パターン候補を列挙する．頻出パターンにノードを付加する操作を拡張と呼ぶ（図 1 に拡張の例を示す）．拡張によりサイズが 2 のパターンの候補が得られるが，これらの候補のうち，出現頻度が閾値以上のパターンのみを頻出パターンとして残す．以降，サイズを 1 ずつ増やしながら拡張を行い，頻出パターンが得られなくなるまで繰り返す．これにより，出現頻度が閾値以上の木構造パターンを効率的に得ることができる．パターンの出現頻度は，パターンの最右葉のノードに対応するコーパス中のノードを保持しておき，これに基づき計算する．

3.3 依存構造の結合度の評価

FREQT により，木構造集合において閾値以上の頻度で出現するパターンを得ることができる．これらのパターンのうち，対応する単語列がコロケーションであるか否かを評価する必要がある．コロケーションの評価尺度として，本手法では自己相互情報量に基づく尺度を用いる．

3.3.1 自己相互情報量に基づく 2 単語間の結合度の評価

依存関係にある単語間の結合度を評価する手法として，Lin は自己相互情報量に基づく手法を提案している [1]．2 つの要素 x，y に対して，自己相互情報量は次式で定義される．

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

$P(x,y)$ は要素 x と y が同時に出現する確率，$P(x)$，$P(y)$ はそれぞれ要素 x，y が出現する確率である．$PMI(x,y)$ が正のとき，x と y は同時に出現しやすいことを表し，負の時，同時に出現しにくいことを表す．

Lin の手法では，依存関係にある 2 単語の自己相互情報量を，依存関係が与えられた条件のもとでの確率に基づき計算する．いま，依存する語と依存される語をそれぞれ，d, h とする．また，依存関係の種類を t とすると，依存関係にある 2 単語間の自己相互情報量は以下の式により定義される．

$$\begin{aligned} PMI(h,d|t) &= \log \frac{P(h,d|t)}{P(h|t)P(d|t)} \\ &= \log \frac{\frac{f(h,d,t)}{f(t)}}{\frac{f(h,t)}{f(t)}\frac{f(d,t)}{f(t)}} \\ &= \log \frac{f(t)f(h,d,t)}{f(h,t)f(d,t)} \end{aligned} \tag{2}$$

ここで $f(\cdot)$ は出現頻度を表す．

3.3.2 自己相互情報量に基づく依存構造の結合度の評価

Lin の手法では，2 単語で構成される依存構造の結合度しか評価できない．そこで本節では，3 つ以上の単語から構成される依存構造において結合度を評価する手法を提案する．この手法では，依存構造を 2 つに分割して自己相互情報量を計算し，その自己相互情報量の平均をとってその木構造の結合度とする．

最初に依存構造の分割について説明する．依存構造を分割するには，依存構造中のエッジを一つ選択し，親ノードに連結した部分と，子ノードに連結した部分に分ければよい．以下では，e をエッジとするとき，親ノードに連結した部分を α_e，子ノードに連結した部分を β_e と書く．図 2 に，依存構造の分割の例を示す．

次に，依存構造に対する結合度の計算について説明する．基本的な考え方は Lin の手法と同じである．依存する語 d，依存される語 h にはそれぞれ α_e，β_e が対応する．すなわち，依存構造 T をエッジ e で分割し

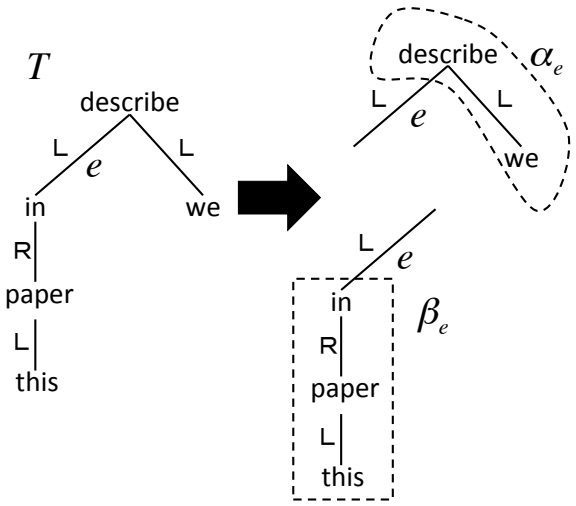

図 2: 依存構造の分割の例

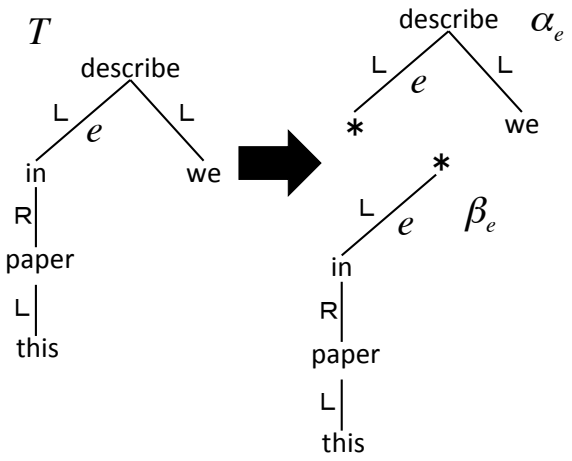

図 3: ＊の導入

たときの α_e と β_e の自己相互情報量は以下の式のようになる．

$$PMI(\alpha_e, \beta_e|t) = \log \frac{f(t)f(\alpha_e, \beta_e, t)}{f(\alpha_e, t)f(\beta_e, t)} \quad (3)$$

$f(t)$ は，依存構造コーパス中のエッジをスキャンすれば，容易に求められる．また，$f(\alpha_e, \beta_e, t) = f(T)$ であるので，FREQT の結果をそのまま利用できる．$f(\alpha_e, t)$，及び $f(\beta_e, t)$ については，別途求める必要があるが，これについては次節で述べる．

最後に，この依存構造の結合度について述べる．依存構造 T のエッジの集合を E とするとき，結合度は以下のように定義する．

$$Assoc(T) = \frac{1}{|E|} \sum_{e \in E} PMI(\alpha_e, \beta_e|t) \quad (4)$$

この式は，2 単語で構成される依存構造に対して適用すると，Lin の式と等しくなる．すなわち，この式は，Lin の手法の自然な拡張ということができる．

3.3.3 依存構造パターンとエッジの共起頻度

前節で述べた結合度を計算するためには，$f(\alpha_e, t)$，及び $f(\beta_e, t)$ を求めることが必要である．本節では，これを求めるために FREQT に変更を加える．

まず，$f(\alpha_e, t)$ について述べる．これを求めるために，本手法では FREQT に特殊な記号＊を導入する．この記号は任意の単語にマッチすることを表す．共起頻度 $f(\alpha_e, t)$ を求めることは，図 3 に示す様なパターンの頻度を求めることに相当する．パターンを拡張するとき，＊をラベルの持つノードを付加することにより，このようなパターンの頻度を求めることができる．ただし，＊を使用した拡張を無制限に行うと効率的でないため，以下の様な制約を設ける．

- ＊をラベルに持つノードに対して，いかなるノードも付加しない．
- パターンが＊をラベルに持つノードを含むとき，＊をラベルに持つノードを付加しない．

次に，$f(\beta_e, t)$ について述べる．FREQT では，パターンの最右葉に対応するコーパス中のノードをすべて保持している．したがって，コーパス中のノードを上にたどることにより，パターンのルートに対応するノードを求めることができる．β_e のルートに対応するコーパス中のノードとその親ノードを結ぶエッジをすべてスキャンすれば，$f(\beta_e, t)$ を求めることができる．

3.3.4 コロケーションの抽出

本節では，前節で提案した結合度を用いて，コロケーションを抽出する手法について述べる．基本的には，結合度の高いパターンを選択し，それらを単語列に復元するのであるが，いくつかの制約を導入する．

一つ目の制約は，結合度が極大であるもののみをコロケーションとする制約である．Silva らは，単語 n グラムの結合度が極大であるかどうかによりコロケーションを判定する手法を提案しているが [5]，この手法では，対象となる n グラムから端の単語を取り除いた $n-1$ グラム，及び端に単語を追加した $n+1$ グラムのいずれの結合度よりも，対象の n グラムの結合度が大きい場合，これをコロケーションとしている．本稿で提案する手法では，このアイディアを依存構造に適用する．すなわちコロケーションとなる依存構造パターン T は以下の条件を満たすものとする．

- すべての $T' \in Sub(T) \cup Super(T)$ に対して，$Assoc(T) > Assoc(T')$．

ここで，$Sub(T)$ は T から葉ノード，あるいは根ノードを一つ取り除いたパターン集合，$Super(T)$ は T の任意のノードに子ノードを追加したパターン，あるいはルートに親ノードを追加したパターンの集合である．

表 1: 抽出されたコロケーション

thank ... reviewers for ... comments
S ENSEVAL ... data
% precision and ... % recall
Proceedings of ... Workshop
From ... point of view
differences ... are statistically significant
supported by ... grant
thank ... for ... helpful comments
Let ... and let
until ... is reached
not only for ... but also
would you ... ?
difference ... is statistically significant
not only for ... but ... for
as “ ... ”
agreement between ... annotators
of “ ... ”
there is ... room for improvement
thank ... for ... discussions
How do ... ?

2 つ目の制約は，木構造の閉包性に関する制約である．Martens らの手法では，閉じたパターンのみをコロケーションの候補としている [3]．ここで，パターン T が閉じているとは，次の条件を満たすことをいう．

- すべての $T' \in Super(T)$ に対して，$f(T) - f(T') > 0$

$f(T) = f(T')$ であるような T' が存在することは，T に対してある単語が必ず隣接することを意味する．本手法では，この制約を強め，コロケーションは以下の条件を満たすものとする．

- すべての $T' \in Super(T)$ に対して，$f(T) - f(T') > m$

4 実験

提案手法の有効性を確認するために，実験を実施した．実験には，ACL Anthology Corpus[6] から取り出した 619913 文を使用し，提案手法によりコロケーションを抽出した．提案手法を適用するには，文に対して依存構造を与えなければならないが，本実験では，各英文に対して Charniak Parser[7] により句構造を付与し，Pennconverter[8] を用いて付与された句構造を依存構造に変換した．出現頻度の閾値は 50 とした．

3 つ以上の単語からなるパターンとして 59213 個のパターンが抽出されたが，そのうち，極大性の制約，閉包性の制約を満たしたものは，8694 個であった．文中において単語が離れて出現しているパターンのうち，結合度が上位 20 位であったものを表 1 に示す．論文でよく使用される言い回しが抽出されている．また，“not only for ... but also” のようなコロケーションが抽出できたことは，本手法が，単語が離れて出現するコロケーションの抽出に有効であることを示している．

5 おわりに

本稿では，依存構造に基づきコロケーションを抽出する手法を提案した．本手法では，文を依存構造に基づく木構造として記述し，その木構造から木構造パターンを抽出することにより，依存関係で連結された単語列を獲得する．獲得された単語列に対して，自己相互情報量に基づき依存構造の結合度を評価し，コロケーションを抽出する．

今後，提案手法の詳細な評価をする予定である．また，本手法では自己相互情報量をベースに結合度を定義したが，本手法により得られる頻度情報があれば，その他の尺度をベースとした結合度を定義することが可能である．これらとの比較実験についても検討したい．

参考文献

[1] Lin: Automatic Identification of Non-compositional Phrase, Proc. of the 37th Annual Meeting of the Association for Computational Linguistics, pp. 317-324, 1999.

[2] 葛原ら: 構文構造に基づく英語表現の自動獲得とその評価, 言語処理学会第 17 回年次大会発表論文集, pp380-383, 2011.

[3] Martens and Vandeghinste: An Efficient, Generic Approach to Extracting Multi-Word Expressions from Dependency Trees, Proc. of the 2010 Workshop on Multiword Expressions: from Theory to Applications, pp.85-88, 2010.

[4] Asai et al.: Efficient Substructure Discovery from Large Semi-Structured Data, Proc. of 2nd SIAM Inter. Conf. on Data Mining, pp.158-174, 2002.

[5] Silva and Lopes: A Local Maxima Method and a Fair Dispersion Normalization for Extracting Multi-word Units from Corpora. Proc. of 6th Meeting on Mathematics of Language, pp.369–381, 1999.

[6] Schäfer et al.: Towards an ACL Anthology Corpus with Logical Document Structure. An Overview of the ACL 2012 Contributed Task, Proc. of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries, pp.88-97, 2012.

[7] Charniak and Johnson: Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking, Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics , pp.173-180, 2005.

[8] http://fileadmin.cs.lth.se/nlp/software/pennconverter/pennconverter.jar