構造を持った定型表現の自動獲得と機械翻訳での利用

望月 道章　　中澤 敏明　　黒橋 禎夫
京都大学大学院情報学研究科
{mochizuki, nakazawa, kuro}@nlp.kuee.kyoto-u.ac.jp

1　はじめに

現在の機械翻訳では単語よりも大きなフレーズを単位として翻訳を行うことが一般的である。しかし、ここでのフレーズは対訳コーパスから自動的に推定された単語対応をヒューリスティックなルールを用いて拡張することで得られたものであり、そのフレーズが各言語において、まとまりとしてもっともらしいかどうかは考慮されていない。そのため、定型表現の翻訳を行う際に問題が起こる。定型表現とは複数の単語が組み合わさって一つの意味を持つ表現であり、構成する単語を個別に扱ってしまうと翻訳誤りの一因となる。例えば定型表現の一つである“ (〜する) 方 が 良い ”は、“ 方 ⇔ direction ”と“ 良い ⇔ good ”のように個別に翻訳すると翻訳誤りになってしまう。

この問題を解決するためにはフレーズのもっともらしさを考慮する方法が考えられ、既にいくつか提案されている [3, 1]。しかしこれらの研究では単語列上で連続した表現しか考慮しておらず、中国語の“ 在 〜 中 ”(“ 〜 に おいて ”) のような単語列上は連続しない定型表現を扱うことができない。そこで本研究では依存構造木から定型表現を自動的に獲得し、知識として機械翻訳で利用する手法を提案する。定型表現の獲得はその表現の出現頻度や周辺語の異なり数に基づいたスコアを用いて行う。依存構造木を用いることで単語列では不連続であっても、直接の依存関係が存在していれば定型表現として獲得することができる。また、自動獲得された定型表現を対訳文内の単語・句アライメントで利用することにより、精度の向上を目指す。

2　依存構造木からの定型表現の獲得

本手法では任意の表現に対してスコア付けを行い、その値が閾値以上の表現を定型表現として獲得する。定型表現はまとまりで頻繁に出現し、連接する単語の種類が多いと考えられる。この特徴を取り入れた指標として C-value [2] を拡張したものを用いる。

2.1　C-value

C-value とは単語列を対象としたコーパス中のコロケーションを判定するためのスコアであり、以下の式で定義される。

$$C\text{-}value(a) = \begin{cases} log_2(|a|) \cdot f(a) & (T_a = \phi) \\ log_2(|a|) \cdot \left(f(a) - \frac{\sum_{b \in T_a} f(b)}{\#(T_a)} \right) & (otherwise) \end{cases}$$

a は対象とする表現、$f(a)$ と $|a|$ はそれぞれ a の頻度と a を構成する単語数である。T_a は a を内部に含むより大きな表現の集合であり、その異なり数を $\#(T_a)$ とする。式の形から頻度 ($f(a)$) が高い表現であっても周辺の単語の種類 ($\#(T_a)$) が少なければ、値が小さくなることがわかる。

C-value では大きさに関係なく a を含む全ての表現を T_a として扱っている。例えば、“ in spite ”の C-value は“ in spite of ”、“ incresed in spite ”、“ in spite of the ”などを T_a として計算する。しかし、本研究では図 1 のように a よりも一単語大きい表現だけを T_a とし、文頭側と文末側で別々に C-value の計算を行い、両方が閾値以上の表現を獲得する。なお、図の X、Y は任意の単語を表す。つまり、“ in spite ”の計算を行う際は“ in spite of ”と“ incresed in spite ”のように文頭又は文末側に一単語が接続した表現だけを T_a とする。“ in spite ”はほとんどの場合“ in spite of ”に含まれて出現するため、文末側の値が低くなり、定型表現として獲得されず、“ in spite of ”のみが獲得される。

2.2　依存構造木からの獲得

依存構造木では文頭側と文末側の代わりに root 側と leaf 側で T_a を区別し、計算を行う。root 側に関しては単語列と同様に計算できるが、leaf 側については単語列の場合と異なるため、T_a をさらに詳細に区別する必要がある。そこで本研究では以下の変更を加え、図 2 のように T_a を区別した。なお、今後は依存構造木上の単語はノードと呼ぶ。

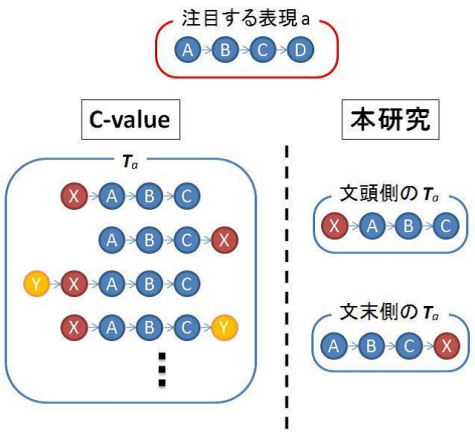

図 1: 単語列の T_a

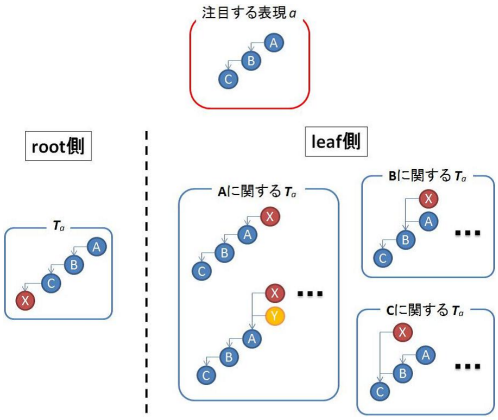

図 2: 依存構造木の T_a

ノードごとに C-value を計算する

単語列の場合、単語は常に注目する表現の端のノードに接続していた。しかし依存構造木では全てのノードに単語が接続する可能性があり、同じ単語でも異なるノードに接続することが考えられる。このような異なりを全て T_a の一つとしてしまうと、あるノードに接続しやすい単語があっても他のノードに接続する単語の種類が多いため、$\#(T_a)$ が大きくなり誤って獲得されてしまう。そこで、図 2 に示すように a の各ノードごとに T_a を区別しそれぞれについて計算を行い、その最小値を leaf 側のスコアとする。それにより、あるノードに接続しやすい単語が存在すれば、そのノードの C-value は小さくなるので、定型表現としては獲得されず、全てのノードについて接続する単語の種類が多い表現のみを獲得することができる。

同じノードに接続する単語をまとめて扱う

依存構造木では一つのノードに複数の単語が接続する場合があり、そのような表現をどの表現の T_a として扱うかが問題になる。本研究では接続している単語をまとめて扱い、図 2 に示すように a の任意のノードに複数の単語が接続する表現も T_a として計算を行う。

また、同じ単語でも接続するノードに対し単語列上で前から接続している場合と後ろから接続している場合で区別して扱っている。

本研究のように依存構造木を利用した表現獲得の研究には葛原ら [7] や Martens ら [4] の研究がある。葛原らは本研究と同じようにある表現のノードごとにスコアを計算することで獲得を行っている。しかし、英文作成を支援する表現の獲得が目的であり、節や句なども含んだ大きな表現も獲得されている。機械翻訳ではできるだけ小さい単位を扱う方が望ましいため、本研究とは獲得したい表現の大きさが異なる。また、Martens らはいくつかのスコアを用いて表現の獲得しているが、獲得した表現の具体的なアプリケーションへの応用は行っていない。

2.3 定型表現の獲得実験

提案手法による定型表現の獲得実験を行った。利用したコーパスは内山・井佐原らの方法により作成した JST 日英抄録 (約 100 万文対) [6] と日中科学論文 (約 60 万文対) であり、各言語を単言語のコーパスとみなして獲得を行った。また、単語の区別に利用した情報は以下である。

- 日本語: 代表表記, 品詞, 活用形
- 英語: 原形, 品詞
  (“ it ”以外の代名詞, 三単現, 複数形は区別しない)
- 中国語: 表層語, 品詞

今回は獲得の対象を六単語以下の表現に限定し、閾値は 3000 に設定した。また、獲得されたものの中には機械翻訳で扱うには不適切な表現があったので、以下に示す簡単なルールでフィルタリングを行った。ただし、中国語のフィルタリングは行っていない。

日本語 (名詞または指示詞) + 助詞
leaf が (“ する ”または名詞性接尾辞)

英語 冠詞を含む 2 単語
be 動詞と前置詞が接続

獲得された定型表現の例を表 1 に示す。“ ことができる ”や“ in order to ”など翻訳で有用な定型表現が獲得できていることが分かる。また、“ 在 ～ 中 ”や“ as ～ as ”など単語列上では不連続な定型表現も本手法により獲得できることが確認できた。

表 1: 獲得した定型表現の例

日本語	英語	中国語
に ついて	this paper	在 ～ 中
と して	based on	不 能
こと が できる	as ～ as	本 研究
に ついて 述べた	in order to	就 是
本稿 で は	as a result	在 ～ 内

表 2: 定型表現のスコア

定型表現	スコア
こと が でき	22830
が でき ます	22605
が でき	33572
ます か	20714

3 対訳文アライメントでの定型表現の利用

3.1 ベースラインシステム

ベースラインシステムとしては中澤らの用例ベース機械翻訳システム [5] を用いた。このモデルでは依存構造木上で統計的句アライメントを行っている。依存構造木を用いることで、言語構造が大きく異なる言語対でも柔軟に対応することができる。アライメント手法を簡単に説明すると、まず既存の統計的単語アライメントモデルにより単語レベルでの対応を推定し、これをヒューリスティックなルールで依存構造木上での句対応にマッピングする。これを初期状態とし、句対応確率と句の依存関係の確率を考慮して EM アルゴリズムにより繰り返しモデル推定を行う。EM アルゴリズムの途中により大きな句を獲得するステップがあることが特徴である。

3.2 定型表現の利用

機械翻訳では定型表現を構成する単語を個別に扱うと翻訳誤りの原因になってしまうため、定型表現はまとめて扱う必要がある。しかし、定型表現を構成する単語のアライメントが誤っているために依存構造木上で対応先が不連続になり、まとめて扱えない場合がある。その例を図 3 に示す。ここでは、(■) が対応関係を表し、薄い青と濃い青の部分が正解の対応である。定型表現である“ 方 が 良い ”が“ 方 ⇔ me ”と“ 良い ⇔ should ”という対応を持っているため、“ 方 が 良い ”の対応先は依存構造木上で不連続になってしまう。そこで、本研究では定型表現はまとまりで一つの意味を持つので対応先でもまとまっていると考え、定型表現を構成する単語の対応先が依存構造木上で不連続になる対応を禁止するという制約を用いた。具体的には、定型表現を構成する単語の対応先が不連続になってしまう場合、対応確率が低い方を利用しない。こうすることで、図 3 の“ 方 ⇔ me ”がなくなり、“ 方 が 良い ”をまとめて扱うことができる。

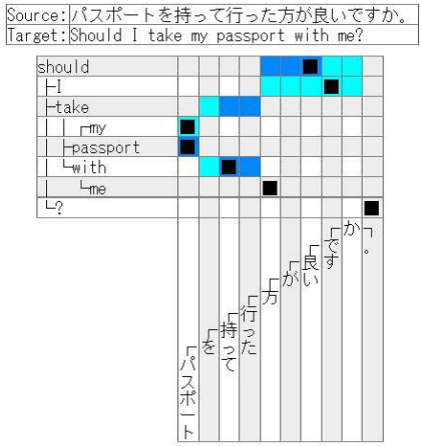

図 3: アライメントの誤り例

また、文内のどのまとまりを定型表現とするかも問題となる。本研究では前章で獲得した定型表現のうち文内に存在するものは基本的に全てを定型表現として採用する。ただし、候補がオーバーラップした場合は長い表現を優先する。もし、同じ長さであった場合は C-value の高い方を採用する。この時の C-value は root 側と leaf 側の平均を取った値である。例えば、獲得された定型表現の C-value が表 2 であった場合、“ こと が でき ます か ”という文では“ こと が でき ”と“ ます か ”の二つの定型表現が採用される。

4 実験

4.1 実験設定

定型表現を利用したアライメントを行い、精度への影響を調べた。実験には 2 章と同じコーパスを用いた。アライメントの評価には人手で正解を与えた日英 480 対訳文と日中 500 対訳文を利用し、以下の式で示される Precision、Recall、Alignment Error Rate(AER) を用いた。AER はアライメントの総合的な精度を示す指標であり、その値が低い程精度が良い。

$$Precision = \frac{|A \cap P|}{|A|} \qquad Recall = \frac{|A \cap S|}{|S|}$$

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

表 3: アライメントの精度

	定型表現	Pecision	Recall	AER
日英	なし	81.48	68.27	25.43
	あり	82.50	67.74	25.32
	GIZA++	81.17	62.19	29.25
日中	なし	84.72	78.82	18.15
	あり	85.16	78.62	18.03
	GIZA++	83.87	75.48	20.29

A がシステムの出力、P と S が正解である。S(Sure) は必ず必要な正解であり、図 3 の濃い青である。P(Possible) は英語の冠詞や日本語の助詞などのようにあっても誤りではない正解であり、図 3 の薄い青である。

4.2 結果と考察

定型表現を利用したアライメントの精度を表 3 に載せる。なお、参考ために GIZA++による双方向のアライメント結果をヒューリスティックに統合したものの精度も載せる。結果を比較すると定型表現を利用した方がわずかではあるが精度が向上していることが分かる。また、日英、日中ともに Recall が低下し、Precision が上昇する傾向が見られる。

実際のアライメント結果の例を図 4 に示す。定型表現である“ で は ”をみると、定型表現を利用する前の対応は“ は ⇔ been ”、“ 現在 で ⇔ at present ”であり依存構造木上で不連続である。しかし、定型表現を利用すると“ は ⇔ been ”の対応が採用されなくなり、“ 現在 で は ”という大きな対応が新たに獲得されている。ここから制約が有効に働いていることが分かる。

しかし、言語構造の違いなどによって定型表現の対応が正しくても対応先が不連続となってしまう場合がある。その場合、正しい対応が禁止されてしまい、Recall 低下の原因となっている。定型表現の情報をどのようにアライメントの制約として利用するべきかについては今後さらに検討する必要がある。

5 おわりに

本研究では定型表現をコーパスから自動的に獲得し、アライメント時に制約として利用する手法を提案した。定型表現はコロケーション獲得の指標である C-value を拡張することで依存構造木から獲得しており、単語列上では連続しない表現も獲得できる。また、アライメントでは定型表現を構成する単語の対応先が依存構造木上で不連続になる対応を禁止するという制約を用いた。実験は日英、日中間で行い、そのどちらでも依存構造木から獲得した定型表現がアライメントの精度向上に有効であることを確認した。

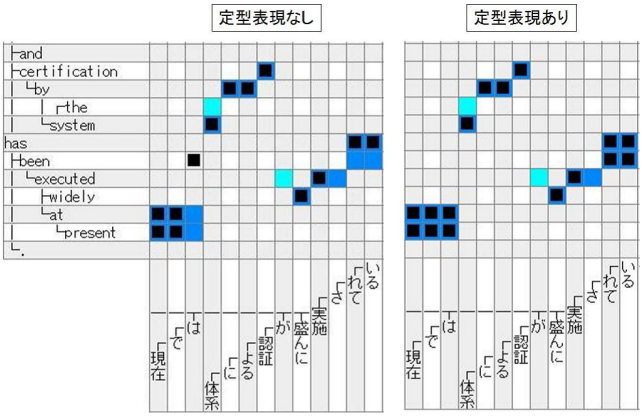

図 4: アライメントの結果例

今後は、制約により誤って禁止される問題を解決するために定型表現の利用方法を検討するとともに定型表現を利用した翻訳を行いその有効性を検証する予定である。

参考文献

[1] Xiangyu Duan, Min Zhang, and Haizhou Li. Pseudo-word for phrase-based machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 148–156, 2010.

[2] Katerina T. Frantzi and Sophia Ananiadou. Extracting nested collocations. In *Proceedings of the 16th conference on Computational linguistics*, pp. 41–46, 1996.

[3] Zhanti Liu, Haifeng Wang, Hua Wu, and Sheng Li. Improving statistical machine translation with monolingual collocation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 825–833, 2010.

[4] Scott Martens and Vincent Vandeghinste. An efficient, generic approach to extracting multi-word expressions from dependency trees. In *Proceedings of the Multiword Expressions: From Theory to Applications(MWE 2010)*, pp. 85–88, 2010.

[5] Toshiaki Nakazawa and Sadao Kurohashi. Fully syntactic ebmt system of kyoto team in ntcir-8. In *Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies(NTCIR-8)*, pp. 403–410, 2010.

[6] Masao Utiyama and Hitoshi Isahara. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 72–79, 2003.

[7] 葛原和也, 加藤芳秀, 松原茂樹. 構文構造を利用した英語論文からの表現の自動獲得. 研究報告自然言語処理 (NL), pp. 1–7, 2010.