機械学習を用いた機械翻訳用モダリティコーパスの修正

村田 真樹[0]　内山 将夫　内元 清貴　馬 青　井佐原 均
総務省 通信総合研究所

1　はじめに

近年，種々のタグつきコーパスが作成され，タグつきコーパスを利用した研究も盛んに行なわれている．しかし，タグつきコーパスには誤りが含まれており，このことが各研究の進捗の妨げとなることもある．このため，コーパス中の誤りを修正することは，重要な研究課題となる[1]．われわれは，コーパス修正の研究を形態素・構文コーパスにおいてすでに行なっており[3]，その研究では50万個の形態素データのうちエラーが多いとシステムで予測した4,000個の形態素データを調べると2,000個程度の誤りを検出できそうであるという成果をすでにあげている．この手法の汎用性の確認および手法の改良のために，われわれはあらたに現在作成中のモダリティコーパスにおいてコーパス修正の研究を再度行なった．本稿はこの結果について述べる．

ここで対象とするするモダリティコーパスは，文献[4, 5]などにおいて行なっているモダリティ表現の日英翻訳を行なう際の教師データとなるものである．(本稿でのモダリティはテンスアスペクトを含む広義なものを指すことにする．) 機械翻訳においてモダリティ表現は困難な問題として知られている．従来は，モダリティ表現は人手による規則により翻訳されていた．しかし近年になって用例ベースなどの機械学習手法によるモダリティの翻訳がなされはじめており，本研究で扱うモダリティコーパスはこの種の機械学習手法に基づく機械翻訳では必須となるものである．

以降，2節でモダリティコーパスについて述べ，3節でコーパス修正の方法を4節でコーパス修正の実験を述べる．

2　機械翻訳用モダリティコーパス

本節では，モダリティコーパスについて述べる．まず，このモダリティコーパスの一部を図1に示す．図のようにこのコーパスは日英の対訳コーパスからなり，英語側の文には以下の二か所のタグが付与されている[2]．

, この子どもはああ言えばこう言うから小憎らしい
This child always talks back to me,and this <v>is</v> why I <vj>hate</vj> him.

d 彼がああおくびょうだとは思わなかった
I <v>did not think</v> he was so timid.

c ああ忙しくては休む暇もないはずだ
Such a busy man as he <v>cannot have</v> any spare time.

図 1: モダリティコーパスの一部

- 英語の主節の動詞部分を <v>,</v> のタグで囲む．
- 日本語の主節の動詞に対応する英語の動詞部分を <vj>, </vj> のタグで囲む．

また，日本語側の文の先頭に “c” や “d” といった記号がふられているが，これらはこの対訳データのモダリティを意味する (例えば，“c” は canを，“d” は過去形を意味する[3])．モダリティの分類としては以下の34種類のものを用いた．以下のものは，対訳の英語文の動詞がどのような形になっているかによって定められる．

1. {現在形，過去形} と {進行形，進行形でない} と {完了，完了でない} のすべての組み合わせ (8種類)
2. 命令形 (1種類)
3. 助動詞相当語句 (be able to の現在形と過去形, be going to の現在形と過去形, can, could, have to の現在形と過去形, had better, may, might, must, need, ought, shall, should, used to, will, would の19種類)
4. 名詞句 (1種類)
5. 分詞構文 (1種類)
6. 動詞省略 (1種類)
7. 間投詞、挨拶文など (1種類)
8. 日本語と英語で動詞の対応がとれない場合 (1種類)
9. 作業不可 (1種類)

このモダリティの分類は英語の表層表現に基づいて定めたものであり，日本語文だけを与えてこの分類を

[0]murata@crl.go.jp

[1]コーパスの自動修正の先行文献はない．コーパス中の誤りを検出する研究としては，ブースティングを用いるもの[1]や anomaly detection を用いるもの[2]などがある．

[2]「日本語に対応する英語の動詞部分」と「英語の主節の動詞部分」が一致する場合は「英語の主節の動詞部分」のタグのみ付与した．また，「日本語に対応する英語の動詞部分」の方はそれほど綿密にタグ付与は行なっておらず，「日本語に対応する英語の動詞部分」と「英語の主節の動詞部分」が一致しない場合にもタグ付与をしなかった場合もある．

[3]図1の一つめのデータには “,” があるが，これは<vj>を用いるときに使われるもので，“,” の左に<v>で囲まれた動詞に対するモダリティが右に<vj>で囲まれた動詞に対するモダリティが記述される．このコーパスでは現在形の出現が多いのでその場合はタグをふらなかった．このため，“,” の左右が空欄となってこの部分には “,” だけが付与されている．

推定できれば，モダリティ表現の日英翻訳のできあがりとなる．このため，機械学習に基づくモダリティ推定の研究では，このモダリティの分類を示すタグと日本語文のみが用いられる．

上記の仕様で，コーパスの作成を外注した．対訳データは，講談社和英辞典 [6] から取った約 40,000 文の例文と電子協で作成している白書のデータ [7](約 6,000 文) の二種類のものを用いた．外注先の会社では，英語文への "<v>" などのタグの付与およびその英語文の動詞を見てモダリティの分類のタグの付与を人手で行なった．また，複数回のチェック作業を行なってもらい，外注先の会社としては誤りはまったくないという状態までこの作業を行なってもらった．

3 コーパスの修正の方法

ここでは，前節で人手で作成したコーパスを機械学習の手法を用いて修正することを試みる．本稿のコーパス修正の方法は，文献 [3] の方法と同じく，コーパス中の誤り修正の対象となるタグ[4]の生起確率を求め，この確率を用いてコーパス修正を行なう．

まず，各タグの生起確率の求め方であるが，本研究では最大エントロピー法に基づく方法と決定リストに基づく方法の二種類を試した[5]．

- 最大エントロピー法に基づく方法 [8, 9]

  これは，あらかじめ設定しておいた素性の集合を $F(\ni f_j(1 \leq j \leq k))$ とするとき，式 (1) を満足しながらエントロピーを意味する式 (2) を最大にするときの確率分布 $p(a,b)$ を用いる方法である．

$$\sum_{a\in A, b\in B} p(a,b)g_j(a,b) = \sum_{a\in A, b\in B} \tilde{p}(a,b)g_j(a,b) \quad (1)$$
$$for\ \forall f_j\ (1 \leq j \leq k)$$

$$H(p) = -\sum_{a\in A, b\in B} p(a,b)\ log\,(p(a,b)) \quad (2)$$

  ただし，A, B は出力値 (本稿の場合は対象としているタグ) と文脈の集合を意味し，$g_j(a,b)$ は 文脈 b に素性 f_j があってなおかつ a を出力する場合 1 となりそれ以外で 0 となる関数を意味する．また，$\tilde{p}(a,b)$ は，既知データでの (a,b) の出現の割合を意味する．

  式 (1) は確率 p と出現を意味する関数 g をかけることで期待値を求めることになっており，右辺の既知データの期待値と左辺の求める確率分布に基づいて計算される期待値が等しいことを制約として，エントロピー最大化 (分布の平滑化) を行なって確率分布を求めるものとなっている．

- 決定リストに基づく方法

  これは，あらかじめ設定しておいた素性 $f_j(\in F, 1 \leq j \leq k)$ のうちいずれか一つのみを文脈として確率値を求める方法であり，ある文脈 b で a を出力する確率は以下の式によって与えられる[6]．

$$p(a|b) = p(a|f_{max}) \quad (3)$$

  ただし，f_{max} は以下の式によって与えられる．

$$f_{max} = argmax_{f_j\in F}\ max_{a_i\in A}\ \tilde{p}(a_i|f_j) \quad (4)$$

  また，$\tilde{p}(a_i|f_j)$ は素性 f_j を文脈に持つ場合の a_i の出現の割合である．決定リストに基づく方法は簡便ではあるが，ある一つの素性のみを文脈とするので，確率推定の方法としては少々貧弱なものとなっている．

本研究では確率を求める際の文脈となる素性としては，以下のものを用いた．(1 箇所につき 26 (5 + 10 + 10 + 1) 個の素性が生じる．)

- 英語文の最初の<v>/<vj> の前方の 1 ～ 5gram の文字.
- 英語文の最初の<v>/<vj> の後方の 1 ～ 10gram の文字.
- 英語文の最後の<v>/<vj> の前方の 1 ～ 10gram の文字.
- 英語文の文末の 1gram の文字.

ただし，英語文で疑問文など動詞句が文中の二か所以上に分解されている場合は左からこの順で</v> から<v> の部分を消してから上記の素性の抽出を行なった[7]．

次にコーパス中の各タグが誤っているかいなかの判定であるが，これは元のタグおよびそれ以外の場合のタグ (本稿では残る 33 種類のタグ) の生起確率を求め，元のタグの生起確率が最も大きい場合元のタグは正しいと判定し，元のタグ以外のタグの生起確率が最も大きい場合元のタグは誤っていると判定することによって行なわれる．次に誤っていると判断されたタグについては，そのとき生起確率が最も大きいタグを修正後のタグとしてふりなおすということを行なう (実際にはこのふり直しは人手で確認の上，行なう)[8]．

上記の方法だけでもコーパスの修正はできるが，本稿のような確率値が求まる手法の場合は以下で述べるようにコーパス修正の確信度を定義できるので，コーパス修正の候補の各箇所をこの値によってソートし

[4]本稿では "<v>" のタグの修正は対象とせず，モダリティの分類のタグのみを修正の対象とする．

[5]本稿では各タグの生起確率の求め方としては最大エントロピー法や決定リスト法を用いるが，コーパス修正としてはもっと強力な確率推定の方法を用いてもよい．

[6]決定リストを使った確率値の推定方法は文献 [3] でも述べていたが，文献のものと本稿のものは若干異なる．また，本稿のものの方が自然なものでありかつ，予備的な実験では本稿のものの方がコーパス修正の精度がよいという結果を得ている．

[7]本稿のコーパスは，日本語文から英語文のモダリティを推定するためのものであるので，素性は日本語文から取り出すべきだと一見思いがちであるが，わざわざそのような精度がさがるようなことはしなくてもよい．英語文のモダリティ表現にあたる分類をふるため，また人手でのコーパス作成も英語文の動詞部分を見ていたので，機械学習によるコーパス修正も英語文の動詞部分を中心に調べる本文で述べたような素性でいいのである．

[8]上記のコーパス修正作業は，機械学習 (本稿では最大エントロピー法および決定リスト手法) を用いてコーパスのタグを再推定しふりなおすのと全く等価である．

て，コーパス修正の確信度が高いところから修正していった方が便利である．

本稿では，このソートに用いるコーパス修正の確信度としては以下の二つのものを試した．

- **方法 1** — 修正後のタグの生起確率を確信度とする方法
- **方法 2** — 修正前のタグの非生起確率を確信度とする方法

ただし，本稿では，非生起確率は 1 から生起確率をひいたものとする．方法 1 は修正後のタグの生起確率が大きければよいと考える方法で，方法 2 は修正前のタグの生起確率が小さければよいと考える方法である．

これでおおかたコーパス修正の方法の説明は終わったが，あと機械学習の方法で確率の値を算出する際には，クローズで算出するかオープンで算出するかの二通りのやり方がある．

- クローズによる確率値の算出
- オープンによる確率値の算出

クローズによる確率値の算出は，修正すべきか判断するタグの部分のデータも含めて確率値を算出する方法で，オープンによる確率値の算出は，修正すべきか判断するタグの部分のデータを含めずに確率値を算出する方法である．本稿のオープンによる確率値の算出方法には 10 分割のクロスバリデーションを利用した[9]．

4 コーパス修正の実験

前節で述べた方法で実際にコーパス修正の実験を行なった．ここでは紙面の都合上講談社和英辞典のデータ (約 40,000 文) を用いて行なった実験についてのみ述べる[10]．この実験は作業不可のタグがついた文は除いて行ない，総数 39,718 個のモダリティタグのデータに対して行なった．実験結果を表 1と表 2にあげる．「ランダム 300 個」は無作為に抽出した 300 個のシステム修正箇所での精度を意味し，上位 X 個は方法 1 または方法 2 によるソートを行なったデータでの上位 X 個のシステム修正箇所での精度を意味する．検出精度は，システム修正箇所において誤りの検出を成功した割合を意味し，修正精度は，システム修正箇所においてタグの修正を成功した割合を意味する．表で抽出総数はシステムがタグ誤りと判定しタグを修正した箇所の総数を意味し，この修正作業を修正と見ずに，タグの推定作業として見た場合には，抽出総数はシステムの推定誤りの総数を意味する．

実験結果の表 1と表 2から以下のことがわかる．

- 検出精度と修正精度はほとんど同程度の精度であり，誤り検出だけでなく修正も同時にやってしまった方がよいことがわかる[11]．
- 概ね最大エントロピー法の方が決定リストよりもコーパス修正の精度が高い．しかし，クローズでの確率推定での上位精度は決定リストも最大エントロピー法に匹敵する精度をあげている．
- クローズでの確率推定を用いる方法の方が，オープンでの確率推定を用いる方法よりも上位精度が高い．しかし，抽出総数はオープンの方が高く，例えば，最大エントロピー法では上位 200 個まで調べた段階で，オープンの方の正しく修正した箇所の数が上回る．
- 方法 1 と方法 2 によるソートでは，概ね方法 1 の方が上位精度が高くなる傾向がある．(われわれの先行研究 [3] では方法 2 しか試していなかったが，方法 1 がよいことが今回の研究であらたにわかった．)
- 「ランダム 300 個」での精度と上位精度を比較すると圧倒的に上位精度の方が高い．方法 1, 方法 2 のいずれにせよ，コーパス修正の確信度に基づいてソートすることが重要であるとわかる．

以上の結果を踏まえると，現状の結果ではコーパス修正としては以下の方略をとるのがよいと思われる．

- まずクローズでの確率推定と方法 1 を用いてコーパス修正を高精度に行なう．次に最大エントロピー法によるオープンでの確率推定と方法 1 を用いてさらに多くのコーパス修正を行なう．

5 おわりに

本稿では，機械翻訳用モダリティコーパスを対象として，機械学習を用いたコーパス修正を行ない，実際にこれによって高品質のモダリティコーパスを作成した．今後このコーパスを用いてモダリティの日英翻訳の研究を行なう予定である [5]．

われわれのコーパス修正の方法は，以下の利点を持っている．

- 確率値を算出するのでこの確率値をコーパス修正の確信度として用いることで修正候補箇所をソートでき，修正の確信度が大きいところからコーパスを修正できる．
- 機械学習を用いる方法なので，機械学習がもともと持っている利点を継承する．
  - 機械学習手法と同程度の汎用性をもち，様々なコーパスの修正に利用できる．
  - 人手で規則を作成する必要がなく，素性を適切に設定するだけでコーパス修正ができる．

[9] 決定リスト法によるオープンでの確率推定では元タグの事例を含めずに確率推定するために元タグの確率が 0 になったり，修正後タグの確率が 1 になったりしやすく，そのような事例が多い場合確率値が同一になり順序関係を決めることができずソートで並べ替えることが困難になる．この場合は，その確率値を求めるのに用いる素性を文脈としてとる事例の総数が多いものほど，ソートで上位に並ぶようにしている．

[10] また，素性の増減による精度の変化の実験も行なったが，ここでは素性を増やすと抽出数が減るが精度があがる傾向があったことだけを述べるにとどめておく．

[11] 精度と別に実際に人手で確認しながら修正をする際にも修正候補がある方がどのように間違っているのかがわかって便利である．コーパス修正に限らず一般に誤りを指摘される場合もどう誤っているのかも示してもらわないと誤っていることに気づかない可能性がある．

表 1: 最大エントロピー法を用いた場合のコーパス修正の精度

		クローズで確率推定 (抽出総数 184 個)				オープンで確率推定 (抽出総数 694 個)			
		検出精度		修正精度		検出精度		修正精度	
ランダム 300 個		69%	(127/184)	68%	(126/184)	28%	(84/300)	26%	(78/300)
方法 1	上位 50 個	100%	(50/ 50)	100%	(50/ 50)	88%	(44/ 50)	88%	(44/ 50)
	上位 100 個	92%	(92/100)	92%	(92/100)	88%	(88/100)	88%	(88/100)
	上位 150 個	77%	(116/150)	77%	(116/150)	80%	(121/150)	79%	(119/150)
	上位 200 個	69%	(127/184)	68%	(126/184)	68%	(136/200)	67%	(134/200)
	上位 250 個	—	—	—	—	60%	(151/250)	59%	(149/250)
	上位 300 個	—	—	—	—	53%	(160/300)	52%	(157/300)
方法 2	上位 50 個	88%	(44/ 50)	88%	(44/ 50)	72%	(36/ 50)	72%	(36/ 50)
	上位 100 個	81%	(81/100)	81%	(81/100)	74%	(74/100)	71%	(71/100)
	上位 150 個	74%	(112/150)	74%	(111/150)	70%	(106/150)	68%	(102/150)
	上位 200 個	69%	(127/184)	68%	(126/184)	67%	(135/200)	65%	(131/200)
	上位 250 個	—	—	—	—	60%	(152/250)	58%	(147/250)
	上位 300 個	—	—	—	—	52%	(157/300)	50%	(152/300)

表 2: 決定リスト法を用いた場合のコーパス修正の精度

		クローズで確率推定 (抽出総数 383 個)				オープンで確率推定 (抽出総数 2,427 個)			
		検出精度		修正精度		検出精度		修正精度	
ランダム 300 個		34%	(104/300)	33%	(101/300)	6%	(18/300)	6%	(18/300)
方法 1	上位 50 個	100%	(50/ 50)	100%	(50/ 50)	56%	(28/ 50)	52%	(26/ 50)
	上位 100 個	92%	(92/100)	92%	(92/100)	43%	(43/100)	40%	(40/100)
	上位 150 個	76%	(115/150)	74%	(112/150)	31%	(47/150)	29%	(44/150)
	上位 200 個	62%	(124/200)	60%	(121/200)	26%	(52/200)	24%	(48/200)
	上位 250 個	51%	(128/250)	50%	(125/250)	22%	(55/250)	20%	(51/250)
	上位 300 個	44%	(132/300)	43%	(129/300)	20%	(61/300)	19%	(57/300)
方法 2	上位 50 個	88%	(44/ 50)	86%	(43/ 50)	66%	(33/ 50)	64%	(32/ 50)
	上位 100 個	86%	(86/100)	84%	(84/100)	48%	(48/100)	46%	(46/100)
	上位 150 個	71%	(107/150)	69%	(104/150)	44%	(66/150)	42%	(63/150)
	上位 200 個	59%	(118/200)	57%	(115/200)	35%	(71/200)	34%	(68/200)
	上位 250 個	50%	(126/250)	49%	(123/250)	30%	(77/250)	29%	(73/250)
	上位 300 個	43%	(129/300)	42%	(126/300)	26%	(80/300)	25%	(76/300)

本稿の研究では，実際の日本語のモダリティ推定に用いる素性とは異なるものを用いたが，推定に用いる素性と同じものを用いてコーパス修正を行なう際には，コーパス修正の研究とタグの推定の研究を同時に行なうことができる[12]．さらには，すでに今まで行なわれている確率に基づいて行なわれている種々のコーパスのタグの推定の研究において，本稿の方法 1 のようなソートを行なうことで，その研究で対象としたコーパスの修正もできると推測される．

コーパス修正の研究はその重要性に比べてあまり議論されてこなかったが，われわれの手法で形態素コーパス [3]，機械翻訳用モダリティコーパスといった多様なコーパスを高精度に修正できたことで，本手法のような確率を利用する方法でたいていのコーパスの修正は可能であると推測される．

[12]各種の機械学習のパッケージにおいて，推定だけでなく本稿の方法に基づいたようなコーパス修正の機能もオプションをつけておくといろいろと便利であるかもしれない．

参考文献

[1] Steven Abney, Robert E. Schapire, and Yoram Singer, Boosting applied to tagging and PP attachment, *EMNLP/VLC-99*, (1999).

[2] Eleazar Eskin, Detecting errors within a corpus using anomaly detection, *NAACL-2000*, (2000).

[3] 村田真樹, 内山将夫, 内元清貴, 馬青, 井佐原均, 決定リスト，用例ベース手法を用いたコーパス誤り検出・誤り訂正，自然言語処理研究会 2000-NL-136, (2000).

[4] 村田真樹, 馬青, 内元清貴, 井佐原均, 用例ベースによるテンス・アスペクト・モダリティの日英翻訳，人工知能学会誌, Vol. 16, No. 1, (2001).

[5] 村田真樹, 馬青, 内元清貴, 井佐原均, サポートベクトルマシンを用いたテンス・アスペクト・モダリティの日英翻訳，電子情報通信学会 言語理解とコミュニケーション研究会 NLC2001, (2001).

[6] 清水護, 成田成寿（編），講談社和英辞典, (講談社, 1976).

[7] 日本電子工業振興協会，自然言語処理システムに関する調査報告書, (2000).

[8] Eric Sven Ristad, Maximum Entropy Modeling for Natural Language, (ACL/EACL Tutorial Program, Madrid,, 1997).

[9] Eric Sven Ristad, Maximum Entropy Modeling Toolkit, Release 1.6 beta, (http://www.mnemonic.com/ software/memt, 1998).