Shift-Reduce 法による依存構造解析におけるビーム幅制御

扇本岳大　　鶴岡慶雅
東京大学 工学部 電子情報工学科　東京大学 工学系研究科
{ougimoto,tsuruoka}@logos.t.u-tokyo.ac.jp

1 はじめに

自然言語処理において基礎的な技術である依存構造解析は、図 1 のように文が与えられたときに、文中の各単語がどの単語に係っているかという係り先やその係り方を表すラベルを予測するタスクである。この技術は機械翻訳、含意関係認識、質問応答など様々な問題に用いられている。依存構造解析の解析手法のひとつである遷移型モデルでの解析では max-violation update による構造化パーセプトロンを用いた学習で高い精度が得られている [1]。しかし、ビームに含まれる候補の数はあらかじめ固定された値となっており、探索の効率性や精度において改善の余地があると考えられる。そこで、本研究では重みベクトルの標準偏差を用いてビームに含まれる候補の数を動的に制御する手法を提案するとともに、その有効性を検証する。

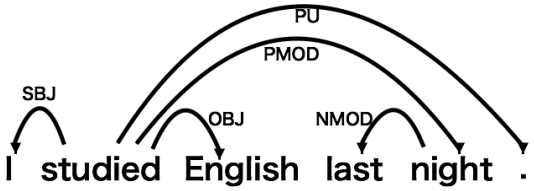

図 1: 依存構造解析

操作	スタック	バッファ	抽出した依存関係
		I studied English last night.	
Shift	I	studied English last night.	
Shift	I studied	English last night.	studied→(SBJ)I
Left(SBJ)	studied	English last night.	
Shift	studied English	last night.	
Right(OBJ)	studied	last night.	studied→(OBJ)English
Shift	studied last	night.	
Shift	studied last night	.	
Left(NMOD)	studied night	.	night→(NMOD)last
Right(PMOD)	studied	.	studied→(PMOD)night
Shift	studied .		
Right(PU)	studied		studied→(PU).

図 2: 依存構造解析例 (ラベルは括弧書きで操作の後や矢印の後に記した)

2 関連研究

2.1 遷移型モデル

依存文法の構文解析器としては大きく分けて二つの種類があり、一つがグラフ型モデル [6] で、もう一方が遷移型モデル [7] である。グラフ型モデルは大域最適化によって構文木全体を評価し最もスコアが高い構文木を出力する方法であるのに対し、遷移型モデルは依存関係情報をスタックとバッファ上での操作列へと変換し、各操作ごとの評価値の合計が最も高い操作列に相当する構文木を出力する方法である。本研究では遷移型モデルを用いて構文解析を行う。遷移型モデルはグラフ型モデルに比べ、豊富な特徴量を用いることができ、高速な割に高い精度が得られるという特徴がある。

遷移型モデルには主に arc-standard モデルと arc-eager モデルがある [8] が、本研究においては arc-standard モデルを使用した。arc-standard モデルにおける操作は以下の三種類である (Left-Reduce と Right-Reduce については各ラベルに対応する操作がそれぞれ一つずつ存在する)。

Shift: バッファの先頭の単語をスタックに入れる

Left-Reduce: スタック 2 番目の単語からスタック先頭の単語への依存関係を抽出し、スタックの 2 番目の単語を削除する

Right-Reduce: スタックの先頭の単語からスタック 2 番目の単語への依存関係を抽出し、スタックの先頭の単語を削除する

初期状態はバッファに全入力単語列が入っていてスタックが空の状態とし、終了条件はバッファが空でスタックにひとつの単語が入った状態である。図 1 の文について実際に構文解析を行った様子を図 2 に示す。

2.2 構造化パーセプトロン

系列ラベリングはある長さ n の入力の系列 $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ に対して、最適であるラベル列 $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ を予測するタスクである。構造化パーセプトロンはこの系列ラベリングにパーセプトロンによる学習を適用したものである。ラベリングの際には、特徴量ベクトルと重みベクトルの内積をスコアとし、スコアが最も大きいラベル列 $\boldsymbol{y}$ を正しいラベル列とする。入力の系列を $\boldsymbol{x}$、出力されるラベル列を $\boldsymbol{y} = (x_1, x_2, ...x_n)$、特徴ベクトルを $\boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y})$、重みベクトルを $\boldsymbol{w}$ としたとき、予測されるラベル列は以下の式のように表される。

$$\arg\max_{\boldsymbol{y}} \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$$

学習では予測が正解だった場合には重みベクトルの更新を行わず、不正解だった場合は以下の式のように重みベクトルの更新を行う。

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y_{correct}}) - \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y_{best}})$$

($\boldsymbol{y_{correct}}$:正解のラベル列、$\boldsymbol{y_{best}}$：予測したラベル列)

2.3 max-violation update

構造化パーセプトロンにおいてすべての候補を探索して最適な系列を発見することは候補の数が膨大となる場合、実際上不可能である。そこで、探索中にスコアが高い方から複数の候補を保持しながら、それぞれの候補について並行して探索を行っていく探索方法であるビームサーチが行われる。依存構造解析においてビームサーチを用いて構造化パーセプトロンによって学習を行うと高い精度が得られることが知られている [10]。その中でも、ビームの中でスコアが最も高い系列と正解の系列のスコアの差が最大となる系列によって学習を行う max-violation update による学習は高速で高い精度が得られている [1]。本研究はこの max-violation update を用いた構造化パーセプトロンによって学習を行う。図 3 にて max-violation update による学習の様子を示す。

2.4 マージンパーセプトロン

正解に対するスコアと予測されたクラスのスコアの差 (マージン) が一定の値以上になるようにパーセプトロン学習を行う学習法をマージンパーセプトロンという [4]。マージンパーセプトロンを用いると、

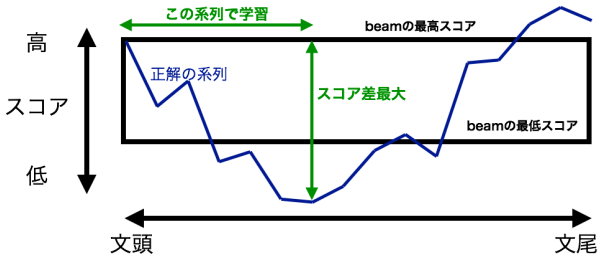

図 3: max-violation update による構造化パーセプトロン

SVM (Support Vector Machine) のようなマージン最大化の手法のように、マージン無しの場合のパーセプトロンよりも学習したモデルの汎化誤差が小さいことが知られている [5]。 正解のクラスのスコアを $score_{correct}$、予測されたクラスのスコアを $score_{best}$ 、マージンを m とすると、マージン無しのパーセプトロンでは $score_{correct} < score_{best}$ の場合に重みベクトルの更新が行われるが、 マージンパーセプトロンでは $score_{correct} - m < score_{best}$ の場合に重みベクトルの更新が行われる。

3 提案手法

従来手法ではあらかじめビームに含まれる候補の数を決めた上で探索を行っている。そのため、ビームの中で最もスコアが高い系列とスコアが近い候補が多くあると、十分に復帰の見込みがあるにもかかわらずビームから外れる候補が出現し、そのことが精度の低下につながっていると考えられる。また、逆にビームの中で最もスコアが高い系列と大幅にスコア差があり、復帰の見込みがほとんどない系列に対しても探索が行われるため、あまり意味のない探索を行っていることになる。

このような事態を防ぐためにはビームに含まれる候補の数をあらかじめ決めておくのではなく、スコアによってビームに含まれる候補の数を動的に制御することが考えられる。そこで、本研究ではスコア幅を定め、ビームの中で最もスコアが高い系列との差がそのスコア幅以下である候補についてはすべて探索を行い、差がスコア幅より大きい候補については探索を行わないことにする。

構造化パーセプトロンによる学習では、学習が進んでいくにつれ、重みベクトルの各成分の絶対値は大きくなっていくため、候補同士のスコアの差がつきやすくなる。そのため、スコア幅はスコア差がつきやすく

なるのと同じオーダーで増加するようにするのが望ましいと考えられる。

そこで、本研究では全重みベクトルの標準偏差にあらかじめ決めておいた定数をかけた値をスコア幅として用いることを提案する。構造化パーセプトロンの性質上、全重みベクトルの平均はゼロベクトルとなるため、全重みベクトルの標準偏差は各成分の二乗和の平方根により求めることができる。この標準偏差を各文の探索前に求め、その値を用いて探索を行う。定数については複数の種類について実際に探索を行い、最も高い精度が得られたものを使用する。

4 実験

4.1 使用データ

データとしては Penn Treebank の Wall Street Journal コーパスを使用し、セクション 2 からセクション 21 までを学習データとし、セクション 22 を開発データとした。Penn Tree Bank は句構造文法によって記述されたデータであるので、CoNLL2007、2008、2009 で使用された変換器 [3] を用いて句構造文法から依存文法へと変換を行った。データ中の品詞については Tsuruoka らによって開発されたタガー [9](精度は約 97.19%) を用いて得られた各単語の品詞情報を使用した。

4.2 特徴量

特徴量については Huang らの研究 [2] に倣い、表 1 に示す 28 種類の特徴量を用いた。ただし、s_i、q_i はそれぞれスタックとバッファの i 番目の単語、α がある一つの単語であるとすると $\alpha.lc$、$\alpha.rc$ はそれぞれ α に係っている単語のうち、判明している中でそれぞれ最も文頭に近い単語と最も文尾に近い単語、$\alpha.w$、$\alpha.t$ はそれぞれ α の単語の種類と品詞を表している。学習を行う前に、正しい係り受け情報が得られるような正解の操作列を用いて全学習データを解析し、各特徴量についてその中で 5 回以上出現したものを特徴量として用いた。

4.3 手法

ベースラインモデルとして、ビームに含まれる候補の個数を一定とし、最終的に予測された系列が正解の場合にはマージンパーセプトロンを行って学習、予測された系列は正解ではないが正解の系列がビームから外れることがなかった場合には通常の構造化パーセプトロン、正解の系列がビームから外れた場合には max-violation update により学習を行うモデルを用いた。提案手法の有効性を調べるため、以下の三つの手法について実験を行った。

表 1: 使用した特徴量

$s_0.w$	$s_0.t$
$s_1.w$	$s_1.t$
$q_0.w$	$q_0.t$
$s_0.w \circ s_0.t$	$s_1.w \circ s_1.t$
$q_0.w \circ q_0.t$	$s_0.w \circ s_1.w$
$s_0.t \circ s_1.t$	$s_0.t \circ q_0.t$
$s_0.w \circ s_0.t \circ s_1.t$	$s_0.t \circ s_1.w \circ s_1.t$
$s_0.w \circ s_1.w \circ s_1.t$	$s_0.w \circ s_0.t \circ s_1.w$
$s_0.t \circ q_0.t \circ q_1.t$	$s_1.t \circ s_0.t \circ q_0.t$
$s_0.w \circ q_0.t \circ q_1.t$	$s_1.t \circ s_0.w \circ q_0.t$
$s_2.t \circ s_1.t \circ s_0.t$	$s_1.t \circ s_1.lc.t \circ s_0.t$
$s_1.t \circ s_1.rc.t \circ s_0.t$	$s_1.t \circ s_0.t \circ s_0.rc.t$
$s_1.t \circ s_1.lc.t \circ s_0.t$	$s_1.t \circ s_1.rc.t \circ s_0.w$
$s_1.t \circ s_0.w \circ s_0.lc.t$	
$s_0.w \circ s_0.t \circ s_1.w \circ s_1.t$	

a ベースラインモデルそのまま。ビームに含まれる候補の個数を一定とした手法。

b ベースラインモデルからビームに含まれる候補の個数を可変として制御に用いるスコア幅を一定とした手法。

c ベースラインモデルからビームに含まれる候補の個数を可変として重みベクトルの標準偏差を用いてスコア幅を定めるようにした手法。

4.4 実験方法

a、b、c の手法に対してそれぞれ複数回学習を行った。学習回数ごとに開発データによって精度を測り、最高の精度が得られる学習回数を調べる。どの手法についてもマージンは 100 で固定とした。

a の手法についてはビームに含まれる候補の数を 20、24、28 の 3 通りについて調べた。

b の手法についてはビームに含まれる候補の数の上限を 100 とし、スコア幅を 100、175、250 の 3 通りについて調べた。

表 2: 実験結果

	UAS	LAS
手法 a	0.91264	0.88161
手法 b	0.91356	0.88236
手法 c	0.91511	0.88470

c の手法についてはビームに含まれる候補の数の上限を 100 とし、重みベクトルの標準偏差にかける定数をを 0.03、0.035、0.04 の 3 通りについて調べた。

4.5 結果

精度は係り先さえ正解ならば正解とみなす基準 UAS と、係り先とラベルの双方が正解でなければ正解としない基準 LAS によって評価を行った。結果は表 2 のようになった。手法 a についてはビームに含まれる候補の数が 24 のとき学習回数 10 回で、手法 b についてはスコア幅が 175 のとき学習回数 9 回で、手法 c については重みベクトルの標準偏差が 0.035 のときに学習回数 12 回でそれぞれ最高の精度が得られた。

4.6 考察

すべてのパターンについて実験を行ってはいないため、確実な検証は行えていないが、基本的にどの手法においても 3 つの実験のうちパラメータ値 (手法 a でビームに含まれる候補の数、手法 b でスコア幅、手法 c で重みベクトル標準偏差にかける定数値) が中央の値のときにおいて最も高い精度が得られている。そのため、今回の実験を行ったパラメータ値の範囲内の値に精度のピークがあると予想される。

今回の実験結果では、手法 a より手法 b のほうが高い精度が得られており、ビームに含まれる候補の数を固定するよりもスコア幅を固定するほうがより高い精度が得られると予想される。また、手法 b よりも手法 c のほうが高い精度が得られており、スコア幅を固定するより重みベクトルの標準偏差によって学習の深度に応じてスコア幅を変動させたほうがより高い精度が得られ、本研究の提案手法が依存構造解析の精度向上に有効であると予想される。

5 おわりに

本稿ではビームに含まれる候補の数を重みベクトルの標準偏差によって制御する手法を提案し、実験によってこの手法が依存構造解析の精度向上に役立つことが予想されることを示した。

今後の課題としては、より詳細な実験によって精度向上が見込まれることを検証し、遷移型モデルでの動的計画法による探索 [2] など、より高い精度が得られるモデルに対しても適用を試みることが挙げられる。また、標準偏差によるスコア幅の制御では学習する文ごとのスコア幅の制御を行うことはできているが、一つの文の探索の中ではスコア幅は一定で制御を行っていない。よって、残った操作の数などに応じて文の中でも動的にスコア幅を制御できる手法を考案する必要がある。

謝辞

本研究は、JST、CREST の支援を受けたものである。

参考文献

[1] Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In *NAACL-HLT*, 2012.

[2] Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *ACL*, 2010.

[3] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for english. In *16th Nordic Conference of Computational Linguistics*, 2007.

[4] Werner Krauth and Marc Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Mathematical and General*, Vol. 20, No. 11, p. L745, 1987.

[5] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz Kandola. The perceptron algorithm with uneven margins. In *ICML*, 2002.

[6] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *ACL*, 2005.

[7] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *IWPT*, 2003.

[8] Joakim Nivre. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, 2004.

[9] Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Kazama. Learning with lookahead: can history-based models rival globally optimized models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, 2011.

[10] Yue Zhang and Stephen Clark. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *EMNLP*, 2008.