B1TB2081

卒業論文

単語分散表現の shift-reduce 型構文解析への利用

小松 広弥

2015 年 3 月 31 日

東北大学
工学部 情報知能システム総合学科

単語分散表現の shift-reduce 型構文解析への利用 *

小松 広弥

内容梗概

依存構文解析は様々な自然言語処理技術の中で，最も基礎的な技術の 1 つである．しかし，単語の意味的，文の構造的な曖昧性から，その精度は約 90 %前半ほどである．本研究では，英語の shift-reduce 型依存構文解析器において，単語の分散表現を素性に利用することで，単語の意味，構文構造的なクラスを捉え，解析の精度が向上することを示す．これは，類似する単語は，その単語に関する依存構文が類似しているという考えに基づく．単語分散表現の構築については，大量の言語データから分布意味論仮説に基づき，周辺単語の統計情報を利用するような一般的な構築手法に加え，解析器の内部状態を利用し，解析器の内部動作に着目した単語分散表現の構築手法を提案する．

キーワード

依存構文解析, shift-reduce 型構文解析, 単語分散表現

*東北大学 工学部 情報知能システム総合学科 卒業論文, B1TB2081, 2015 年 3 月 31 日.

目 次

1　序論

計算機によって大量の自然言語文を処理し，計算機が言語の意味を理解することは，情報検索，質問応答システム，情報抽出などの有用なタスクに生かされている．このような自然言語処理技術で最も基礎部分に当たるものの 1 つとして，依存構文解析がある．この解析を基にして，意味表現解析 [1]，関係知識獲得 [2] など，応用的な自然言語処理技術につながっていく場合が多い．

文内の単語の修飾・被修飾の関係を表す構造を依存構文といい，依存構文解析とは，この依存構文を得る解析である．例えば，”I saw you with her.” という文に対しては，図 1 の様な依存構文が存在する．

一般に計算機による依存構文解析には様々な手法が存在するが，基本的には文内の 2 つの単語間に着目して，その間に依存関係があるかどうかを機械学習によって判定することが多い．そのため，2 つの単語間の情報，主に着目する単語とその周辺単語の表層形，品詞タグから素性を抽出することになる．しかし，この素性をベクトルで表示すると非常に疎で高次元である．特に表層の文字列を利用する素性は，単語の異なりを単位とする次元を考えることになり,「単語の表層文字列が何であるか」という情報しか持っておらず，単語の意味などのまとまりの情報を利用できないことが問題の 1 つとしてあげられる．また，表層文字列のマッチしか考えないため，学習データに存在しない，あるいは低頻度の単語に関して学習が進まないことが 2 つ目の問題として挙げられる．

本研究では，解析器の素性として，意味的，構文構造的な類似度を捉えることができるような単語分散表現を考え，それを単語表層の素性と置き換えることを考える．素性として意味的，構文構造的類似性を導入することにより，単語の類似性による依存構文の類似性を捉えることが可能であることが 1 つ目の利点として挙げられる．また，この類似性を依存構文学習データ以外の大規模なデータから獲得することにより，学習時の未知．低頻度単語を単語の類似性によって補間できることが 2 つ目の利点である．

単語分散表現は，単語間の類似度行列を特異値分解により次元を圧縮することによって構築する．単語間の類似度は分布仮説に基づき，大規模なデータから単語の周辺単語の共起情報の類似度によって定義する．周辺単語として，一般には

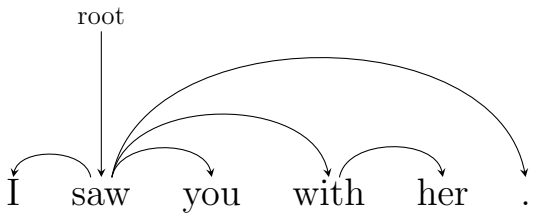

図 1: ”I saw you with her.” に対する依存構文

文内の前後の単語，もしくは依存構文木上の親子の単語をとる事が多いが，本研究では，この 2 つに加え，shift-reduce 型解析器の内部状態における前後の単語を周辺単語とすることを提案する．解析器の動作は内部状態によって決定するため，内部状態を周辺単語とすることで，解析器における動作の類似性を捉えられる．動作の類似性はそのまま解析器の素性として利用するのに適していると考えられる．

評価実験では，これらの単語分散表現を素性として利用することによって，英語における依存構文解析の精度向上が見られた．

2　背景

2.1　依存構文解析

依存関係ラベル $L = \{l_1, \ldots, l_{|L|}\}$ が与えられたとき，文 $x = w_0 \ldots w_n$ に対する依存構文は，方向付きグラフ $G = (V, A)$ で表される．ここで，$V = \{0, \ldots, n\}$ は，単語を表すノード集合，$A \subseteq V \times L \times V$ は，依存関係を表す方向付きアーク集合である．本研究では簡単のためラベルを 1 種類 $L = \{null\}$ とする．例えば，図 1 の文 $x = w_0 \ldots w_5 =$ "I","saw","you","with","her","." に対する依存構文について，$V = \{0, \ldots, 5\}$，$A = \{(1, null, 0), (1, null, 2), (1, null, 3), (3, null, 4), (1, null, 5)\}$ である．1 つの依存関係 $w_i \rightarrow w_j$ に関して，単語 w_j に対して単語 w_i を親，係り元などと，単語 w_i に対して単語 w_j を子，係り先などと呼ぶ．図 1 における 1 つの依存関係 saw $\rightarrow$ I を考えたとき，"saw"が親の単語，"I"が子の単語である．

依存構文解析には大きく分けてグラフベースの解析と遷移ベースの解析が存在する [3]．グラフベースの解析として，最大全域木アルゴリズムを用いるもの [4] や CKY アルゴリズムを用いるもの [5] が代表的である．一方，遷移ベースの解析は shift-reduce 型解析器を用いるもの [6] が代表的である．この研究では遷移ベースの手法の 1 つである shift-reduce 型の解析に着目する．

2.2　shift-reduce 型依存構文解析

shift-reduce 型依存構文解析は，入力の文に対し，左から右に単語を走査し解析を行い，親単語が決定していない単語を処理中の単語としてスタックに積んで処理する手法である．例えば，"I saw you with her." という文に対しては，図 2 のように，"I","saw",... と順に走査し処理していく．

未処理の単語と処理中の単語を格納するスタックを合わせて 1 つの状態として，そこに 3 種類の動作 {`shift`, `reduce-right`, `reduce-left`} を行うことで依存関係を決定する．`shift` は未処理単語の先頭をスタックに積み，次の単語に走査する．`reduce` はスタックの先頭 2 単語に対して依存関係を認め，係り先の単語をスタックから削除する．このとき，`reduce-right` の場合は，スタック先頭の単語が

ステップ	動作	内部状態		
		スタック	未処理単語	アーク
0	-	[ ]	[I saw with her .]	{}
1	shift	[I]	[saw you ...]	{}
2	shift	[I saw]	[you with ...]	{}
3	reduce-left	[saw]	[you with ...]	{(saw, I)}
4	shift	[saw you]	[with her ...]	{(saw, I)}
5	reduce-right	[saw]	[with her ...]	{(saw, I), (saw, you)}
...				

図 2: shift-reduce 型依存構文解析の内部動作

係り先となり，reduce-left の場合は，スタック 2 単語目が係り先となる．これら 3 つの動作のうち 1 つを取ることを 1 ステップとし，これらを繰り返すことによって文全体の依存構文を求める解析である．例えば，”I saw you with her.”という文に対しての依存構文を得る場合，図 2 のようなステップが取られる．1 ステップ目は動作 shift が選ばれ，未処理単語先頭の単語”I”がスタックに積まれ，次の単語”saw”に走査する．3 ステップ目では動作 reduce-left が選ばれ，依存関係 saw → I が認められ，子の単語”I”がスタックから除去される．

解析の動作選択を行うために，1 つの状態に対してスコアを定義する．解析時は最終的にスコアが最大となるような状態を探索する．各動作 {shift, reduce-right, reduce-left} に対するスコアの増分 $\{\xi, \lambda, \rho\}$ は，式 (1-3) で示すように，状態から抽出される素性ベクトルと予め学習された重みベクトルとの内積で定義される．

$$\xi = \mathbf{w} \cdot f_{\texttt{shift}}(S, i) \tag{1}$$

$$\lambda = \mathbf{w} \cdot f_{\texttt{reduce-left}}(S, i) \tag{2}$$

$$\rho = \mathbf{w} \cdot f_{\text{reduce-right}}(S, i) \tag{3}$$

各動作それぞれの素性 $f_{\texttt{shift}}(S, i)$，$f_{\texttt{reduce-left}}(S, i)$，$f_{\texttt{reduce-right}}(S, i)$ は，素性セットと動作を表す素性の結合になっている．素性セットは未処理単語およびス

タック内の単語の表層形及び品詞タグを抽出し，それらの組み合わせによって定義する．例えば図 2 の 2 ステップ目の状態について，スタック 1 単語目と 2 単語目の組合せ素性 s_0.w $\circ$ s_1.w の素性は，式 (4) になる．

$$(s_0.\mathrm{w} = \mathrm{saw}) \circ (s_1.\mathrm{w} = \mathrm{I}) \tag{4}$$

これは，素性ベクトルのある次元が「スタックの先頭の単語が ”saw”であること，かつスタック 2 番目の単語が ”I”であること」を表し，その次元の成分が 1 であることを表す．さらに式 (4) の素性は，動作を表す素性と結合して素性ベクトルの 1 つの素性になる．例えば動作 `shift` を取るときの素性は，式 (5) になる．

$$(s_0.\mathrm{w} = \mathrm{saw}) \circ (s_1.\mathrm{w} = \mathrm{I}) \circ (action = \texttt{shift}) \tag{5}$$

1 つの状態に対する素性ベクトルは，このようなベクトルの足し合わせ，すなわち 0,1 だけの 2 値ベクトルになっている．また，素性の次元は単語の表層形と品詞タグの種類数，またそれらの組合せだけの大きさになっているため，素性ベクトルは非常に高次元で，疎なベクトルである．

素性セットは，Huang[7] のパーザに従うが，表 1 で示すように，未処理の単語，スタック，スタックの単語が依存関係にある単語の表層形及び品詞タグの組合せから成り立っている．ただし，図 4 に示すように，s_i，q_i はそれぞれスタック，未処理単語の i 番目の単語を表し，s_i.lc（s_i.rc）は，スタック内の単語 s_i の最も左（右）の係り先単語である．w.w，w.t はそれぞれ単語 w の表層，品詞タグを表す．

以上を定式化すると，図 3 になる．

重みベクトルは，真の依存構文が付与されたコーパスを用いた教師あり学習により得られる．文 $x = w_0 \ldots w_{n-1}$ と真の依存関係集合 A_{gold} が与えられたとき，アルゴリズム 1 によって動作列 y_{gold} を一意に得ることができる．図 1 の依存構文において，真の動作列は式 (6) である．

$$\begin{gathered}\texttt{shift}, \texttt{shift}, \texttt{reduce-left}, \texttt{shift}, \texttt{reduce-right}, \texttt{shift}, \\ \texttt{shift}, \texttt{reduce-right}, \texttt{reduce-right}, \texttt{shift}, \texttt{reduce-right}\end{gathered} \tag{6}$$

[1]記号”|”はスタックと単語の結合を表す．例えば S = [I saw] に対して $S|$you = [I saw you] である．

状態	$\langle S, i, A, c\rangle$ ただし， S:スタック，i: 未処理単語先頭位置，c:スコア， $(w_a, w_b) \in A$: w_a から w_b への依存関係
入力	$w_0 \ldots w_{n-1}$:文
初期状態	$\langle \emptyset, 0, \emptyset, 0\rangle$
動作	$\texttt{shift}(\langle S, i, A, c\rangle) = \langle S\|w_i, i+1, A, c+\xi\rangle$ [1] --- $\texttt{reduce-right}\ (\langle S\|s_1\|s_0, i, A, c\rangle) = \langle S\|s_1, i, A \cup (s1, s0), c+\lambda\rangle$ --- $\texttt{reduce-left}\ (\langle S\|s_1\|s_0, i, A, c\rangle) = \langle S\|s_0, i, A \cup (s0, s1), c+\rho\rangle$
終了状態	$\langle s_r, n, A, c\rangle$

図 3: shift-reduce 型依存構文解析の定式化

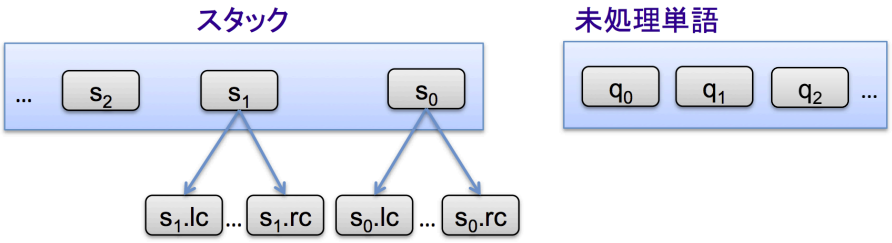

図 4: 内部状態と素性の範囲

	素性セット $\mathbf{f}(S,i)$	$s_0, s_1, \cdots =$ スタック $q_n = w_{i+n}$:未処理単語
1 グラム	s_0.w s_1.w	q_0.w
	s_0.t s_1.t	q_0.t
2 グラム	s_0.w ∘ s_0.t	s_0.w ∘ s_1.w
	s_1.w ∘ s_1.t	s_0.t ∘ s_1.t
	q_0.w ∘ q_0.t	s_0.t ∘ q_0.t
3 グラム	s_0.w ∘ s_0.t ∘ s_1.t	s_0.t ∘ s_1.w ∘ s_1.t
	s_0.w ∘ s_1.w ∘ s_1.t	s_0.w ∘ s_0.t ∘ s_1.w
	s_0.t ∘ q_0.t ∘ q_1.t	s_1.t ∘ s_0.t ∘ q_0.t
	s_0.w ∘ q_0.t ∘ q_1.t	s_1.t ∘ s_0.w ∘ q_0.t
	s_2.t ∘ s_1.t ∘ s_0.t	
	s_1.t ∘ s_1.lc.t ∘ s_0.t	s_1.t ∘ s_1.rc.t ∘ s_0.t
	s_1.t ∘ s_0.t ∘ s_0.rc.t	s_1.t ∘ s_1.lc.t ∘ s_0.t
	s_1.t ∘ s_1.rc.t ∘ s_0.w	s_1.t ∘ s_0.w ∘ s_0.lc.t
4 グラム	s_0.w ∘ s_0.t ∘ s_1.w ∘ s_1.t	

表 1: 素性セット

Algorithm 1 真の動作列を求めるアルゴリズム

入力 :文 x, 真の依存関係集合 A_{gold}

$State \leftarrow \langle \emptyset, 0, \_, \_ \rangle$

$y \leftarrow []$

repeat

  if $\langle S|s_1|s_0, i, \_, \_ \rangle$ and $(s_1, s_0) \in A_{gold}$ and $\nexists w[(s_0, w) \in A_{gold}]$ **then**

    $y.append($`reduce-right`$)$

    $State \leftarrow$`reduce-right`$(State)$

  else if $\langle S|s_1|s_0, i, \_, \_ \rangle$ and $(s_0, s_1) \in A_{gold}$ and $\nexists w[(s_1, w) \in A_{gold}]$ **then**

    $y.append($`reduce-left`$)$

    $State \leftarrow$`reduce-left`$(State)$

  else

    $y.append($`shift`$)$

    $State \leftarrow$`shift`$(State)$

  end if

until State が終了状態

return y

Algorithm 2 構造化平均パーセプトロン

入力: 学習データ $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$, 素性関数 Φ

$\mathbf{w}, \mathbf{aw} \leftarrow \vec{0}$

$n \leftarrow 0$

repeat

  for all example (x, y) in D **do**

    $n \leftarrow n + 1$

    $z \leftarrow PREDICT(x, \mathbf{w})$

    if $z \neq y$ **then**

      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x, y) - \Phi(x, z)$

      $\mathbf{aw} \leftarrow \mathbf{aw} + n(\Phi(x, y) - \Phi(x, z))$

    end if

  end for

until coveraged

return $\mathbf{w} - \mathbf{aw}/n$

2.3 構造化パーセプトロン

パーセプトロンはデータセット D 内から 1 つづつ例 d を取り出し，それに対して重みベクトルを更新するオンライン学習である．1 つの例 d は，入力 x と，正しい出力構造 y からなる．重みベクトルが学習できたとき，予測は素性関数を Φ としたとき，式 (7) によって行われる．

$$\arg\max_{y} \mathbf{w} \cdot \Phi(x, y) \tag{7}$$

Algorithm2 に一般的な構造化平均パーセプトロンのアルゴリズムを示す．$PREDICT$ 関数は入力と現在の重みベクトル $\mathbf{w}$ を用いて予測を行い，その結果の構造 z を返す関数である．予測は，x と y から，Φ によって素性を抽出し，式 (7) によって決定する．現在の重みベクトルによる予測が間違っていれば，重みベクトルに正しい素性を加え，予測されたものを負例として素性から減じることによって，間違えた予測が正しい方向に修正されるように重みベクトルを更新する．また平均化

パーセプトロンは学習データを何回か走査して重みを更新し，その重みベクトルの平均を出力することで性能が向上することが知られている．

shift-reduce 型構文解析の学習を構造化パーセプトロンで行う場合では，文 x と文に対応する真の動作系列 y から解析器の重みベクトルを学習する．動作系列をビーム探索で学習・予測する場合，真の系列が探索の途中でビームから外れてしまうと正しく予測できなくなってしまうため，どの負例で重みベクトルを更新するかが重要になる．この問題に対する学習手法として max-violation 構造化パーセプトロン [12] という手法がある．これは正例に対して，一番予測が間違ったところまでの素性で重みベクトルを更新する手法である．具体的には，系列の各ステップにおいて一番良い予測のスコア，つまりビームの先頭のスコアと真のスコアの差が最大となったときの素性を用いて重みベクトルを更新する．max-violation 構造化パーセプトロンのアルゴリズムを Algorithm3 に示す．ただし関数 $MAX\text{-}VIOLATION$ は，i ステップ目のビーム先頭を B_i^0 としたとき, 式 (8) である．

$$\arg\min_{y^*, z^* \in B_i^0} \Phi(x, y^*) - \Phi(x, z^*) \tag{8}$$

本研究ではこの max-violation 構造化パーセプトロンを用いて学習を行う．

Algorithm 3 max-violation 構造化平均パーセプトロン

入力: 学習データ $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$, 素性関数 Φ

$\mathbf{w}, \mathbf{aw} \leftarrow \vec{0}$

$n \leftarrow 0$

repeat

  for all example (x, y) in D **do**

    $n \leftarrow n + 1$

    $y', z \leftarrow MAX - VIOLATION(x, y, \mathbf{w})$

    if $z \neq y'$ **then**

      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x, y') - \Phi(x, z)$

      $\mathbf{aw} \leftarrow \mathbf{aw} + n(\Phi(x, y') - \Phi(x, z))$

    end if

  end for

until coveraged

return $\mathbf{w} - \mathbf{aw}/n$

3　手法

この章では，shift-reduce 型依存構文解析器の内部状態を利用した単語分散表現の作成手法と，その分散表現を素性として再度学習の素性に組み込む事を考える．

まず，3.1 節で，既存手法の問題点と単語分散表現の解析への素性利用について利点と方法を述べる．次に，3.2 節で，本研究で提案する解析器の動作の類似性を捉えた単語分散表現を含めた，単語分散表現の構築手法について述べる．

3.1　単語類似性の素性利用

既存手法の素性セットは，2.2 節でも述べたように，単語の表層形及び品詞タグの組合せを考えているため，素性空間及び重みベクトルは非常に高次元で，素性は非ゼロ要素が次元に対して非常に少ない疎なベクトルになっている．

この素性の一部として s_0.w の素性とそれに対応する重みベクトル $\mathbf{w}_{s_0.\mathrm{w}}$ を考える（例えば，図 5 上部ではスタックの先頭が”saw”，動作が `shift` の時の素性ベクトルの一部を表している）．この素性ベクトルと重みベクトルは，単語の表層の種類と 3 種類の動作の組合せ数だけの次元が必要になる．単語の異なり数は数万の単位であるため，これらは非常に高次元であることがわかる．つまりこの素性は,「単語の表層文字列が何であるか」という情報しか持っておらず，単語の意味や品詞タグ以上の構文構造に関する情報を利用できないことが 1 つ目の問題点として挙げられる．また単語表層のマッチしか考えないため，学習データにおいて低頻度や未知の単語に対して学習が進まないことが 2 つ目の問題点として挙げられる．

ここで，例えば，”saw” という単語に関して，主語に関する係り先は，”I” のように「見る」という動作を取れるような,「人間・生物」を表す単語が来ることが多く，目的語に関する係り先は，”you” といったような,「実体物」を表す単語が来やすい，といったような依存関係の選好性が存在する．一方，似た意味を持つ”watched” という単語に関しても”saw ”の依存関係と似た依存関係を持つことが考えられる．そこで，単語の意味，構文構造などの類似性を解析の素性として加えることによって，依存関係の類似性を捉えた解析ができることが 1 つ目の問

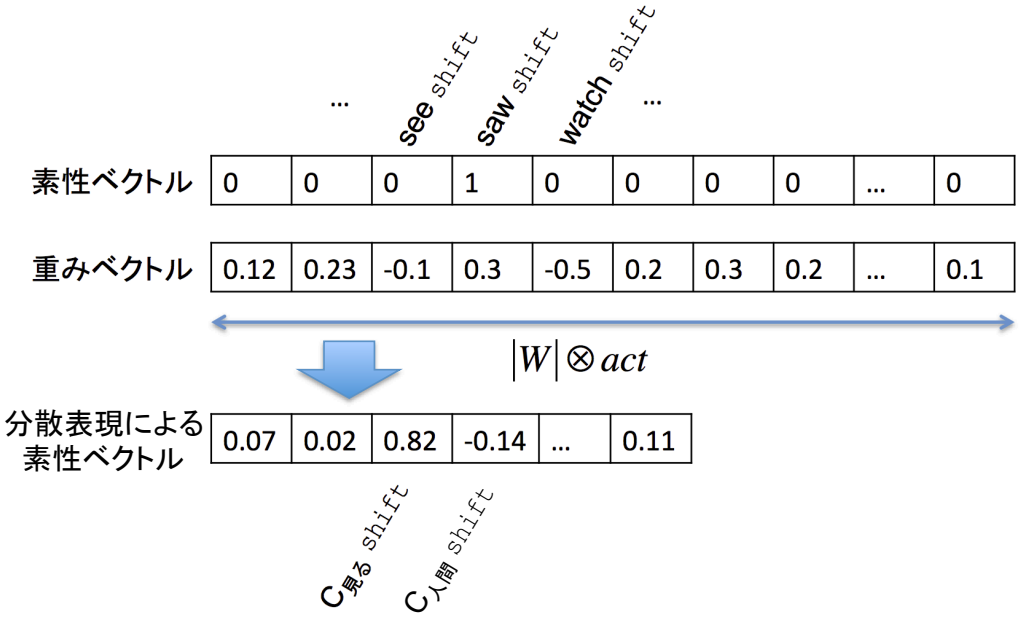

図 5: 素性ベクトル

題を軽減すると考えられる．またこの類似性を学習データ以外のデータから得ることによって学習データにおいて未知・低頻度の単語を類似単語によって補間することで 2 つ目の問題に対処する．

本研究では，素な素性ベクトルを単語の類似性を捉えた密な分散表現に置き換えることを考える．素性を意味的，構文構造的な類似性を捉えた密な空間に抑えることで，先に述べたような依存関係の類似性を捉えられるような解析になると考えられる．また，この単語分散表現を，学習データ以外の大規模な言語データから構築することによって，学習データに存在しない，または低頻度の単語の分散表現を得ることができる．それらの単語を意味的，構文構造的な類似性が利用できる解析器で学習・解析することによって未知，低頻度の単語に対しても適切な構文構造を得ることができると考えられる．

図 5 では，単語の意味に着目し，各次元は，例えば「見る」という意味を持つ単語であることを表す次元,「人間」という意味を持つ単語であることを表す次元,などということを概念的に表現している．ただし，各次元の持つ意味は，必ずし

も人間が一意に解釈できるとは限らない．

3.2 節で，本研究で提案する手法を含めた，解析器の素性として利用する単語分散表現の構築手法について述べる．

3.2 単語分散表現の構築

単語分散表現を構築する一般的な手法として，単語を周辺文脈の分布のベクトルと考え共起の情報を計測し，単語の類似度を考える研究がある [8]．これは，単語の意味は周辺文脈の単語の分布によって決まるという分布仮説 [9] に基づいている．近年では，Mikolov ら [10] や，Socher ら [11] がニューラルネットワークを用いて意味的構成性を捉えた単語ベクトルを作成している．

素性に利用する単語分散表現を構築する手法として，文の周辺単語を利用する方法，依存構文木の周辺単語を利用する方法，さらに解析器の内部状態に基づく手法を考える．

解析器の内部状態に基づく分散表現

本研究で提案する shift-reduce 型依存構文解析器の内部状態を利用した単語分散表現について述べる．

この単語分散表現は，3.1 節で述べたように，ターゲットとする単語がスタックの先頭に来たとき，解析器が取る動作の類似性を捉えたものになることが望ましい．ここで，既存手法の素性（表 1）を考えると，解析器の動作は，高々ターゲット単語の周辺 1 単語の表層形及び品詞タグ，加えて周辺 3 単語とその係り先単語の品詞タグのみで決定しているため，スタック先頭単語をターゲットとしたとき，解析器が取る動作の類似性は，解析器の各ステップにおける周辺単語の分布の類似性によって捉えられると考えられる．

そこで，スタックの先頭単語をターゲットとして，解析器の各ステップにおける周辺単語の分布を計測することによって単語の分散表現を得ることを提案する．すると，単語間の解析器の動作の類似度は分散表現のコサイン類似度で捉えることができる．

周辺単語を 4 つの位置 $\mathrm{p} \in \{\mathrm{s}, \mathrm{s\ in\ win}, \mathrm{q}, \mathrm{q\ in\ win}\}$，に分割し，ターゲットの単語 $w_i \in W$ と，コンテキスト単語 $c_j \in C_p$ から，各単語の出現頻度 $C_w(w_i)$,$C_c(c_j)$ とバイグラム出現頻度 $C(w_i, c_j)$ を計測する．これらの出現頻度により単語-コンテキストの PMI 行列を作成する．図 4 において，ターゲットの単語は s_0，コンテキスト単語は，位置 s においては s_1，位置 s in win においては $\{s_1, s_2, s_3\}$，位置 q においては q_0，位置 q in win においては $\{q_0, q_1, q_2\}$ とする．

$$\mathbf{P}_{\mathrm{p}} = [\mathrm{PPMI}(w_i, c_j)] \in \mathbb{R}^{n \times m}\ \forall \mathrm{p} \tag{9}$$

ただし，$n = |W|$，$m = |C_p|$ であり，共起の指標は PPMI(Positive Pointwise mutual information) を用いる．

$$\mathrm{PPMI}(w_i, c_j) = \max(\log \frac{p(w_i, c_j)}{p(w_i)p(c_j)}, 0) \tag{10}$$

ただし，$p(w_i, c_j) = \frac{C(w_i, c_j)}{\Sigma C(w_i, c_j)}$ は，w_i と c_j が同時に出現する確率，$p(w_i) = \frac{C_w(w_i)}{\Sigma C_w(w_i)}$，$p(c_j) = \frac{C_c(c_j)}{\Sigma C_c(c_j)}$ は，それぞれ w_i と c_j の出現確率である．各 $\mathbf{P}_{\mathrm{p}}$ の列ベクトル $\mathbf{P}^*_{\mathrm{p}}[i:]$ は，ターゲット単語 w_i の各位置における周辺単語の分布を示すベクトルになっている．PMI は低頻度の事例については誤差が大きくなるため，$\mathbf{P}_{\mathrm{p}}$ はノイズが含まれる可能性がある．そのため，各行列 $\mathbf{P}_{\mathrm{p}}$ を特異値分解により $\mathbf{P}^*_{\mathrm{p}} \in \mathbb{R}^{n \times k}$ に次元圧縮を行う．

すべての位置の類似度を考慮して分散表現を構築するために，周辺単語分布を示す各 $\mathbf{P}^*_{\mathrm{p}}$ から単語類似度行列を作成し，その要素積行列 (**multi**) と，連結行列 (**concat**) を考える．

$$\mathbf{S}_{\mathrm{p}} = [\mathbf{sim}(w_i, w_j)] \in \mathbb{R}^{n \times n}\ \forall \mathrm{p} \tag{11}$$

$$\mathbf{S}_{\mathrm{multi}} = [\prod_{p} \mathbf{S}_{\mathrm{p}}[i:j]] \in \mathbb{R}^{n \times n} \tag{12}$$

$$\mathbf{S}_{\mathrm{concat}} = [\mathbf{S}_{\mathrm{s}}|\mathbf{S}_{\mathrm{s\ in\ win}}|\mathbf{S}_{\mathrm{q}}|\mathbf{S}_{\mathrm{q\ in\ win}}] \in \mathbb{R}^{n \times 4n} \tag{13}$$

ただし”|”は行列の結合である．類似度は非負のコサイン類似度を考える．

$$\mathbf{sim}(w_i, w_j) = \max(\frac{\mathbf{P}^*_{\mathrm{p}}[i:] \cdot \mathbf{P}^*_{\mathrm{p}}[j:]}{|\mathbf{P}^*_{\mathrm{p}}[i:]||\mathbf{P}^*_{\mathrm{p}}[j:]|}, 0) \tag{14}$$

S_{multi} と S_{concat} は単語-すべての位置の周辺単語の類似度行列になっている．これを特異値分解によって次元圧縮を行うことで，類似する単語に対応する次元を縮退させ，圧縮後の次元は単語の動作的まとまりを捉えることになる．つまり特異値分解後の類似度行列 $S^*_{\text{multi}}, S^*_{\text{concat}} \in \mathbb{R}^{n \times d}$ の列ベクトルは，列に対応する単語の d 次元の分散表現になる．

文の周辺単語による分散表現

単語の意味は周辺文脈の単語の分布によって決まるという分布仮説に基づき，周辺単語を文の前後 3 単語として，同様の手法で単語分散表現を構築する．

文 $x = w_0 \dots w_n$ において，ターゲットの単語を w_i としたとき，コンテキスト単語を $\{w_{i-1}, w_{i-2}, w_{i-3}, w_{i+1}, w_{i+2}, w_{i+3}\}$ として，出現頻度を計測，PMI により単語の周辺単語分布ベクトルを作成する．PMI 行列はノイズ除去のために特異値分解を施す．PMI 行列から単語類似度行列を作成し，特異値分解により単語分散表現を構築する．

依存構文木上の周辺単語による分散表現

周辺単語を依存構文木上の親子の単語によって分散表現を構築する．

依存構文 $G = (V, A)$ において，ターゲットの単語を w_i としたとき，コンテキスト単語を w_i の親単語 3 世代，子単語 3 世代として，出現頻度を計測，PMI により単語の周辺単語分布ベクトルを作成する．ただし，コンテキストを親と子及びその深さで区別する．この手法も同様に PMI 行列から単語類似度行列を作成し，特異値分解により単語分散表現を構築する．

4　実験

4.1　実験設定

まず，ベースラインとして Huang らの研究 [7] に基づく non-DP shift-reduce 型パーザを実装し，1 文に対する動作列を max violation 構造化パーセプトロン [12] で学習した．ビーム幅は 8 とした．学習，評価は Penn Treebank の Wall Street Journal コーパスの分割データを用い，02 - 21 を学習データ，22 を開発データ，23 をテストデータとした．重みベクトルは 0 ベクトルで初期化し，学習データ全体に対するイテレーションの回数を 10 回としてパーザを学習した．

次に，New York Times の 2007 年 1 月から 6 月までの 6ヶ月分の本文（1,578,645 文）に対して，品詞タグを Stanford POS Tagger[13] で得た上で，学習したベースラインのパーザで解析を行い，3.2 節の手法で 4 種類の単語分散表現（解析器の内部状態に基づく分散表現 (**multi**, **concat**)，文の周辺単語による分散表現 (**linear**)，依存構文木城の周辺単語による分散表現 (**tree**)）を作成した．ただし，出現頻度 50 未満の単語は未知単語として 1 つの単語として考え PMI を計測した．学習・解析時の単語に分散表現が存在しなければ未知単語のベクトルが用いられる．分散表現の次元は 300 次元とした．

また参考として Mikolov ら [10] の手法による公開されている単語分散表現（w2v）とも比較した．この分散表現は Google News データセット約 1000 億トークンで学習されている．

これらの単語分散表現で，3.1 節で述べたように単語の素性を置き換え再度パーザの学習を行った．学習データ，学習アルゴリズム，イテレーションの回数は baseline の方法に準拠した．

評価は記号を除いたすべての単語に対して，係り元の単語の正解率で評価する．

4.2　結果・考察

依存構文解析精度

ベースラインと各分散表現を素性として用いたときの結果を表 2 に示す．

	精度 (%)	
手法	開発セット	テストセット
baseline	90.83	90.90
multi	91.13	90.98
concat	91.34	91.32
linear	91.08	90.94
tree	91.48	91.44
w2v	91.44	91.31

表 2: 依存構文解析の精度

baseline との比較によると，単語分散表現を素性に利用することによって，解析の精度が向上した．本研究で提案した手法を比較すると，**multi** より，**concat** の方がより精度が向上した．**multi** の場合，全てのコンテキストの位置の類似度が大きくならないと単語の類似度が大きくならないため，**concat** より動作の類似性が捉えられていないと考えられる．一方で **concat** は，次元圧縮の段階で類似性の識別に有用なコンテキストの位置を選択することができるので，解析の精度が上がっていると考えられる．

単語分散表現を利用した手法の中では，**tree**，次いで **concat** が高精度であった．これらの手法は，学習データも大きくニューラルネットワークを利用している **w2v** の手法に匹敵している．

単語類似度

ある単語についてそれに類似する上位 5 件の単語を，各単語分散表現のコサイン類似度によって求め，これを比較した（表 3,4）．表 3 において，対象の単語と構文的役割が異なると思われる類似単語を下線で示した．また，表 4 において，対象の単語と意味クラスが異なると思われる類似単語を下線で示した．

表 3 より，**linear** の手法より **multi**，**concat**，**tree** の手法のほうが構文的役割が捉えられていると考えられる．**linear** の手法は文の周辺単語を見ているため，例えば助動詞に対して”not”などの，隣接しやすい単語の類似度が高くなってし

まう．対して，他の手法は，周辺単語として依存構文を見ているため，依存構文的に似ている単語が類似し，構文的役割が捉えられているといえる．

一方，表 4 より，**tree** の手法より **multi**，**concat**，**linear** の手法のほうが意味クラスを捉えられていると考えられる．単語の意味は，主に文においての周辺単語に依存していると考えられる．

提案手法 **multi**，**concat** は，両者の特徴を捉えている．これは，解析初期のステップでは，主に文においての周辺単語がカウントされ，終盤のステップに行くにつれ，依存構文上の周辺単語がカウントされているからであると考えられる．

この結果を踏まえ，再度解析の精度について考えると，解析の精度は構文的役割が捉えられていることが重要である可能性がある．

(a) might: 助動詞

concat	**multi**	**linear**	**tree**
could	should	should	should
should	must	n't	does
would	can	could	must
can	could	how	could
must	may	if	did

(b) gave: 他動詞 (V A to B)

concat	**multi**	**linear**	**tree**
took	turned	took	took
brought	took	came	came
got	brought	went	got
turned	found	added	showed
asked	added	sent	turned

表 3: (a)might (b)gave に対する類似度上位 5 単語
ただし，構文的役割が異なると思われる単語を下線で示す.

(c) noon

concat	**multi**	**linear**	**tree**
a.m.	9:30	11:30	a.m.
11:30	8:30	9:30	sundays
pm	4:30	10:30	saturdays
9:30	10:30	7:30	mondays
6:30	7:30	1:30	10:30

(d) foods

concat	**multi**	**linear**	**tree**
product	ingredient	organic	goods
ingredients	drinks	menu	technologies
drinks	wines	beer	airlines
food	food	wine	brands
wine	wine	coffee	chips

表 4: (c)noon (b)foods に対する類似度上位 5 単語
ただし，意味が異なると思われる単語を下線で示す.

5　関連研究

依存構文解析の素性として，単語の意味，構文構造などの情報を用いる研究は多く行われている．

Koo ら [14] は Brown アルゴリズム [15] で求められるクラスタ情報を素性として用いることでグラフベースの手法の依存構文解析の精度が向上している．このクラスタ情報は，n グラム単語の出現頻度情報によって階層的クラスタリングを行った結果であり，0,1 の 2 ビットによるビット列で単語が表現される．素性には先頭の数個のビット列をそのまま素性として用いている．Banasal ら [16] は，連続値のベクトルを作成する様々な手法の単語ベクトルを用いて単語クラスタ情報を得て，依存構文解析に利用する研究である．しかしこのクラスタ情報も，連続値のベクトルから階層的クラスタリングにより，2 ビットのビット列に直してから，先頭数個のビット列を素性として用いているものであり，グラフベースの手法の依存構文解析で精度向上が見られる．以上の 2 つの手法は，階層的クラスタリングに基づくビット列を利用しているため，単語がどのクラスタに属するか，という情報を解析器に与えていることになる．

Andreas ら [17] は外部データから構築した単語分散表現をそのまま素性として組み込むことは言語処理のさまざまなタスクにおいて，1. 学習データにない単語を類似単語によって補間できること，2. 類似単語の統計をプールして使えること，3. 分散表現がそのまま素性として使うのに適していること，の 3 つの利点があることを述べていて，その効果を確率的文脈自由文法を用いた依存構文解析の精度向上によって示している．

6　結論

shift-reduce 型依存構文解析の素性に単語分散表現を利用することによって，解析の精度が向上することが確認できた．これは，単語の類似度，特に構文構造的な類似度によって，依存構文の選好性を学習・解析に利用できていることによると考えられる．

また解析器の内部状態に基づく単語分散表現の構築手法を提案した．この分散表現も，一般的に用いられる文上，依存構文木上の周辺単語に基づく手法と同様に解析器の精度に貢献することが確認できた．また定性的評価であるが，この分散表現は意味・構文構造的類似度を同時に捉えていることを確認した．内部状態の周辺単語の取り方には，係り先単語として除かれた単語を使ったり，ターゲットとの距離の情報を入れるなど改善の余地があると考えられる．これによって解析の精度は更に向上する可能性がある．

謝辞

本研究を進めるにあたり，ご指導頂いた乾健太郎教授，岡崎直観准教授に感謝致します．研究活動，本論文の執筆全般に渡り，直接のご指導，適切な助言をくださった田然研究特任助教に感謝致します．日常の議論や研究会で様々な知識や示唆をくださった乾・岡崎研究室の皆様に感謝致します．

参考文献

[1] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1–10, Singapore, August 2009. Association for Computational Linguistics.

[2] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[3] Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency Parsing. Morgan and Claypool, 2009. [Pedagogical].

[4] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pages 523–530, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[5] Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics, 1996.

[6] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. Comput. Linguist., 34(4):513–553, December 2008.

[7] Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 1077–1086, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[8] Dekang Lin. Automatic retrieval and clustering of similar words. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2, ACL '98, pages 768–774, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[9] Zellig Harris. Distributional structure. Word, 10(23):146–162, 1954.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 3111–3119. Curran Associates, Inc., 2013.

[11] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2012.

[12] Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT '12, pages 142–151, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[13] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60, 2014.

[14] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In Proceedings of ACL-08: HLT, pages 595–603, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[15] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. Comput. Linguist., 18(4):467–479, December 1992.

[16] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 809–815. Association for Computational Linguistics, 2014.

[17] Jacob Andreas and Dan Klein. How much do word embeddings encode about syntax? In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 822–827, Baltimore, Maryland, June 2014. Association for Computational Linguistics.