B5IM2017

修士論文

分散表現による外部知識の自然言語解析への適用

小松 広弥

2017年2月10日

東北大学 大学院
情報科学研究科 システム情報科学専攻

本論文は東北大学 大学院情報科学研究科 システム情報科学専攻に
修士 (情報科学) 授与の要件として提出した修士論文である。

小松 広弥

審査委員：

乾 健太郎 教授　（主指導教員）
木下 賢吾 教授
大町 真一郎 教授
岡崎 直観 准教授　（副指導教員）

分散表現による外部知識の自然言語解析への適用*

小松 広弥

内容梗概

自然言語解析は自然言語の文から，その文が持つ構文的，意味的構造を得るものであり，多くの自然言語処理の応用の基礎となる．自然言語解析は一般的に教師あり学習によって構築されるが，その性能向上のためには訓練データとは異なるあらゆる文に対して解析が行えるように，素性の設計や教師データの拡充を行う必要がある．自然言語解析の素性には，単語文字列の異なりに注目した単語表層素性が効果的でよく用いられるが，訓練時に未知の単語に対して学習ができないことや，既知の単語に対して過学習してしまうことが問題として挙げられる．また，文が持つ構文，意味構造の複雑さから，多量の訓練データを人手で用意することも非常に手間がかかる．本論文では，まず単語表層素性の欠点に対応するため，単語分散表現を用いた分散表現素性を提案する．また意味表現解析においては，意味表現の背後にあるフレームワークに基づいて部分的な意味表現を学習に加えることを提案する．実験では，依存構文解析と意味表現解析に対しての性能向上が見られた．また分散表現素性の利点と，意味表現解析の課題点を分析した．

キーワード

依存構文解析，Abstract Meaning Representation 解析，単語分散表現，Proposition Bank

*東北大学 大学院情報科学研究科 システム情報科学専攻 修士論文, B5IM2017, 2017 年 2 月 10 日.

目 次

図 目 次

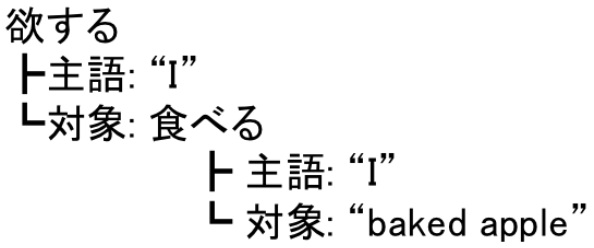
表 目 次

1　はじめに

自然言語解析とは，文字列で表現される我々が用いることば，自然言語の文から，文が持つ文法的，構文的構造や意味的な構造を得るものである．自然言語解析の例として，依存構文と呼ばれる文内の単語間の修飾関係を表す構文に変換する**依存構文解析**や，意味表現と呼ばれる述語の意味的なクラスや述語が取り得る項の種類などを特定した構造に変換する**意味表現解析**などが存在する．

自然言語解析は，文字列である自然言語を計算機に扱いやすいような構造として表現できるようになるため，多くの自然言語処理のシステムやアプリケーションの基盤となっている．

例えば，ロボットに向かって “*I want to eat a baked apple.*” という文が発話された場面を考える．この文には，以下のような意味的な構造があると考えられる．



計算機がこのような構造を得ることができると，例えば以下のような自然言語処理を用いたアプリケーションなどに応用できる．

- 質問応答システム: 人間の質問 “*What does the speaker want to eat?*” に対して,「欲する」の対象である「食べる」の対象が「*a baked apple*」なので，計算機が “*A baked apple*“と答えることができる．
- 対話システム: この発話に対し，計算機が “*A baked apple is delicious, isn't you?*” などとコミュニケーションすることができる．
- 推薦システム: 話者が「欲する」ものは食べるものなので，計算機がインターネットからレシピを検索して提示することができる．

このように，様々な自然言語処理タスクの基盤となる自然言語解析は，パイプライン的に利用する事が多く後の処理に大きく影響するため，その精度が重要である．

一般に自然言語解析器は，文とその文が表す構造が対になったデータを用いて，教師あり機械学習手法によって構築される．一般に機械学習の性能向上のためには，素性の設計や教師データの拡充などが行われるが，いずれも訓練データとは異なるあらゆる文に対して解析を行えるようにすること，すなわち汎化性能を上げることを目的としている．

自然言語解析の素性は，単語文字列の異なりに注目した，単語が何であるかを表す素性がよく用いられる．本論文では，これを単語表層素性と呼んでいる．この素性は非常に効果的であるが，訓練時に未知の単語に対して重みベクトルが学習されないことや，既知の単語に対して過学習してしまう可能性があることが問題点として挙げられる．また，意味表現解析においては，意味表現の構造の複雑さから多量の訓練データを用意することは人手のコストが高いことが問題として考えられる．

本論文では，外部のテキストからなる単語分散表現や，意味表現が土台としているフレームワークの情報を解析器に与えて，より汎用的で高性能な自然言語解析を行えるようにすることを目的とする．単語分散表現とは単語を低次元の空間上のベクトルとして表現する手法で，構文的，意味的に類似する単語同士は近いベクトルを持つ性質がある．また，意味表現が土台としているフレームワークは，例えば「食べる」の対象は食べ物を表す単語であるといったような述語の種類に応じて項に入る単語の種類を提示している．これにより例えば，単語分散表現から “*apple*” と “*banana*” は似た単語であり “*eat*” と “*apple*” の間に修飾関係があれば，“*eat*” と “*banana*” の間にも修飾関係があるだろうと予測したり，意味表現が土台としているフレームワークが持つ “*eat*” の対象は食べ物であるという情報から，対象がどの単語であるか予測することができる．

本論文では上記を目的として，単語分散表現による分散表現素性を提案する (4節)．また，訓練データに加えて，意味表現が土台としているフレームワークを反映したアノテーション付テキストを新たな訓練データとして加えることを提案す

る (5 節).

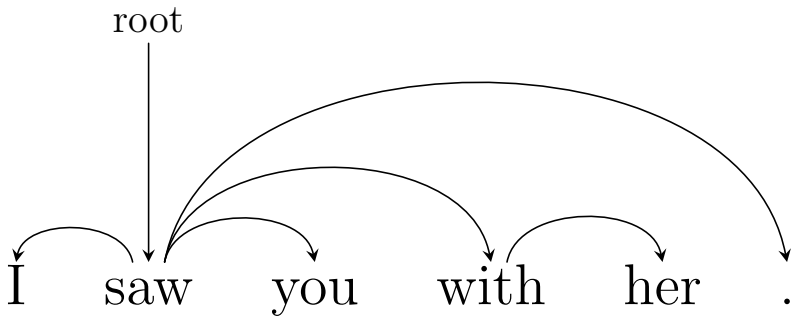


図 1: *“I saw you with her.”* に対する依存構文.

2 背景

2.1 依存構文

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，根付き有向木 $G = (V, A)$ で表される．ここで，$V = \{w_0 \ldots w_n\}$ 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．本論文の依存構文解析においては，簡単のためラベルを 1 種類 $L = \{\text{null}\}$ 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，$V = \{$ *“I”, “saw”, “you”, “with”, “her”, “.”* $\}$，$A = \{($ *“saw”*, null, *“I”*$)$, $($ *“saw”*, null, *“you”*$)$, $($ *“saw”*, null, *“with”*$)$, $($ *“with”*, null, *“her”*$)$, $($ *“saw”*, null, *“.”*$)\}$ である．ラベルが *null* のとき，1 つの依存関係を (w_i, w_j) と省略して書くことがある．1 つの依存関係 (w_i, w_j) に関して，単語 w_j に対して単語 w_i を親 (parent)，係り元 (head, governor) などと，単語 w_i に対して単語 w_j を子 (child)，係り先 (dependent) などと呼ぶ．図 1 における 1 つの依存関係 (saw, 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 [1]．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 [2] や CKY アルゴリズムを用いるもの [3] 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 [4] が代表的である．この研究では遷移ベー

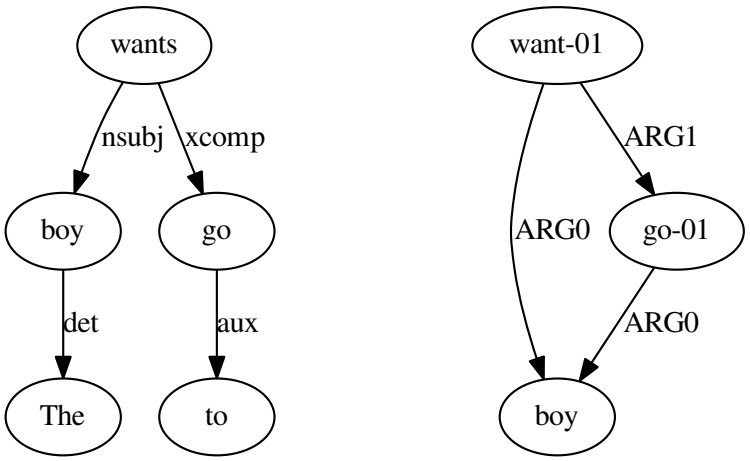


図 2: “The boy wants to go.” に対する依存構文と AMR.

スの手法の 1 つである shift-reduce 型の解析に着目する.

2.2 Abstract Meaning Representation (AMR)

Abstract Meaning Representation (AMR)[5] は，様々な英語の文の意味構造を表現する意味表現言語である．AMR は,「誰が何に対して何をしたか」を捉えることができる．それぞれの文は根付き有向非巡回グラフで表現でき，ノード,エッジに対して，それぞれ**概念 (concept)**，**関係 (relation)** のラベルが付けられる．述語に付けられる概念，およびそれに関連する関係は，**Proposition Bank (PropBank)**[6, 7, 8] とよばれるオントロジに基いている.

“ *The boy wants to go.*” という文に対する依存構文と AMR を図 2 に示す．この AMR は,

- 「欲する」イベント want-01 が存在し
- その「欲する人 (ARG0)」は boy であり,

- 「欲する物 (ARG1)」は「行く」イベント go-01 である．
- さらに go-01 の「行く人 (ARG0)」は，want-01 の ARG0 と同じ boy である

という意味を表現している．AMR を定式化すると，関係ラベル集合 $L = \{l_1, \ldots, l_{|L|}\}$ と概念ラベル集合 $C = \{c_1, \ldots, c_{|C|}\}$ が与えられたとき，AMRG は $G = (V, E)$ で定義される．

- 概念ノード集合 V: 表現する文に存在する概念の集合．各要素 v_i は 1 つの概念ラベル c_x を持つ．
- 関係エッジ集合 E: 表現する文に存在する概念同士を結ぶ関係の集合．各関係 e_j は 1 つの関係ラベル c_y を持つ．

AMR 解析には，グラフベースの手法と遷移ベースの手法が存在する．代表的なグラフベースの手法 [9] は，単語から概念の特定を行い，最大全域木アルゴリズムを応用して概念間の関係の特定を行う．一方遷移ベースの手法 [10, 11, 12] は，依存構文構造に基づいて単語を走査し，依存構文構造を変形させていくことで AMR を得る．

本研究では遷移ベースの手法に着目し，ベースラインとしている．

2.3 Proposition Bank (PropBank)

Proposition Bank (PropBank) は，テキストに対して基礎的な意味関係である述語項関係を付与すること，また意味関係が付与されたテキストを構築することを目的としたプロジェクトである [6, 7, 8].

PropBank は，述語クラスとその項の種類の定義と，その定義に基づくアノテーション付テキストを提供している．ここでは例に沿ってそれぞれの役割を説明する．

述語クラスとその項の定義の例を図 3 に示す．図 3 には，“want, desire” の意味を持つ述語を表す “want-01” と，“eat, consume” の意味を持つ述語を表す “eat-01” を示していて，“eat-01” の “ARG0” の項には “consumer, eater” を表す単語が，“ARG1” の項には “meal” を表す単語が入ることを示している．

Class want-01	want, desire
ARG0	Wanter
ARG1	thing wanted
ARG2	beneficiary
ARG3	in-exchange-for
ARG4	from

Class eat-01	eat, consume
ARG0	consumer, eater
ARG1	meal

図 3: PropBank の述語クラスと項の定義.

eat-01	His eating carrots constantly has tinted his skin a suspiciously bright orange hue.
ARG0	His
REL	eating
ARG1	carrots
ARGM-tmp	constantly

図 4: PropBank のアノテーション付テキスト.

またアノテーション付テキストの例を図 4 に示す．図 4 は，“*His eating carrots constantly has tinted ...*” という文が “eat-01” のイベントを表現していて，イベントを表す述語 “REL” が “*eating*” であり，項 “ARG0” が “*His*”，項 “ARG1” が “*carrots*”，時間を表す項 “ARGM-tmp” が “*constantly*” であることを示している．アノテーション付テキストは定義された項の他に，時間を表す項 “ARGM-tmp” や場所を表す項 “ARGM-loc” のように，任意の述語が取り得る項も同時に表現している．

2.4 単語分散表現

単語分散表現とは，単語を低次元の空間上のベクトルで表現したものである．単語分散表現は，分布仮説 [13] と呼ばれる「同じ使われ方をする，同じ周辺文脈の分布を持つ単語は類似の意味を持つ」という仮説に基づいている．伝統的には，単語に対しその周辺に現れる単語の頻度を計測し，各単語を行，文脈に出現

する単語を列として，共起頻度や共起確率，自己相互情報量 (Pointwise Mutual Information; PMI) などを行列の要素とする単語文脈行列を考え，その行ベクトルを単語分散表現とみなしていた．

しかし，単語文脈行列はその次元が語彙数に基づくため非常に高次元で疎である．そこで，単語文脈行列を潜在意味解析することで低次元な分散表現を得る手法が存在する．この手法は，まず単語文脈行列 $M \in \mathcal{R}^{m \times n}$ に特異値分解を行い，M を 3 つの行列で表現する．

$$M = U \Sigma V^T \tag{1}$$

ここで $U \in \mathcal{R}^{m \times m}$ はユニタリ行列，$\Sigma \in \mathcal{R}^{m \times n}$ は M の r 個の特異値を対角成分とする対角行列，$U \in \mathcal{R}^{n \times n}$ はユニタリ行列である．そして，特異値を $d(< r)$ 個で打ち切った対角行列 $\Sigma_d \in \mathcal{R}^{m \times n}$ を用いて，$M\sqrt{\Sigma_d}$ の行ベクトルを d 次元の単語分散表現とする手法である [14].

Mikolov らは，ニューラルネットワークを利用した単語ベクトルの効率的な学習法を提案した [15]．単語に対して，単語ベクトルと文脈ベクトルを用意し，単語ベクトルが周辺文脈の文脈ベクトルをその内積の大きさで予測できるように学習を行った．

これらの単語分散表現は，周辺文脈の分布の近い単語，すなわち意味的に類似する単語同士は，空間上で近いベクトルで表現されている．また，ベクトルの加減法によって「man に対して king ならば，woman に対しては何か」という問に対して，$v_{\mathrm{king}} - v_{\mathrm{man}} + v_{\mathrm{woman}} \approx v_{\mathrm{queen}}$ となるような，意味の加減算や単語の類推が可能である．

本研究では，単語分散表現が類似単語同士が近いベクトルを持つ性質を利用し，解析の素性として組み込むことを提案している．

3　関連研究

3.1　依存構文解析に関する研究

依存構文解析において汎化性能を向上させるために，類似単語の情報を素性として利用する手法はいくつか挙げられる．

Koo ら [16] は Brown アルゴリズム [17] で求められるクラスタ情報を新しく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．先頭からのビット列が一致する単語同士ほどより小さな同じクラスタに属していることになるため，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．

Bansal ら [18] は，単語分散表現を用いて単語クラスタ情報を得て，素性として追加している．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．本研究では，単語の類似情報をビット列によるクラスタに落とし込まずに分散表現のまま導入し性能が向上した．

Andreas ら [19] 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) 類似単語の統計を集めて使えること，(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仮説し，それぞれ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によって確認している．Andreas らの手法では，コーパス全体で学習したときの解析精度に向上が見られなかったが，本研究では精度の向上が見られた．

近年では，ニューラルネットワークを用いた依存構文解析に関する研究も多く存在し [20, 21, 22, 23, 24, 25]，これらは顕著な性能を示しているが，ニューラル

ネットワークを用いた手法は，パラメータ調整の複雑さや，解析器の予測した根拠がブラックボックスになってしまうことが問題として挙げられる．本研究では，単語分散表現によって素性を表現したことが性能向上につながっているため，これらの手法も素性を低次元で密なベクトルで表現することが性能向上につながっているのではないかと推測できる．

3.2 AMR 解析に関する研究

Wang ら [11, 12] は，AMR のフレームワークである PropBank アノテーションから学習された意味役割付与システム [26] の出力を，遷移型 AMR 解析の素性として追加している．Brandt ら [27] も，ニューラルネットワークを利用した意味役割付与を開発し，同様に遷移型 AMR 解析の素性として追加している．意味役割付与とは，PropBank と同様のラベル付けを一般の文に対して自動的に行うものである．そのため，PropBank アノテーションの情報を間接的に利用していると言え，意味役割付与システムのエラーが伝播する可能性がある．

本研究では，PropBank アノテーションを直接訓練データとして変換していることになり，AMR のフレームワークを直接解析器に組み込んでいると考えられる．

Puzikov ら [28] は，Wang ら [10, 11] の遷移型 AMR 解析の分類モデルに順伝播型ニューラルネットワークを利用している．この解析器は，単語や品詞タグのような基本的な素性を低次元で密な分散表現にマップしニューラルネットワークの入力とし，さらに分散表現も同時に学習している．組合せ素性はニューラルネットワークのパラメータ，すなわち基本的な分散表現素性の一次写像として学習される．

本研究では，既存手法の素性セット [10, 11] に基づいて，外部テキストから構築された単語分散表現を固定して素性として加えている．組合せ素性はベクトル同士の演算として表現される．Puzikov ら [28] はベースラインに対して分散表現を素性として加えた場合に性能の向上が見られなかったのに対して，本研究では性能向上が見られた．

Barzdins ら [29] は，文から AMR の文字列表現を直接予測するような，アテンション付きリカレントニューラルネットワークを構築している．この解析器は，

単体での性能は Wang らの手法 [11] に劣るが，Barzdins らはこの解析器をアンサンブル学習に利用している．ニューラルネットワークは低次元で密なベクトルで文を表現し，文の意味的な汎化を狙っていると考えられる．

4　依存構造解析に対する単語分散表現の適用

依存構文は文内の単語間の修飾関係を表す構文であり，多くの自然言語処理のタスクにおいて基盤の処理になっている．依存構文解析のエラーは後続のタスクに伝播するため，その精度は非常に重要である．一般に依存構文解析は，教師あり分類問題のタスクとして定式化される．NLP の機械学習において頻繁に用いられる,「単語がなにであるか」を表す**単語表層素性**は非常に強力であるが，単語表層素性をベクトルで表示したとき，非常に高次元で疎なベクトルである．この節では，単語表層素性の代わりに**分散表現素性**を用いる手法を提案する．単語表層素性は訓練データ外のテキストから構築された単語分散表現を利用するので，単語の意味，構文的クラスタ情報を解析に利用できると期待される．

実験では，Shift-Reduce 型依存構文解析 [30, 31] において単語表層素性を分散表現素性で置換した際に解析精度が向上することを示した．また，この向上の要因として，Andreas ら [19] が仮説したように，単語分散表現が (1) 訓練時に未知の単語を既知の単語と結びつける効果，(2) 類似する単語同士で解析器の動作を類似させる効果があることを分析した．

4.1　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．

はじめに，解析器の内部状態 S を，$S = (\mathbf{q}, \mathbf{s}, \mathbf{A})$ で定義する．

- キュー $\mathbf{q}$: 未処理の単語を格納する．
- スタック $\mathbf{s}$: 処理中の単語を格納する．
- 集合 A: 構築された依存関係集合

ある状態に対して，動作を選択することで状態を遷移させる．典型的な arc-standard システム [32, 4, 30] では動作は三種類である．

ステップ	動作	内部状態 S スタック $\mathbf{s}$	キュー $\mathbf{q}$	依存関係 A
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 5: shift-reduce 型依存構文解析の内部動作.

1. Shift: $\mathbf{q}$ の第一要素を取り除き， $\mathbf{s}$ の先頭に加える.
2. Reduce-Left: $\mathbf{s}$ の第一要素 s_0 と第二要素 s_1 の間に依存関係 (s_0, s_1) を認め，s_1 を $\mathbf{s}$ から除去する.
3. Reduce-Right: $\mathbf{s}$ の第一要素 s_0 と第二要素 s_1 の間に依存関係 (s_1, s_0) を認め，s_0 を $\mathbf{s}$ から除去する

文 $x = w_0 \dots w_n$ の依存構文を得る場合，まず状態を $([w_0 \dots w_n], [], \{\})$ で初期化し，最終状態 $([], [s], A)$ になるまで，動作を適応し状態を遷移させることを繰り返す.

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 5 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-Left, Reduce-Right} に対するスコアの増分 $\{\xi, \lambda, \rho\}$ は，式 (2-4) 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.

$$\xi = \mathbf{w} \cdot f_{\texttt{Shift}}(S) \tag{2}$$

$$\lambda = \mathbf{w} \cdot f_{\texttt{Reduce-Left}}(S) \tag{3}$$

$$\rho = \mathbf{w} \cdot f_{\texttt{Reduce-Right}}(S) \tag{4}$$

素性セットは Huang ら [30] に従い，キュー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，スタック内の要素を先頭から $s_0, s_1, \ldots$，キュー内の要素を先頭から $q_0, q_1, \ldots$ と表すとき，単語の表層形 $s_0\texttt{w}$ や $q_0\texttt{w}$，単語の品詞タグ $s_0\texttt{t}$，単語の表層と品詞タグの組合せ $s_0\texttt{wt}$ などを素性として用いる．他の素性としては，s_{0l} を s_0 の最左の子単語，s_{0r} を s_0 の最右の子単語としたとき，それらの品詞タグ $s_{0l}\texttt{t}$ や, 上記素性の組み合わせ $s_0\texttt{w}q_0\texttt{w}$ などが用いられる．

ある状態においてスタックやキュー内の具体的な単語や品詞タグが与えられたときは，$\texttt{w}$ と $\texttt{t}$ の添字を用いて具体的な素性を表現する．例えば図 5 の 3 ステップ目の状態からは，$q_0\texttt{w}_{you}$, $s_0\texttt{t}_{\text{VBD}}$, $s_0\texttt{w}_{saw}\texttt{t}_{\text{VBD}}$, $s_{0l}\texttt{t}_{\text{PRP}}$, $s_0\texttt{w}_{saw}q_0\texttt{w}_{you}$ のような素性が得られる．このとき，例えば $q_0\texttt{w}_{you}$ は，キューの先頭要素の単語の表層形が “*you*” であること，$s_0\texttt{w}_{saw}\texttt{t}_{\text{VBD}}$ は，スタックの先頭要素の単語の表層形が “*you*” で，その品詞タグが “VBD” であることを示している．

これらの素性をベクトルで表示したとき，ある要素だけが 1 でその他の要素が 0 の one-hot ベクトルになっている．また，これらの素性の結合である $f_{\texttt{Shift}}(S, i)$, $f_{\texttt{Reduce-Left}}(S, i)$，$f_{\texttt{Reduce-Right}}(S, i)$ は，0,1 だけの 2 値ベクトルであり，そ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．

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，arc-standard システムにおける動作列 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 (5) である．

$$\begin{gathered}\texttt{Shift}, \texttt{Shift}, \texttt{Reduce-Left}, \texttt{Shift}, \texttt{Reduce-Right}, \texttt{Shift}, \\ \texttt{Shift}, \texttt{Reduce-Right}, \texttt{Reduce-Right}, \texttt{Shift}, \texttt{Reduce-Right}\end{gathered} \tag{5}$$

	$s_0, s_1, \cdots =$ スタック 素性セット $\mathbf{f}(S,i)$ q_n :キュー
1 単語からなる単語表層素性	$s_0\mathtt{w}$ $s_1\mathtt{w}$ $q_0\mathtt{w}$
＋品詞タグとの組合せ	$s_0\mathtt{w}s_0\mathtt{t}$ $s_1\mathtt{w}s_1\mathtt{t}$
	$q_0\mathtt{w}q_0\mathtt{t}$
	$s_0\mathtt{w}s_0\mathtt{t}s_1\mathtt{t}$ $s_0\mathtt{t}s_1\mathtt{w}s_1\mathtt{t}$
	$s_0\mathtt{w}q_0\mathtt{t}q_1\mathtt{t}$ $s_1\mathtt{t}s_0\mathtt{w}q_0\mathtt{t}$
	$s_1\mathtt{t}s_{1rc}\mathtt{t}s_0\mathtt{w}$ $s_1\mathtt{t}s_0\mathtt{w}s_{0lc}\mathtt{t}$
2 単語からなる単語表層素性	$s_0\mathtt{w}s_1\mathtt{w}$
＋品詞タグとの組合せ	$s_0\mathtt{w}s_1\mathtt{w}s_1\mathtt{t}$ $s_0\mathtt{w}s_0\mathtt{t}s_1\mathtt{w}$
	$s_0\mathtt{w}s_0\mathtt{t}s_1\mathtt{w}s_1\mathtt{t}$
品詞素性	$s_0\mathtt{t}$ $s_1\mathtt{t}$ $q_0\mathtt{t}$
	$s_0\mathtt{t}s_1\mathtt{t}$ $s_0\mathtt{t}q_0\mathtt{t}$
	$s_0\mathtt{t}q_0\mathtt{t}q_1\mathtt{t}$ $s_1\mathtt{t}s_0\mathtt{t}q_0\mathtt{t}$
	$s_2\mathtt{t}s_1\mathtt{t}s_0\mathtt{t}$
	$s_1\mathtt{t}s_{1lc}\mathtt{t}s_0\mathtt{t}$ $s_1\mathtt{t}s_{1rc}\mathtt{t}s_0\mathtt{t}$
	$s_1\mathtt{t}s_0\mathtt{t}s_{0rc}\mathtt{t}$ $s_1\mathtt{t}s_{1lc}\mathtt{t}s_0\mathtt{t}$

表 1: 依存構文解析における素性セット.

4.2 単語分散表現による単語表層素性の次元削減

単語表層素性を 1 つ以上の単語表層を含む素性と定義する．例えば，$q_0\mathtt{w}_{you}$ $s_0\mathtt{w}_{saw}\mathtt{t}_{\mathrm{VBD}}$，$s_0\mathtt{w}_{saw}q_0\mathtt{w}_{you}$ は単語表層素性である．ここでは単語表層素性を単語分散表現によって置換する手法を説明する．

本手法の概念図を図 7 に示す．単語表層素性は 4.1 節でも述べたように，one-hot なベクトルで表現され，それにより単語表層素性に対する重みベクトルは単語ごとに異なる値が発火するようになる．また訓練データにない未知語に対しては，未知語に対する重みベクトルの値が学習されないため，解析の際スコアが計算されない．本手法では，単語表層素性の one-hot ベクトルを単語分散表現で置換することを提案する．単語分散表現を素性に用いることにより類似する単語同士で

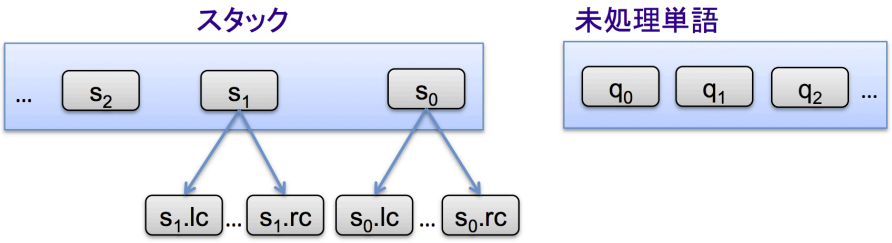


図 6: 内部状態と素性の範囲.

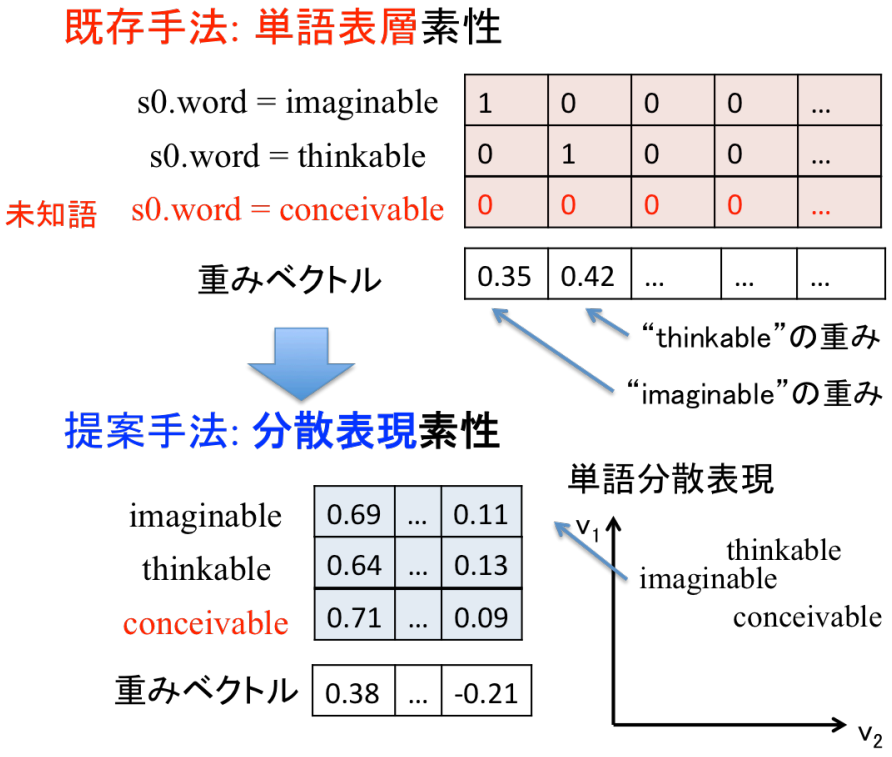


図 7: 分散表現素性.

解析器のスコアが類似し，また未知語に対してもスコアが計算されるようになる．

単語表層素性の組合せパターンに対して，分散表現素性の構築方法を詳しく説明する．

1 単語からなる単語表層素性　$\mathtt{w}$ を単語，s を素性セットの任意のシンボルとするとき $s\mathtt{w}$ を 1 単語からなる単語表層素性とする．$s\mathtt{w}$ は one-hot ベクトルである．

$\mathtt{e} = (v_i)_{1 \leq i \leq d}$ を単語 $\mathtt{w}$ の d 次元単語分散表現としたとき，単語表層素性を $s\mathtt{w}$ を式 6 で表される，基底 $se_1, \ldots, se_d$ の線形結合 $s\mathtt{e}$ によって置換する．

$$s\mathtt{e} := \sum_{i=1}^{d} v_i \cdot (se_i) \tag{6}$$

例えば単語 “saw” の単語分散表現が $\mathtt{e}_{saw} = (0.6, \ldots, 0.2)$ のとき，単語表層素性 $s_0\mathtt{w}_{saw}$ は式 7 で置換される．

$$s_0\mathtt{e}_{saw} := 0.6 s_0 e_1 + \ldots + 0.2 s_0 e_d. \tag{7}$$

ここで，$s_0e_1, \ldots, s_0e_d$ は，素性テンプレート $s_0\mathtt{w}$ に対して置換される．すなわち任意の単語 x に対して，単語表層素性 $s_0\mathtt{w}_x$ は，$s_0e_1, \ldots, s_0e_d$ の d 次元の素性として表現される．よって，単語表層素性 $s_0\mathtt{w}_x$ は単語の語彙数分だけ存在したのに対し，置換後の単語分散表現による素性は，d 種類となり，素性の次元を削減できる．

1 単語の単語表層素性と品詞タグの組合せ素性の場合は，品詞タグの素性は one-hot 素性のままとして扱う．例えば，$s_0\mathtt{t}_{\mathrm{VBD}}\mathtt{w}_{saw}$ の素性は，式 8 で表される $s_0\mathtt{t}_{\mathrm{VBD}}\mathtt{e}_{saw}$ によって置換される．

$$s_0\mathtt{t}_{\mathrm{VBD}}\mathtt{e}_{saw} := 0.6 s_0\mathtt{t}_{\mathrm{VBD}} e_1 + \ldots + 0.2 s_0\mathtt{t}_{\mathrm{VBD}} e_d. \tag{8}$$

この場合も，$s_0\mathtt{tw}$ の組合せ素性は，単語の語彙数と品詞タグの種類数の積だけの種類数があったのに対して，置換後の単語分散表現による素性 $s_0\mathtt{t}_{\mathrm{VBD}}\mathtt{e}_{saw}$ は，次元数と品詞タグの種類数の積だけの大きさになる．

2 単語以上からなる単語表層素性　$s_0\mathtt{w}q_0\mathtt{w}$ のような，2 単語以上からなる単語表層素性は，対応する単語の分散表現を組み合わせることで素性を構築する．ここでは 2 単語の組み合わせを考えるが，3 単語以上の組合せも同様に考えることができる．$\mathsf{e}_1 = (u_i)_{1\leq i\leq d}$ と $\mathsf{e}_2 = (v_i)_{1\leq i\leq d}$ をそれぞれ単語 $\mathtt{w}_1$ と $\mathtt{w}_2$ の単語分散表現とするとき，次の演算による素性で，単語表層素性 $s\mathtt{w}_1\mathtt{w}_2$ を置換する．

- Outer product ($\otimes$):

$$s(\mathsf{e}_1 \otimes \mathsf{e}_2) := \sum_{i=1}^{d}\sum_{j=1}^{d} u_i v_j \cdot (se_i\tilde{e}_j),$$

例えば，$\mathsf{e}_{saw} = (0.6, \ldots, 0.2)$，$\mathsf{e}_{you} = (0.5, \ldots, 0.8)$ のとき，$s_0\mathtt{w}_{saw}q_0\mathtt{w}_{you}$ は次で置換される

$$\begin{aligned} s_0q_0(\mathsf{e}_{saw} \otimes \mathsf{e}_{you}) := (0.6\times 0.5)s_0q_0e_1\tilde{e}_1 + \ldots \\ + (0.6\times 0.8)s_0q_0e_1\tilde{e}_d + \ldots \\ + (0.2\times 0.8)s_0q_0e_d\tilde{e}_d. \end{aligned}$$

- Sum ($+$):

$$s(\mathsf{e}_1 + \mathsf{e}_2) := \sum_{i=1}^{d}(u_i + v_i)\cdot(se_i).$$

同様の例において，$s_0\mathtt{w}_{saw}q_0\mathtt{w}_{you}$ は次で置換される

$$s_0q_0(\mathsf{e}_{saw} + \mathsf{e}_{you}) := (0.6+0.5)s_0q_0e_1 + \ldots + (0.2+0.8)s_0q_0e_d.$$

- Concatenation ($\oplus$):

$$s(\mathsf{e}_1 \oplus \mathsf{e}_2) := \sum_{i=1}^{d} u_i\cdot(se_i) + \sum_{j=1}^{d} v_j\cdot(s\tilde{e}_j).$$

同様の例において，$s_0\mathtt{w}_{saw}q_0\mathtt{w}_{you}$ は次で置換される

$$s_0q_0(\mathsf{e}_{saw} \oplus \mathsf{e}_{you}) := 0.6s_0q_0e_1 + \ldots + 0.2s_0q_0e_d + 0.5s_0q_0\tilde{e}_1 + \ldots + 0.8s_0q_0\tilde{e}_d.$$

- Elementwise Product ($\circ$):

$$s(\mathsf{e}_1 \circ \mathsf{e}_2) := \sum_{i=1}^{d} (u_i v_i) \cdot (se_i).$$

同様の例において，$s_0\mathtt{w}_{saw}q_0\mathtt{w}_{you}$ は次で置換される

$$s_0q_0(\mathsf{e}_{saw} \circ \mathsf{e}_{you}) := (0.6 \times 0.5)s_0q_0e_1 + \ldots + (0.2 \times 0.8)s_0q_0e_d.$$

単語表層素性 $s_0\mathtt{w}_x$，$q_0\mathtt{w}_y$ が高次元な one-hot ベクトルである考えると，2 単語の表層素性 $s_0\mathtt{w}_xq_0\mathtt{w}_y$ は，$s_0\mathtt{w}_x$ と $q_0\mathtt{w}_y$ の外積で表現されるため，Outer product が自然な演算である．実際，4.3 節内の実験では Outer Product が 4 種類の演算の中で最も良い性能であった．しかし，外積は d 次元の分散表現に対して d^2 次元の素性になる一方で，Sum と Elementwise Product は d 次元，Concatenation は $2d$ 次元の素性となり，より低次元で組合せを表現できる．また，Elementwise Product は Outer Product の要素の一部であり，外積の対角成分にあたる要素の素性としての効果を確認できる．

4.3 実験

4.3.1 実験設定

Huang ら [31] の解析器を再実装し，全ての単語表層素性を 4.2 節で述べた方法によって分散表現素性に置換した．学習，解析における探索はビームサーチを行い，ビーム幅は 8 とした．また任意の状態に対して動作を選択する分類器の学習には max-violation パーセプトロン [31] を学習アルゴリズムとして用いた．訓練，開発，テストデータは Huang らのシステム [1] が利用している Penn Treebank の標準分割を用い，Stanford Tagger[2] を利用して品詞タグ付けを行った．評価は，記号を除く親単語の正答率 (Unlabeled Attachment Scores; UAS) を指標とした．また，単語分散表現の効果を明らかにするために，訓練データに存在しない単語が

[1] http://acl.cs.qc.edu/~lhuang/
[2] http://nlp.stanford.edu/software/corenlp.shtml

含まれる文を開発データから 148 文抽出し，これらの文に対する性能も比較した．これを未知データセットする．

単語分散表現は Levy らの手法 [33, 14] に基づき，New York Times コーパス [3] (2007 年 1 月-2007 年 6 月, 約 150 万文) を用い，出現頻度が 50 回以上の単語について 300 次元の単語分散表現を構築した．構築の手法は，2.4 節で紹介したように，対象の単語に対して文脈に現れる単語の頻度から，非負 PMI 行列を作成し，それを特異値分解することで単語分散表現を得る．今回は文脈に取る単語に応じて 3 種類の単語分散表現を比較した．

- Plain 対象単語に対して，文の両側 3 単語を文脈単語とした．
- Tree 対象単語に対して，Huang らの解析器で得られた依存構文上で，親子 3 つ以内の単語を文脈単語とした．
- State Huang らの解析器の全てのステップにおける内部状態で，s_0 の位置にある単語を対象単語として，$\{s_1, s_2, s_3, s_{0l}, s_{0r}, s_{1l}, s_{1r}, q_0, q_1, q_2\}$ の位置にある単語を文脈単語とした．

Tree や State の単語分散表現は，依存構文解析のためのより構文的な情報が分散表現に含まれることを期待している．

4.3.2 結果

各解析器の結果を表 2 に示す．まず，Outer の演算で組合せ素性を表現し，State 文脈によって構築された単語分散表現を用いたときが，最高性能を示していることがわかった．また，その他の演算手法，その他の文脈による分散表現を用いたときも，ベースラインの性能より向上していることがわかった．特に未知データセットに対しては，大きな向上が得られた．

次に，単語ビット列表現 [16, 18] で単語表層素性を置換する手法と比較を行った．単語ビット列表現は，前述の単語分散表現から階層的クラスタリングによって構築したものと，Bansal ら [18] によって構築されたものを用いた [4]．ビット列

[3] https://catalog.ldc.upenn.edu/LDC2008T19

[4] http://ttic.uchicago.edu/~mbansal/

は先頭から 4, 6, 8, 12, 16, 20 文字目までを用いた．これらの結果は有意な性能向上が見られなかった．

さらに，参考として，Chen らのニューラルネットワークに基づく解析器 [22] の結果を示す．Chen らもこの研究と同様に，高次元で疎な素性の代わりに，単語分散表現を用いて低次元で密な素性を用いることで汎化効果を期待している．今回は，単語分散の初期化にランダムな 300 次元の分散表現を利用した場合と，前述した 3 種類の単語分散表現を用いた場合で比較を行った．結果として，単語分散表現の初期化の効果は見られたが，3 種類の単語分散表現で比較したとき，STATE 文脈によって構築された単語分散表現の効果が見られなかった．この結果から，STATE 文脈によって構築された単語分散表現は，Huang らの解析器に特有の分散表現であったと考えられる．また，今回の結果では，Chen らの解析器は Huang らの解析器より悪い性能を示したが，この結果からニューラルネットワークのデータに対するハイパーパラメータの調整のコストが高いことや，人手で調整した素性にも有用な情報が含まれていることが伺える．

	開発	テスト	未知
Huang ら (ベースライン) [31]	91.93	91.68	89.01
State 単語分散表現において，異なる演算を比較:			
Outer	**92.57***	**92.20***	**90.27***
Sum	92.25*	91.85	90.10*
Concatenation	92.18	91.86	89.96
Elementwise	92.18	91.84	89.91
Outer 演算において，異なる単語分散表現を比較:			
Plain	92.33*	91.78	90.08*
Tree	92.37*	92.09*	89.82
State	**92.57***	**92.20***	**90.27***
ビット列素性:			
Plain	91.71	91.20	89.18
Tree	90.38	90.07	88.00
State	91.31	90.96	89.04
Bansal ら [18]	92.06	91.75	90.13
ニューラルネットワーク [22]:			
Random	86.37	86.19	81.06
Plain	90.68	90.48	87.02
Tree	91.06	90.82	87.38
State	91.03	90.57	87.88

表 2: UAS による解析性能. 太字は各データセットにおける最高性能を示している．アスタリスク (*) によって示された数字は，ペアブートストラップ検定においてベースラインと比較して統計的に有意である．($p < 0.05$)

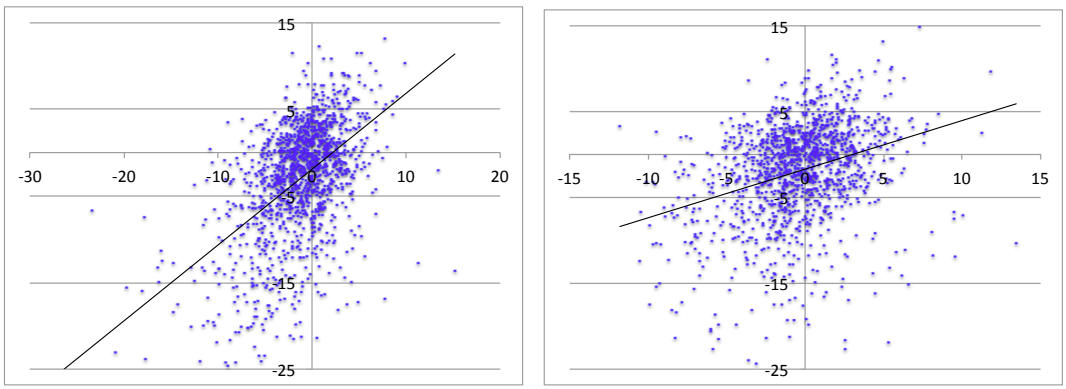


図 8: 高頻度語（左）と中頻度語（右）x における，$s_0\mathtt{w}_x$ に対する重み（X）と $s_0\mathtt{e}_x$ に対する重み（Y）の相関図.

4.3.3 分析

単語分散表現素性を用いた解析器はベースラインの単語表層素性に基いて素性の次元を削減する事ができたのだろうか．図 8 は， 高頻度語と中頻度語それぞれについて，単語を x としたとき，ベースラインで学習した単語表層素性 $s_0\mathtt{w}_x$ に対するスコアを X に，分散表現による素性 $s_0\mathtt{e}_x$ に対するスコアを Y にプロットしたものである．$s_0\mathtt{w}_x$ に対するスコアとは対応する重みベクトルの値 $W(s_0\mathtt{w}_x)$ であり，$s_0\mathtt{e}_x$ に対するスコアは $s_0\mathtt{e}_x$ と 重みベクトル $(W(s_0e_1), \ldots, W(s_0e_d))$ の内積で求められる．また図 8 には，近似直線を同時に示しており，$s_0\mathtt{w}_x$ に対するスコアと $s_0\mathtt{e}_x$ に対するスコアの間に正に相関があることを示している．この図から，単語分散表現素性による解析器とベースラインの単語表層素性による解析器は互いに相関がある，つまりふたつの解析器は互いに似た動作が起きていると考えられる．よって表 2 で示した有意な性能向上は，ベースラインで正解した解析をそのまま維持しつつ，ベースラインで誤った解析を訂正できたからであると考えられる．

それでは，なぜ単語表層素性で誤った解析を分散表現で訂正できたのだろうか．図 9 の上段は，任意の 2 単語間について，単語分散表現のコサイン類似度を X 軸に, 1 単語からなる単語表層素性に対する重みベクトルのコサイン類似度を Y 軸に示したものである．この図から，ベースラインに対する類似単語の重みベクト

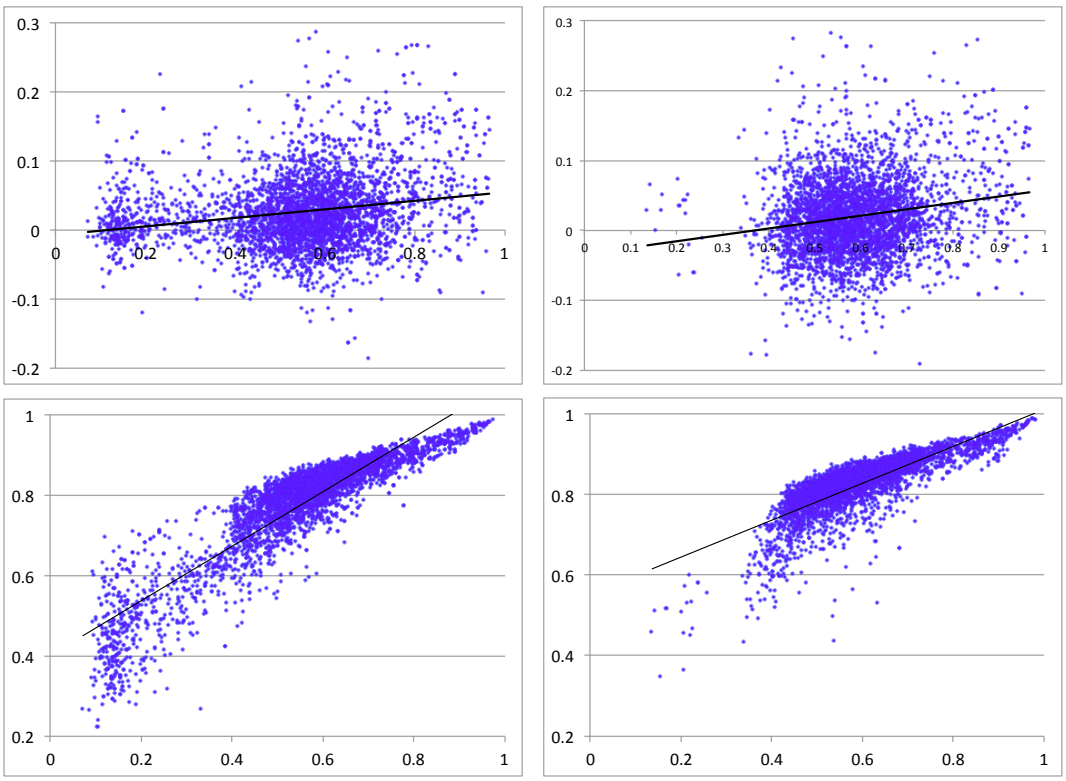


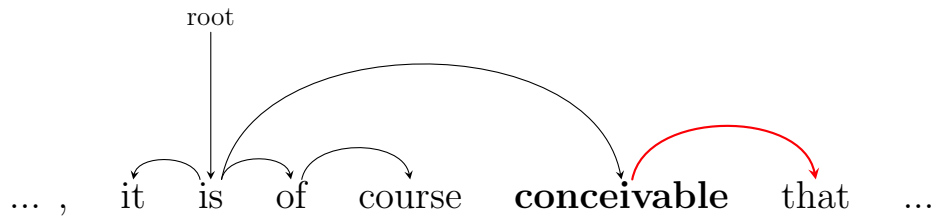
図 9: 高頻度語（左）と中頻度語（右）の任意の 2 単語 x, y において，X 軸に x, y の単語分散表現のコサイン類似度，Y 軸に x, y の素性のコサイン類似度をプロットした相関図.

ルはわずかに類似することがわかる．一方図 9 の下段は，任意の 2 単語間について，単語分散表現のコサイン類似度を X 軸に，単語分散表現素性に対する重みベクトルのコサイン類似度を Y 軸に示したものである．この図によると，単語の類似度と単語分散表現素性に対する重みベクトルは強く相関していることがわかる．すなわち，分散表現素性による解析器は，素性の次元を削減したことによって，類似単語に対してほぼ同じ解析が行われている．この性質から，Andreas ら [19] も述べているように，単語分散表現により素性の次元を削減することは，

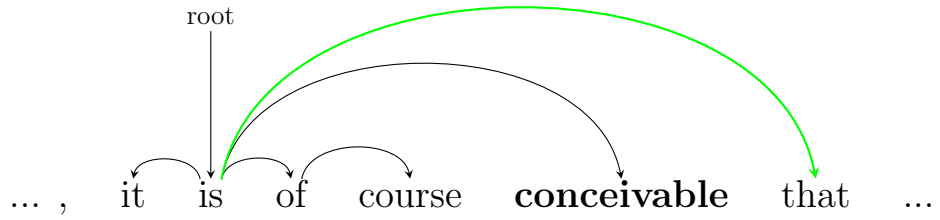
1. 未知の単語を既知の単語と結びつける効果,
2. 既知の単語の中で解析器の動作を類似させる効果

があると考えられる.

(a) Using Lexical Features (Red is wrong)



(b) Using Embedding Features (Green is correct)



"*While it is possible that the Big Green initiative will be ruled unconstitutional, it is of course conceivable that in modern California it could slide through.*"

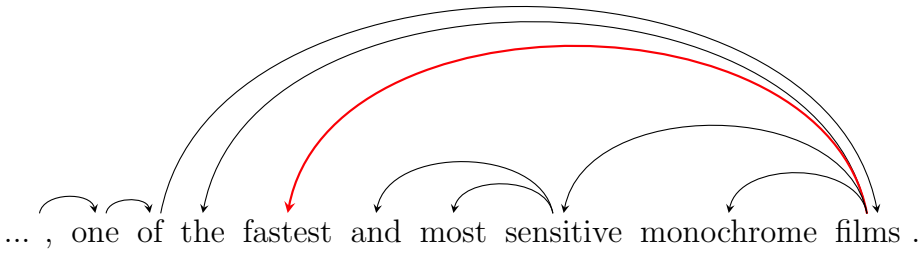
図 10: 未知単語（太字の単語）を含む文に対する性能向上.

未知の単語を含む文に対しての向上は，表 2 の未知データセットに対する性能向上から見て取れる．図 10 は，未知の単語を含む文の具体的な例を示している．この例において，"*conceivable*" は訓練データに現れない未知の単語であった．そのためベースラインの解析器では，未知単語に対して正しく解析することができなかった（図 10(a)）．しかし，"*conceivable*" の単語分散表現は，"*subjective*" や "*undeniably*" などと類似していたため，分散表現素性を用いることで，既知の単語の情報を利用して正しく解析することができた（図 10(b)）.

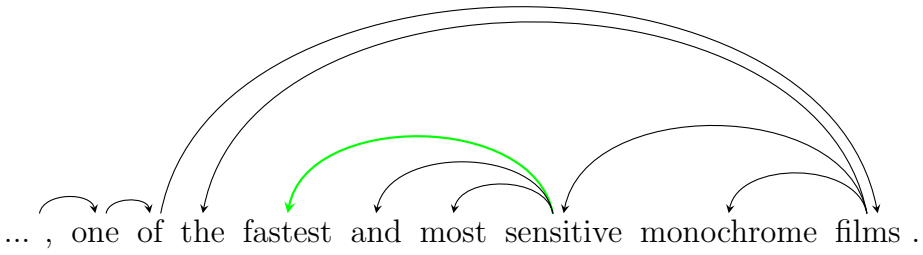
既知の単語に関する効果を明らかにするために，形容詞の並列構造を例に用いる．具体的には，解析器の内部状態において，

- s_1t と s_0t が JJ, JJS, JJR のいずれかの品詞タグであり，
- s_{0l}t が CC, Comma のいずれかの品詞タグである

(a) Using Lexical Features (Red is wrong)



(b) Using Embedding Features (Green is correct)



"*The Rochester, N.Y., photographic giant recently began marketing T-Max 3200, one of the fastest and most sensitive monochrome films.*"

図 11: 形容詞の並列構造に対する性能向上.

ときを考える．例えば，"*black and white*" という文に対して，$s_0\mathtt{w}$ ="*white*"，$s_1\mathtt{w}$ = "*black*"， $s_{0l}\mathtt{w}$ = "*and*" となる内部状態がこれに当てはまる．"*black and white*" の依存構文は，$V = \{(white, and), (white, black)\}$ となるため，この内部状態では，真の動作としては `Reduce-Left` が選ばれる．確かに訓練データにおいて，この内部状態においては 98.8%が `Reduce-Left` が真の動作であった．しかし，ベースラインの解析器で New York Times コーパスを解析したとき，`Reduce-Left` が選ばれた割合は減少し 96.7%であった．これは形容詞の並列構造の規則を解析器が捉えられていないと考えることができる．一方，単語分散表現素性を用いた解析器では，`Reduce-Left` が選ばれた割合は 99.4%に増加した．これはベースライン

に比べてパーミュテーションテストにおいて統計的に有意であった．この観測から，単語分散表現素性を用いた解析器は，類似単語に対して類似する動作を行うことで，並列構造の規則を強く獲得していることがわかる．形容詞の並列構造に対して性能向上した具体的な文例を図 11 に示した．

図 12 は，ベースラインの解析器と単語分散表現素性を用いた解析器の学習において，訓練データの大きさを少なくしたときの UAS の値を示したものである．訓練データを故意に少なくし未知の単語を増やすことで，単語分散表現素性の利点をより活かせると考えられる．図 12 が示すように，訓練データを小さくしたときでも，単語分散表現素性を用いた解析器はベースラインの解析器の性能を上回った．しかし，どちらの解析器も，訓練データの大きさを少なくするに従って性能は減少した．単語分散表現は訓練データをすべて用いたときの解析器から構築されているため，少なからず構文構造の情報が含まれていると考えられるが，この結果から，単語分散表現には構文構造的な情報が多くエンコードされていないことが予測できる．つまり，単語分散表現は解析の性能向上に役立つが，その理由としては，単語分散表現がより効果的に単語の汎化を行えるからであり，より重要な構文構造的な情報を含んでいるからではないと予測できる．

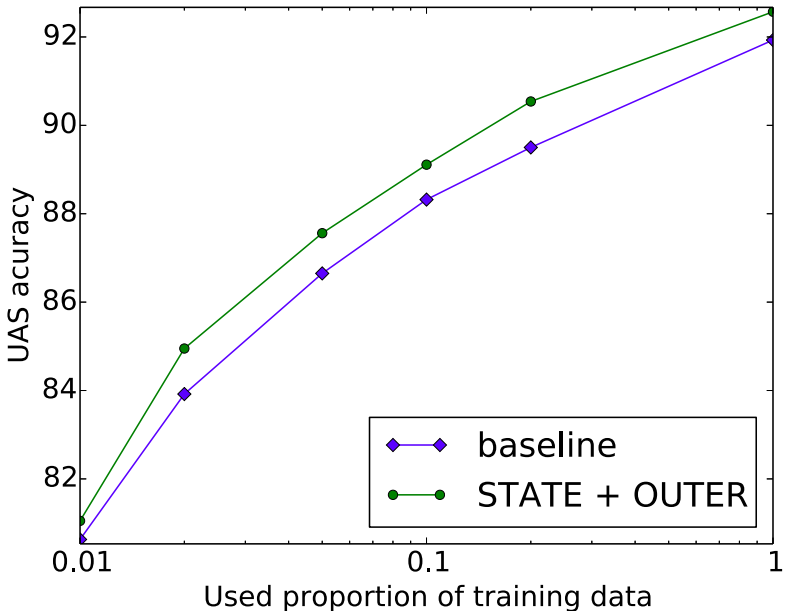


図 12: 訓練データ量を減少させたときのそれぞれの解析器の UAS 値.

5　意味解析おけるオントロジと分散表現の利用

述語の意味的なクラスや述語が取り得る項の種類などを定義するものをオントロジと呼ぶ．オントロジに基づく意味表現は質問応答や談話関係解析などのより深い言語理解を行うシステムのための入力として利用されるため，意味表現の解析は非常に重要である．ここでは，意味表現の背後にあるオントロジの情報を意味解析に組み込む事を考える．今回は意味表現に **Abstract Meaning Representation** (AMR)[5] を用いる．AMR は **Proposition Bank** (PropBank)[6, 7, 8] と呼ばれるオントロジを背後に持つ．PropBank は述語のクラスと述語に対してその項に入るべき単語のクラスを規定し，述語と項のアノテーション付きテキストを提供する．

この研究では，AMR 訓練データの他に PropBank のアノテーション付テキストを AMR の学習に組み込むことで，述語のクラスや項の学習データを増やし様々なパターンに対して学習が行えることを期待している．また，より学習を汎化させるために 4 節で提案した分散表現素性を同時に組み込む．

実験では，PropBank アノテーションを学習に加えることで性能が向上することを示した．また，分散表現素性を同時に用いることでさらなる性能向上が確認できた．

5.1　遷移型 AMR 解析

Wang らによる遷移ベースの AMR 解析 [10, 11, 12] は，文から依存構文解析によって依存構文を得たあと，依存構文木のノードを順にたどることによって概念や関係の特定を行い，AMR 特有の構造を得るような解析器である．

解析器の内部状態 S は $S = (\sigma, \beta, A)$ で定義される．

- キュー σ: 未処理のノードを格納する．σ の先頭要素を σ_0 とし，$\sigma = \sigma_0|\sigma'$ と表す．
- キュー β: 主に σ_0 の子ノードを格納する．同様に，β の先頭要素を β_0 とし，$\beta = \beta_0|\beta'$ と表す．

- AMR 構造 G

文 x に対して依存構造 D が得られるとき，初期状態から順に動作を行うことによって状態を遷移させ，終了状態までこれを繰り返すことで解析を行う．初期状態 S_{init} は $S_{init} = (\sigma_{init}, [].G_{init})$ とする．

- σ_{init} は依存構文木 D を後順深さ優先探索で走査し，ノードを格納する．例えば，図 2 においては，$\sigma_{init} = [the, boy, to, go, wants]$ である．
- G_{init} は依存構文 D の単語をノード，依存関係をアークとする．

状態遷移のための動作は 9 種類である．

- `DeleteNode` は，ノード σ_0 をグラフ G から取り除き，σ からも取り除く．この動作は，機能語などの依存構文には存在するが AMR には存在しない単語の葉ノードを取り除く．図 13 は，“*The boy wants.*” という文に対して，σ_0 =“*The*”，σ_1 =“*boy*” のとき，`DeleteNode` によって AMR には存在しない “*The*” ノードをグラフから取り除いている．
- `NextNode(c)` は，まずノード σ_0 に対して概念ラベル c を付与する．そして，σ からその先頭要素 σ_0 を取り除き，新たに σ の先頭要素となった σ_1 の子ノードで β を初期化する．概念ラベルは主にこの動作を通して付けられる．図 14 は，“*The boy wants.*” という文に対して，図 13 に続いて，`NextNode`(boy) によって σ_0 =“*boy*” のノードに，概念ラベル “boy” を付与している．
- `NextEdge(r)` は，エッジ (σ_0, β_0) に関係ラベル r を付与し，β からその先頭要素 β_0 を取り除く．依存関係にあるノード間の関係ラベルは，主にこの動作によって付けられる．図 15 は，“*The boy wants to go*” という文に対して，σ_0 =“*wants*”，β_0 =“*boy*” の間のエッジに，`NextEdge`(ARG0) によって関係ラベル r を付与している．
- `Swap(r)` は，σ_0 と β_0 の依存関係の方向を入れ替え，β_0 を新たな親とし，エッジ (β_0, σ_0) に関係ラベル r を付与する動作である．その後，β_0 を β から取り除き，σ の第 2 要素に挿入する．この動作で依存構文と AMR でヘッドが異

なる場合を修正できる．図 16 は，“*South Korea and Israel oppose*” という文に対して，Swap(op1) によって，β_0 =“*and*” を，AMR の構造に合うように σ_0 =“*Korea*” の親に移動させている．

- Reattach(k,r) は，エッジ (σ_0, β_0) を取り除き，新たに関係ラベル r のエッジ (k, β_0) を加える．この動作で依存構文と AMR で係り先が異なる場合を修正できる．また並列構文の処理を *Swap* に続いて行える．図 17 は，“*South Korea and Israel oppose*” という文に対して，Swap(op1) に続いて，Reattach(***and***,op2) によって β_0 =“*Israel*” の親を “*and*” に修正している．

- ReplaceHead は，σ_0 と関連するエッジをグラフから取り除き，β_0 と置き換える．その後，β_0 を σ の先頭要素とし，新たに β に子ノードを展開する．この動作は，葉にない AMR に不要なノードを取り除くことができる．図 18 は，“*live in Singapore*” という文に対して，AMR に不要な σ_0 =“*in*” を ReplaceHead によって除去している．

- Reentrance(k,l) は．新たに関係ラベル r のエッジ (k, β_0) を，グラフに加える．その他の状態は変化させない．この動作は主に照応関係をつけるときに用いられる．図 19 は，“*The police wants to arrest*” という文に対して，β_0 =“*police*” を照応する単語 “*arrest*” を探し，Reentrance(***arrest***, ARG0) によって，その関係を付与している．

- Merge は，σ_0 と β_0 をそれぞれ σ，β から取り除きそれらを 1 つのノード $\tilde{\sigma}$ として，σ の先頭に挿入する．この動作は固有表現や句動詞などを 1 つのノードにまとめることができる．図 20 は，“*Micheal Karras*” という人物の固有表現を Marge によって 1 つの概念としてまとめている．

- Infer(c) は，σ の先頭要素 σ_0 と第二要素 σ_1 の間にエッジがある場合，その間に新たに概念ラベル c のノード σ_a を追加する．すなわち，グラフには σ_a のノードと，(σ_1, σ_a)，(σ_a, σ_0) のエッジが追加され，(σ_1, σ_0) は取り除かれる．また，σ_a は σ の 2 番目に挿入される．このノードは単語にはないが AMR に存在するノードを作る事ができる．図 21 は，“*minister*” という単

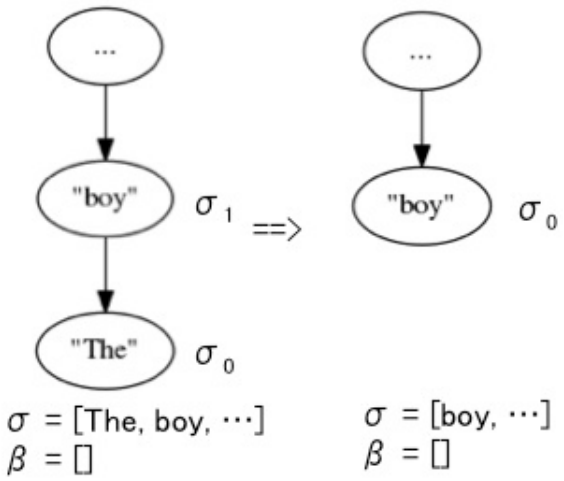


図 13: DeleteNode

語に対して “have-org-role-91” という概念を新たに作り出している．これは，“*minister*” は「minister という役職を持つ人」ということを暗に意味しているからである．

文の依存構文と真の AMR が与えられたとき，初期状態から終了状態まで，1 つの状態に対して真の動作が求められるので，状態に対して動作を正しく選択できるような分類器を学習できる．

5.2 オントロジ情報の解析器学習への利用

この節では，5.1 節で述べた遷移型 AMR 解析において，AMR のオントロジ，すなわち PropBank の情報を AMR の解析に導入する方法を説明する．

PropBank の述語と項の定義が AMR のベースになっていることと，その定義に基づいたアノテーション付テキストが存在することを考えると，PropBank のアノテーション付テキストは部分的な AMR と考えることができる．AMR の訓練データに加えて，訓練データとは他のテキストにつけられた部分的な AMR を学習に利用する事によって，エッジが張られるべきノードを特定したり，エッジに対する関係ラベル，ノードに対する概念ラベル付けに対し学習量の増加させ，ロングテイルなエッジやノードの学習が可能になると考えられる．

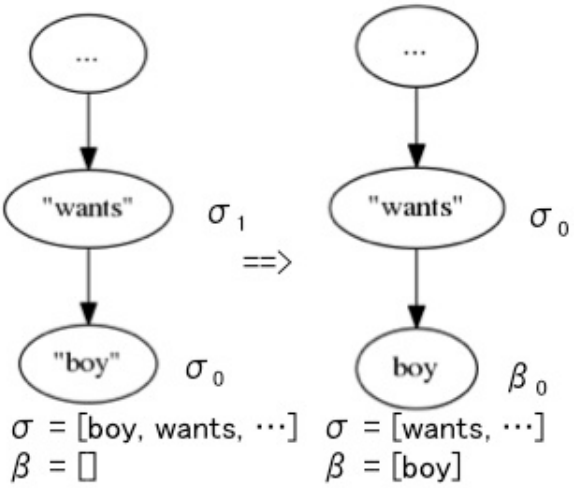


図 14: NextNode(c)

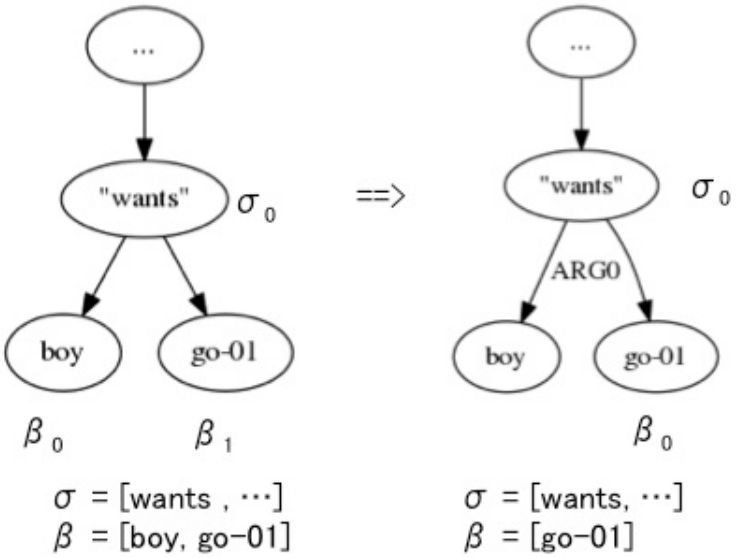


図 15: NextEdge(r)

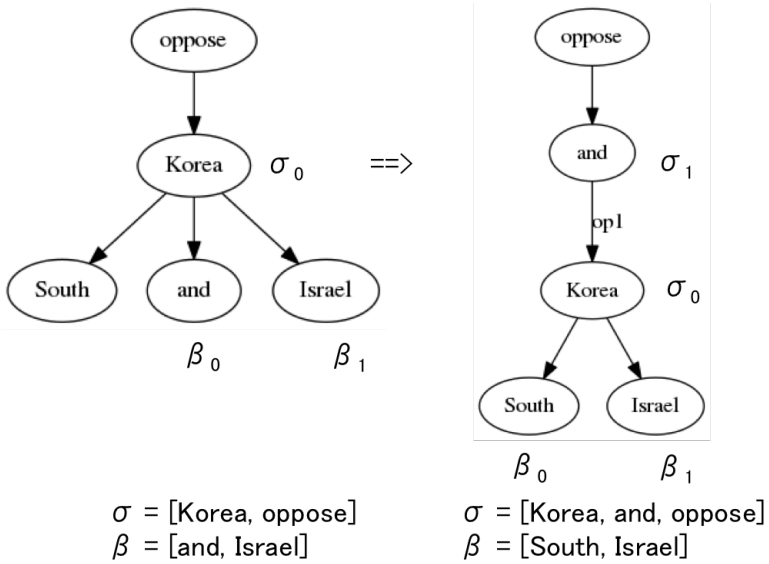


図 16: Swap(r)

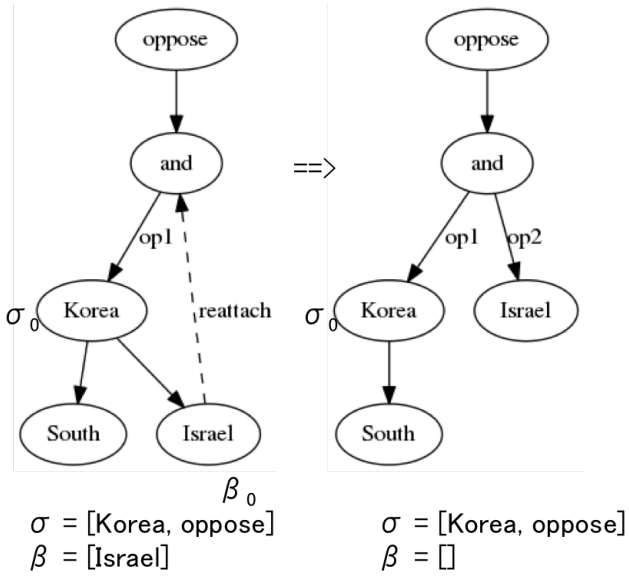


図 17: Reattach(k,r)

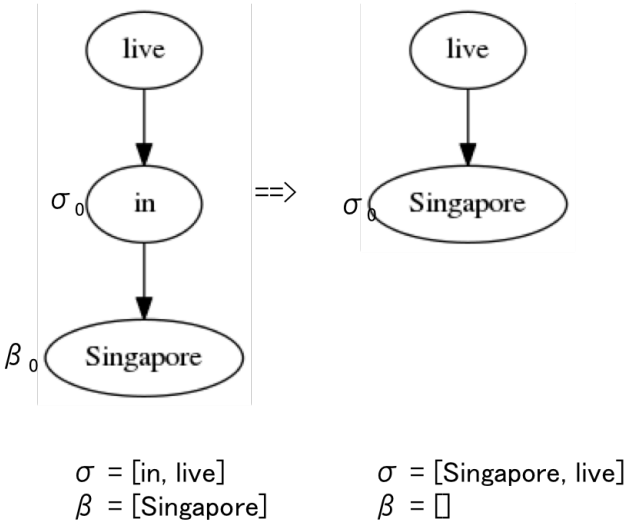


図 18: ReplaceHead

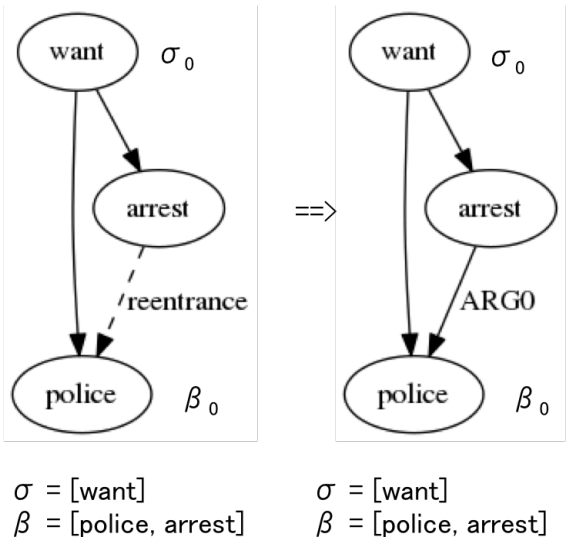
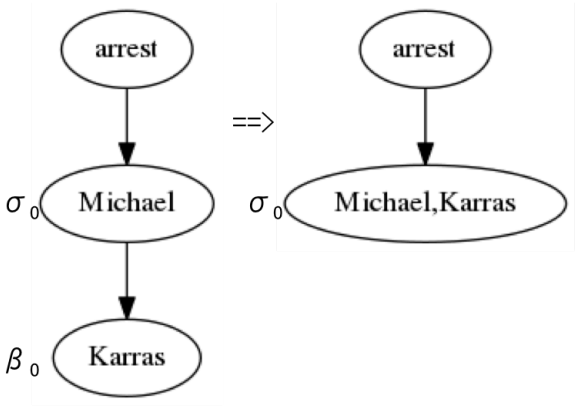
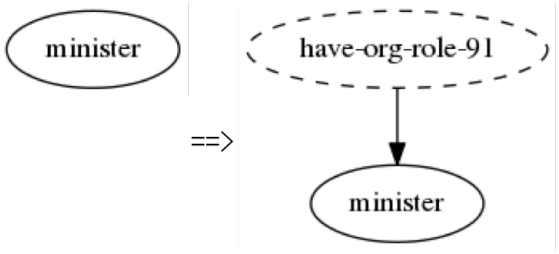


図 19: Reentrance(k,r)



σ = [Michael, arrest] σ = [Michael_Karras, arrest]
β = [Karras] β = []

図 20: Merge



σ = [minister] σ = [minister, habe-org-role-91]
β = [] β = []

図 21: Infer(c)

AMR は固有表現や修飾関係，モダリティなどあらゆる意味関係を表示しているのに対して，PropBank アノテーションは一部の述語を中心に述語と項の関係のみを表示しているので，訓練データとして利用するときには文から到達可能な一部の状態に対する動作しか学習できない．本節では PropBank アノテーションから復元可能な状態に対して真の動作を定義する手法を説明する．

文 $\mathbf{x} = w_1 \dots w_n$ に対して述語単語，PropBank のフレーム，スロット，項の単語の 4 つ組 $pb = (f, v, s, a)$ が与えられたとき，フレームと項の依存構文上の位置関係により，部分的な AMR とみなすことができる．

1. v は概念ラベル f のノードとなる．

2. v から a へ直接依存関係がある場合，すなわち，l_* を任意の依存関係ラベルとして依存構文に (v, l_*, a) が存在する場合，関係ラベル s のエッジ (v, a) が存在する．

3. v から a へ前置詞を挟んで依存関係がある場合，すなわち，p_* を任意の前置詞として依存構文に $(v, \mathrm{prep}, p_*), (p_*, \mathrm{pobj}, a)$ が存在する場合，関係ラベル s のエッジ (v, a) が存在する．

4. a から v へ直接依存関係がある場合，すなわち，l_* を任意の依存関係ラベルとして依存構文に (a, l_*, v) が存在する場合，s の逆関係ラベル s^{-1} のエッジ (a, v) が存在する．ここで逆関係ラベルとは，例えば “ARG0” 対して “ARG0-of” のようにフォーカスを入れ替えるようなラベルである．

5. v と a が依存構文上で兄弟関係にある場合，すなわち，文内の任意の単語を x_*，依存関係ラベルを l_{*1}, l_{*2} として，依存構文に $(x_*, l_{*1}, v), (x_*, l_{*2}, a)$ が存在する場合，照応関係を表すような関係ラベル s のエッジ (v, a) が存在する．

これらのパターンに対して，状態を定め，その状態から AMR を得るための動作を得ることができる．まず，状態は文から到達可能なものにするため，初期状態から β に σ_0 の子ノードを展開し，先頭から順に取り除いていく．$\beta = []$ となった場合 σ の先頭要素を取り除き σ_0 の子ノードを展開することを繰り返す．その過程において，述語単語，PropBank のフレーム，スロット，項の単語の 4 つ組

$pb = (f, v, s, a)$ に対して，依存関係と状態が以下のパターンに当てはまる場合，AMR を得るための真の動作が定義できる．

1. 状態が $\sigma_0 = v$，$\beta = []$ のとき，`NextNode(`f`)`
2. v から a へ直接依存関係がある場合，
   状態が $\sigma_0 = v$，$\beta_0 = a$ のとき，`NextEdge(`s`)`
3. v から a へ前置詞を挟んで依存関係がある場合，
   状態が $\sigma_0 = v$，$\beta_0 = p$ のとき，$\beta_0 = a$ に置換して `NextEdge(`s`)`
4. a から v へ直接依存関係がある場合，
   状態が $\sigma_0 = a$，$\beta_0 = v$ のとき，`NextEdge(`s^{-1}`)`
5. v と a が依存構文上で兄弟関係にある場合，
   状態が $\beta_0 = a$ のとき，`Reentrance(`v`,`s`)`

例として以下の 4 例を考え，PropBank アノテーション付テキストから復元できる動作を確認する．

- $\mathbf{x}$ = “*The boy went*”，pb = (go-01, *went*, ARG0, *boy*) に対して，
  (i) 状態 $S = (went|\sigma', boy|\beta', G)$ のとき，`NextEdge(`ARG0`)`
  (ii) 状態 $S = (went|\sigma', [], G)$ のとき，`NextNode(`go-01`)`
- $\mathbf{x}$ = “*go to school.*”，pb = (go-01, *go*, ARG1, *school*) に対して，
  (i) 状態 $S = (go|\sigma', to|\beta', G)$ のとき，$S = (go|\sigma', school|\beta', G)$ として `NextEdge(`ARG1`)`
  (ii) 状態 $S = (go|\sigma', [], G)$ のとき，`NextNode(`go-01`)`
- $\mathbf{x}$ = “*the washing machine*”，pb = (wash-01, *washing*, ARG0, *machine*) に対して，
  (i) 状態 $S = (washing|\sigma', [], G)$ のとき，`NextNode(`wash-01`)`
  (ii) 状態 $S = (machine|\sigma', wash\text{-}01|\beta', G)$ のとき，`NextEdge(`ARG0-of`)`

- $\mathbf{x}$ = "*The boy wants to go*"，pb = (go-01, *go*, ARG0, *boy*) に対して，
  (i) 状態 $S = (\sigma, boy|\beta', G)$ のとき，`Reentrance(`*go*，ARG0)
  (ii) 状態 $S = (go|\sigma', [], G)$ のとき，`NextNode(`go-01`)`

5.3 実験

5.3.1 実験設定

ベースラインとして Wang ら [12] が公開している実装[5] を利用し，5.2 節の手法で PropBank アノテーションから得られる，状態に対する真の動作を学習に加えた．Wang ら [12] にならい，学習，解析における探索は，ビーム幅 1 の貪欲サーチを行い，状態に対して動作を選択する分類器の学習には構造化平均化パーセプトロンを用いた．AMR の訓練，開発，テストデータには LDC2015E86: DEFT Phase 2 AMR Annotation R1[6] を利用した．これは，新聞記事やインターネット掲示板などの文に対して，AMR が付けられたコーパスである．PropBank アノテーションは OntoNotes Release 5.0[7] を用いた (**ALL**: 273,278 インスタンス) が，これには話し言葉ドメインの文も含まれるため，ドメインを新聞記事である Wall Street Jornal に限定したもの (**WSJ**: 103,630 インスタンス) とも比較した．ここでインスタンスとは，それぞれのフレームが付与された述語の数である．ただし，受動態のフレーム "be-01" や完了形のフレーム "have-02" など AMR に存在しない機能語のフレームを取り除くため，PropBank の頻度上位 100 単語に存在するが，AMR の訓練データには存在しないフレームは学習対象外とした．それぞれ SyntaxNet[23][8] で品詞タグ付けと依存構文を得た．評価は，概念と関係の適合率と再現率からなる F1 値を指標とした (Smatch Score[34])．

素性セットは Wang ら [10, 11, 12] に従うが，学習データの汎化を行い述語の選好性をより捉えやすくすることを期待して，分散表現素性を**追加**した設定とも比較した (**Embed**)．分散表現素性は Wang らが用いた単語表層素性に基づいて 4.2

[5] https://github.com/c-amr/camr
[6] https://catalog.ldc.upenn.edu/LDC2015E86
[7] https://catalog.ldc.upenn.edu/LDC2013T19
[8] https://github.com/tensorflow/models/tree/master/syntaxnet

	開発	テスト
ベースライン [12]	**65.67**	63.15
+ PropBank WSJ	65.66	**64.00**
+ PropBank ALL	65.41	63.60
ベースライン + Embed	65.06	63.52
+ PropBank WSJ	**65.66**	**64.56**
+ PropBank ALL	65.55	64.37

表 3: Smatch Score による解析性能．太字は分散表現素性を用いた場合と用いない場合について，各データセットにおける最高性能を示している．

	概念	関係
ベースライン	76.85	52.77
ベースライン + WSJ	**77.22**	**53.57**
ベースライン + Embed	77.07	53.13

表 4: テストセットに対する各解析器における概念，関係別の適合率，再現率からなる F1 値．太字は概念，関係の F1 値が高い方の解析器を示している．

節の手法で構築するが，単語表層素性は取り除かず両素性を学習時に用いた．単語分散表現については，Mikolov らが公開している skip-gram の実装 [9] を利用し，英語 Wikipedia[10] から 100 次元の分散表現を構築した．

5.3.2 結果

各解析器の Smatch Score を表 3 に示す．ベースラインに加えて PropBank からの学習を行った場合，開発セットではベースラインが最高性能を示したのに対し，テストセットでは PropBank からの学習を行った場合に最高性能を示した．学習のイテレーションの調整を開発セットで行っていることを考えると，PropBank からの学習を行った場合の方が 実際の解析時などに様々な文に対応できると考

[9] https://code.google.com/archive/p/word2vec/
[10] https://dumps.wikimedia.org/enwiki/20151201/

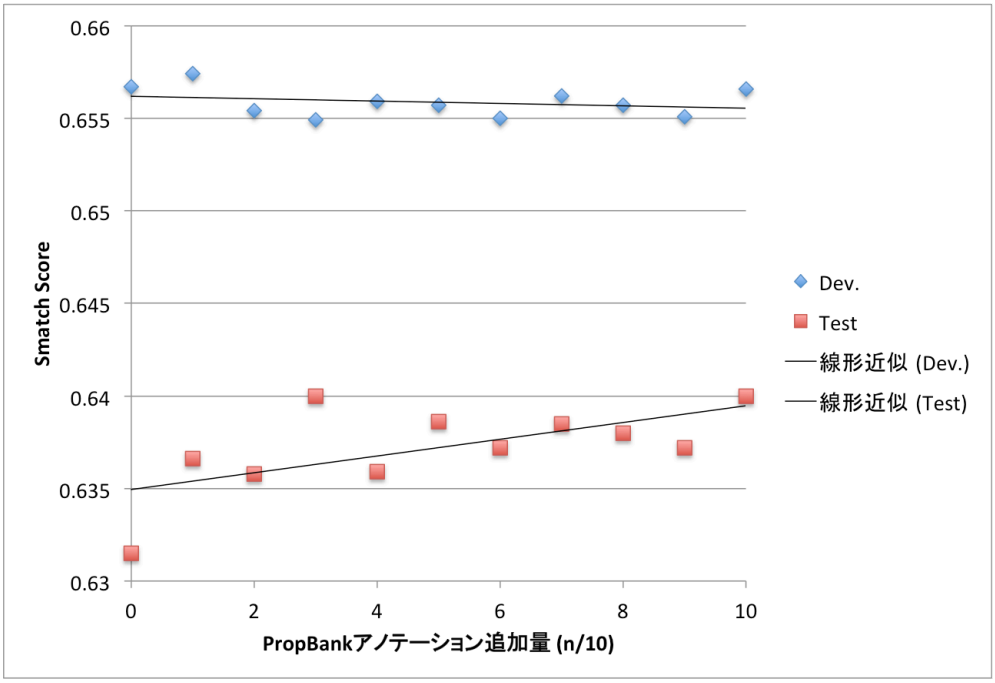


図 22: PropBank 学習データ量に対する AMR 解析性能.

えられる．PropBank のドメイン間の比較について，ALL の設定では，WSJ と比べて僅かにスコアが低いことが確認できた．学習に加えるドメインは実際の訓練データのドメインに合わせて適切に選択する必要があると考えられる．

分散表現素性を用いた場合，ベースラインと比較してテストセットに対する性能向上が見られた．また，分散表現素性を用いたモデルに対してさらに PropBank アノテーションを学習に加えた場合，今回用いた実験設定の中でテストセットに対する最高性能が得られた．分散表現素性を用いた場合の性能向上は，4 節と同様に未知語に対する効果や，類似単語同士の解析器の動作に対する効果があると考えられるが，特に AMR 解析における分散表現素性の効果と課題について 5.3.3 節で考察する．

5.3.3 分析

図 22 は，ベースラインに対して PropBank アノテーションを加えた際に，PropBank アノテーション (WSJ) の量を変化させたときの開発，テストセットに対す

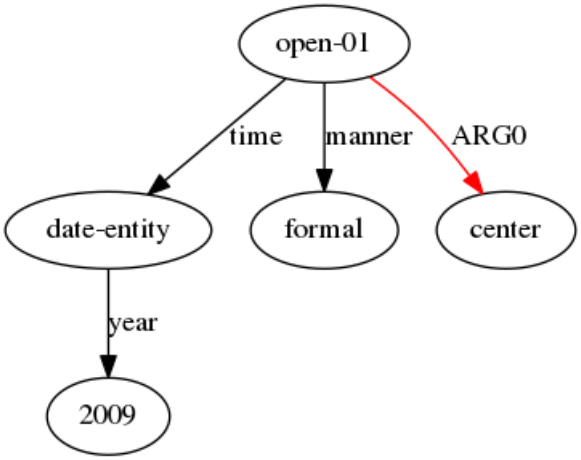
	ベースライン	ベースライン + WSJ	ベースライン + Embed
DeleteNode	**89.9**	89.8	89.5
NextNode	**95.7**	95.6	95.5
NextEdge	92.2	92.2	**92.3**
Swap	50.8	51.4	**51.7**
Reattach	39.2	**39.3**	38.4
ReplaceHead	80.7	80.9	**82.3**
Reentrance	**20.7**	18.4	19.6
Merge	68.6	72.7	**79.7**
Infer	**26.9**	26.1	18.3

表 5: 各解析器における各動作の適合率，再現率からなる F1 値．太字は各動作で F1 値が高い方の解析器を示している．

る性能の変化をプロットしたものである．また，それぞれに対する近似直線も合わせて示している．図 22 より，テストセットに対する性能が PropBank アノテーションのデータ量に従って向上していることから，PropBank アノテーションが AMR 解析の性能に寄与していると言える．また，AMR の訓練データを増加させずとも PropBank アノテーションの量を更に増やすことで AMR 解析の性能が向上すると期待できる．一方で，表 3 に示した PropBank アノテーションのドメインの違いも考慮すると，適用するテストデータや文に合わせて，話し言葉，書き言葉等のドメインは適切に選択する必要がある．

AMR 解析における PropBank や単語表層素性の効果を確認するために，概念，関係別の評価値（表 4）と動作ごとの評価値（表 5）をベースラインと比較した．表 4 には，Cai ら [34] の評価用スクリプトから，概念，関係別に適合率，再現率を計算し F1 値を示している．表 5 には，テストデータを解析する際に到達する全ての状態に対する動作の適合率と再現率からなる F1 値を示している．文に対して AMR が与えられれば，その文からできる状態に対して真の動作を求めることができるため，表 5 の値は，正解を真の動作として予測を分類器の出力した動作としたものである．ただし，NextEdge，NextNode などのパラメータを持つ動

AMR (ベースライン)



AMR (ベースライン+WSJ, 正解)

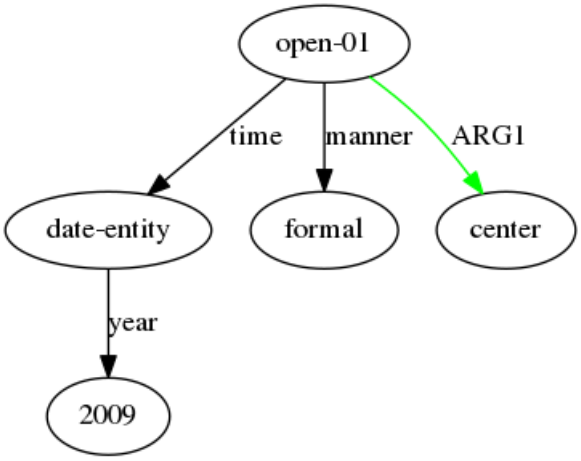
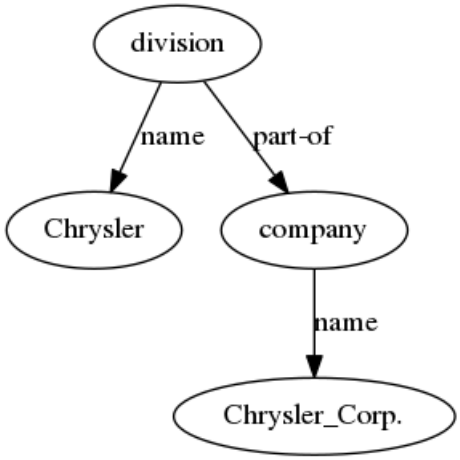
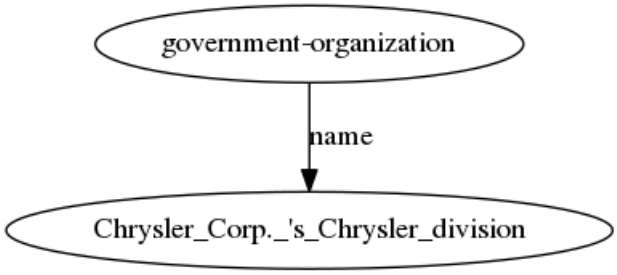


図 23: “*The center will formally open in 2009.*” に対する PropBank を学習に用いない場合と用いた場合の AMR 解析結果.

AMR (正解)



AMR (ベースライン)



AMR (ベースライン+Embed)

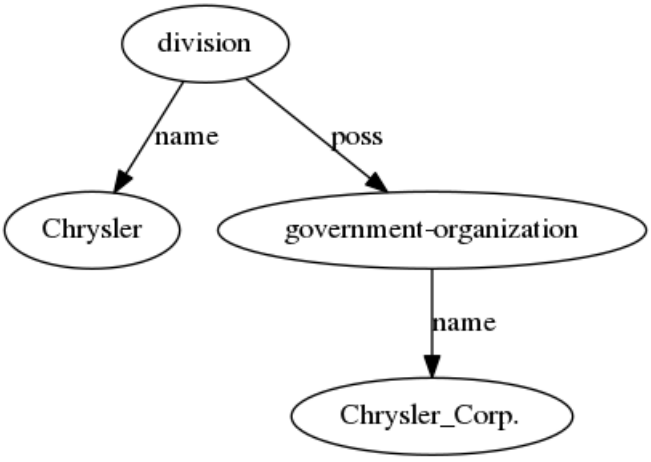


図 24: "*Chrysler Corp. 's Chrysler division*" に対する分散表現素性を用いない場合と用いた場合の AMR 解析結果.

作において，そのパラメータの区別はしていない．

PropBankを学習データに加えたとき，表5より，Swap，Reattach，ReplaceHead，Mergeの動作において性能が向上した．NextEdge，NextNodeの動作はベースラインからあまり変化していないが，パラメータを持つ動作においてそのパラメータの区別をしていないことと，表4の概念と関係のF1値が大きく向上していることを考えると，PropBankの情報は主に概念や関係のラベルを判定することに貢献していると考えられる．関係ラベル付によって性能向上した例を図23に示す．文“*The center will formally open in 2009.*”のAMRの概念“open-01”と“center”の間の関係ラベルについて，ベースラインでは“ARG0 (opener)”と誤判断したのに対して，PropBankを用いた場合には“ARG1 (thing opening)”と正しく判定できた．

分散表現素性を用いた場合，表5より，NextEdge，Swap，ReplaceHead，Mergeの動作において性能が向上した．特にMergeに関して顕著な性能向上が見られた．Mergeの性能向上により，解析ができた例を図24に示す．文“*Chrysler Corp. 's Chrysler division*”において，分散表現素性を用いない解析器では固有表現の区切りを発見できなかったのに対して，分散表現素性用いた解析器の場合，概念ラベル，関係ラベルの誤りはあれど，固有表現の区切りを特定しAMRの構造を当てることが可能になったと考えられる．

一方表5より，Inferはベースラインにおいても性能が悪かったが，PropBankや分散表現素性を用いた場合に更に性能が悪化した．PropBankや単語分散表現素性が効果的に機能しなかったInferに関連する例の一つとして，“*With this bridge, the distance would be very small.*”という文を考える．この文に対しての依存構文とAMRを図25に示す．図25より，このAMRは依存構文の単語に対応しない概念“cause-01”が現れている．Wangら[12]の解析器の動作に照らし合わせると，“*small*”から“cause-01”をInferの動作によって作り出す必要がある．この文からAMRを導出する場合，“small”という単語が必ず「引き起こす」という意味を持つわけではなく，“*will be A with B*”が「BがAを引き起こす」といったような，単語の組合せによる意味の変化，すなわち構成性を考える必要がある．単語表層素性や単純な分散表現素性を用いた場合，これを捉えることは難しい．今

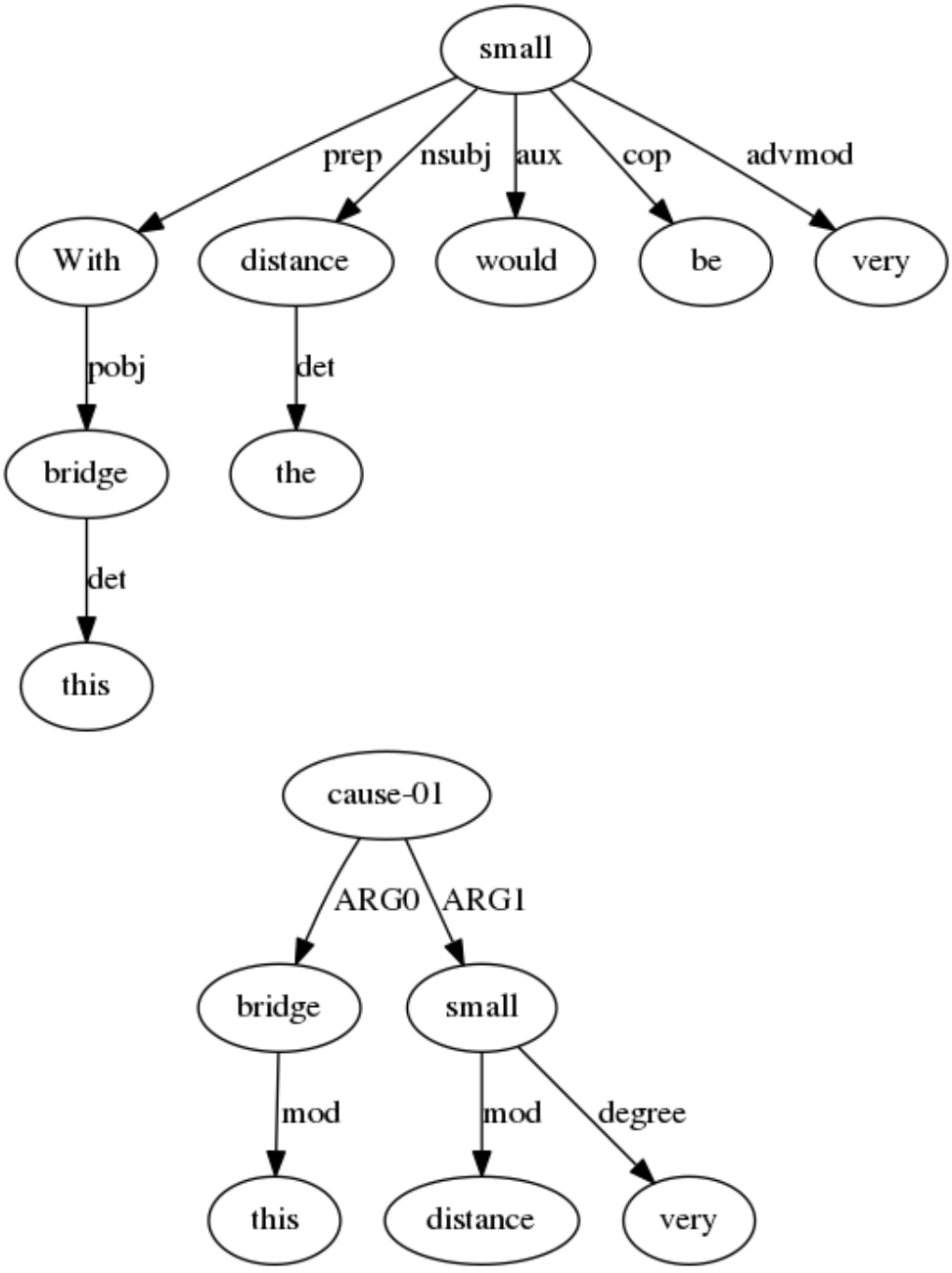


図 25: “*With this bridge, the distance would be very small.*” に対する依存構文（上）と AMR（下）.

後の課題としては，AMR の解析においてはこのような構成性が捉えられる分散表現を素性として用いるべきではないかと考えられる．すなわち “*would be small with*” の分散表現が，“*cause small*” と類似するような分散表現を素性として組み込むべきではないかと考えられる．

6　終わりに

本論文では，頑健で汎用的な自然言語解析器の構築のために，まず単語分散表現を用いた分散表現素性を提案した．この分散表現素性を依存構文解析に対して適用したところ，依存構文解析の性能向上に寄与することを確認した．この分散表現素性を用いる利点として，単語分散表現が (1) 訓練時に未知の単語を既知の単語と結びつける効果，(2) 類似する単語同士で解析器の動作を類似させる効果があることを分析した．

次に，意味表現解析である AMR 解析において，AMR の背後にあるオントロジの情報を部分的な AMR 構造として学習データに取り入れることで，AMR 解析の性能向上に寄与することを確認した．AMR 解析においても本論文で提案した分散表現素性を用いたとき，さらなる性能向上が確認できた．AMR 解析における PropBank による学習は概念や関係のラベル付などに，また分散表現素性は，AMR 解析に特有な処理である単語を除去したり結合したりする処理などに貢献することを分析した．さらに，AMR 解析が誤った事例を定性的に分析し，意味的な構造を捉える場合は単語の組合せによる意味の変化を捉える必要があり，AMR 解析の今後の課題を考察した．

謝辞

本研究を進めるにあたり，多くの皆様のご協力，ご助言をいただきましたことに，ここに心より感謝申し上げます.

主指導教員である乾健太郎教授には，ご多忙の中，研究活動だけでなく進路に関することなど多くのご指導，ご助言を頂きましたことに心より感謝申し上げます．副指導教員である岡崎直観准教授には，同じく研究活動に関して多くのご助言を頂きましたことに心より感謝申し上げます．ご多忙の中審査委員をお引き受けくださいました，木下賢吾教授，大町真一郎教授に心より感謝申し上げます．研究方針や研究手法，論文執筆に関しまして，直接のご指導を頂いた田然研究特任助教に心より感謝申し上げます．研究会や日々の議論におきまして，多くのアドバイスを頂きました乾・岡崎研究室の皆様に感謝申し上げます.

最後になりましたが，学校生活におきまして関わってくださいましたすべての皆様に感謝致します.

参考文献

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

[2] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP HLP*, pp. 523–530, 2005.

[3] 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.

[4] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 2008.

[5] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of LAW7-ID,*, 2013.

[6] Paul Kingsbury and Martha Palmer. From treebank to propbank. In *Proceedings of LREC-2002*, May 2002.

[7] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, Vol. 31, No. 1, pp. 71–106.

[8] Daniel Gildea and Martha Palmer. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL*, pp. 239–246, 2002.

[9] Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of ACL*, pp. 1426–1436, 2014.

[10] Chuan Wang, Nianwen Xue, and Sameer Pradhan. A transition-based algorithm for amr parsing. In *Proceedings of NAACL-HLT*, pp. 366–375, 2015.

[11] Chuan Wang, Nianwen Xue, and Sameer Pradhan. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of ACL-IJCNLP*, pp. 857–862, 2015.

[12] Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. CAMR at semeval-2016 task 8: An extended transition-based AMR parser. In *Proceedings of SemEval-2016*, pp. 1173–1178, 2016.

[13] Zellig Harris. Distributional structure. *Word*, Vol. 10, No. 23, pp. 146–162, 1954.

[14] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Proceedings of TACL*, 2015.

[15] 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 NIPS*, pp. 3111–3119. Curran Associates, Inc., 2013.

[16] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, 2008.

[17] 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.*, Vol. 18, No. 4, pp. 467–479, December 1992.

[18] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*, 2014.

[19] Jacob Andreas and Dan Klein. How much do word embeddings encode about syntax? In *Proceedings of ACL*, 2014.

[20] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, Vol. 12, , 2011.

[21] Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. Parsing with compositional vector grammars. In *Proceedings of ACL*, 2013.

[22] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, 2014.

[23] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In *Proceedings of ACL-IJCNLP*, 2015.

[24] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL-IJCNLP*, 2015.

[25] Taro Watanabe and Eiichiro Sumita. Transition-based neural constituent parsing. In *Proceedings of ACL-IJCNLP*, 2015.

[26] Sameer S Pradhan, Wayne H Ward, Kadri Hacioglu, James H Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of HLT-NAACL*, pp. 233–240, 2004.

[27] Lauritz Brandt, David Grimm, Mengfei Zhou, and Yannick Versley. Icl-hd at semeval-2016 task 8: Meaning representation parsing - augmenting amr parsing with a preposition semantic role labeling neural network. In *Proceedings of SemEval-2016*, pp. 1160–1166, 2016.

[28] Yevgeniy Puzikov, Daisuke Kawahara, and Sadao Kurohashi. M2l at semeval-2016 task 8: Amr parsing with neural networks. In *Proceedings of SemEval-2016*, 2016.

[29] Guntis Barzdins and Didzis Gosko. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. In *Proceedings of SemEval-2016*, pp. 1143–1147, 2016.

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

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

[32] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, 2003.

[33] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in NIPS*, 2014.

[34] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of ACL*, pp. 748–752, 2013.

発表文献一覧

受賞一覧

- The 29th Pacific Asia Conference on Language, Information and Computation (PACLIC-29) Best Paper Award (Computation), October 2015.
- 平成 26 年度東北大学総長賞, March 2015.

国際会議論文

- Yuki Igarashi, Hiroya Komatsu, Sosuke Kobayashi, Naoaki Okazaki and Kentaro Inui. Tohoku at SemEval-2016 Task 6: Feature-based Model versus Convolutional Neural Network for Stance Detection. In Proceedings of the International Workshop on Semantic Evaluation (SemEval '16), pp.401-407, June 2016.
- Hiroya Komatsu, Ran Tian, Naoaki Okazaki and Kentaro Inui. Reducing Lexical Features in Parsing by Word Embeddings. In Proceedings of The 29th Pacific Asia Conference on Language, Information and Computation (PACLIC-29), pp.106-113, October 2015.

国内会議・研究会論文

- 小松 広弥, 田 然, 岡崎 直観, 乾 健太郎. 単語分散表現の shift-reduce 型構文解析への利用. 情報処理学会研究報告 自然言語処理（NL）May 2015.