文の潜在グラフ表現の学習：ニューラル機械翻訳での検証

橋本和真　鶴岡慶雅
東京大学 工学系研究科
{hassy,tsuruoka}@logos.t.u-tokyo.ac.jp

1　はじめに

自然言語処理における文の構文解析は，句構造や係り受けの解析という形で長年研究されている．それらの構文解析は，人手のアノテーションに基づく機械学習によって実現されることが主流である．構文情報は，関係抽出や機械翻訳といったタスクの素性として精度向上に寄与してきた．また，近年のニューラルネットワークに基づく研究においても，文を単語列でなく木構造で表すことの有効性が示唆されている [2].

しかし，構文解析は単一言語の文法に基づいており，必ずしも適用先のドメインやタスクに適しているとは限らない．最近ではタスクに特化した句構造を強化学習により獲得する手法が提案されているが，「二分木」という強い制約が存在する [6]．木構造はその表現のし易さから，ニューラルネットワークにおいて幅広く用いられているが，離散的に木構造として文を表現することが最適とも限らない．

そこで本研究では，適用先のタスクに特化した，文の潜在的なグラフ表現を学習するモデルを提案する．このグラフ表現は，単語から単語への重み付きアークの集合であり，タスクの目的関数の最適化により直接学習される．提案モデルは，タスク依存のモデルと，タスク非依存の構文解析モデルを連結することで実現され，構文解析モデルを既存のアノテーションで事前学習することも可能である．実験では，ニューラル翻訳を用いた英日翻訳タスクに提案手法を適用してグラフ表現の学習の効果を検証し，人手の係り受けのアノテーションとは違う依存関係が学習されることを確認した．

2　文の潜在グラフ表現の学習

2.1　係り受け解析に基づくグラフ表現

ここでは，係り受け解析を基にモデルを考える．係り受け解析では，特別なノード ROOT を根とした木構造で文を表現する．各ノードは文中の各単語に対応し，それぞれのノード間に係り受けラベルが付与される．単語数 N の文の各単語 w_i $(1 \leq i \leq N)$ の親ノード H_{w_i} はヘッドと呼ばれ，その係り受けラベル l_{w_i} を用いて

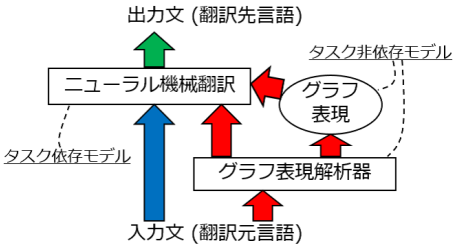

図 1: 潜在グラフ表現を用いたニューラル機械翻訳．

文は (w_i, H_{w_i}, l_{w_i}) の三つ組みの集合で表現される．

本研究では，この単語間の依存関係のモデル化における，「木構造」という強い制約を取り除き，親ノードの全候補に対する確率分布として表現する．つまり，単語間の関係を離散的に決定するのではなく，重み付きで複数の候補の重ね合わせを考えることになる．これにより，各単語に着目して複数の単語間の関係を表現することができる．具体的には，前述の三つ組みを $(w_i, p(H_{w_i}|w_i), p(l_{w_i}|w_i))$ に変更する．単語 w_j $(j \neq i)$ が親ノードになる確率は，$p(H_{w_i} = w_j|w_i)$ として表現される．これを，文の潜在グラフ表現と呼ぶ．

このような表現を獲得するために，係り受け解析を，各単語の親ノード同定タスクとしてモデル化した手法 [3] を利用する．元の手法との違いは，チャンキング層を除いて品詞タグ付けの層と係り受け解析の層からなる 2 層の双方向 LSTM リカレントニューラルネットワークを用いること，文末を表す EOS を各文に加え，EOS に対応する係り受け解析の隠れ層を ROOT を表すベクトルとすること，である．

2.1.1　単語ベクトル表現

まず，入力文の各単語 w_i はその単語ベクトル $v_{\mathrm{dp}}(w_i) \in \mathbb{R}^{d \times 1}$ と文字 n グラムベクトル $c(w_i) \in \mathbb{R}^{d \times 1}$ の連結 $x(w_i) = [v_{\mathrm{dp}}(w_i); c(w_i)] \in \mathbb{R}^{2d \times 1}$ に変換される．d はベクトルの次元である．各単語中に存在する文字 n グラムを単語ベクトル同様にベクトル化され，それらを平均することで $c(w_i)$ が計算される．これらの単語と文字 n グラムベクトルはモデルパラメータとして学習される．

2.1.2 品詞タグ付け層

1 層目では, ベクトル化された単語を双方向 LSTM により処理することで各単語の文脈付きの表現 $h_i^{(1)} \in \mathbb{R}^{2d\times 1}$ を獲得する:

$$h_i^{(1)} = [\overrightarrow{h}_i^{(1)}; \overleftarrow{h}_i^{(1)}], \quad (1)$$

ここで, $\overrightarrow{h}_i^{(1)}, \overleftarrow{h}_i^{(1)} \in \mathbb{R}^{d\times 1}$ は順方向・逆方向の LSTM により計算された単語 w_i に対応する隠れ層である. 双方向 LSTM の計算式は先行研究 [3] で用いられているものに従い, 各遷移では対応する単語の表現ベクトル $x(w_i)$ を入力する.

次に, 単語レベルのタグを予測するために $h_i^{(1)}$ をクラス分類器に入力する:

$$p_i^{(1)} = \mathrm{softmax}(W_s^{(1)}\mathrm{ReLU}(W_h^{(1)}h_i^{(1)})), \quad (2)$$

ここで, タグの種類数を $C^{(1)}$ とすると, $p_i^{(1)} \in \mathbb{R}^{C^{(1)}\times 1}$ はタグの確率分布であり, $W_s^{(1)} \in \mathbb{R}^{C^{(1)}\times 2d}$, $W_h^{(1)} \in \mathbb{R}^{2d\times 2d}$ は重み行列である. バイアスの表記は省略されている. このタグ付け器は, 人手のアノテーションに基づいて学習することも可能であり, 次の係り受け解析層に入力することで学習することも可能である.

2.1.3 係り受け解析層

1 層目の情報に基づいて, 2 層目では双方向 LSTM を用いて単語の依存関係をモデル化する. 2 層目の順方向 LSTM への入力は, 単語の表現ベクトル $x(w_i)$ と 1 層目の LSTM の隠れ層 $\overrightarrow{h}_i^{(1)}$ に加え, タグ付けの結果 $W_\ell^{(1)}p_i^{(1)} \in \mathbb{R}^{d\times 1}$ を用いる. ここで, $W_\ell^{(1)} \in \mathbb{R}^{d\times C^{(1)}}$ は, 各タグを表現する列ベクトルをまとめた行列である. この入力により, 双方向 LSTM による各単語の文脈付きの表現 $h_i^{(2)} = [\overrightarrow{h}_i^{(2)}; \overleftarrow{h}_i^{(2)}]$ を得る.

次に, 各単語 w_i の親ノードが w_j である確率を以下のように計算する:

$$p(H_{w_i} = w_j|w_i) = \frac{\exp\left(h_j^{(2)} \cdot (W_{dp}h_i^{(2)})\right)}{\sum_{k\neq i}\exp\left(h_k^{(2)} \cdot (W_{dp}h_i^{(2)})\right)}, \quad (3)$$

ここで, $W_{dp} \in \mathbb{R}^{2d\times 2d}$ はパラメータ行列であり, $h_{N+1}^{(2)}$ を ROOT の表現ベクトルとして対応させる. これにより, 各単語が他の単語とどの程度の重みで依存関係があるのかをモデル化できる. このモデルは人手のアノテーションにより学習が可能 [3] であるが, 本研究のように, 他のタスクへの入力として使用することでタスクに特化した依存関係を学習することも可能である.

最後に, 依存関係のラベルを予測する:

$$p_i^{(2)} = \mathrm{softmax}(W_s^{(2)}\mathrm{ReLU}(W_h^{(2)}[h_i^{(2)}; y(H_{w_i})])), \quad (4)$$

ここで, $y(H_{w_i}) \in \mathbb{R}^{2d\times 1}$ は, 人手のアノテーションで教師有り学習する際は正解の親ノードの隠れ層を入力する. タスク特化の学習を行う場合には, $\sum_{j\neq i} p(H_{w_i} = w_j|w_i)h_j^{(2)}$ として重み付き和で計算する.

2.2 ニューラル機械翻訳への適用

2.2.1 エンコーダ

本稿では, 以上のモデルにより得られる文のグラフ表現をニューラル機械翻訳に適用する. ここでは, アテンションに基づくニューラル機械翻訳モデル [5] を利用する. まず, 入力文の各単語 w_i を d' 次元の単語ベクトル $v_{mt}(w_i) \in \mathbb{R}^{d'\times 1}$ に変換する. ただし, この $v_{mt}(w_i)$ はグラフ表現解析器の $v_{dp}(w_i)$ とは異なる.

次に, 各単語を順方向 LSTM に入力することで各単語に対応する隠れ層 $h_i^{(en)} \in \mathbb{R}^{d'\times 1}$ を計算する. この LSTM の遷移の際の入力には, 単語ベクトル $v_{mt}(w_i)$ と, グラフ表現解析モデルの隠れ層 $h^{(2)_i}$ の連結 $[v_{mt}(w_i); h_i^{(2)}] \in \mathbb{R}^{(d'+2d)\times 1}$ を用いる. つまり, 全体として見ると 2 層の双方向 LSTM と 1 層の順方向 LSTM からなる 3 層構造とみなすことができる.

系列データをリカレントニューラルネットワークで扱う場合には, 離れた単語間の関係を明示的には考慮していないが, ここではグラフ表現解析器の出力を基に, 離れた単語間の関係を明示的に導入する. 具体的には, 文末の EOS を除く各単語に関して,

$$dep(w_i) = \tanh(W_{dep}[h_i^{(en)}; z(H_{w_i}); p_i^{(2)}]), \quad (5)$$

ここで, $z(H_{w_i}) = \sum_{j\neq i} p(H_{w_i} = w_j|w_i)h_j^{(en)}$ は w_i の親ノード候補に対応する隠れ層ベクトルの重み付き和である.

2.2.2 デコーダ

デコーダは 1 層の LSTM から成り, その初期状態はエンコーダの最後の (EOS に対応する) 隠れ層 $h_{N+1}^{(en)}$ とそのメモリセルで初期化される. このデコーダの t 番目の隠れ層 $h^{(dec)_t} \in \mathbb{R}^{d'\times 1}$ が与えられたとき, t 番目の出力単語を予測するために, まずは元のモデル [5] と同様にエンコーダの隠れ層の重み付き和を計算する (アテンション):

$$a_t = \sum_{i=1}^{N+1} s(i,t)h_i^{(enc)}, \quad (6)$$

ここで, $s(i,t)$ は, t 番目の出力単語を予測する際の, エンコーダの i 番目の隠れ層の寄与分を示す係数であり, 以下のように計算される:

$$s(i,t) = \frac{\exp\left(h_t^{(dec)} \cdot h_i^{(enc)}\right)}{\sum_{j=1}^{N+1}\exp\left(h_t^{(dec)} \cdot h_j^{(enc)}\right)}. \quad (7)$$

さらに, 離れた単語間の関係を考慮するために新たに導入した隠れ層にもアテンションを当てる:

$$a'_t = \sum_{i=1}^{N} s'(i,t) dep(w_i), \tag{8}$$

この係数 $s'(i,t)$ は式 (7) と同様に計算される. これらを用いることで, 単語予測のための隠れ層 $\tilde{h}_t^{(dec)} \in \mathbb{R}^{d' \times 1}$ を計算する:

$$\tilde{h}_t^{(dec)} = \tanh(\tilde{W}[h_t^{(dec)}; a_t; a'_t]), \tag{9}$$

ここで, $\tilde{W} \in \mathbb{R}^{d' \times 3d'}$ は重み行列であり, この $\tilde{h}^{(dec)_t}$ は, t+1 番目のデコーダの隠れ層を計算する際の入力として, 出力単語のベクトルと合わせて使用される. 出力単語 w_t の確率分布は, $p(w'_t) = \mathrm{softmax}(W^{(dec)}\tilde{h}_t^{(dec)})$ と計算され, このモデルの学習は, 学習データ中の正解単語の出力確率に関する負の対数尤度を最小化することで行われる. 学習の高速化のため, BlackOut サンプリング [4] を用いる.

この学習により, 全てのモデルパラメータが同時学習される. 特に, 式 (5), (8) の導入により, どの単語間を関連付けるのが良いか, ということが適用先のタスク, ここでは機械翻訳に特化されて学習される.

3 実験設定

3.1 データ

本稿では, ASPEC 英日翻訳のタスク[1]を用いた. 以前の実験設定 [2] に従い, 英語の単語分割は構文解析器 Enju の出力に従い, 日本語の単語分割には KyTea を用いた. 学習データ 300 万文のうち, 単語数が 50 以下のものを選択し, 1,346,946 文ペアを用いた. 開発データは 1,790 文であり, テストデータは 1,812 文である. 評価実験では, 下記の 2 種類のデータセットを用いた.

小データセット モデルの様々な設定の比較をするために, 上述の学習データのうち 10 万文ペアのみの小さなデータセットを用意した. 英語と日本語の単語, 英語の文字 n グラムの語彙は, それぞれ小データセットに 2 回以上出現するもので構築した. 英語で文字 n グラムを使用する場合には 2, 3, 4 グラムを用いた. この結果, 翻訳先言語 (日本語) の語彙数は 23,532 となった.

大データセット 全データを用いた実験では, 英語の単語と文字 n グラムはそれぞれ 3 回以上出現するもので構築し, 日本語の単語は 5 回以上出現するもので構築した. この結果, 翻訳先の語彙数は 65,680 となった.

[1] http://lotus.kuee.kyoto-u.ac.jp/WAT/.

	BLEU	Perplexity
提案手法	28.50±0.21	11.98±0.20
提案手法+事前学習	28.41±0.35	12.06±0.13
A. グラフ表現の利用無し	28.45±0.13	12.08±0.04
B. グラフ表現の重み一定	28.25±0.17	12.36±0.12
C. 解析器の直接接続無し	27.93	13.58
D. 解析器無し	25.43	15.27

表 1: 小データでの開発データの翻訳精度. ± が付いているものは, 5 回の実験結果の平均と標準偏差である.

3.2 モデルパラメータの学習と翻訳

グラフ表現解析器の単語ベクトル, 双方向 LSTM の隠れ層などの次元は $d = 100$ とした. 英語の単語ベクトルと文字 n グラムベクトルは先行研究 [3] で, 英語 Wikipedia を用いて事前学習されたもので初期化した. その他の重み行列は $[-\sqrt{\frac{6.0}{row+col}}, +\sqrt{\frac{6.0}{row+col}}]$ の範囲の一様乱数で初期化した. ここで, row と col はそれぞれ行列の行と列の数に対応する. ただし, バイアスと softmax のパラメータは 0 で初期化し, LSTM の忘却ゲートのバイアスは 1 で初期化した.

ニューラル機械翻訳に関しては, 単語ベクトル, LSTM の隠れ層などの次元は, 小データでは $d' = 256$ とし, 大データでは $d' = 512$ とした. バイアス, softmax, LSTM の忘却ゲートの他のパラメータは $[-0.1, +0.1]$ の一様乱数で初期化した. BlackOut のサンプリング数は小データでは 2,000 とし, 大データでは 2,500 として, ミニバッチ中の各文ごとに負例を共有して高速化を達成した[2]. 学習にはモメンタム付きの確率的勾配降下法を用い, ミニバッチサイズは 128, 勾配の大きさは 1 にクリップした. モメンタム係数は, 小データでは 0.75, 大データでは 0.7 とした. 初期学習率 1.0 から始めて, 開発データ上での Perplexity が悪化するたびに学習率を 0.5 倍することで調整を行った. また, 式 (9) に dropout を係数 0.2 で適用した.

翻訳はビームサーチで行い, そのスコアには文長の統計量に基づく手法 [2] を用いた. また, 各出力文の長さでスコアを割り, 最終的なスコアとして用いた. ビーム幅は各設定ごとに調整したが, 小データでは 12, 大データでは 50 程度がよいことが多かった.

4 結果

表 1 に, 小データを用いた場合の, 開発データにおける BLEU スコアと Perplexity を示す. Penn Tree Bank の WSJ コーパスの品詞タグ付けと係り受け解析 (Stanford Dependency v3.3.0) の学習データで, グラフ表現

[2] "Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz" の 16 スレッド並列で, 小データは 2 時間, 大データは 4 日程度で学習が終了する.

	BLEU	RIBES
提案手法	38.66	82.17
Kyoto-U [1]	38.20	82.39

表 2: テストデータでの翻訳精度の比較.

解析器を 1 エポックだけ事前学習してから, 提案手法の学習を行った結果, 翻訳スコアとしては有意な効果は確認できなかった. ただし, 事前学習により学習の初期にロスが落ちていくのが速いという違いが確認できたため, そこでの過学習を防ぐことによる精度向上の余地があると考える. 実際, 実験に必要なエポック数が, 事前学習無しの場合に比べて 2〜3 少ないことが多く, 翻訳スコアは非常に良いか悪いかの両極端であり, ばらつきが大きい.

比較手法 A は, 式 (5) を使用しない場合, つまり, 単純に単語ベースで多層 LSTM を用いたモデルであるが, 翻訳スコアにおいては提案手法による有意な改善は確認できなかった. 比較手法 B は, グラフ表現の重みを定数 $\frac{1}{\text{文長}}$ に固定したものであるが, 動的に重みを学習する提案手法のほうがわずかに良いことがわかる. 比較手法 C は, グラフ表現解析器の双方向 LSTM をニューラル機械翻訳モデルに接続しないモデルであり, 翻訳モデルを 1 層に保ちつつ, グラフ表現のみを学習する. 一方で比較手法 D は, グラフ解析器のすべてを取り除いた, 元の 1 層 LSTM による単語ベースモデルである. C と D の比較により, 離れた単語間の関係を明示的に考慮することが有用である可能性が示唆されるが, 翻訳モデル本体を多層化して強力にすると, その効果が薄れるということが見て取れる. つまり, 提案手法の優位性を示すためには, 可能な限り強力なベースラインを構築することが重要である. また, BLEU スコアはモデルの違いよりも, ビームサーチなどの変更により大幅なスコア変動が起こることがあり, どのように翻訳結果を評価すべきか, という課題は常に存在する.

次に, 大データを用いた場合の, テストデータでの結果を表 2 に示す. 同データセットでの最高精度のシステムの結果を比較として載せているが, 提案手法の結果はそれと同等のスコアを達成している. Kyoto-U のシステムは 1,000 次元の多層構造などを用いており, それと比較すると提案手法はより小さなモデルで同等の翻訳スコアを達成しており, 効果的に動作していると言える. 今後は, この高精度の状況において, 潜在的なグラフ表現を獲得することの特徴を調査していく.

5 学習されたグラフ表現の分析

最後に, 学習された文のグラフ表現に関する結果を調査する. まず, 表 2 に示す提案手法で, Penn Tree Bank の WSJ コーパスの開発データをグラフ表現に変換し, 各単語に重みが最大である単語を親ノードとして扱うことで係り受け解析の評価を行ったところ, ラベル無し精度 Unlabeled Attachment Score は 14.7%であった. また, 正解の親ノードが, 2, 3 番目以内に重みが大きい単語に含まれるのは, それぞれ 24.5%, 32.7%であった. これにより, ある程度は人手のアノテーションと一致する依存関係を獲得していることがわかる.

このグラフ解析器を WSJ コーパスで事前学習を行うと, 各単語の親ノード候補の単語に対する重みは 1 に近い値になる, つまり, ほとんど決め打ちになることが多い. 一方で, 適用先のタスク (ニューラル機械翻訳) に特化して学習を行うとそのような傾向は見られず, 様々な単語と関連付けようとする傾向が見られた.

6 おわりに

本稿では, ニューラル機械翻訳に特化した, 文の潜在的なグラフ表現を学習することを試みた. 小さなデータでの翻訳スコアとしては, ベースラインと比較して有意な向上は見られなかったが, 今後はより大きなデータでの比較等をして, 提案手法の有用性を調査していく.

謝辞

本研究は、JST、CREST の支援を受けたものである。CPU でのニューラルネットワーク学習の高速化の知見に関して、Yuchen Qiao 氏と田浦健次郎氏に感謝いたします。

参考文献

[1] F. Cromieres, C. Chu, T. Nakazawa, and S. Kurohashi. Kyoto University Participation to WAT 2016. In *WAT*, 2016.

[2] A. Eriguchi, K. Hashimoto, and Y. Tsuruoka. Tree-to-Sequence Attentional Neural Machine Translation. In *ACL*, 2016.

[3] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. *arXiv*, 2016.

[4] S. Ji, S. V. N. Vishwanathan, N. Satish, M. J. Anderson, and P. Dubey. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *ICLR*, 2016.

[5] M. T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*, 2015.

[6] D. Yogatama, P. Blunsom, C. Dyer, E. Grefenstette, and W. Ling. Learning to Compose Words into Sentences with Reinforcement Learning. *arXiv*, 2016.