依存構造グラフを用いたニューラル回答選択

高橋 拓誠　谷口 友紀　三浦 康秀　大熊 智子
富士ゼロックス株式会社
{takahashi.takumi, Taniguchi.Tomoki, Yasuhide.Miura, Ohkuma.Tomoko}@fujixerox.co.jp

1 はじめに

回答選択タスクは，質問文に対して回答候補の中から適切な回答を選ぶタスクであり，質問応答の重要なタスクの 1 つとして取り組まれている．質問に対して正解の回答を得るために，質問と回答候補対を入力とした深層学習に基づく手法が多く提案されている [1, 9, 10]．

本タスクでは，質問に対して適切な回答を得るために，質問に対する回答候補の適切さを計算する．しかしながら，回答候補が長文になるにつれて，質問に対応する部分を特定することが困難となり，回答選択の精度が低下してしまうことが報告されている [1, 10]．一般的に，長文になるほど依存構造を示す構文木は深くなることが知られているが，長文に対応するためには，この深い依存構造を理解するための枠組みが重要であると言える．

Iyyer ら [4] は文の依存構造を捉えるために，構文解析結果に対して Recursive Neural Network(RNN) を適用することで質問応答で高い精度を示した．しかし，木構造に対して RNN を用いる場合，事前に得た構文解析の結果にネットワークの構造が固定されてしまう制約がある．その結果，構文解析誤りを訂正できないことに起因して質問応答の性能が低下してしまう問題が生じる．この問題に対して，動的に構文構造を学習出来る仕組みを適用することで改善が期待出来るが，質問応答で学習可能な構文構造を取り入れた手法はない．

本論文では，質問と回答候補の構文的な依存構造を考慮して回答を選択するモデルを提案する．構文解析器を用いて得られた依存関係ラベルをもとに，依存構造グラフを生成して文内の単語間の依存関係を表現する．さらに，依存関係ラベルは学習可能なベクトルとして表現するとともに，Self-Attention を用いることで修正可能な構文構造として学習を行う．

本論文の貢献は以下の三点である．

- 構文構造を依存構造グラフとして表現することで，単語間の構文的な関係性を考慮しながら回答選択するニューラルネットワークのモデルを提案する．
- 従来の質問と回答の対応関係を利用した手法や，構文情報を利用した手法と比較して，提案手法の性能が優れていることを示す．
- 特に長文の回答候補を持つ質問に対して，従来手法と比較して提案手法の性能が向上することを示す．

2 提案手法

本研究では，質問 Q に対して回答候補 $\{A_1, ..., A_N\} \in A$ から適切な回答を選択するために，ランキング学習に基づくモデルを採用する．図 1 に提案手法の概要を示す．モデル

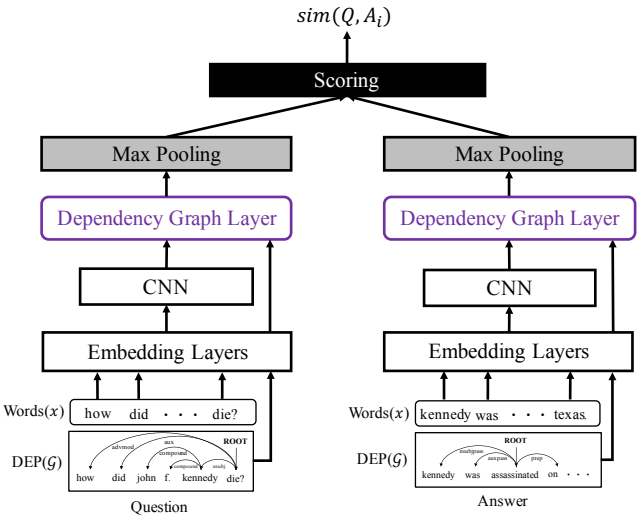

図 1: 提案手法の概要図

全体は，Embedding Layers，CNN，Dependency Graph Layer，Max Pooling，Scoring から構成される．以降の節では，各層の処理について説明する．

2.1 依存構造グラフの構築

本研究では，入力文 x に含まれる各単語 $(x_1, ..., x_M)$ をノード，x_i と x_j 間の依存関係ラベル rel_{ij} をエッジとした依存構造グラフ $\mathcal{G}$ を構築する．依存関係ラベル rel_{ij} は，事前に構文解析器[1]を用いて抽出した．最終的に，依存構造グラフ $\mathcal{G}$ のエッジを ID に変換し構築した隣接行列 $\boldsymbol{A} \in \mathbb{R}^{M \times M}$ を次節の Embedding Layers への入力とする．

2.2 Embedding Layers/CNN

Embedding Layers では，入力文 x に含まれる各単語 $(x_1, ..., x_M)$ を分散表現に変換する処理と，依存構造グラフの隣接行列 $\boldsymbol{A}$ の各要素を分散表現に変換する処理を行う．単語の分散表現を得るために，単語埋め込み行列 $\boldsymbol{E}_w$ により分散表現 $\boldsymbol{w}_i \in \mathbb{R}^{d_w}$ を得る．隣接行列 $\boldsymbol{A}$ は，単語 x_i と x_j の依存関係ラベルを示す要素 A_{ij} を依存関係ラベル埋め込み行列 $\boldsymbol{E}_r$ により分散表現 $\boldsymbol{r}_{ij} \in \mathbb{R}^{d_r}$ に変換する．最終

[1] spaCy: https://spacy.io/

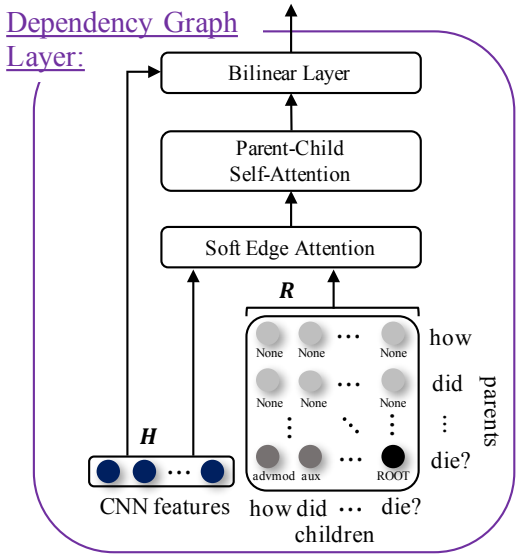

図 2: Dependency Graph Layer の構成図．$\boldsymbol{R}$ 上の埋め込み表現 “None” は親子間に接続がない状態，“ROOT” は ROOT の係り先単語であることを表す．

的に，入力文 x は二階テンソル $\boldsymbol{W}_w \in \mathbb{R}^{d_w \times M}$，隣接行列 $\boldsymbol{A}$ は三階テンソル $\boldsymbol{R} \in \mathbb{R}^{d_r \times M \times M}$ として表現される．

続いて，局所的な系列情報を考慮するために，$\boldsymbol{W}_w$ に対して Convolutional Neural Network(CNN) を適用して隠れ層 $\boldsymbol{H} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_M\} \in \mathbb{R}^{d_h \times M}$ を得る．ここで，d_h は畳み込みフィルターの枚数を表す．

2.3 Dependency Graph Layer

図 2 に示す Dependency Graph Layer は，単語の系列情報 $\boldsymbol{H}$ およびグラフ構造情報 $\boldsymbol{R}$ を入力として，系列情報と構文情報を相補的に計算する．具体的には，以降の節に記述する Soft Edge Attention，Parent-Child Self-Attention，Bilinear Layer により計算される．

2.3.1 Soft Edge Attention

提案手法の依存構造グラフは，単語 x_i と x_j 間の依存関係ラベルをエッジとしたグラフとして表現される．しかし，構文解析をもとに構築した依存構造グラフのエッジのパターンが少ないことから，そのまま Self-Attention を適用してもほとんどのエッジの重みは一様となってしまう．そこで，事前に単語 x_i が x_j の親である確率を系列情報 $\boldsymbol{H}$ から計算する．具体的には，Hashimoto ら [3] の手法をもとに，単語 x_i が x_j の親であるかどうかを示す確率的なエッジ $p(i|j)$ を

$$p(i|j) = \frac{\exp(m(j,i))}{\sum_{k,k\neq j} \exp(m(j,k))} \tag{1}$$

$$m(j,i) = \boldsymbol{h}_j^T (\boldsymbol{W}_p \boldsymbol{h}_i) \tag{2}$$

のように計算する．ここで，$\boldsymbol{W}_p$ は重み行列である．最終的に，単語 x_i と x_j のエッジを $\boldsymbol{e}_{ij} = p(i|j)\boldsymbol{r}_{ij}$ と計算し，次節の Parent-Child Self-Attention への入力とする．

2.3.2 Parent-Child Self-Attention

親ノード P_i に関して接続可能性のあるすべての子ノードを考慮した表現 $\boldsymbol{g}_{P_i}$ を Self-Attention を用いて計算する．このとき $\boldsymbol{g}_{P_i}$ は，依存構造グラフ $\mathcal{G}$ 上の親ノード P_i に関して接続しうるすべての子ノード $C_t \in \{C_1, ..., C_M\}$ を結ぶエッジ e_{it} の接続重みを考慮して

$$\boldsymbol{g}_{P_i} = \sum_t \alpha_{it} \boldsymbol{e}_{it} \tag{3}$$

$$\alpha_{it} = \frac{\exp(\boldsymbol{v}_\alpha^T \boldsymbol{u}_{it})}{\sum_k \exp(\boldsymbol{v}_\alpha^T \boldsymbol{u}_{ik})} \tag{4}$$

$$\boldsymbol{u}_{it} = \tanh(\boldsymbol{W}_\alpha \boldsymbol{e}_{it} + \boldsymbol{b}_\alpha) \tag{5}$$

のように計算される．ここで，$\boldsymbol{W}_\alpha$ は重み行列，$\boldsymbol{v}_\alpha$ は重みベクトル，$\boldsymbol{b}_\alpha$ はバイアス項である．

同様に，子ノード C_j に関して接続可能性のある親を考慮した表現 $\boldsymbol{g}_{C_j}$ を Self-Attention を用いて計算する．最終的に，単語 x_i に関する構文情報素性を $\boldsymbol{g}_i = [\boldsymbol{g}_{P_i}; \boldsymbol{g}_{C_i}]$ とし，$\boldsymbol{G} = \{\boldsymbol{g}_1, ..., \boldsymbol{g}_M\} \in \mathbb{R}^{2d_r \times M}$ を次節の Bilinear Layer への入力とする．なお，; はベクトルの結合を表す．

2.3.3 Bilinear Layer

構文構造を考慮するために，以下の式に従い系列情報と構文情報の両方を考慮した行列 $\boldsymbol{S} \in \mathbb{R}^{d_h \times 2d_r}$ を計算する．最終的に，$\boldsymbol{S}$ の全ての行 $(1 \leq k \leq d_h)$ に対して Max Pooling を適用することで，最も関連した構文情報を含む系列情報素性 $\boldsymbol{s} \in \mathbb{R}^{d_h}$ を得る．なお，$\boldsymbol{W}_d$ は重み行列である．

$$\boldsymbol{S} = \tanh(\boldsymbol{H}\boldsymbol{W}_d\boldsymbol{G}^T) \tag{6}$$

$$s_k = \max_{1 \leq l \leq 2d_r} [\boldsymbol{S}_{k,l}] \tag{7}$$

2.4 目的関数

本研究では，Pair-wise ランキング学習に基づきネットワーク全体の最適化を行う．また，質問 q と回答候補 a の正解らしさを表すスコアは，$\boldsymbol{s}_q$ と $\boldsymbol{s}_a$ のコサイン類似度 $sim(\boldsymbol{s}_q, \boldsymbol{s}_a)$ により計算する．このとき，提案手法の目的関数は，

$$L = \max\{0, m - sim(\boldsymbol{s}_q, \boldsymbol{s}_{a+}) + sim(\boldsymbol{s}_q, \boldsymbol{s}_{a-})\} \tag{8}$$

と定義される．なお，m はマージン，$\boldsymbol{s}_q$，$\boldsymbol{s}_{a+}$，$\boldsymbol{s}_{a-}$ は式 7 より得られた質問，正解の回答，不正解の回答の素性である．不正解の回答は，学習時の各バッチごとに 50 個の負例を回答候補全体からネガティブサンプリングした後，質問 q に対して最も高いスコアを持つものを選んだ．

3 実験

3.1 実験設定

提案手法の性能を評価するために，質問と回答候補の両方が文で構成されたデータセットである WikiQA[11] を使用した．WikiQA のデータセットの統計量を表 1 に示す．なお，評価には MAP(Mean Average Precision) と MRR(Mean Reciprocal Rank) を使用した．

先行研究 [11] に従い，提案手法では入力文の単語数の上限を 40 に設定した．依存構造グラフは，2.1 節に記述した

表 1: WikiQA データセットの内訳

	質問数	平均語数 (Q)	平均語数 (A)
学習	873	6.43	26.04
開発	126	6.86	25.14
評価	243	6.50	25.49

方法に従い構築した．グラフのエッジには，構文解析器によって得られる 46 種類の依存関係ラベルを利用した．ただし，単語 x_i と x_j に依存関係ラベルが付与されない場合は “None”，x_i と x_j がともに上限単語数以下かつ文末以降のノードである場合は “EOS”，x_i が ROOT の係り先である場合は自己参照している要素に “ROOT” というラベルをそれぞれ付与した．単語の分散表現は 300 次元，依存構造グラフのエッジの分散表現は 40 次元，畳み込みフィルターの数は 1500 に設定した．また，畳み込みを行う際のウィンドウサイズを 5 に設定した．先行研究 [1] に従い，バッチサイズは 20，目的関数のマージン m は 0.1 に設定した．

学習には Adam を用いて，学習率の初期値は 0.001 に設定した．また，過学習を避けるため，ドロップアウトを 0.5 の割合で CNN と Dependency Graph Layer に適用した．学習に際し，開発データにおける MAP，MRR のいずれかが 2 エポック以上向上しなかった場合，モデルの学習を打ち切り，最後に更新したモデルを用いて評価データによる評価を行った．

3.2 比較手法

実験では，提案手法を以下の既存手法と比較した．なお，QA-CNN は提案手法の Dependency Graph Layer の効果を比較するために，3.1 節に記述した実験設定に従い評価した．

- **QA-CNN**: 提案手法の Dependency Graph Layer を除外したベースライン [1].
- **AP-CNN**: Santos ら [1] の質問回答間の Attention を計算する手法.
- **NTI**: 木構造に基づく LSTM により回答選択する手法 [6].
- **MVFNN**: 質問回答間の Attention だけでなく，質問タイプ，主動詞，質問のコンテキストに対する Attention を利用した手法 [9]．現状の回答選択タスクにおいて最高精度を示している．
- **MVFNN(w/o Co-attn)**: Sha ら [9] の手法のうち，質問回答間の Attention を除外した手法.

提案手法の Dependency Graph Layer の効果を比較するために，QA-CNN をベースラインとして用意した．さらに，既存研究の構文情報を用いた手法として，NTI および MVFNN(w/o Co-attn)，質問回答間の Attention を計算する手法として AP-CNN と MVFNN を用意した．

3.3 実験結果

表 2 に実験結果を示す．提案手法はベースラインである QA-CNN と比較して各スコア 6pt 以上の改善が見られる．また，質問回答間の対応関係を Attention で計算するモデルである AP-CNN と比較しても，提案手法の結果が良いことが確認できる．さらに，構文情報を考慮した手法である NTI や，複数の Attention を利用した手法である MVFNN(w/o Co-attn) の性能も上回っていることが分かる．しかしながら，現状最高精度を示している MVFNN には及ばない結果であった．原因として，提案手法では質問回答間の Attention を導入していないことが考えられる．

表 2: WikiQA における評価実験の結果．* は質問回答間の Attention を用いたモデル，† は構文情報を用いたモデルを表す．

	MAP	MRR
QA-CNN	0.6563	0.6697
NTI†	0.6742	0.6884
MVFNN† (w/o Co-attn)	0.7018	0.7130
提案手法 †	<u>0.7202</u>	<u>0.7326</u>
AP-CNN*	0.6886	0.6957
MVFNN*†	**0.7462**	**0.7576**

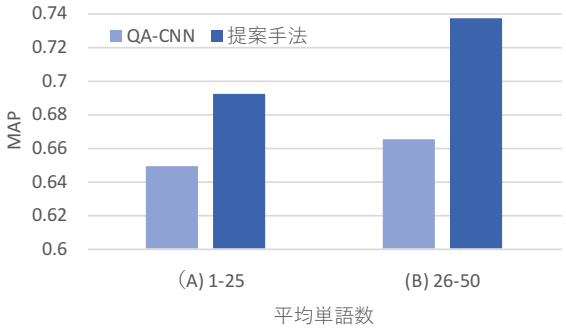

図 3: 短文および長文に対する回答選択の性能

3.4 分析

3.4.1 長文に対する回答選択の性能

回答候補の長さが回答選択に与える影響を調査するために，表 1 の評価データを回答候補の平均単語数が (A)1〜25 単語，(B)26〜50 単語で構成された 2 種類の問題に分割した．分割したデータセットにおける評価対象の質問数は，(A) の条件で 113 件，(B) の条件で 130 件である．

図 3 に評価結果を示す．(A) の短い回答候補を持つ問題においては，提案手法が QA-CNN と比較して 4.36pt ほど高い評価値を得た．

一方，(B) の長い回答候補を持つ問題においては，提案手法が QA-CNN と比較して 7.25pt ほど高い結果を示している．この結果から，提案手法では特に長文に対して上手く正解を得ることが出来ると分かる．

3.4.2 依存構造グラフの可視化

学習した Dependency Graph Layer の接続重み α(式 4) は，依存構造グラフ上のノードを接続するエッジの重要度を表す．構文解析器によって得られた構文解析結果の例および学習した接続重み α の例を図 4 に示す．

この例では，“who is the founder of twitter” という質問に対する回答 “twitter was created in march 2006 by jack dorsey and by july, the social networking site was

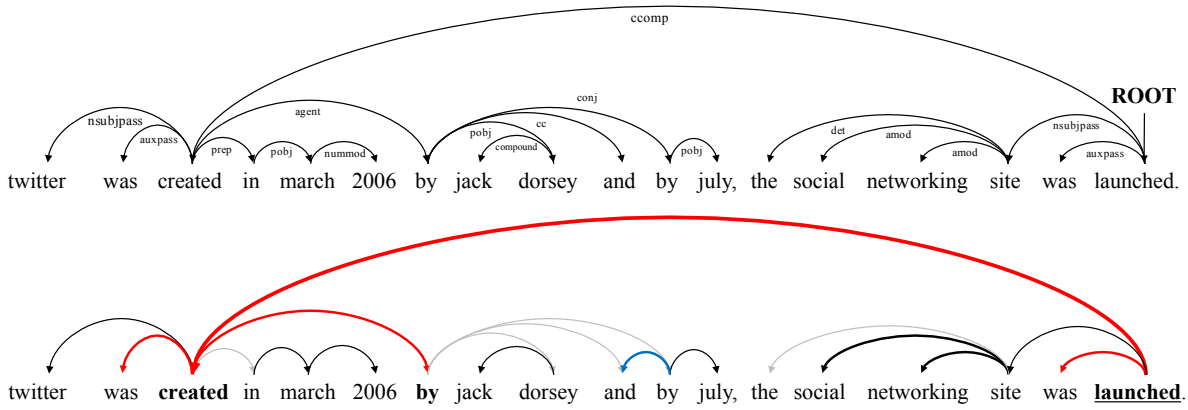

図 4: 上: 構文解析器による構文解析結果の例．下: Dependency Graph Layer の接続重み α の可視化．構文解析器により得られたエッジのうち，接続重み α が平均値以上のエッジは黒色，特に接続重みが強いエッジを赤色，重みが平均値未満のエッジを灰色で示している．また，構文解析結果には出現しておらず，重みが平均値以上だったエッジを青色で示している．

launched.” を可視化している．上図と下図を比較すると，接続重みは構文解析結果を再現しつつ，その中で回答選択に必要なパスに強い重みが付与されていることが分かる．実際に，ROOT の係り先である “launched” からは “created”，さらに “created” からは “by” への接続重みが強く付与されていた．これは，我々が回答選択する際に手がかりとするような重要な箇所とも直感的に合致する．

4 関連研究

本論文では，回答選択タスクにおいて長文に対する精度が低下する問題 [1, 10] に焦点を当てて議論した．長文に対する頑健性は，文のペア関係を理解するためのタスクにおいて重要な課題であると言える．

長文に対して高い精度を得るための方法として構文情報を用いることが挙げられるが，1 節で述べたアルゴリズム [4] の他に，動的に構文構造を学習する手法もいくつか存在する．Niculae ら [7] は入力文に対する候補木集合を推定し，すべての候補木に対して TreeLSTM を適用することで，動的に依存木構造を計算するモデルを提案した．Du ら [2] は係り先，係り元，依存関係ラベルの三つ組から構成される埋め込み表現を用いて，含意判定タスクの既存研究 [8] を拡張した．いずれも動的に構文構造を学習出来るアルゴリズムではあるが，含意判定タスクを対象としており，回答選択は対象としていない．一方で，含意判定で最高精度を示すアルゴリズムが回答選択では必ずしも高い精度を得られないことも検証されており [5]，本研究では回答選択タスクに焦点を当てて文の依存構造を考慮する手法を提案した．

5 おわりに

本論文では，依存構造グラフを用いて文の依存構造を考慮しながら回答選択するニューラルネットワークを提案した．WikiQA を用いた評価では，ベースラインから大幅に性能改善するとともに，いくつかの構文情報を考慮する手法に対しても良い結果が得られることが示された．特に，長文に対する回答選択において顕著に性能向上した．

今後の課題として，質問回答間の対応関係を考慮できるように拡張することが挙げられる．特に，構文構造を考慮した上で質問回答間の重要な箇所を注目させるような手法を検討したい．

参考文献

[1] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, Vol. abs/1602.03609, , 2016.

[2] Qianlong Du, Chengqing Zong, and Keh-Yih Su. Adopting the word-pair-dependency-triplets with individual comparison for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 414–425. Association for Computational Linguistics, 2018.

[3] Kazuma Hashimoto and Yoshimasa Tsuruoka. Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 125–135. Association for Computational Linguistics, 2017.

[4] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 633–644. Association for Computational Linguistics, 2014.

[5] Wuwei Lan and Wei Xu. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3890–3902. Association for Computational Linguistics, 2018.

[6] Tsendsuren Munkhdalai and Hong Yu. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 11–21. Association for Computational Linguistics, 2017.

[7] Vlad Niculae, André F. T. Martins, and Claire Cardie. Towards dynamic computation graphs via sparse latent structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 905–911. Association for Computational Linguistics, 2018.

[8] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255. Association for Computational Linguistics, 2016.

[9] Lei Sha, Xiaodong Zhang, Feng Qian, Baobao Chang, and Zhifang Sui. A multi-view fusion neural network for answer selection. In *AAAI*, 2018.

[10] Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, Vol. abs/1511.04108, , 2015.

[11] Yi Yang, Scott Wen-tau Yih, and Chris Meek. Wikiqa: A challenge dataset for open-domain question answering. ACL - Association for Computational Linguistics, 2015.