知識グラフエンベディングのためのリレーションパスルールによるトランスダクティブデータ拡張

廣瀬雄士 [1]　新保仁 [2]　渡辺太郎 [1]

[1] 奈良先端科学技術大学院大学 先端科学技術研究科，[2] 千葉工業大学 人工知能・ソフトウェア技術研究センター

hirose.yushi.hx4@is.naist.jp,shimbo@stair.center,taro@is.naist.jp

1 はじめに

実世界の情報をデータ化する手法として知識グラフが知られている．知識グラフはエンティティを頂点とし，リレーションを辺のラベルとする有向グラフである．辺を表すヘッドエンティティ，リレーション，テールエンティティの三つ組 (e_h, r, e_t) は，トリプレットと呼ばれ，知識グラフはトリプレットの集合ともみなせる．知識グラフの例として，YAGO [1], Freebase [2]，WordNet [3] があり，Q & A [4]，エンティティリンキング [5] などの自然言語処理タスクで用いられている．

通常，知識グラフは実世界の全てのトリプレットが記述されているわけではない．欠損したトリプレットを予測するタスクを，知識グラフ補完と呼び，主要な手法として，エンベディングモデル [6], [7], [8] とリレーションパスルールモデル [9] が研究されてきた．エンベディングモデルは既知の知識グラフからエンティティ，リレーションの特徴ベクトルを学習し，それらの間の演算により得られるスコアによって欠損したトリプレットの真理値を予測する．リレーションパスルールモデルは，知識グラフ上のエンティティ間のリレーション系列 (リレーションパス) を一階述語論理的なルールと考えて，欠損トリプレットの真理値を予測するモデルである．

両者には長所と短所がある．エンベディングモデルは，知識グラフ全体から特徴を学習できるが，知識グラフの経路情報のモデル化が明示的に行われていない．リレーションパスルールモデルは，ルールを用いているので偽陽性が少ないが，エンティティ間にリレーション系列が存在しなければそのトリプレットを予測することはできない．

本研究では両者の長所を活かすように，リレーションパスルールによって予測されたトリプレットを訓練データに追加し，その拡張データを用いてエンベディングモデルの学習を行う．データの拡張はトランスダクティブ学習 [10] の設定で行う．これによりクエリの予測に有益なデータを拡張することで精度の向上が期待できる．

データセットとして WN18RR，FB15k-237 を用い，通常のデータを用いてエンベディングモデルを学習させた場合と比較すると，拡張データを用いた場合の方が予測精度が良い結果が得られた．通常の知識グラフ補完のためにトランスダクティブ学習の設定を用い，予測精度への効果を調べたのは，私の知る限り本研究が初めてである．

2 背景

知識グラフ補完は，テストクエリ $(e_h, r, ?), (?, r, e_t)$ が与えられた時，? に入る真のエンティティが，エンティティ集合 E の中で何番目に真である可能性が高いかを予測するタスクとしてよく定式化される．本研究も同様なタスクを考える．

2.1 エンベディングモデル

エンベディングモデルはエンティティとリレーションの特徴ベクトルを用いた演算によってトリプレットのスコアが定義されるモデルであり，その値が大きいトリプレットほど，真である可能性が高いと考える．各特徴ベクトルは，通常確率的勾配降下法によって学習される．初期モデルとして RESCAL [6] が知られており，近年精度が高いモデルとして TuckER [8] が知られている．各モデルのスコア関数 f_{RESCAL} および f_{TuckER} は以下のように表される．

$$f_{\mathrm{RESCAL}}(e_h, r, e_t) = \mathbf{e}_h^\top \mathbf{R} \mathbf{e}_t \tag{1}$$

$$f_{\mathrm{TuckER}}(e_h, r, e_t) = \mathbf{W} \times_1 \mathbf{e}_h \times_2 \mathbf{r} \times_3 \mathbf{e}_t \tag{2}$$

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

ここで $\mathbf{e}_h, \mathbf{e}_t, \mathbf{r}, \mathbf{R}, \mathbf{W}$ はそれぞれ学習されるパラメータを示し，順にヘッド，テールエンティティ，リレーションのエンベディングベクトル，リレーション行列，全トリプレット共通のコアテンソルを表す．$\times_n$ はテンソルの n-mode 積を表す．

2.2 リレーションパスルール

リレーションの系列をリレーションパスと呼ぶ．すなわち $r_1, \ldots, r_n$ が n 個のリレーションであるとき，$path_i := (r_1, \ldots, r_n)$ は長さ n のリレーションパスである．また，$(e_1, r_1, e_2), (e_2, r_2, e_3), \ldots, (e_n, r_n, e_{n+1})$ が知識グラフ中のトリプレットであるとき，e_1 と e_{n+1} の間に $path_i$ が存在する，といい，$path_i(e_1, e_{n+1})$ と書く．

リレーションパスを用いた予測の例としてクエリ (Bob, nationality, ?) を考える．(Bob, bornIn, NYC), (NYC, isCityOf, USA) が知識グラフ上で既知なら，Bob,USA の間に (bornIn, isCityOf) というリレーションパスが存在することを用いて，USA がクエリの真のエンティティに含まれると予測することができる．

リレーションパス $path_i = (r_1, r_2, ..., r_n)$ を用いてリレーション r を予測するとき，述語論理を用いて次のようなルールとしてみることもできる．

$$\exists e_1, e_2, \ldots e_{n+1} : (e_1, r_1, e_2) \wedge (e_2, r_2, e_3) \wedge \ldots \wedge (e_n, r_n, e_{n+1}) \rightarrow (e_1, r, e_{n+1}) \tag{3}$$

ルールの尤もらしさは，先行研究 [11] の Partial Completeness Assumption（PCA）という仮定に基づくと，式のような確信度スコアで示される．式はリレーション r の予測に用いる $path_i$ の確信度スコアを表し，e_t' は知識グラフ上のいずれかのエンティティを表す．

$$\mathrm{Conf}_{\mathrm{PCA}}(path_i) = \frac{|\{(e_h, e_t) \mid path_i(e_h, e_t) \wedge (e_h, r, e_t)\}|}{|\{(e_h, e_t) \mid path_i(e_h, e_t) \wedge (e_h, r, e_t')\}|} \tag{4}$$

2.3 データ拡張による知識グラフ補完

リレーションパスルールを用いてデータ拡張を行い，エンベディングモデルの学習に用いるモデルには UniKER [12] が存在する．この手法により，エンベディングモデルの精度は大きく向上するが，データ拡張のために様々な手法が必要である．例えば，拡張するトリプレットをうまく見つけるために，“forward chaining”やパスの補完などがある．またこの手法ではエンベディングスコアもデータ拡張のための情報として用いている．そしてエンベディングの学習→データの拡張→拡張された新たなデータによるエンベディングの学習，というように繰り返しエンベディングモデルを学習する必要がある．

3 トランスダクティブ学習設定によるデータ拡張

トランスダクティブ学習設定では，テストクエリ $(e_h, r, ?)$ が学習時に既知であると考える．このため、テスト時にあらゆるクエリを想定しないといけない通常の知識グラフ補完タスクよりは容易な問題設定になっている．しかし，現実的には予測したいクエリがあらかじめ分かっていて，その予測精度を向上することの方が求められることも考えられる．

本手法ではテストデータのクエリ $(e_h, r, ?)$ が与えられたとき，テールとして該当しそうなエンティティ e_t' をリレーションパスルールによって予測し，トリプレット (e_h, r, e_t') を訓練データに追加する．そうすることでクエリの答えであるテールエンティティか，それと似た特徴を持つテールエンティティを訓練データに追加することが期待できる．その後，拡張データを用いてエンベディングモデルを訓練し，テストの評価に用いる．拡張データで学習されたエンベディングモデルを拡張データモデルと呼ぶこととする．

3.1 リレーションパスルールのマイニング

リレーションパスルールは先行研究 [13] の手法に倣って双方向ランダムウォークによって取得する．トリプレット (e_h, r, e_t) について考える．このマイニング手法では，e_h, e_t それぞれを始点としたグラフ上のランダムウォークが行われ，ウォークしたお互いの終点が同じエンティティだった場合に，リレーション r の予測に用いられるパスが取得される．

各リレーションに対して，マイニングされたパスのうち，確信度スコアがトップ N 個のパス $path^r_{i=1\ldots N}$ を用いてデータ拡張を行う．テストデータの各クエリ $(e_h, r, ?)$ に対して，全テールエンティティ $e_t' \in E$ を用いて，e_h と e_t' の間に各 $path^r_{i=1\ldots N}$ が存在するかを調べる．存在するパスの中で，式 4 の確信度スコアが最も大きいものを (e_h, r, e_t') のパススコアとし，各クエリのパススコアが大きいトリプレットを訓練データに追加する．このとき $topS$ と $confThold$ という閾値を導入する．各クエリに対して，最初の訓練データを含めて多くとも $topS$

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

表 1　知識グラフの各データセット統計情報

データセット	$\|E\|$	$\|R\|$	train	valid	test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

E, R はエンティティ，リレーションの集合を表す．

個までのトリプレットを訓練データに追加する．また $scoreThold$ より小さいパススコアを持つトリプレットは訓練データに追加しないようにする．

4　実験

4.1　実験設定

データセットには WN18RR [14] と FB15k-237 [15] を用いた．データセットの統計情報は表 1 に示す．

データ拡張には各リレーションで確信度スコアトップ 1000 までのパスを用いる．またハイパーパラメータを検証用データで調整するために，テストデータだけでなく検証用データもデータ拡張のためのクエリとして用いた．エンベディングモデルには RESCAL,TuckER を用い，そのハイパーパラメータ等の値は付録に示す．ハイパーパラメータの探索範囲は，ベースモデルと拡張データモデルで等しく，検証用データの MRR が最大になるようグリッドサーチする．

また各データ (e_h, r, e_t) に対して逆リレーション r^{-1} を用いた (e_t, r^{-1}, e_h) を追加する．そして学習と評価は各トリプレットのテールエンティティを予測する手法を用いる [16].

4.1.1　損失関数

損失関数には KvsAll [17] という負例の取り方と，二値交差エントロピー損失を用いる．このときミニバッチ内の一つの訓練事例は (e_h, r) となり，このペアに対する損失関数は以下のように表される．

$$L = -\frac{1}{|E|}\sum_{e' \in E}^{|E|}(\mathbf{y}_{e'} \log \sigma\left(f(e_h, r, e')\right) + (1 - \mathbf{y}_{e'}) \log\left(1 - \sigma\left(f(e_h, r, e')\right)\right)) \tag{5}$$

ここで $\mathbf{y}_{e'}$ は，トリプレット (e_h, r, e') が訓練データに含まれるなら 1，それ以外は 0 となるラベルを表し，f は学習に用いるスコア関数，σ は活性化関数であるシグモイド関数を表す．

4.2　実験結果

ベースモデルと拡張データモデルによるテストクエリの予測結果を表 2 に示す．MRR は予測された真のエンティティのランクの逆数の平均を表し，Hits@K は真のエンティティのランクが K 位内であるテストクエリの割合を表す．

表 2　拡張データモデルの実験結果

		WN18RR		FB15k-237	
	モデル	MRR	Hits@10	MRR	Hits@10
RESCAL	ベース	0.462	0.518	0.350	0.536
	拡張データ	**0.485**	**0.565**	**0.352**	**0.538**
TuckER	ベース	0.464	0.517	**0.354**	**0.540**
	拡張データ	**0.495**	**0.567**	**0.354**	**0.540**

5　分析・考察

5.1　予測精度に関する考察

表 2 の実験結果から，拡張データがエンベディングモデルの精度を向上させることがわかる．また，FB15k-237 で拡張データモデルがベースモデルに対してあまり精度の改善が見られなかった原因の一つに，適切なハイパーパラメータを設定できなかった可能性が考えられる．拡張データによって訓練データの数が変化するので，最適なエンベディングの次元やドロップアウトの値が変化したと思われる．

5.2　予測結果の違いに関する分析

一つのテストデータに対して，そのクエリによって生成される拡張データを，そのテストデータの拡張データと呼ぶことにする．

拡張データに真またはそれに類似したテールエンティティを含んだ結果，予測が改善されたのか分析する．ベースモデル・拡張データモデルによって予測される各検証用データのランクを調べ，それぞれのモデルで 10 位以内かそれ未満で予測されたデータ数と，その内拡張データを持つもの，拡張データに含まれるものを示したのが，表 3,4 である．エンベディングモデルには RESCAL を用いた．

まず WN18RR について表 3 のデータをみる．拡張データに含まれる検証用データは全て，拡張データモデルで 10 位以内に予測できている．そのうち 93 個はベースモデルでは 10 位以内に予測できなかったもので，拡張データモデルの予測精度が向上に寄与している．

FB15k-237 も WN18RR と同じような傾向が見られるが，拡張データモデルによって新たに 10 位内に予測できるようになったトリプレットの数が，予測できなくなった数と同じくらい存在する．

両方のデータセットで拡張データを持つ検証用

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

表 3 各モデルによる検証用データの予測ランク比較（WN18RR）

		ベースモデルランク					
		10 位以内			10 位未満		
		全データ	have 拡張データ	in 拡張データ	全データ	have 拡張データ	in 拡張データ
拡張モデルランク	10 位以内	3,013	2,532	2,408	375	185	93
	10 位未満	114	61	0	2,566	1,144	0

表 4 各モデルによる検証用データの予測ランク比較（FB15k-237）

		ベースモデルランク					
		10 位以内			10 位未満		
		全データ	have 拡張データ	in 拡張データ	全データ	have 拡張データ	in 拡張データ
拡張モデルランク	10 位以内	17,642	5,592	4,320	1,532	141	57
	10 位未満	1,280	126	0	14,616	537	3

表中の“have 拡張データ”は拡張データを持つ検証用データ，“in 拡張データ”は拡張データに含まれる検証用データを示す．

データのうち，そのデータ自身が拡張データに含まれなくても精度が上昇したものが存在する．この原因が，拡張データにクエリの真のテールエンティティに類似したエンティティを含んだためかどうか次の節で確かめる．

5.3 エンベディングコサイン類似度比較

以下のように検証用データと拡張データのテールエンティティのエンベディングコサイン類似度を分析した．検証用データ (e_h, r, e_t) の拡張データのテールエンティティの集合を $D(e_h, r)$ とする．$D(e_h, r)$ の各テールエンティティ e'_t に対して，E 中の全エンティティとのエンベディングのコサイン類似度を求める．各 e'_t について e'_t と e_t のコサイン類似度が E 中で何番目か求める．各 e'_t と e_t のコサイン類似度ランクの中で最も小さいものを，$D(e_h, r)$ と e_t との最小コサイン類似度ランクとする．

検証用データで拡張データに含まれない，かつ拡張データ持ち検証用データのうち，以下 3 タイプのデータを比較のために取り出す．

- タイプ 1：10 位以内の予測精度が向上したデータ（予測ランクがベースモデルで 10 位未満かつ拡張データモデルで 10 位以内のデータ）
- タイプ 2：10 位以内の予測精度が悪化したデータ（予測ランクがベースモデルで 10 位以内かつ拡張データモデルで 10 位未満のデータ）
- タイプ 3：10 位以内に変わらず予測できていないデータ（予測ランクがベースモデル，拡張データモデル共に 10 位未満のデータ）

各タイプに含まれる検証用データそれぞれについて，その拡張データのテールエンティティとの最小コサイン類似度ランクを求める．その後各タイプごとに，検証用データの最小コサイン類似度ランクの MRR を求める．このようにして求められた 3 タイプの MRR を，各タイプに含まれる検証用データ数と共に表 5 に示す．

表 5 3 タイプの検証用データの最小コサイン類似度 MRR

	WN18RR		FB15k-237	
	類似度 MRR	データ数	類似度 MRR	データ数
タイプ 1	**0.139**	92	**0.127**	84
タイプ 2	0.0303	61	0.0392	126
タイプ 3	0.0169	1144	0.0422	534

両データセット共にタイプ 1 が最も最小コサイン類似度 MRR が高く，類似したエンティティの拡張データの学習が，拡張データモデルの精度向上に寄与したと考えられる．

6 おわりに

トランスダクティブ設定の下で，リレーションパスによるデータ拡張を行い，エンベディングモデルの学習と予測を行った．予測結果は拡張データなしの場合を上回り，分析の結果，多くの場合トランスダクティブ学習の設定がより良いデータの拡張につながることがわかった．また本手法は他のエンベディングモデルへの応用も期待できる．

トランスダクティブではなく，通常の設定でのリレーションパスルールによるデータ拡張は今後の課題とする．

参考文献

[1] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, 2007.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, 2008.

[3] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, Vol. 38, No. 11, p. 39–41, November 1995.

[4] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, p. 105–113, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Deep semantic match model for entity linking using knowledge graph and text. *Procedia Computer Science*, Vol. 129, pp. 110 – 114, 2018.

[6] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, Vol. 11, pp. 809–816, 2011.

[7] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[8] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Empirical Methods in Natural Language Processing*, 2019.

[9] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference*, pp. 3–20. Springer, 2018.

[10] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, p. 148–155, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[11] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, Vol. 24, No. 6, pp. 707–730, 2015.

[12] Ke wei Cheng, Ziqing Yang, Ming Zhang, and Y. Sun. Uniker: A unified framework for combining embedding and horn rules for knowledge graph inference. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[13] Matt Gardner and Tom Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1488–1498, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[14] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings, 2018.

[15] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[16] Timothee Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 2863–2872, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[17] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

A 付録

A.1 マイニングされたパス数

本研究で用いたマイニングされたパスの各リレーションごとの平均数・標準偏差を表に示す．

表 6 マイニングされたパス数

データセット	WN18RR	FB15k-237
各リレーションの取得パス数の平均・標準偏差	28k ± 26k	29k ± 14k

A.2 エンベディングモデル実験設定とハイパーパラメータ

エンベディングモデル実験設定とハイパーパラメータの値を表 7 に示す．表中の{}内の値は探索に用いた値である．

表 7 エンベディングモデルのハイパーパラメータ

データセット	WN18RR	
モデル	RESCAL	TuckER
最適化アルゴリズム	Adam [18]	
学習率	{0.005,0.003,0.001}	{0.01, 0.005, 0.003, 0.001, 0.0005}
学習率の減衰率	{1, 0.995, 0.99}	
入力，隠れ層 1，隠れ層 2 ドロップアウト	(0.2,0.2,0.3)	
バッチサイズ	128	
エンティティエンベディング次元	200	
リレーションエンベディング次元	200^2	30
ラベルスムージング	0.1	
エンベディング初期化方法	Uniform[-0.31,0.31]	Xavier normal [19]
最大エポック数	500	

データセット	FB15k237	
モデル	RESCAL	TuckER
最適化アルゴリズム	Adam	
学習率	{0.003, 0.001, 0.0005}	{0.01, 0.005, 0.003, 0.001, 0.0005}
学習率の減衰率	{1, 0.995, 0.99}	
入力, 隠れ層 1, 隠れ層 2 ドロップアウト	(0.3,0.4,0.5)	
バッチサイズ	128	
エンティティエンベディング次元	200	
リレーションエンベディング次元	200^2	200
ラベルスムージング	0.1	
エンベディング初期化方法	Xavier normal	
最大エポック数	500	

A.3 データ拡張の域値

データ拡張の域値 $topS, scoreThold$ は，WN18RR で $topS = 5, scoreThold = 0.1$，FB15k-237 で $topS = 5, scoreThold = 0.6$ を用いた．

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).