Hypernym Discovery via a Recurrent Mapping Model

Yuhang Bai
BDBC and SKLSDE
Beihang University, China
baiyh@act.buaa.edu.cn

Richong Zhang*
BDBC and SKLSDE
Beihang University, China
zhangrc@act.buaa.edu.cn

Fanshuang Kong
BDBC and SKLSDE
Beihang University, China
kongfs@act.buaa.edu.cn

Junfan Chen
BDBC and SKLSDE
Beihang University, China
chenjf@act.buaa.edu.cn

Yongyi Mao
School of EECS
University of Ottawa, Canada
ymao@uottawa.ca

Abstract

Hypernym discovery aims to identify all possible hypernyms of a given term. The most recent hypernym discovery models exploit multiple mapping functions to project a term to different semantic spaces and then aggregate these embeddings to a general representation for further classification. We refer to this model as a *parallel style* model. In this work, we observe that there are hierarchical relations between a target terms' hypernyms. However, these hierarchical relations were not sufficiently considered in the previous *parallel style* model. To leverage the hierarchical relations, we propose a *sequential style* model that recurrently maps the query words to their hypernyms, starting from the most specific ones to the less specific ones. Empirical studies on SemEval-2018 Task 9 confirm the effectiveness of the presented model.

1 Introduction

Hypernymy, namely "is-a" relation, is a vital lexical-semantic relation in natural languages, which relates general terms to their instances or subtypes. In a hypernymy relation, we name a specific instance or subtype *hyponym* and its related general term *hypernym*. For instance, (*apple*, *fruit*) is in hypernymy relation, where *apple* is a hyponym and *fruit* is one of its hypernyms. Due to its general representation ability of semantic relations, hypernymy becomes an essential concept in modern natural-language research, and hypernymy detection becomes a fundamental component in many natural language processing (NLP) tasks, such as taxonomy construction (Snow et al., 2006; Navigli et al., 2011), semantic search (Hoffart et al., 2014; Roller et al., 2014; Roller and Erk, 2016), textual entailment (Dagan et al., 2013; Bowman et al., 2015; Yu et al., 2020) and question answering (Yahya et al., 2013; Gupta et al., 2018).

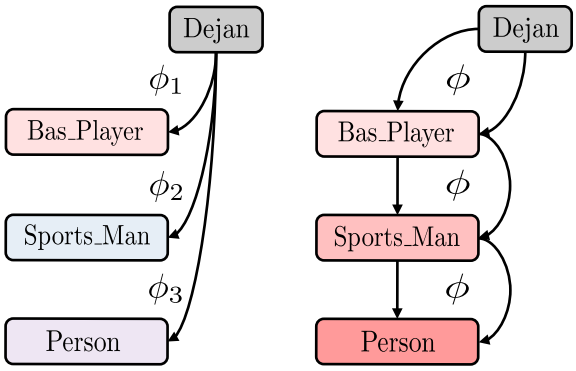


Figure 1: The parallel modeling (left) and sequential modeling (right). The Dejan is the name of a basketball player. The Bas_Player denotes the `Basketball Player` hypernym.

One branch of existing works builds the hypernymy-relation-identification problem as a "detection" task, which is only interested in whether a given term pair "is" or "is not" in hypernymy relation. These works formulate hypernym detection as a binary classification task. This hypernym detection task has been studied for years and plenty of models have been successfully applied in this task (Held and Habash, 2019a; Le et al., 2019).

This paper focuses on another problem named hypernym "discovery", which is different from the detection task. Given an input term, the hypernym discovery task retrieves a ranked list of its suitable hypernyms from a large corpus. For training, some hyponyms with their gold hypernym lists are provided. SemEval-2018 Task 9 (Camacho-Collados et al., 2018) is the only benchmark for this task. Existing studies working on this task mainly build a parallel mapping model (Bernier-Colborne and Barriere, 2018; Fu et al., 2014; Ustalov et al., 2017; Yamane et al., 2016). It introduces mul-

*Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2912–2921
August 1–6, 2021. ©2021 Association for Computational Linguistics

tiple parallel projections, each with independent parameters, to extract features and score the given hyponym/hypernym pair, as shown in the left picture of Figure 1. This model structure is motivated by the assumption that each fine-grained "is-a" relation should be modeled by a specific projection. However, the parallel mapping model may be ineffective due to uncountable "is-a" relation types in real-world applications. It may also suffer from overfitting due to its large model capacity. It also ignores the relations between different projections because they are independently learned.

To overcome the limitations of the parallel mapping model, we propose a recurrent mapping model. Our model is motivated by the observation that a hypernymy term may be produced by the "hypernymy transformation", which transforms a term to its closely related hypernym via a projection, as shown in Figure 1 (right). In the figure, we want to identify hypernyms of the term `Dejan`. The higher-level hypernym `Person` may be transformed from a `Dejan–Bas_Player–Sports_Man–Person` path. The parallel modeling in Figure 1 (left) may not have the ability to capture these sequential relations. Thus we devise a recurrent mapping model for these sequential relations. Note that the projection in our recurrent mapping model is shared among all hops. We assume that a higher-level hypernym term can be generated by operating multiple hypernymy transformations recurrently from the given term. In this way, we build the sequential relations between the transformed terms and largely reduce the parameters used in projection.

We also consider the types of hyponyms when building our recurrent mapping model. As pointed out in previous work (Bernier-Colborne and Barriere, 2018; Camacho-Collados et al., 2018), hyponyms are divided into two types, namely, the concept type and the entity type. This type information is available in the dataset. According to the type of a hyponym, the hypernymy relation can be divided into "subclass-of" (e.g. A guitar is an instrument) and "instance-of"(e.g. Rome is a city). The former represents the hypernymy relation between two concepts, and the latter connects an entity-hyponym with a concept-hypernymy. We first exploit two projections to obtain type-enhanced representation for different types of hyponyms, then feed the type-enhanced representation to a unified recurrent mapping model. In this way, we provide appropriate additional model capacity to handle different types of hyponyms and simultaneously we can utilize all data with hyponyms that belong to both types to train our model.

Our recurrent model outputs a representation vector at each hop. These vectors indicate hypernym representations from different hierarchy levels corresponding to the original hyponym. While scoring a candidate hypernym, we exploit an attention mechanism to aggregate hypernym representations from each hierarchy level. The attention weight of a level is viewed as the probability of the candidate hypernym lying in that level.

In summary, the contributions of this work are as follows:

- We propose a recurrent mapping model that utilizes a shared mapping unit to model the inherent hierarchical dependencies between hypernyms.
- To exploit the hyponym-type information, we use an independent projection matrix for each type to map hyponyms of different types to hypernym space.
- We utilize the attention mechanism into the aggregation module to obtain learnable weight.

2 Related Work

Earlier research on hypernym detection mainly focuses on unsupervised methods, which can be categorized into pattern-based methods and distributional methods. Pioneered by Hearst (1992), the pattern-based methods pre-define some common patterns that indicate hypernym relation, for example, word phrase "such as", "especially". Words occurring together in these pre-defined patterns will be extracted as hyponym/hypernym pairs. This method is quite intuitive. However, one serious problem is sparsity, since many hyponym/hypernym pairs never co-occur explicitly in the corpus, let alone in specific patterns. As a result, this method can provide high accuracy, at the cost of low recall. Seitner et al. (2016) try to improve this method by proposing an extended set of patterns, while (Snow et al., 2004; Shwartz et al., 2016) put forward methods to learn such lexical-syntactic patterns automatically. To further alleviate the sparsity problem, distributional models are proposed. Based on distributional inclusion hypothesis (DIH, Geffet and Dagan (2005)), these

models represent every word as a distributional vector. Hyponym/hypernym pairs that never co-occur in the corpus can be captured based on the relation on their distributional vectors.

Most recent studies on hypernym discovery are supervised methods. Fu et al. (2014) was inspired by the well-known example "$V(king) - V(queen) \approx V(man) - V(woman)$", where V(w) is the embedding of the word w. The authors observed that the same linguistic regularities are preserved between hyponym and hypernyms. Thus they supposed that a hyponym can be projected to its hypernym. Besides the uniform linear projection, they also proposed piece-wise linear projections to model fine-grained hypernymy relations. Ustalov et al. (2017) makes use of negative examples to regularize the model, which further improves the performance of the model. Yamane et al. (2016) jointly learns the clusters and projections, the number of clusters can be determined depending on the learned projections and vice versa.

Some other work (Bernier-Colborne and Barriere, 2018; Held and Habash, 2019b) utilize a mix of both unsupervised and supervised methods. Dash et al. (2020) argued that hypernym relation can be represented as strict partial order relation (transitive, irreflexive and asymmetric) and they introduced a model which takes strict partial order relation as soft constraint.

3 Problem Formulation

Hypernym discovery task aims to identify all hypernym terms of a given hyponym term. Formally, let $\mathcal{V}$ be the set of all terms, and $\mathcal{X}$ and $\mathcal{Y}$ denote the set of all hyponym terms and candidate hypernym terms, respectively. Both $\mathcal{X}$ and $\mathcal{Y}$ are subsets of $\mathcal{V}$. For a given hyponym $x \in \mathcal{X}$, a hypernymy detector is expected to find all hypernym terms $y \in \mathcal{Y}$ that make (x, y) a hypernymy relation.

4 Methodology

In this section, we introduce the proposed Recurrent Mapping Model (RMM). It consists of three components: a type enhanced representation module that maps the hyponym embedding via different projections, a recurrent mapping module that transforms the term features into multiple concept-level semantics and an aggregation module that aggregates hypernym representations from each hierarchy level and compute a final score via the aggregated vector. We next describe each component in detail.

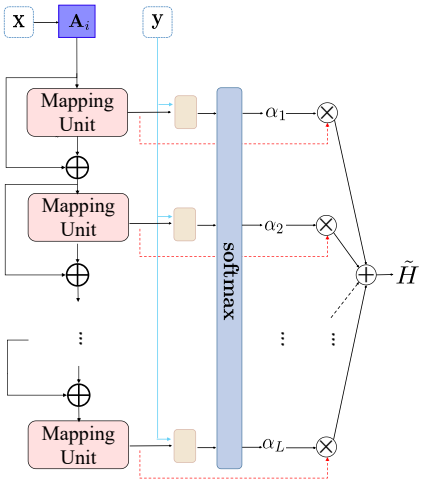


Figure 2: Overview of RMM model.

4.1 Type Enhanced Representation

Type information is essential for hypernym discovery. It implies different hypernym types that provide the hyponyms' attributes. Utilizing type information may increase the hypernym-discovery performance. Specifically, let $\mathbf{x} \in \mathbb{R}^{d\times 1}$ be the embedding of hyponym term x, for its type indexed by i, we introduce a projection $\mathbf{A}_i \in \mathbb{R}^{d\times d}$ to map the hyponym embedding as a type enhanced representation $\mathbf{t} \in \mathbb{R}^{d\times 1}$ as follows:

$$\mathbf{t} = \mathbf{A}_i\mathbf{x} \tag{1}$$

In this way, we provide appropriate additional model capacity to handle different types of hyponyms and simultaneously we can utilize all data with hyponyms that belong to both types to train our model.

4.2 Recurrent Mapping Module

One motivation of this paper is that we assume the candidate hypernym terms may come from different concept levels. The hypernym term at a higher concept-level can be obtained by transforming from a lower-level hypernym term. To model this transformation process, we build a recurrent mapping module with a shared projection function. Specifically, let $\mathbf{h}_l \in \mathbb{R}^{d\times 1}$ denote the transformed concept semantics at the l^{th} level (or the term representation after l-hop transformations). The transformation from a l^{th}-level concept semantics to a

$(l+1)^{\text{th}}$-level is then formulated as

$$\hat{\mathbf{h}}_{l+1} = \mathbf{W}_{\phi}\mathbf{h}_l \tag{2}$$

In Equation 2, the semantic representation of the primary concept level is exactly the type-enhanced representation, that is $\mathbf{h}_0 = \mathbf{t}$. $\mathbf{W}_{\phi} \in \mathbb{R}^{d\times d}$ is the learnable projection matrix which is shared during each transformation. We recurrently make L transformations from the original hyponym embedding. We hope this recurrent mapping process captures the relationships of different concept-levels and associates the given hyponym term with its hypernym terms at different concept levels.

Theoretically, when the number of layers increases in a neural network, its model capabilities become large and it should produce lower training error. However, the gradients may vanish after they are propagated through many layers, thus degrade the model performance (He et al., 2016). Consequently, optimizing the projection matrix in Equation 2 is difficult when the maximum number of transformations L becomes large. Following the previous work (He et al., 2016), we introduce residual network (ResNet) to overcome the gradient vanishing problem as follows:

$$\mathbf{h}_{l+1} = \hat{\mathbf{h}}_{l+1} + \mathbf{h}_l \tag{3}$$

ResNet improves the training efficiency by reformulating the $(l+1)^{\text{th}}$-level representation with reference to the l^{th}-level representation. In addition, the ResNet also forces the model to *remember* the past information during multi-hop transformation.

4.3 Aggregation Module

After obtaining the multi-hop representations of the given hyponym term, we obtain a final representation via letting multi-hop representations attend to the candidate embedding.

Concretely, by calculating dot product between representations from each hop and candidate hypernym, we can get a weight vector. The attention weight of a level is viewed as the probability of the candidate hypernym lying in that level. The weight vector and the final score of the candidate term embedding $\mathbf{y}$ being the input's hypernym are calculated as follows:

$$\begin{aligned} \alpha_l &= \frac{\exp(\mathbf{h}_l \cdot \mathbf{y})}{\sum_{k=1}^{L} \exp(\mathbf{h}_k \cdot \mathbf{y})} \\ \widetilde{H} &= \sum_{l=1}^{L} \alpha_l \cdot \mathbf{h}_l \\ s_y &= \widetilde{H} \cdot \mathbf{y} \end{aligned} \tag{4}$$

This mechanism allows the model to adaptively assign weights for representations from each hop based on the candidate terms. We expect this aggregation strategy to provide appropriate scores that correctly rank the true hypernym terms ahead of other candidate terms.

4.4 Loss Function

Hypernymy discovery is viewed as a ranking problem. Existing models optimize this ranking problem using a pair-wise loss function. They first score the candidate hypernym terms by training a binary classifier that identifies whether a given (x, y) pair supports a hypernymy relation, then rank the candidates by descending order of their matching scores. Although pair-wise loss function is efficient, it learns to score each candidate independently (Shi and Weninger, 2017). Instead of optimizing the model via pairwise loss, we introduce the list-wise loss function that learns to score the candidates collectively.

Let $\mathcal{X}_{train}$ be the set of hyponym terms in the training set. For each hyponym term x in the training set, let $\mathcal{Q}_x$ denote a set of candidate hypernyms, which consists of only gold hypernym terms of x, and $\mathcal{C}_x$ denotes a controllable number of negative candidates. Then we compute the cross-entropy loss with the sampled positive and negative candidates as follows:

$$\mathcal{L} = -\sum_{x\in\mathcal{X}_{train}} \frac{1}{|\mathcal{Q}_x|} \sum_{y\in\mathcal{Q}_x} \log \frac{\exp(s_y)}{\sum_{y'\in\mathcal{Q}_x\cup\mathcal{C}_x}\exp(s_{y'})} \tag{5}$$

where $\frac{1}{|\mathcal{Q}_x|}$ is a normalizing term that balances the learning of all gold candidate scores. The number of negative candidates $|\mathcal{C}_x|$ is optional in practice, we leave it as a hyperparameter. We may set $\mathcal{C}_x = \mathcal{Y} - \mathcal{Q}_x$ with enough computation source available. In this way, we simultaneously optimize the scores for a collection of candidates, improving the training efficiency.

5 Experiments

5.1 Data Sets

We evaluate the performance of our model on SemEval-2018 Task 9 [1]benchmark for hypernym discovery. This shared task consists of five different subtasks covering both general-purpose (multiple languages-English, Italian, and Spanish) and domain-specific (Music and Medicine domains) tasks. For each subtask, a large textual corpus, a vocabulary including all valid hypernyms and a training and testing set of hyponyms and its gold hypernyms are provided. In this paper, we consider the three English subtasks: 1A (general), 2A (medical) and 2B (music). The summarized statistics of the datasets are shown in Table 1. For more details, we refer the reader to the original SemEval-2018 Task 9 (Camacho-Collados et al., 2018) paper.

Three metrics were used for the performance evaluation.

Mean Average Precision (MAP) For a given query word, average precision(AP) is the average of the correctness of each obtained hypernym from the search space. MAP is the mean of this value among all queries in the data set.

Mean Reciprocal Rank (MRR) Since MAP ignores the exact rank of the true hypernyms, we introduce the Mean Reciprocal Rank (MRR) metric which focuses on the top results performance. Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks over all queries. The reciprocal rank of an individual query is the reciprocal of the rank in which the first true hypernym is returned.

Precision at K (P@K) Precision at K is the proportion of the top-K results that are true hypernyms of a given query.

Following the same evaluation procedures as previous studies (Bernier-Colborne and Barriere, 2018; Held and Habash, 2019b; Dash et al., 2020), the scorer script provided by SemEval-2018 Task 9 is exploited for evaluating our proposed model and comparing fairly with other recent models.

5.1.1 Compared Models

We compare our model with baseline models: MFH (Camacho-Collados et al., 2018), vTE (Camacho-Collados et al., 2018), 300-sparsans(Berend et al., 2018), and NLP-HZ (Qiu et al., 2018). 8 SPON (Dash et al., 2020), Hybrid of SVD & NN (Held and Habash, 2019b).

subtask	corpus size	#train	#test
1A	16G	1500	1500
2A	800M	500	500
2B	500M	500	500

Table 1: Data set statistics

Besides, we also compare our model with recent models. Brief descriptions of these models are given as follows.

CRIM (Bernier-Colborne and Barriere, 2018) In the CRIM model, multiple parallel projections are introduced to map the queries to different spaces. A logistic regression function is then applied to compute the final score. In addition, this module is combined with an unsupervised system that identifies hypernym based on specific Hearst-style patterns. The final output of CRIM is the combination of both supervised and unsupervised models.

SPON (Dash et al., 2020) In the SPON model, non-negative activations and residual connections are exploited to enforce asymmetry and transitive as soft constraints.

Hybrid of SVD & NN (Held and Habash, 2019b) This model is a hybrid system which exploits both unsupervised and supervised approaches at the same time.

In their proposed supervised module, the nearest neighbor approach is used. Given a hyponym, candidate hypernyms which are its nearest neighbors are returned. A similarity cut-off point is trained on tuning data, such that if there is no neighbor with a similarity greater than the cutoff point, the model simply returns the most frequent set of hypernyms from the entire training set.

5.2 Implementation

We trained 200-dimensional word embeddings via the standard skip-gram word2vec algorithm (Mikolov et al., 2013) on the provided textual corpus. The representations of the hyponyms and hypernyms are directly initialized by the pre-trained word2vec embeddings. In the training process of our model, hyponym embeddings are fixed and hypernym embeddings are learnable. To avoid overfitting, dropout is applied after each mapping function. Besides, an early stop strategy is also used. Thus, if MAP on the validation set does not

[1]https://competitions.codalab.org/competitions/17119

Model	1A			2A			2B		
	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5
MFH	8.77	21.39	7.81	28.93	35.80	34.20	33.32	51.48	35.76
vTE	10.60	23.83	9.91	18.84	41.07	20.71	12.99	39.36	12.41
Sparsans	8.95	19.44	8.63	17.94	37.56	17.06	12.08	25.14	11.73
NLP-HZ	9.37	17.29	9.19	20.04	28.27	20.39	11.37	19.19	11.23
Hybrid	15.97	34.07	15.00	37.85	**64.47**	**40.19**	54.62	**77.24**	55.08
CRIM	19.78	36.10	19.03	34.05	54.64	36.77	40.97	60.93	41.31
SPON	20.20	36.95	19.40	33.50	50.60	35.10	54.70	71.20	56.30
RMM	**27.12** ±0.12	**39.07** ±0.42	**23.41** ±0.22	**38.56** ±0.34	54.89 ±0.63	37.17 ±0.33	**63.86** ±0.06	74.75 ±0.52	**61.61** ±0.25

Table 2: Performance comparison on different models on the benchmark datasets. In the first column, Hybrid denotes the Hybrid of SVD/NN model and Sparsans represents the 300-sparsans model. The results of RMM are average from 3 runs of experiments. Other reported results are from their corresponding original paper.

increase after 200 continuous epochs, training will be terminated. The max epoch is set to 1000. In addition, gradient clipping is used in the weight updating process, with a clip of $1e^{-4}$. We use the Adam optimizer with beta1=beta2=0.9 and with a learning rate of $2e^{-4}$ for all datasets. We choose two separate embedding transformation matrices for two different query types. When initializing these projection matrices in the mapping function, we add random noises of Gaussian distribution (zero mean and $\frac{1}{200}$ variance) to an identity matrix. Finally, we implement our model using PyTorch on a Linux machine with a GPU device Tesla V100 SXM2 32GB.

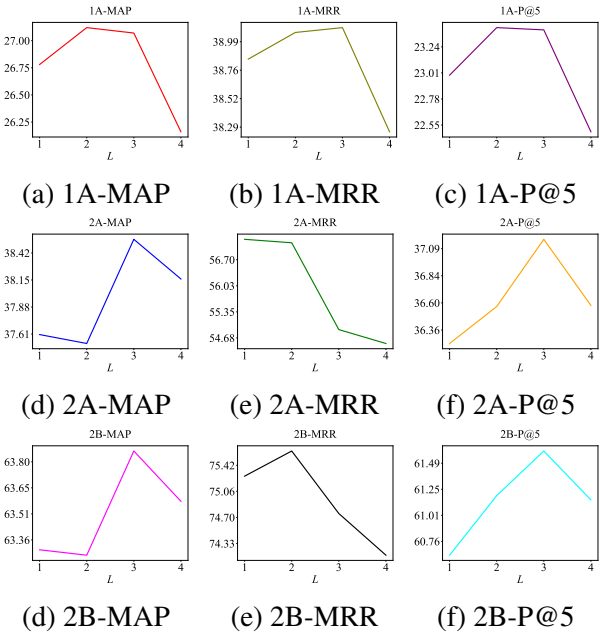
5.3 Results

5.3.1 Overall Results

Table 2 shows the MAP, MRR and P@5 performance of our model and the other baseline models across multiple hypernym discovery sub-tasks. The value of L is tuned over the validation set, we used L=2 for subtask 1A and L=3 for 2A and 2B. Note that, to avoid the performance randomness, the performance of our model is the average of three random runs. For the other compared models, the reported performance is taken from their original paper. The values printed in **bold** font are the top-performing models in the comparison.

In the table, it is clear that our recurrent mapping model (RMM) outperforms almost all existing baseline models on all the general English hypernym discovery tasks. More specifically, on all sub-tasks, RMM outperforms any supervised hypernym discovery models on all metrics. The only model that RMM does not fully beat is the Hybrid of SVD/NN model, which uses both unsupervised and supervised approaches. We note however that RMM scores best 6 out of the 9 metrics across the compared methods.

In addition, RMM outperforms the most recent supervised models, i.e. CRIM and SPOM by a significant margin. This performance suggests that the true hypernym rank is generally higher than other candidate words using RMM.



(a) 1A-MAP (b) 1A-MRR (c) 1A-P@5

(d) 2A-MAP (e) 2A-MRR (f) 2A-P@5

(d) 2B-MAP (e) 2B-MRR (f) 2B-P@5

Figure 3: The comparison of RMM with different mapping units. In each sub-figure, x axis represents the number of mapping units and the y axis represents the performance of the corresponding model. The results reported is an average of three experiments.

5.3.2 The Impact of the Number of Mapping Units

We now examine the impact of the chosen number of mappings on RMM. In our recurrent mapping process, we use mapping functions from a specific term to a more general term it belongs to. The number of mappings can be understood as the number of projections from the original query. Thus, this value is a hyper-parameter in RMM. We let L denote the number of mapping units.

As discussed above, we speculate that the correct hypernym might be chosen by different mapping units (MU). Here we present experimental results to support this claim and show how this value affects the performance of RMM. We vary L from 1 to 4 to observe the performance. The experimental results are shown in Figure 3. Note that all values in the figure are the average over three runs.

The results indicate that RMM is sensitive to L. RMM shows a general increasing performance when increasing L. This might be due to that the hypernym semantics transformation is captured by our model. We find that RMM performs best at $L = 3$ for 2A, 2B and $L = 2$ for 1A. This phenomenon is consistent with the typical hypernymy transformation situation in the data set. We observe that, in general, the true hypernym list is often in the form of a two or three layers hierarchical structure. Here we list two examples of the hierarchical structure: `guitar` $\rightarrow$ `stringed instrument`$\rightarrow$ `musical instrument`; `alternative rock`$\rightarrow$ `rock music`$\rightarrow$ `music` $\rightarrow$ `music style`. So that RMM will be effective when choosing L as 2 or 3.

It also can be noted that when choosing $L = 4$, RMM achieves a lower performance on all data sets. This result confirms our claim and shows the effectiveness of the sequential structure exploited in RMM.

5.3.3 Ablation Study

To more precisely evaluate RMM and to comprehensively analyze the contribution of each component of our model, we conduct an additional experiment of ablation studies.

Specifically, this experiment involves three components, ResNet connection between mapping units, separate transformation function for query type and the attention mechanism to identify the importance of mapping units. By removing or modifying each of them individually, we are able to observe their effects on our model.

The experiment was performed on the same datasets along with the same experimental setup and hyperparameters as in the main experiment.

Without Residual Mechanism RMM uses a residual mechanism to overcome the gradient vanishing problem and to improve the model performance. In this ablation setup, we directly remove this ResNet connection and refer to this setup as $\mathrm{RMM}_{\text{w/o ResNet}}$.

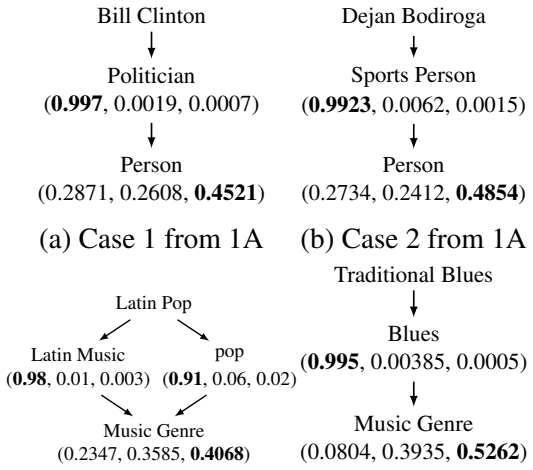
Without Separating Query Types Before multi-hop mapping, RMM model exploits two different learnable projection matrices for query types of `entity` and `concept` to transform original embedding to "is-a" embedding. In this ablation experiment, we unify these two projection matrices and refer to this setup as $\mathrm{RMM}_{\text{w/o QType}}$. It's worth noting that hyponyms in 2A(music) are all of concept type, thus unifying these two projection matrices can get a close performance as before.

Without Attention Mechanism RMM model makes use of an attention mechanism to aggregate hypernym representations from all mapping units according to candidate hypernyms. In this ablation experiment, to verify its contribution, we remove this weighting mechanism and instead use a mean approach, which simply averages the representations of the output of each unit to aggregate semantics from all mapping units. We refer to this setup as $\mathrm{RMM}_{\text{w/o Att}}$.

Table 3 shows the result of our ablation study. It shows that all components are critical for RMM. Specifically, we can see that removing ResNet degrades the model performance, which proves ResNet can avoid the information loss between units. Without using the query type, model performance degrades as well. It's notable that CRIM (Bernier-Colborne and Barriere, 2018) also utilized the type information of hypernyms by training separate logistic regression classifiers for different types of hyponyms. However, their ablation study suggests that their type modeling actually degrades the model performance which is contrary to the results shown in our ablation study. It indicates that the type information is better modeled in our model. From the last raw of Table 3, we can conclude that taking the average of each mapping unit's output decreases the model performance.

Model	1A			2A			2B		
	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5
RMM	27.12	39.07	23.41	38.56	54.89	37.17	63.86	74.75	61.61
$\text{RMM}_{\text{w/o ResNet}}$	25.78	37.08	22.41	35.80	55.65	34.52	62.39	74.38	60.21
$\text{RMM}_{\text{w/o QType}}$	24.76	37.14	21.22	38.23	54.32	36.73	61.71	69.89	60.18
$\text{RMM}_{\text{w/o Att}}$	25.91	37.63	22.37	36.60	57.04	35.00	62.90	75.74	60.27

Table 3: Ablation study.



(a) Case 1 from 1A (b) Case 2 from 1A

(c) Case 3 from 2B (d) Case 4 from 2B

Figure 4: Case Studies. The values in the bracket represent the weights on the first, the second and the third mapping units filtered by hypernym.

5.3.4 Case Study

In this subsection, we present a detailed result analysis on 4 randomly chosen cases from our testing sets, with the aim to validate our motivation of the recursive structure of RMM being capable of capturing the near hypernym first and far hypernym later. We observe the weights on different units and wish to examine if RMM indeed assigns higher weights to relative-lower units when the true hypernym is near the query word and vice versa. The results are shown in Figure 4.

From the figure we can observe that RMM is able to assign a higher weight to the first mapping units for its first hop hypernym. This is seen in all 4 cases by assigning a weight more than 0.9 to the first unit for their immediate hypernym. For example, in Figure 4(a) the hypernym `Politician` for query `Bill Clinton`. On the contrary, for a far hypernym of a query, a higher weight is on the last mapping units. For example, in Figure 4(c), the hypernym `Music Genre` for query `Latin Pop`, a higher weight is on the third units.

This result confirms the capability of RMM in capturing the latent hypernymy transformation and hierarchical dependencies between hypernyms.

6 Conclusion

Hypernym discovery is a basic task in natural language processing. Existing studies focus on designing better models for discovering better mapping functions from hyponyms to hypernyms. However, the latent semantic transformation between the hypernyms of one hyponym is not considered. In this study, both the mapping and the semantic transformation between hypernyms are considered by a recursive mapping model. In addition, with the attention mechanism, different levels of transformations are softly mixed in the final representation for the final classification task. Empirical studies on a public hypernym discovery task verify the superiority of the presented recursive model.

This study is a first attempt on modeling the transformation between hypernyms and we only achieve preliminary progress. The better usage of this will definitely promote the effectiveness of hypernym discovery. In practice, this transformation can be extracted and graph convolutional network (GCN) or other neural networks can be exploited for explicitly this information. Also, the combination of unsupervised and supervised models has shown advantages. However, most of these hybrid models are two separate processes and the supervised part highly depends on the pre-defined "is A" patterns. To build a uniform hybrid model still remains an open problem. We will study these open problems in our future work.

Acknowledgement

This work is supported partly by the National Natural Science Foundation of China (No. 61772059), by the Fundamental Research Funds for the Central Universities, by the Beijing S&T Committee (No. Z191100008619007) and by the State Key Laboratory of Software Development Environment (No. SKLSDE-2020ZX-14).

References

Gábor Berend, Márton Makrai, and Péter Fldiák. 2018. 300-sparsans at semeval-2018 task 9: Hypernymy as interaction of sparse attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*.

Gabriel Bernier-Colborne and Caroline Barriere. 2018. Crim at semeval-2018 task 9: A hybrid approach to hypernym discovery. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 725–731.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP 2015*, pages 632–642.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. 2018. Semeval-2018 task 9: Hypernym discovery. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018); 2018 Jun 5-6; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 712–24.* ACL (Association for Computational Linguistics).

I Dagan, D Roth, F Zanzotto, and M Sammons. 2013. *Recognizing Textual Entailment:Models and Applications*. Morgan Claypool.

Sarthak Dash, Md Faisal Mahbub Chowdhury, Alfio Gliozzo, Nandana Mihindukulasooriya, and Nicolas Rodolfo Fauceglia. 2020. Hypernym detection using strict partial order networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7626–7633.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114.

Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Geo Jain, and Shubhashis Sengupta. 2018. Can taxonomy help? improving semantic question matching using question taxonomy. In *COLING 2018*, pages 499–513.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR 2016*, pages 770–778.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Coling 1992 volume 2: The 15th international conference on computational linguistics*.

William Held and Nizar Habash. 2019a. The effectiveness of simple hybrid systems for hypernym discovery. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

William Held and Nizar Habash. 2019b. The effectiveness of simple hybrid systems for hypernym discovery. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3362–3367.

Johannes Hoffart, Dragan Milchevski, and Gerhard Weikum. 2014. STICS: searching with strings, things, and cats. In *SIGIR 2014*, pages 1247–1248.

Matthew Le, Stephen Roller, Laetitia Papaxanthos, Douwe Kiela, and Maximilian Nickel. 2019. Inferring concept hierarchies from text corpora via hyperbolic embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI 2011*, pages 1872–1877.

Wei Qiu, Mosha Chen, Linlin Li, and Luo Si. 2018. NLP_HZ at SemEval-2018 task 9: a nearest neighbor approach. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana. Association for Computational Linguistics.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *EMNLP 2016*, pages 2163–2172.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING 2014*, pages 1025–1036.

Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. 2016. A large database of hypernymy relations extracted from the web. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 360–367.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *AAAI 2017*, pages 1236–1242.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. *arXiv preprint arXiv:1603.06076*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL 2006*.

Dmitry Ustalov, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Negative sampling improves hypernymy extraction based on projection learning. *arXiv preprint arXiv:1707.03903*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *CIKM 2013*, pages 1107–1116.

Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. 2016. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1871–1879.

Changlong Yu, Hongming Zhang, Yangqiu Song, Wilfred Ng, and Lifeng Shang. 2020. Enriching large-scale eventuality knowledge graph with entailment relations. In *AKBC 2020*.