Leveraging WordNet Paths for Neural Hypernym Prediction

Yejin Cho[*,1] **Juan Diego Rodriguez**[*,2] **Yifan Gao**[3] **Katrin Erk**[1]

[1]Department of Linguistics [2]Applied Research Laboratories [3]Department of Mathematics
The University of Texas at Austin
{ycho, yifan233, katrin.erk}@utexas.edu
juan.rodriguez@arlut.utexas.edu

Abstract

We formulate the problem of hypernym prediction as a sequence generation task, where the sequences are taxonomy paths in WordNet. Our experiments with encoder-decoder models show that training to generate taxonomy paths can improve the performance of direct hypernym prediction. As a simple but powerful model, the *hypo2path* model achieves state-of-the-art performance, outperforming the best benchmark by 4.11 points in hit-at-one (H@1).

1 Introduction

Hypernymy, or the IS-A relation, is one of the most important lexical relations. It is used to create taxonomies of terms and it is the main organizational criterion of nouns and verbs in WordNet (Fellbaum, 1998). Learning hypernymy is also important in practice, as knowing a word's hypernyms gives an approximation of its meaning, and enables inferences in downstream tasks such as question answering and reading comprehension. Predicting hypernymy is still a challenging task for word embeddings (Pinter and Eisenstein, 2018; Bernier-Colborne and Barriere, 2018; Nickel and Kiela, 2018) and previous studies have shown that it is more difficult to predict hypernymy than other lexical relations (Balažević et al., 2019; Allen et al., 2019).

Hypernymy prediction is often evaluated against a given taxonomy, typically WordNet (Fellbaum, 1998). The main hypothesis that we pursue in this paper is that knowledge of this taxonomy, in particular of taxonomy paths, will be helpful for hypernymy prediction. So we introduce two simple encoder-decoder based models for hypernym prediction that make use of information in the full taxonomy paths.

There has been much recent work on modeling lexical relations based on distributed representations (Pinter and Eisenstein, 2018; Bernier-Colborne and Barriere, 2018; Nickel and Kiela, 2018). However, the task formulations and evaluation datasets have differed widely, making it hard to compare different approaches. We focus on evaluating on hypernymy, rather than jointly on many relations, which can mask strong performance differences across relations. We evaluate our encoder-decoder models against several previous models that have not been evaluated in the same setting before. Like many other approaches, we use WordNet as the basis for our experiments. We formulate the task as the task of finding the correct point to attach a new node (*synset*) to the WordNet taxonomy. We build on the existing WN18RR dataset (Dettmers et al., 2018), but filter its hypernymy pairs to produce *WN18RR-hp*, a subset that is leak-free with respect to approaches that use taxonomy paths during training, as we do.

We find that one of our new models, *hyper2path*, achieves state-of-the-art performance on hypernym prediction on WN18RR-hp, exceeding the best performance of benchmark models by 4.11 points in accuracy of the highest-ranked prediction (hit-at-one, H@1). In particular, we observe the greatest performance gain in noun hypernym prediction, where it improves over the best benchmark by 5.17 H@1 points.

* Equal contribution.

	Hyponym	Generated hypernym path	Gold hypernym
✓	pizza.n.01	dish.n.02 → nutriment.n.01 → food.n.01 → . . . → entity.n.01	dish.n.02
✓	alps.n.01	range.n.04 → geological_formation.n.01 → . . . → entity.n.01	range.n.04
✓	whisper.v.01	talk.v.02 → communicate.v.02 → interact.v.01 → act.v.01	talk.v.02
✗	proletarian.n.01	*worker.n.01 → person.n.01 → causal_agent.n.01 → . . . → entity.n.01	commoner.n.01
✗	austerity.n.01	*punishment.n.01 → social_control.n.01 → . . . → entity.n.01	self-discipline.n.01
✗	compulsive.n.01	*sick_person.n.01 → unfortunate.n.01 → person.n.01 → . . . → entity.n.01	person.n.01

Table 1: We frame hypernym prediction as a sequence generation problem. Given a query hyponym (e.g., *pizza.n.01*), the ***hypo2path rev*** model generates its taxonomy path, from its direct hypernym (*dish.n.02*) to the root node (*entity.n.01*). ✓ and ✗ indicate a correct and an incorrect prediction, respectively. In each example, an underlined synset corresponds to what the model predicted as a direct hypernym.

2 Hypernym Prediction

Several tasks related to hypernymy have been proposed under different names: extracting is-a relations from text (*hypernym discovery*) (Hearst, 1992; Snow et al., 2005; Camacho-Collados et al., 2018), binary classification of whether two given words are in a hypernym relation (*hypernym detection*) (Weeds et al., 2014; Shwartz et al., 2016; Roller et al., 2018), and constructing or extending a taxonomy (*taxonomy induction*) (Snow et al., 2006; Jurgens and Pilehvar, 2016). Another recently introduced task is hierarchical path completion (Alsuhaibani et al., 2019), where, given a hypernym path of length 4 from WordNet, the task is to predict the correct hyponym(s).

While hypernymy has long been studied in computational lexical semantics, another thread of recent research on hypernymy comes from the literature on knowledge base completion (Bordes et al., 2013; Nickel and Kiela, 2017; Pinter and Eisenstein, 2018; Dettmers et al., 2018). Here, hypernymy is considered as one of multiple different semantic relations between two nodes in a graph. Extending from this line of research, we also consider the **relation prediction** task in a semantic graph, but only focus on one relation of interest, hypernymy.

Like previous work in knowledge base completion (Bordes et al., 2013; Nickel and Kiela, 2017; Pinter and Eisenstein, 2018; Balažević et al., 2019), we take WordNet as our experimental space, so we learn hypernymy between *synsets* rather than raw lemmas. A synset is a basic lexical unit in WordNet, defined as a set of *lemmas* that are synonymous to each other. A synset thus also functions as one of the senses for each of the lemmas in the set. For example, the synset *mark.n.01* (a number or letter indicating quality) consists of the three lemmas *mark*, *grade*, and *score*. Given a new synset, which we call the *source node*, our task is to predict its direct hypernym or *target node* from among the synsets in WordNet. For example, for the source node *woolly_daisy.n.01*, the model should identify *wildflower.n.01* as the target node in the graph.

While some previous approaches have predicted indirect hypernyms using the transitive closure of WordNet (Vendrov et al., 2016; Li et al., 2019), we focus on predicting direct hypernyms. Datasets for indirect hypernymy often include hypernyms that are too generic and not informative enough, where some semantically distant concepts are trivially mapped to the root node (*entity.n.01*) or a high-level hypernym near the root. We also restrict ourselves to modeling hypernym relations specifically, unlike much work on WN18RR which learns hypernymy as one of 11 lexical relations (Bordes et al., 2013; Pinter and Eisenstein, 2018; Balažević et al., 2019). As we discuss in Section 5, we observe that hypernymy is more effectively learned when trained on its own.

3 Models

In this section we introduce our two new path-based models, *hypo2path* and *Path Encoder*[1], along with the four benchmark models.

3.1 Path Generators: *hypo2path* and *hypo2hyper*

In our first model, we treat hypernym prediction as a sequence generation task: given a hyponym, the goal is to generate the entire path in the WordNet taxonomy starting from the root node (*entity.n.01* for nouns)

[1]Code and data are available at `https://github.com/scarletcho/hypernym-path-generation`.

and ending with the direct hypernym. For example, *flock.n.02* should be mapped to its hypernym path by generating *entity.n.01* → *abstraction.n.06* → *group.n.01* → *biological_group.n.01* → *animal_group.n.01*. So the model is tasked to translate source synsets to target synset sequences. We denote this model as ***hypo2path***. Our intuition behind this model is that training with a more difficult objective (i.e., entire hypernym path prediction rather than direct hypernym prediction) may result in a stronger model.

We use a standard LSTM-based sequence-to-sequence model (Sutskever et al., 2014) with Luong-style attention (Luong et al., 2015). This encodes a synset embedding of a hyponym into a hidden state, which is taken as the initial state of the LSTM decoder. The decoder generates synsets sequentially, conditioned on previously generated synsets. While the attention mechanism assigns weights to the source tokens, here we only have a single source token (i.e., a query hyponym). In our task, the attention mechanism serves as a way to avoid "forgetting" the source hyponym while decoding long paths[2].

Reversing the order of the source or target sequences can the improve performance of encoder-decoder models, since the encoded hidden state is closer to the first target token (Sutskever et al., 2014; Gillick et al., 2016). Motivated by this, we experiment with a model variant called ***hypo2path rev*** in which we reverse the target path to generate a sequence of hypernyms starting from the direct hypernym of the source. This frames every generation step as direct hypernym prediction, which the decoder may more easily learn. Examples of generated reversed paths are shown in Table 1.

In order to determine whether generating an entire hypernym path as an auxiliary task helps to accurately predict a synset's direct hypernym, or whether generating only the direct hypernym is enough, we also perform experiments with ***hypo2hyper***, a variant of *hypo2path*. Here the encoder-decoder model is trained to only generate a direct hypernym (i.e., both the source and target sequences are of length 1).[3]

3.2 Path Encoder

We also examine the reverse approach: training an LSTM encoder that learns vector representations of hypernym paths. Given a query hyponym, we construct an embedding of a hypernym path (from the root node down), and the model is tasked to distinguish the gold path (which ends at its direct hypernym) from distractor paths. We construct path embeddings using a bidirectional LSTM followed by a fully-connected layer. The output corresponding to the last hidden state is the encoded path vector. We denote this model as ***Path Encoder***.

Given the training set $\mathcal{S}$ consisting of pairs (x, p) of a hyponym and a path,[4] we train the model by minimizing the Euclidean distance of encoded path vectors V_p and the embedding vectors of the hyponym V_x. In addition, the model is trained to maximize the distance of V_p and $V_{x'}$ from the negative examples $\{(p, x')|(*, x') \in \mathcal{S}\}$ which are generated by randomly pairing a hyponym with a random path. The model is optimized with the following ranking loss function:

$$L = \max\{0, \|V_p - V_x\|_2 - \|V_p - V_{x'}\|_2 + \gamma\}$$

where γ is a positive margin hyperparameter.

During prediction, the model first encodes all hypernym paths $\{p|(p, *) \in \mathcal{S}\}$. Then for each query hyponym x the path that minimizes $\|V_p - V_x\|_2$ is returned as the predicted hypernym path. From this path, we take the predicted direct hypernym of x for evaluation, just as we do for *hypo2path*.[5]

3.3 Benchmark Models

TransE In the TransE model (Bordes et al., 2013), a semantic relationship is interpreted as a vector translation in embedding space. Given a triplet (*s*, *r*, *t*) of a source node *s*, a relation *r*, and a target node *t*, the model learns embeddings for the nodes and relation such that the target vector *t* is near $s + r$.

[2]Attention improved the performance of *hypo2path* on nouns by about 3 points in H@1 (averaged over 5 runs).

[3]We also experimented with settings in-between *hypo2path* and *hypo2hyper*, where the model is trained to predict only truncated hypernym paths of maximum length 2 (for verbs) or 5 (for nouns and instance nouns). We do not report these results, since performance was similar to training with the full path.

[4]When multiple paths for a given hyponym exist in the graph, each of these paths is paired with its hyponym *x*.

[5]As with the *hypo2path* experiments, we also tried encoding truncated paths of length 2 and 5. We omit these results, since they were similar to encoding the full path.

M3GM Max-Margin Markov Graph Models (M3GM) (Pinter and Eisenstein, 2018) exploit graph motif properties in WordNet (e.g., number of cycles of length 2 or 3) to predict different semantic relations. As a global feature model, M3GM reranks the top *N* candidates predicted by a local distributional feature model such as TransE (Bordes et al., 2013).

CRIM Bernier-Colborne and Barriere (2018) proposed a hybrid system which exploits both unsupervised pattern-based hypernym discovery and supervised *projection learning* (Ustalov et al., 2017). The core idea of the supervised algorithm is to learn multiple projection matrices which map a query embedding to a target hypernym. Their system ranked first on the three subtasks in SemEval-2018 Task 9 (Camacho-Collados et al., 2018).

Text2edges The approach most similar to ours is (Prokhorov et al., 2019), which represents each hyponym using its textual definition from WordNet and maps it to its taxonomy path from the root to its parent node. Given the definition of a query hyponym, a bidirectional LSTM encoder-decoder with attention is used to generate the taxonomic path starting from the root node. For example, the definition of *swift* ("a small bird that resembles a swallow and is noted for its rapid flight") is mapped to the sequence '*animal*, *chordate*, *vertebrate*, *bird*, *apodiform bird*'. Their best system, *text2edges* with pre-trained ConceptNet numberbatch embeddings (Speer et al., 2017), uses a reduced set of artificial edge label symbols rather than the original node labels.

Similar to our approach, text2edges uses a sequence-to-sequence model with an attention mechanism. However, there are several important differences. First, it encodes the definition of an input hyponym, while our prediction conditions only on the vector representation of the synsets themselves without looking at their definitions. Obtaining a definition of an unknown word is not always feasible, especially for domain-specific jargon and neologisms that are frequently used but seldom defined in dictionaries. On the other hand, computing their embeddings is a less challenging task when using approaches such as fastText (Bojanowski et al., 2017), which also works for words seen only once because it interpolates from embeddings of each word piece.[6] Another key difference is that it can only apply to *rooted tree graphs* with a single root. For this reason, it cannot be trained on the verb taxonomy in WordNet, which has more than one root.

3.4 Other Path Encoding Approaches

We next discuss some related path encoding approaches which we do not compare in our experiments. First, Das et al. (2017) proposed a model for link prediction that is similar to *Path Encoder*. Given a multi-relational knowledge base, their task is to assign the correct relation to those entity pairs linked by an entity-relation path but without a direct relation between them. Here, sequences of entities and relations are encoded with an LSTM and the relation whose vector is closest to the encoded path is returned. In our task, however, concepts are linked by only one relation (hypernymy or instance hypernymy). This prevents us from drawing additional information from other relation paths between synsets.

Alsuhaibani et al. (2019) also use a path-based model, but predict hyponyms rather than hypernyms in WordNet. Their model learns hierarchical embeddings over the taxonomy, where each leaf node is represented as the sum of the embeddings of its first four hypernyms (i.e., one direct and three indirect hypernyms in the taxonomy path). The model predicts the hyponym of a given path by maximizing the distance between the sum of the hypernym vectors and candidate hyponym vectors. However, this approach cannot be used for hyponyms that never appear in the taxonomy.

4 Experimental Setup

4.1 Dataset: WN18RR-hp

WN18RR (Dettmers et al., 2018) is a filtered version of the WN18 dataset (Bordes et al., 2014), a subset of WordNet 3.0 which only contains synsets with at least fifteen connections to other synsets. WN18RR

[6]The obtained embedding of an undefined word can be trivially identified as a synset embedding with a single lemma (i.e., without any other synonyms), obviously because there are no other known synonyms. Single-lemma synsets are commonly observed in WordNet (33.6% of nouns, 32.6% of instance nouns, and 41.7% of verbs in our dataset (WN18RR-hp)).

removed seven inverse relations (such as hyponymy) from WN18 that caused a test leakage problem.

For this work, we only use two relations in WN18RR: *hypernymy* and *instance hypernymy*. Instance hypernymy can be thought of as a special type of hypernymy; it only holds between an *instance* that is a terminal node (e.g., specific persons, countries, and geographic entities) and its hypernym (common noun). Also, we evaluate verb and noun hypernymy separately because verb hypernymy (troponymy) in WordNet is conceptually distinct from noun hypernymy, as it expresses a manner relation rather than an IS-A relation (Fellbaum, 2002).

To avoid giving an unfair advantage to the path-based models, we filtered both validation and test sets to only include hyponym queries that are unseen anywhere in the full taxonomy paths of the training data. By eliminating the queries observed during path training, we made sure that all evaluated queries are equally new to both path-based models (e.g., hypo2path) and non-path models (e.g., hypo2hyper). We also exclude hyponyms from the test and validation sets which appear as hyponyms in the training set[7] to prevent the models from merely copying. We denote this subset as **WN18RR-hp**.

In sum, we use three different types of hypernym relation sets (noun, instance noun, verb) in our experiments. The number of examples in WN18RR-hp is shown in Table 2.

Number of pairs	Train	Valid	Test
Noun	27946	647	676
Instance noun	2921	76	79
Verb	6849	187	206

Table 2: Data statistics of hyponym-hypernym pairs in WN18RR-hp.

The WordNet taxonomy is a directed acyclic graph where many hypernyms have multiple paths to the root. When training path models (i.e., *hypo2path*, *Path Encoder*, and *text2edges*), we included every existing path to a query's parent as individual target instance(s).

4.2 Evaluation Metrics

We use two different measures that represent the accuracy of model predictions: a hard accuracy measure (hit at one, H@1) and a "ballpark match" (soft accuracy) measure (WuP).

H@1 score As one of the most commonly used evaluation metrics for the relation prediction task, the hits-at-*k* (H@*k*) is the proportion of correct predictions (*hits*) within the top *k* ranked predictions. As the most intuitive and practical measure of each model's performance, we only consider the top first prediction accuracy (H@1), and not others with larger *k* (e.g., H@10).

Wu & Palmer similarity (WuP) The Wu and Palmer score (Wu and Palmer, 1994) is a similarity measure between two nodes in a taxonomy that ranges between 0 and 1. To quantify how close they are, it considers how deep the two nodes and their closest common ancestor are in a taxonomy. For instance, the WuP score for *orange.n.01* and *lemon.n.01* is 0.75, while it is only 0.35 for *orange.n.01* and *car.n.01*. To assess how close a prediction is to the gold hypernym, we compute the WuP score between them using NLTK's implementation.[8] We report an averaged WuP score of each system.

4.3 Synset Embeddings

Averaged Lemma Vectors Following Pinter and Eisenstein (2018)[9], we computed an embedding for each synset by averaging the pretrained fastText embeddings (Bojanowski et al., 2017)[10] of its synonyms

[7]These cases exist because some queries have multiple hypernyms. WN18RR allows such queries with multiple gold targets to appear in train and evaluation sets.

[8]When two nodes are identical, NLTK's implementation of WuP score does not necessarily return 1.0. We added an ad-hoc check to make sure we get 1.0 WuP score in such cases.

[9]We used `embed_from_words.py` released by the first author at `https://github.com/yuvalpinter/m3gm`

[10]`https://fasttext.cc/docs/en/pretrained-vectors.html`

(*lemmas*). If a synset contained any multi-word lemma, the words within the lemma were averaged. We used these synset embeddings for all our reported experiments.[11]

4.4 Model Details

Baselines We include two simple baselines, *closest vector* and *closest co-hyponym*, in order to gauge reasonable lower bounds for our metrics. The *closest vector* baseline is obtained by predicting the hypernym whose vector is closest (using the Euclidean norm) to the given synset, choosing from among all the hypernyms in the training set. On the other hand, *closest co-hyponym* is obtained by predicting the hypernym of the closest hyponym in the training set, i.e., under the assumption that nearby vectors are co-hyponyms.

TransE, M3GM, CRIM, and text2edges Our replications of the benchmark models are based on the original source code[12] keeping the default hyperparameters except for a few things: For TransE and CRIM, we tuned for the best number of training epochs. We increased the early stopping threshold to five epochs for TransE, as the model stopped too early with the default setting. Also, we only trained the supervised part of CRIM, since the unsupervised part of CRIM requires an external corpus for training. All models except text2edges used fastText embeddings to compute synset embeddings. For text2edges, we replicated their best system which takes the pretrained ConceptNet numberbatch embeddings to represent words in node definitions.

For all models except M3GM, we trained a separate model for each relation of WN18RR-hp (noun, instance noun, verb). On the other hand, we trained all three relations in a single M3GM model, as it employs graph motif features of different relations. We trained this multi-relational M3GM model as a re-ranker of TransE that was also trained on all three relations.[13] We did not run a post-hoc tuning for graph score weights in M3GM.

Path Generators: *hypo2path* and *hypo2hyper* We implemented a sequence-to-sequence model with Luong attention in Keras, which we used for the *hypo2path* and *hypo2hyper* experiments. We used a single-layer unidirectional LSTM with 256 hidden units and a dropout rate of 0.3 for both the encoder and the decoder. We trained the network with teacher forcing and used the Adam optimizer with a learning rate of 0.001 and batch size of 256. The embedding layer was frozen during training. Synsets without pretrained embeddings were assigned random vectors with elements sampled uniformly from $[-.25, .25]$. Greedy decoding was used to generate sequences.[14] We did not perform any hyperparameter tuning for these models.

Path Encoder The *Path Encoder* model was implemented in PyTorch, using a single-layer bidirectional LSTM to encode the path and a fully-connected layer to map the output to the target embedding space. The dimension of the LSTM cell was 1024 for nouns and instance nouns and 512 for verbs.

The learning rates in $\{0.01, 0.001\}$ and margins in $\{0.1, 0.3, 0.5, 0.7\}$ were tuned differently for noun, instance noun, and verb experiments. We used the same dropout rate, batch size and choice of optimizer as in the *hypo2path* experiments.

5 Results and Discussion

Results Overall, *hypo2path rev* shows the highest aggregate (micro-averaged) H@1 (dev: 24.43, test: 25.59) across the three hypernymy relations (nouns, instance nouns, and verbs), while CRIM has the best aggregate ballpark correctness (WuP) scores that are closely followed by *hypo2path rev* (Table 6).

[11]We also trained *hypo2path* with randomly initialized embeddings (trained with the rest of the network), and observed a large drop in H@1. This suggests there is information in the embeddings which cannot be learned solely from the hypernym generation task, in line with observations of Pinter and Eisenstein (2018).

[12]TransE and M3GM: `https://github.com/yuvalpinter/m3gm/`
CRIM: `https://github.com/gbcolborne/hypernym_discovery/`
text2edges: `https://github.com/VictorProkhorov/Text2Path/`

[13]This multi-relational TransE model is different from the single-relational TransE models reported in the results section.

[14]Preliminary experiments on nouns showed no improvement when using beam search (with beam widths up to 6).

	Validation		Test	
	H@1	**WuP**	**H@1**	**WuP**
Closest vector	9.74	54.36	10.50	54.65
Closest co-hyponym	17.62	60.34	18.64	59.92
TransE	13.91	61.50	11.69	61.36
M3GM	18.24	61.66	18.20	61.73
CRIM	22.41	65.32	19.53	65.29
text2edges	17.31	**67.67**	16.42	66.41
Path Encoder	20.56	63.84	22.63	63.59
hypo2hyper	22.87	64.55	23.82	64.47
hypo2path	22.26	65.79	23.82	**66.53**
hypo2path rev	**23.65**	65.32	**24.70**	65.98

Table 3: Scores for **nouns**.

	Validation		Test	
	H@1	**WuP**	**H@1**	**WuP**
Closest vector	29.87	57.70	20.25	59.25
Closest co-hyponym	50.00	74.41	49.37	77.47
TransE	54.55	81.02	54.43	83.34
M3GM	48.05	78.42	58.23	81.09
CRIM	66.23	86.12	67.09	86.67
text2edges	69.74	**88.05**	72.15	**88.53**
Path Encoder	62.34	82.88	49.37	75.96
hypo2hyper	**70.13**	86.78	**73.42**	87.75
hypo2path	66.23	85.00	72.15	86.24
hypo2path rev	**70.13**	87.23	**73.42**	87.18

Table 4: Scores for **instance nouns**.

	Validation		Test	
	H@1	**WuP**	**H@1**	**WuP**
Closest vector	3.21	34.79	1.94	30.58
Closest co-hyponym	8.02	39.21	5.34	36.24
TransE	3.21	35.71	3.40	35.65
M3GM	3.21	31.95	2.43	30.47
CRIM	**12.30**	**46.25**	**9.71**	**43.66**
Path Encoder	6.42	36.45	5.34	35.60
hypo2hyper	10.16	41.65	7.28	39.23
hypo2path	7.49	37.82	8.25	39.07
hypo2path rev	8.56	41.01	9.22	39.66

Table 5: Scores for **verbs**.

	Validation		Test	
	H@1	**WuP**	**H@1**	**WuP**
Closest vector	10.08	50.62	9.56	50.09
Closest co-hyponym	18.35	57.17	18.47	56.52
TransE	15.11	57.83	13.56	57.91
M3GM	17.64	56.95	18.3	56.92
CRIM	23.99	**63.14**	21.48	**62.63**
Path Encoder	21.14	59.80	21.31	58.87
hypo2hyper	24.21	61.7	24.56	61.23
hypo2path	22.9	61.65	24.66	62.53
hypo2path rev	**24.43**	62.15	**25.59**	62.34

Table 6: Aggregated scores across all three groups.

For both nouns (Table 3) and instance nouns (Table 4), *hypo2path rev*[15] is a clear winner in terms of H@1. Despite being a simple model, it achieves the best H@1 with notable improvements over more complex benchmarks. We observe large gains (5.17 points) on nouns over CRIM and some gains (1.27 points) on instance nouns over text2edges. With respect to the ballpark accuracy (WuP), *hypo2path* shows similar performance to CRIM and text2edges on nouns, while text2edges does slightly better on instance nouns.

For nouns, the reversed *hypo2path* model ('*hypo2path rev*') achieved the best performance on H@1, while the non-reversed and reversed versions performed similarly in terms of WuP scores. Without reversing the path, the model's H@1 slightly degraded, and is closer to *hypo2hyper*'s results. The performances of *Path Encoder* followed closely after the proposed three encoder-decoder models.

On instance nouns, performances of different models are overall much higher than on nouns. The best results are observed from the *hypo2hyper*, *hypo2path rev*, and text2edges models. That *hypo2hyper* and *hypo2path rev* perform similarly is not surprising: instance hypernymy is less likely to be learned from path generation, as it is a special type of hypernymy that only holds between a leaf and its parent node.

On the other hand, none of the models[16] does well on verb hypernymy (Table 5). CRIM achieved the highest scores overall for verbs. Consistent with the experiments on nouns and instance nouns, *hypo2path rev* had relatively strong performance, with 9.22 H@1 on the test set, which is comparable to the best H@1 (9.71). However, the best H@1 is still below 10% and the best WuP score is not much higher than the closest co-hyponym baseline.

Discussion Despite being trained with about ten times less data, scores for instance hypernymy are generally much higher than for noun hypernymy. Our finding that it is easier to predict hypernyms for individual entities than for common nouns is consistent with previous work (Boleda et al., 2017; Camacho-Collados et al., 2018; Balažević et al., 2019; Nguyen et al., 2019).

On the other hand, scores for hypernymy amongst verbs are very low, despite having about twice as much training data as instance hypernyms. This could be due both to the fact that verbs have more

[15]We ran experiments across five different random seeds for *hypo2path rev* and found the standard deviation to be very low (noun: [0.82_{dev}, 0.97_{test}], instance noun: [2.52_{dev}, 2.08_{test}], verb: [0.29_{dev}, 1.17_{test}]) indicating that the model is quite stable.

[16]text2edges is not included since it cannot run on the verb data.

Error type	Examples Query hyponym	Gold hypernym	Prediction	Confounding hyponym	%
Indirect hypernym	chant.n.01	religious_song.n.01	music.n.01		14.4
	folk_singer.n.01	singer.n.01	musician.n.01		
Co-hyponym	tart.n.03	pastry.n.02	pie.n.01		3.6
	knitwear.n.01	clothing.n.01	apparel.n.01		
Polysemy (Shared lemmas)	nut.n.03 [**nut**]	block.n.01	seed.n.01	nut.n.01 [**nut**]	17.4
	chest.n.02 [**chest**]	box.n.01	external_body_part.n.01	breast.n.01 [breast, **chest**]	
Multi-word lemmas with shared words	night_porter.n.01 [**night** porter]	doorkeeper.n.03	evening.n.01	guest_night.n.01 [guest **night**]	15.4
	carpet_beater.n.01 [**carpet** beater, rug beater]	beater.n.02	cleaning_implement.n.01	carpet_sweeper.n.01 [**carpet** sweeper, sweeper]	

Table 7: Four typical error patterns and examples observed from *hypo2path rev* evaluated on nouns (validation set). Altogether, they account for 50.8% of the total errors. Words within square brackets after a synset are the set of lemmas of the synset. The boldfaced are the shared lemmas/words.

complex semantics than nouns and to the structure of the WordNet verb hierarchy; while the noun subgraph is a single tree rooted at *entity.n.01*, the verb subgraph consists of 599 shallow trees. The WordNet verb hierarchy also has a number of annotation errors described in Richens (2008). In addition, verbs are more polysemous than nouns (Fellbaum, 1990), and the hypernym relation for verbs (troponymy) encompasses a diverse set of heterogeneous subsumption relations (Fellbaum, 2002; Richens, 2008).

Our results also suggest that it may be more effective to learn hypernymy separately from other lexical relations, rather than in a multi-relational setup. For example, M3GM trained with all 11 relations of WN18RR achieved a high aggregate validation H@1 of 39.88 in our replication , but a much lower H@1 for hypernymy (1.19) and instance hypernymy (3.74). These were evaluated on the original WN18RR, following Pinter and Eisenstein (2018), so these scores are not comparable to Tables 3, 4, 5, 6. Unlike Pinter and Eisenstein (2018), we evaluated M3GM in one direction (i.e., only hypernym prediction, rather than both hyponym and hypernym prediction).[17] These results are in line with the H@10 scores reported by Nguyen et al. (2019) and Balažević et al. (2019), which are substantially lower for hypernymy than for the other lexical relations in WN18RR.

6 Error Analysis

Here we examine the predictions of the best performing model, *hypo2path rev*, on the validation set for nouns in WN18RR-hp (647 synsets).

Path Validity Regardless of whether the predicted direct hypernym was correct or not, we observe that every generated path, from each predicted hypernym to the root, is actually a valid path in WordNet. This is not surprising, since all such paths appear in the training data. While the model always correctly generated valid paths in the graph, they did not necessarily start at the correct node (i.e., failing to predict the gold direct hypernym).

Nearby Nodes For noun hypernymy, 14.4% of the errors are due to predicting an **indirect hypernym** (Table 7). The remaining incorrect predictions are not on the path from the hyponym to the root: these include **co-hyponyms** ("siblings", or nodes that share the same parent), and "cousins" (nodes that share the same non-parent ancestor). 3.6% of incorrect predictions are co-hyponyms (also in Table 7). About half of all predicted cousins had a common ancestor with the query hyponym that was within four steps.

Similar Synset Embeddings Some synsets were similar enough to mislead the model: **polysemous lemmas** were shared across different synsets (17.4% of the total errors) and some **multi-word lemmas** had shared words (15.4%) (Table 7). In these cases, the model incorrectly returned a hypernym of a "confounding hyponym" that has a similar representation to the query. This type of error is attributable

[17]When evaluated on the both directions, we obtained 42.95 aggregate validation H@1, which is close to their reported performance, 43.26. Interestingly, their hyponym prediction validation H@1s are overall much higher (hypernymy: 16.18, instance hypernymy: 41.12) than those of hypernym prediction.

to the way synset embeddings are computed (i.e., averaging lemma vectors) which we adopted from Pinter & Eisenstein (2018).

Lemma Overlap and Polysemy Although none of the queries (synsets) are shared between the training and evaluation (validation/test) sets, some lemma-level overlaps exist (5.4% of training set overlaps with the validation set). We checked whether our model is taking advantage of any lemma overlap by computing the correlation between prediction correctness and lemma overlap rate (i.e., how many lemmas of a query hyponym synset are already seen anywhere in the training set) of each predicted pair in the validation set. If the correlation is positive, this indicates that the the model did get some hints from lemmas seen from training set.

However, we observed a significant negative correlation (-0.228; p-value<0.001). This is related to the point on polysemy discussed above. In nearly half of the validation instances where a query had a lemma overlap with the training set, the model incorrectly predicted the hypernym of a confounding hypernym.

Rare Synsets Is hypernym prediction more difficult for infrequent synsets (i.e., synsets whose lemmas rarely appear in the corpus from which the word vectors were derived)? We define a synset's frequency as the average of the frequencies of its lemmas.[18]

We find that the 164 synsets with frequency under 2,000 have an H@1 score of 15.2 (an 8.4 point drop). To further quantify the effect of synset frequency on performance, we ranked and binned every prediction by synset frequency (Figure 1). There is a clear upward trend, suggesting that methods designed to learn better embeddings with sparse data (Herbelot and Baroni, 2017) could improve performance substantially.

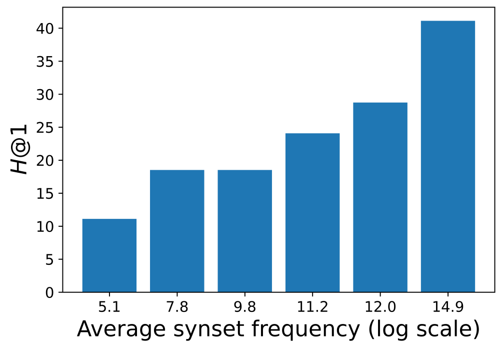

Figure 1: H@1 for hyponyms of different frequency within equal-sized bins of size 108.

7 Conclusion and Future Work

In this paper we have considered the *hypernym prediction* task, the task of identifying the correct direct hypernym in a taxonomy of a synset given its embedding. In terms of evaluation, we have focused on both "exact match" (H@1) and "ballpark match" (WuP) metrics, and we have for the first time provided a comparison of existing models that had not previously been evaluated with the same metrics or on the same datasets.

We have introduced two simple encoder-decoder based models for hypernym prediction that make use of information in the full taxonomy paths, finding that in particular *hypo2path rev* shows state-of-the-art performance on the WN18RR-hp dataset. For nouns, it achieves improvements of 5.17 test H@1 over the best benchmark model. For verbs, we find that no model achieves a high performance. Instance

[18]We extracted the frequency statistics of the English Wikipedia (`wiki.en.bin`) fastText word vectors using function `get_words` from `https://github.com/facebookresearch/fastText` (v. 0.9.1).

nouns, on the other hand, are the easiest to predict. Encouragingly, we find that “ballpark match” (WuP) for instance nouns is at over 87 points.

There are several directions for future work. Encoding lemmas separately and with attention, rather than with a single embedding, could allow the attention mechanism to assign lower weights to less informative, misleading, or polysemous lemma names (which were related to over a third of all errors). One potential way to handle polysemy could involve encoding both synsets and glosses, using either classical word embeddings or contextualized word embeddings. Another extension of this work is to use multi-task learning with multiple decoders and different tasks, similar to Luong et al. (2016). For example, in addition to learning to decode a path of hypernyms leading to a given synset, a model could also generate the synset’s co-hyponyms or its hypernym’s *lexname*[19].

Our results suggest that hypernym prediction is challenging for words which occur less frequently in the corpus used to compute embeddings. Methods are needed which can learn more effectively from low-frequency data, where the “unknown word” does not appear very often (Herbelot and Baroni, 2017; Kabbach et al., 2019).

Acknowledgments

This work was supported by the ARL:UT Independent Research and Development Program. The authors acknowledge the Texas Advanced Computing Center (TACC) at the University of Texas at Austin for providing computing resources used to obtain the results in the paper. We thank Jessy Junyi Li and anonymous reviewers for providing helpful comments and feedback for this work.

References

Carl Allen, Ivana Balazevic, and Timothy M Hospedales. 2019. On understanding knowledge graph representation. *arXiv preprint arXiv:1909.11611.*

Mohammed Alsuhaibani, Takanori Maehara, and Danushka Bollegala. 2019. Joint learning of hierarchical word embeddings from a corpus and a taxonomy. In *The Conference of Automatic Knowledge Base Construction.*

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*, pages 4465–4475.

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 (SemEval-2018)*, pages 725–731.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Gemma Boleda, Abhijeet Gupta, and Sebastian Padó. 2017. Instances and concepts in distributional space. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 79–85.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

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).

[19]Every synset in WordNet is assigned a category label from a list of 45 semantic fields called *lexnames* (Miller et al., 1990). These are high-level concepts such as *event, feeling, location* for nouns and *change, creation, cognition* for verbs.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Christiane Fellbaum. 1990. English verbs as a semantic net. *International Journal of Lexicography*, 3(4):278–301.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT press.

Christiane Fellbaum. 2002. On the semantics of troponymy. In *The Semantics of Relationships*, pages 23–34. Springer.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 304–309.

David Jurgens and Mohammad Taher Pilehvar. 2016. Semeval-2016 task 14: Semantic taxonomy enrichment. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1092–1102.

Alexandre Kabbach, Kristina Gulordava, and Aurélie Herbelot. 2019. Towards incremental learning of word embeddings using context informativeness. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 162–168.

Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. 2019. Smoothing the geometry of probabilistic box embeddings. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Minh-Thang Luong, Quoc Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2180–2189.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6341–6350. Curran Associates, Inc.

Maximillian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 3776–3785.

Yuval Pinter and Jacob Eisenstein. 2018. Predicting semantic relations using global graph properties. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1751.

Victor Prokhorov, Mohammad Taher Pilehvar, and Nigel Collier. 2019. Generating knowledge graph paths from textual definitions using sequence-to-sequence models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1968–1976.

Tom Richens. 2008. Anomalies in the WordNet verb hierarchy. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 729–736.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, pages 1297–1304.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Dmitry Ustalov, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Negative sampling improves hypernymy extraction based on projection learning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 543–550, Valencia, Spain. Association for Computational Linguistics.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259. Dublin City University and Association for Computational Linguistics.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.