CRIM at SemEval-2018 Task 9: A Hybrid Approach to Hypernym Discovery

Gabriel Bernier-Colborne
g.b.colborne@gmail.com

Caroline Barrière
caroline_barriere@yahoo.ca

Abstract

This report describes the system developed by the CRIM team for the hypernym discovery task at SemEval 2018. This system exploits a combination of supervised projection learning and unsupervised pattern-based hypernym discovery. It was ranked first on the 3 sub-tasks for which we submitted results.

1 Introduction

The goal of the hypernym discovery task at SemEval 2018 is to predict the hypernyms of a query given a large vocabulary of candidate hypernyms. A query can be either a concept (e.g. *cocktail* or *epistemology*) or a named entity (e.g. *Craig Anderson* or *City of Whitehorse*). Two types of data were provided to train the systems: a large unlabeled text corpus and a small training set of examples comprising a query and its hypernyms. More details on this task may be found in the task description paper (Camacho-Collados et al., 2018).

The system developed by the CRIM team for the task of hypernym discovery exploits a combination of two approaches: an unsupervised, pattern-based approach and a supervised, projection learning approach. These two approaches are described in Sections 2 and 3, then Section 4 describes our hybrid system and Section 5 presents our results.

2 Pattern-Based Hypernym Discovery

Pattern-based approaches to relation extraction have been discussed in the literature for quite some time (see surveys by Auger and Barrière (2008) and Nastase et al. (2013)). They can be used to discover various relations, including domain-specific ones (Halskov and Barrière, 2008) and more general ones, such as hypernymy. The pattern-based approach to hypernym discovery was pioneered by Hearst (1992), who defined specific textual patterns (e.g. *Y such as X*) to mine hyponym/hypernym pairs from corpora.

This approach is known to suffer from low recall because it assumes that hyponym/hypernym pairs will occur together in one of these patterns, which is often not the case. For instance, using the training data of sub-task 1A, we found that the majority of training pairs never co-occur within the same paragraph in corpus 1A, let alone within a pattern that suggests hypernymy.

To increase recall, we extend the basic pattern-based approach to hypernym discovery in two ways. First, we identify co-hyponyms for each query and add the hypernyms discovered for these terms to those found for the query. These co-hyponyms are identified using patterns, and filtered based on distributional similarity using the embeddings described in Section 3.3. Furthermore, we discover additional hypernyms using a method based on the following assumptions: most multi-word expressions are compositional, and the prevailing head-modifier relation is hypernymy.

The co-hyponym patterns we use are limited to enumeration patterns (e.g. X_1, X_2 *and* X_3). For hypernyms, we use an extended set of Hearst-like patterns which we selected empirically (e.g. *Y such as X*, *Y other than X*, *not all Y are X*, *Y including X*, *Y especially X*, *Y like X*, *Y for example X*, *Y which includes X*, *X are also Y*, *X are all Y*, *not Y so much as X*).

Our pattern-based hypernym discovery algorithm can be defined as follows: given a query q,

1. Create the empty set Q, which will contain an extended set of queries.

2. Search for the co-hyponym patterns in the corpus to discover co-hyponyms of q. Add these to Q and store their frequency (number of times a given co-hyponym was found using these patterns).

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 725–731
New Orleans, Louisiana, June 5–6, 2018.

3. Score each co-hyponym $q' \in Q$ by multiplying the frequency of q' by the cosine similarity of the embeddings of q and q'. Rank the co-hyponyms in Q according to this score, keep the top n,[1] and discard the rest.

4. Add the original query q to Q.

5. Create the empty set of hypernyms H_q.

6. For each query $q' \in Q$, search for the hypernym patterns in the corpus to discover hypernyms of q'. Add these to H_q.

7. Add the head of each term in H_q to this set, as well as the head of the original query q.

8. Score each candidate $c \in H_q$ by multiplying its normalized frequency[2] by the cosine similarity between the embeddings of c and q, and rank the candidates according to this score.

Although the pattern-based search for both co-hyponyms and hypernyms can find terms not included in the provided vocabulary (which could also be useful), we discarded out-of-vocabulary terms because we had not learned embeddings for them.

3 Learning Projections for Hypernym Discovery

Several supervised learning approaches based on word embeddings have recently been developed for the task of hypernym detection and the related task of hypernym discovery (Camacho-Collados, 2017). The general idea is to learn a function that takes as input the word embeddings of a query q and a candidate hypernym h and outputs the likelihood that there is a hypernymy relationship between q and h. To discover hypernyms for a given query q (rather than classify a given pair of words), we apply this decision function to all candidate hypernyms, and select the most likely candidates (or all those classified as hypernyms).

This decision function can be learned in a supervised fashion using examples of pairs of words that are related by hypernymy and pairs that are not. The supervised model can take as input a combination of the embeddings of q and h, and different ways of combining the embeddings for this purpose have been used (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). In related work, models have been proposed that learn to project the embedding of q such that its projection is close to that of its hypernym h (Fu et al., 2014; Yamane et al., 2016; Espinosa-Anke et al., 2016). This has been termed *projection learning* (Ustalov et al., 2017). The decision function is then based on how close the projection of q is to a given candidate h.

Fu et al. (2014) introduced a model that learns multiple projection matrices representing different kinds of hypernymy relationships. In this model, each (q, h) pair is first assigned to a cluster based on their vector offsets, then projection matrices are learned for each cluster. Based on this work, Yamane et al. (2016) proposed a model that jointly learns the clusters and projection matrices.

We use a similar method to learn projections for hypernym discovery, but our approach differs from that of Yamane et al. (2016) in several ways: our model performs a soft clustering of query-hypernym pairs rather than a hard clustering, and we modified the training algorithm in several ways.

3.1 The Model

Given a query q and a candidate hypernym h, the model retrieves their embeddings $e_q, e_h \in \mathbb{R}^{d \times 1}$ using a lookup table. These embeddings were learned beforehand on a large unlabeled text corpus (i.e. the corpora provided for this task). The embedding e_q is then multiplied by a 3-D tensor containing k square projection matrices $\phi_i \in \mathbb{R}^{d \times d}$ for $i \in \{1, \ldots, k\}$, producing a matrix $P \in \mathbb{R}^{k \times d}$ containing the projections of e_q:

$$P_i = (\phi_i \cdot e_q)^T \quad (1)$$

The model then checks how close each of the k projections of e_q are to e_h by taking the dot product:

$$s = P \cdot e_h \quad (2)$$

The column vector $s \in \mathbb{R}^{k \times 1}$ is then fed to an affine transformation and a sigmoid activation function (in other words, a logistic regression classifier) to obtain an estimate of the likelihood that q and h are related by hypernymy:

$$y = \sigma(W \cdot s + b) \quad (3)$$

To discover the hypernyms of a given query, we compute the likelihood y for all candidates and select the top-ranked ones.

[1] We set $n = 5$ empirically.

[2] Frequencies were normalized in the range $[0.05, 1.0]$.

3.2 The Training Algorithm

We train the model using negative sampling: for each positive example of a (query, hypernym) pair in the training data, we generate a fixed number m of negative examples by replacing the hypernym with a word randomly drawn from the vocabulary.[3] We then train the model to output a likelihood (y) close to 1 for positive examples and close to 0 for negative examples. This is accomplished by minimizing the binary cross-entropy of the positive and negative training examples. For a particular example, this is computed as follows:

$$H(q, h, t) = t \times \log(y) + (1 - t) \times \log(1 - y)$$

where q is a query, h is a candidate hypernym, t is the target (1 for positive examples, 0 for negative), and y is the likelihood predicted by the model. If we sum H for every example in the training set D (containing both the positive and negative examples), we obtain the cost function $J = \sum_{(q,h,t)\in D} H(q, h, t)$. This function is minimized by gradient descent, using the Adam optimizer[4] (Kingma and Ba, 2014).

A few details of the setup we use for training are worth mentioning:

- We use a fixed number of projections (k) rather than the dynamic clustering algorithm of Yamane et al. (2016). For our official runs, we used $k = 24$.
- The word embeddings are normalized to unit-length before training.
- For the initialization of the projection matrices, we add random noise to an identity matrix, which means that at first, the projections of a query are simply k randomly corrupted copies of the query's embedding.
- We train the model on random mini-batches containing 32 positive examples and $32 \times m$ negative examples (m being the number of negative examples).
- Dropout is applied to the embeddings e_q and e_h and the query projections P. For regularization, we also use gradient clipping, as well as early stopping.
- We sample positive examples using a function based on the frequency of the hypernyms in the training data, such that we subsample (q, h) pairs where h occurs often in the training data. The probability of sampling (q, h) is given by:

$$P(q, h) = \sqrt{\frac{\min_{h' \in D}(\text{freq}(h'))}{\text{freq}(h)}}$$

  where $\text{freq}(h)$ returns the frequency of h in the training data.[5]
- The word embeddings are optimized (or "fine-tuned") during training.
- We use a multi-task learning setup whereby we train two separate logistic regression classifiers, each with their own parameters W and b, and use one for queries that are named entities, and the other for queries that are concepts.[6] The rest of the parameters (i.e. the projection matrices ϕ) are shared.

The various hyperparameters mentioned above were tuned on the trial set (i.e. development set) provided for sub-task 1A.

3.3 The Word Embeddings

We learned term embeddings for all queries and candidates using the pre-tokenized corpora provided for sub-tasks 1A, 2A, and 2B. We preprocessed the corpora by converting all characters to lower case and replacing multi-word terms found in the vocabulary (candidates and lowercased queries) with a single token, starting with trigrams, then bigrams.[7] We then used the skip-gram algorithm with negative sampling (Mikolov et al., 2013) to learn the embeddings.

3.4 Data Augmentation

For one of our 2 runs, we experimented with a method to add synthetic examples to the positive examples in the training set provided (D). This

[3] Different ways of selecting the negative examples for this purpose have been proposed. See Ustalov et al. (2017).

[4] We use $\beta_1 = \beta_2 = 0.9$.

[5] On the training data of subtask 1A, this produces a sampling probability of 0.06 for the most frequent hypernym (*person*), while the least frequent hypernyms have a sampling probability of 1.

[6] Multi-task learning was not used on subtask 2A, because all queries were concepts.

[7] It is worth noting that a small number of candidates (e.g. less than 0.1% of candidates on corpus 1A) had a frequency of 0 in this preprocessed corpus, so we could not train an embedding for these candidates. These appear to be terms that always occur within a longer candidate.

was meant to provide additional training data and avoid overfitting the embeddings of the words in the training set. We use 2 heuristics to generate these synthetic examples:

1. Given a positive example $(q, h) \in D$, add (q', h) to the positive examples, where q' is the nearest neighbour of q, based on the cosine similarity of the embeddings of all the words in the vocabulary. This was motivated by the observation that nearest neighbours were often co-hyponyms.

2. Given a query q and the set H_q containing the hypernyms of q according to the training data, compute the α nearest neighbours of each hypernym in H_q, and for each neighbour that is shared by at least 2 of the hypernyms in H_q, add that neighbour to H_q.[8]

Negative examples are generated for each of the synthetic examples, as with the actual positive examples in the training set.

4 Hybrid Hypernym Discovery

Our hybrid approach to hypernym discovery combines supervised projection learning and unsupervised pattern-based hypernym discovery (see Sections 2 and 3). To combine the outputs of the 2 systems, we take the top 100 candidates according to each,[9] normalize their scores and sum them, then rerank the candidates according to this new score. This reranking function favours candidates found by both systems, but also gives a chance to strong candidates found by a single system.

5 Experiments and Results

We submitted 2 runs on 3 of the 5 sub-tasks: 1A (general), 2A (medical), and 2B (music). The system outputs its top 15 predictions in all cases. The difference between the 2 runs is that for run 1, we used data augmentation (see Section 3.4) to train the supervised system – the same unsupervised output was used for both runs. We also submitted one run for cross-evaluation (training on 1A, but testing on 2A or 2B). First, we added the queries and candidates of 2A or 2B to those of 1A before training embeddings on the corpus of 1A.[10] These embeddings were used to train the supervised model on the 1A training data. We then combined the predictions of the supervised and unsupervised models on test set 2A/2B.[11]

A summary of our system's results is shown in Table 1. This table shows the mean average precision (MAP), mean reciprocal rank (MRR) and precision at rank 1 (P@1) of our system and those of the 2 strongest baselines which were computed by the task organizers. The first is a supervised baseline[12] and the second is based on the most frequent hypernyms in the training data. For more details, see (Camacho-Collados et al., 2018).

Our hybrid system was ranked 1st on all three sub-tasks for which we submitted runs. As shown in Table 1, the scores obtained using this system are much higher than the strongest baselines for this task. Furthermore, it is likely that we could improve our scores on 2A and 2B, since we only tuned the system on 1A.

If we compare runs 1 and 2 of our hybrid system, we see that data augmentation improved our scores slightly on 1A and 2B, and increased them by several points on 2A.

Our cross-evaluation results are better than the supervised baseline computed using the normal evaluation setup, so training our system on general-purpose data produced better results on a domain-specific test set than a strong, supervised baseline trained on the domain-specific data.

Table 1 also shows the scores we would have obtained on the test set if we had used only the unsupervised (pattern-based) or supervised (projection learning) parts of our system. Note that the unsupervised system outperformed all other unsupervised systems evaluated on this task, and even outperformed the supervised baseline on 2A.

Combining the outputs of the 2 systems improves the best score of either system on all test sets, sometimes by as much as 10 points.

Notice also that the results obtained using only the supervised system indicate that data augmentation had a positive effect on our 2A scores only (compare runs 1 and 2), although our tests on the

[8] We use $\alpha = 2$.

[9] Analyzing the individual and combined recall of the two systems at various ranks indicated that using more than 100 candidates would not increase recall.

[10] Several of the domain-specific queries that were added to the vocab were not found in corpus 1A. We decided to only use the output of the unsupervised system in these cases.

[11] Note that the unsupervised system used the corpora of 2A/2B, but no supervised learning was carried out on the training data of 2A/2B.

[12] This is a "vanilla" version of TaxoEmbed (Espinosa-Anke et al., 2016).

	Test set 1A			Test set 2A			Test set 2B		
	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1
Hybrid run 1	19.78	36.10	29.67	34.05	54.64	49.20	40.97	60.93	48.20
Hybrid run 2	19.54	35.94	29.67	31.54	46.19	41.40	40.88	60.18	47.60
Supervised run 1	19.09	34.53	28.33	30.63	44.18	40.00	38.24	53.45	38.20
Supervised run 2	19.11	34.99	28.80	28.51	37.63	34.40	39.95	57.34	43.00
Unsupervised	7.36	15.44	10.73	21.74	47.85	38.60	15.86	34.98	28.60
BaselineSUP	10.60	23.83	19.73	18.84	41.07	35.40	12.99	39.36	33.20
BaselineMFH	8.77	21.39	19.80	28.93	35.80	32.60	33.32	51.48	36.20
Hybrid cross-eval	N/A	N/A	N/A	27.18	49.51	43.20	21.02	37.55	29.20
Supervised cross-eval	N/A	N/A	N/A	22.89	38.30	30.60	16.80	29.28	19.20
BaselineSUP cross-eval	N/A	N/A	N/A	11.66	23.83	17.20	6.31	16.54	9.60

Table 1: Our system's results on test sets 1A, 2A, and 2B. The runs we submitted are the hybrid runs. The supervised and unsupervised runs were produced by using our 2 sub-systems separately. BaselineSUP is a strong, supervised baseline and BaselineMFH is the most-frequent-hypernym baseline. Cross-evaluation results were obtained by training the supervised system on 1A and evaluating on 2A or 2B.

trial set suggested it would also have a positive effect on our 1A scores. Given this observation, we find it somewhat surprising that run 1 is the best on all 3 test sets when we use the hybrid system. One possible explanation is that adding the synthetic examples makes the errors of the supervised system *more different* from those of the unsupervised system, and that this in turn makes the ensemble method more beneficial, but we haven't looked into this.

5.1 Ablation Tests

To assess the influence of different aspects of the supervised system and its training algorithm, we carried out a few simple ablation tests on subtask 1A. The baseline for these tests is our supervised projection learning system – we did not apply pattern-based hypernym discovery for any of these tests. We used the setup of run 1 (with data augmentation) and used the trial set for early stopping. We conducted the following tests (one by one, without combining any of the ablations):

1. No subsampling: we sample positive examples uniformly from the training set.
2. No MTL: instead of multi-task learning (MTL), we use a single classifier for both named entities and concepts.
3. Random init: the weights of ϕ are initialized randomly, instead of adding random noise to an identity matrix.
4. Single projection: $k = 1$ instead of 24.
5. Single neg. example: $m = 1$ instead of 10.
6. Frozen embeddings: the word embeddings are not fine-tuned during training.

The results obtained on test set 1A are shown in Table 2.[13] These results show that 2 of the techniques we used, namely subsampling and multi-task learning, actually harmed our system's performance on test set 1A, although our experiments on the trial set suggested that they would be beneficial. This may be due to the small size of the trial set (i.e. 50 queries) or some difference in the underlying distributions of the trial and test sets.

	MAP	MRR	P@1
Baseline	19.05	34.36	27.93
No subsampling	19.47	35.56	29.33
No MTL	19.22	35.09	28.67
Random init	16.54	30.60	24.93
Single projection	12.88	26.50	23.33
Single neg. example	11.58	21.75	15.20
Frozen embeddings	8.01	17.15	11.67

Table 2: Results of ablation tests on test set 1A. The baseline is our supervised system (run 1).

On the other hand, fine-tuning the word embeddings during training seems to be one of the keys to the success of this approach, as are the use of multiple projection matrices, and the sampling of

[13] Note that the baseline results are slightly different than those shown in Table 1 for run 1 (supervised), because we retrained the model to get the results on the trial set, and a random number generator used during training was not seeded with a fixed value.

Query	Predictions
Suzy Favor Hamilton	**athlete**, **sportsperson**, **person**, competitor, sport, olympic sport, ...
wicketkeeper	**cricketer**, **sportsperson**, **athlete**, **competitor**, footballer, **person**, ...
aquamarine	**stone**, crystal, precious stone, **pebble**, gem, **rock**, **gemstone**, ...
tenpence	monetary unit, **metal money**, **note of hand**, person, silver coin, **coin**, ...
vegetarian	dessert, dish, recipe, veggie, food product, organic food, meal, salad, ...
Local Group	voluntary association, locale, coalition, club, country, mapmaking, ...
Swarthmore	university, college, educational institution, school, student, ...
hypostasis	figure of speech, intellection, philosophy, ordinary language, ...

Table 3: Examples of predictions made by our system (run 1) on the test queries of 1A. Correct predictions are in bold. Midline separates high-accuracy examples from low-accuracy examples.

multiple negative examples for each positive example. The way we initialize ϕ also seems to have helped quite a bit.

It is important to remember that a more thorough exploration of hyperparameter space would produce results very different from those of simple ablation tests such as these.

It is worth noting that our supervised model outperforms the supervised baseline provided for this task (see Table 1) even when it exploits a single projection matrix, however the difference in scores between these 2 systems is only 2 or 3 points, depending on the evaluation metric.

We should also note that the supervised model is prone to overfitting, and we found early stopping to be particularly important.

5.2 Qualitative Analysis

We manually inspected the results of run 1 on some of the 1A test queries to get a better idea of the ability of the model to discover hypernyms, and to identify potential sources of errors. Table 3 shows a few of these test cases. Below we will outline a few of our observations, and will refer repeatedly to examples in Table 3.

The model is indeed able to discover valid hypernyms for both concepts and named entities. It seems that it can even handle very low-frequency queries in some cases (*Suzy Favor Hamilton* occurs only 5 times in the corpus), but we have not had the chance to investigate how sensitive the model is to term frequency.

Lexical memorization (Levy et al., 2015) can sometimes be observed. For example, *person* is the most frequent hypernym in the 1A training data, and the model often predicts this candidate incorrectly, even when its other top predictions are completely unrelated (e.g. *tenpence*).

The model can discover hypernyms of different senses of the same query (e.g. *aquamarine*, for which the top 15 predictions contain the valid hypernyms *spectral color* and *primary color*), and it sometimes discovers hypernyms for senses that are not represented in the gold standard (e.g. there is a college named *Swarthmore*, and *hypostasis* has senses related to linguistics and philosophy). It is likely that the senses that dominate the model's top predictions for a given query are its most frequent senses in the corpus.

The case of *vegetarian* suggests that syntactic ambiguity is a source of errors: the predicted hypernyms include some that might be considered valid for the query *vegetarian food*, where *vegetarian* is an adjective, but not for the noun *vegetarian*.

Lastly, the model sometimes confuses concepts and named entities (e.g. *Local Group*, which refers to a group of galaxies). Preserving the true case of the characters in the corpus would mitigate this issue.

6 Concluding Remarks

Our approach to hypernym discovery combines a novel supervised projection learning algorithm and an unsupervised pattern-based algorithm which exploits co-hyponyms in its search for hypernyms. This hybrid approach produced very good results on the hypernym discovery task, and was ranked first on all 3 sub-tasks for which we submitted results.

Acknowledgments

This research was conducted while both authors were affiliated with the Computer Research Institute of Montréal (CRIM), and was supported by the Natural Sciences and Engineering Research Council of Canada.

References

Alain Auger and Caroline Barrière. 2008. Pattern-based approaches to semantic relation extraction: A state-of-the-art. *Terminology, Special Issue on Pattern-based Approaches to Semantic Relation Extraction*, 14(1):1–19.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.

José Camacho-Collados. 2017. Why we have switched from building full-fledged taxonomies to *simply* detecting hypernymy relations. *CoRR*, abs/1703.04178.

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)*, New Orleans, LA, United States. Association for Computational Linguistics.

Luis Espinosa-Anke, Jose Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. 2016. Supervised distributional hypernym discovery via domain adaptation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 424–435.

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)*, volume 1, pages 1199–1209.

Jakob Halskov and Caroline Barrière. 2008. Web-based extraction of semantic relation instances for terminology work. *Terminology*, 14(1):20–44.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Vivi Nastase, Preslav Nakov, Diarmuid Ó Séaghdha, and Stan Szpakowicz. 2013. *Semantic Relations Between Nominals*. Morgan & Claypool, Toronto.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036.

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.

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

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.