Hypernymy Extraction*: Extensive Exploration of Several Methods

FAN, Wenlei
20710754

December, 2020

Abstract

Hypernymy extraction is one the fundamental tasks in many NLP fields such as Question Answering, Textual Entailment as well as the backbone of some models in taxonomy, ontology, and semantic related fields. In this project, I mainly make an attempt to extract hypernymy through two kinds of models, pattern based and learning based. Pattern based methods start from manual hearst patterns, then apply PPMI and SVD techniques on those extracted pairs for further detection. At the same time, distributional methods based on several hypernymy similarity measures are also explored to compare with prior pattern based ones. In supervised learning models, term embeddings and projection learning are tried and both achieve F1 score over 0.7, where more attempts like ablation tests are performed to get a closer look at model performance. Finally, I ended up my extraction research with the Bi-LSTM sequence labeling model which is able to capture semantic contexts. Here we cover a range of extraction methods, from classic to state-of-the-art, and focus on those learning ones. The final results of these methods all get an F1 score higher than 0.7.

*This project mainly focus on two occasions that either the occurrence of hyponym and hypernym are both assumed available or given a query, we extract hypernyms from candidate vocabulary set.

1 Introduction

The problem of hypernymy extraction originates from the simple instance like apple is a kind of fruit, then an upper and lower relationship ***isa*** (apple, fruit) is identified as a kind of hyponymy. Early works mainly mining such relation based on semantic patterns such as Hearst, 1992 [1], which was so pioneering that it has inspired a series of pattern based hyponymy extraction methods. Then patterns got to generated automatically and a series of knowledge base such as Probase, Wikidata, DBpedia were constructed. More recently, classification methods were proposed to extract is-a and not-is-a relationships. Supervised learning is another stream of hypernymy extraction methods where word embedding is often employed.

In this project, I will mainly try 6 categories of models here, with the first three unsupervised and the latter three supervised. In unsupervised approaches, manual acquisition method using Hearst Patterns is the most fundamental one. Based on extracted candidates, metrics like co-occurrence matrix is constructed, then (positive) pointwise mutual information (PPMI) or dimension reduction techniques like Principal Component Analysis (PCA), Singular Value Decomposition (SVD) are able to apply to address scalability problems. Distributional methods extract hypernymy pairs by measuring their similarity in terms of hypernymy relation. These methods have already shown superiority in hypernymy extraction. Word embedding is another kind of popular technique in natural language processing and

prompts learning methods. However, traditional embedding methods only capture the semantic relation that words have close meaning tend to be closer in the vector space. Here, we try a new term embedding which is not only based on co-occurrence, but also tell hypernymy relation. After that, projection learning even goes beyond that and gives more advancement in the extraction procedure. Finally, we implement a bidirectional LSTM model to better encode semantic context by sequence labeling.

Roadmap. The remaining part of this report is organized as follows: section II introduces the models that will be implemented in this project and section III gives the specific implementation details especially for data preprocessing. In section IV, we give the metrics to evaluate the methods tried in this project. And finally we give the evaluation results in section V.

1.1 Datasets

The public benchmark datasets used in this project include Probase, BLESS, SemEval2018-Task9 (Hypernym Discovery) and some other simulated data to research specific models.

1.1.1 Probase

Probase[1] contains concept knowledge generated from web pages automatically. It contains 2.7 million concepts from a corpus of 1.68 billion web pages [16]. This taxonomy is considered the largest, most comprehensive and precise knowledge base currently which is built on probabilistic model. In this project, we trained term embeddings on Probase dataset. The data used could be downloaded directly from the link below.

1.1.2 BLESS

BLESS[2] (Baroni-Lenci Evaluation of Semantic Similarity) dataset contains 200 concrete nouns from a range of classes including animals, clothing, tools and so on. Among that, 100 nouns are animated with semantc relations such as hypernymy, meronymy, random pairs and so on.

Relation Type	**Example** (predicate, related word)
Hypernymy	(yacht, boat)
Cohyponymy	(yacht, sailboat)
Meronymy	(yacht, sail)
Typical attribute	(yacht, expensive)
Typical related event	(yacht, cruise)
Random	(yacht, justice)

Table 1: Relation examples of BLESS dataset.

Task	# of gold standard labeled terms
English (subtask 1A)	3,000
Italian (subtask 1B)	2,000
Spanish (subtask 1C)	2,000
Medical (subtask 2A)	1,000
Music (subtask 2B)	1,000

Table 2: SemEval2018-Task9 Dataset task distribution.

We take advantage of BLESS and Probase in section 2.4, where we trained embeddings on Probase and then trained an SVM classifier on BLESS to perform hypernymy classification.

1.1.3 SemEval2018-Task9

The SemEval2018 Task on Hypernym Discovery[3] is formulated as follows [2]: given a query term, retrieve (or discover) its suitable hypernyms from a target corpus. This task is available for three languages and two specific domains.

The gold standard consists of terms along with their corresponding hypernyms (up to trigrams). Training and testing data are split evenly (50% training - 50% testing).

Specifically, for Domain-Specific Hypernym Discovery subtask 2B, corpus including 100M-word

[1] https://concept.research.microsoft.com/Home/Download.

[2] https://sites.google.com/site/geometricalmodels/shared-evaluation.

[3] https://competitions.codalab.org/competitions/17119#learn_the_details-terms_and_conditions.

Set	1A	1B	1C	2A	2B
Trial	50	25	25	15	15
Training	1,500	1,000	1,000	500	500
Test	1,500	1,000	1,000	500	500

Table 3: # of hyponyms for each dataset in trial, train and test sets

Amazon reviews, music biographies and Wikipedia pages about theory and music genres. We make use of this dataset and its corpus in section 2.5 to do projection learning.

2 Models

In the following, we sequentially introduce Hearst Pattern method, pattern-based methods by means of PPMI and SVD improved ones, distributional models using different hypernymy similarity measures to plainly detect hypernymy pairs. After that, more sophisticated models such as term embeddings and projection learning are studied to encode hypernymy relationship either in the embedding space or vector projection space. Finally, we use a Bi-LSTM sequence labeling model to predict hypernyms through tagged terms.

2.1 Hearst Patterns

This manual acquisition method initiated by Hearst, 1992 [1] was an inspiring one. Extraction was primarily based on semantic patterns which occur frequently in many raw text genres. Such plain method required little pre-encoded knowledge and was quite straightforward to implement in program that it is still considered and extended to this day [3].

A common example goes as 'NP_0[4] such as $\{NP_1, NP_2, \ldots, (\text{and}|or)\}\text{NP}_n$' implies for all NP_i, $1 \leq i \leq n$, there can be concluded that hyponym(NP_i, NP_0). Then (NP_i, NP_0) is the target pair we want to extract in text. See more other hearst patterns in use in section 3.1.

[4] 'NP' refers to Noun Phrase.

2.2 Pattern-Based Approaches

For the first glance, hearst patterns seems to be relatively successful in relation recognition in raw text. However, with semantics getting increasingly richer, there comes at least two drawbacks:

(i) The fact that hypernymy pairs acquired are coarse-grained and low recall precipitates more sophisticated lexical parser.

(ii) Discovery of new patterns are expensive although some traditional methods [1, 4] have shown effective results.

Under such contexts, both updated hearst paterns and improvements in method are developed.

Pattern-based approaches detect hypernymy pairs through pre-extracted ones via hearst patterns. This class of methods rely on the co-occurrence probability of hypernymy pairs in corpus. And their performance in hypernymy related tasks are reported even better than other approaches [3].

Their implementation mainly includes the following steps:

Step1. The extraction probability.

$$p(x, y) = \frac{w(x,y)}{W}$$

,where $w(x, y)$ refers to the counts of hypernymy pair occurred in corpus and W is the total number of extraction.

Step2. Positive Pointwise Mutual Information (PPMI) transformation.

$$ppmi(x, y) = max(0, log\frac{p(x,y)}{p^-(x)p^+(y)})$$

, where $p^-(x)$ refers to the counts of x appeared as a hyponym in pair (x, y) and $p^+(x)$ is the counts of x as a hypernym in pair (x, y).

Step3. Truncated Singular Value Decomposition (SVD).

$$spmi(x, y) = \mathbf{u}_x{}^T \sum_r \mathbf{v}_y$$

, where $\mathbf{u}_x$, $\mathbf{v}_y$ refer to the x-th, y-th row of U, V. $\sum_r$ is the diagonal matrix of truncated singular values with the remaining singular values set to zero.

Through the above three steps, this class of models predict hypernym pairs by giving a similarity measure in terms of hypernymy.

2.3 Distributional Approaches

Distributional similarity measuring models are another class of popular unsupervised hypernymy extraction methods which mainly based on the Distributional Inclusion Hypothesis (DIH). Then the goal of the models is to measure how much the hyponym is included in hypernym semantically via their vector representations in distributional space. Roller et al., 2018 [3] have concluded several frequently used models based on previous papers.

Here we collected these models [3, 5, 6] in a list as follows:

$$WeedsPrec(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i \cdot \mathbf{1}_{y_i>0}}{\sum_{i=1}^{n} x_i}$$

$$ClarkeDE(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} min(x_i, y_i)}{\sum_{i=1}^{n} x_i} \qquad (\mathbf{M})$$

$$invCL(\mathbf{x}, \mathbf{y}) = \sqrt{\mathbf{M}(\mathbf{x}, \mathbf{y}) \cdot (1 - \mathbf{M}(\mathbf{y}, \mathbf{x}))}$$

$$cosine(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\sqrt{\|\mathbf{x}\|_2 \cdot \|\mathbf{y}\|_2}}$$

$$SLQS(\mathbf{x}, \mathbf{y}) = 1 - \frac{E_x}{E_y}$$

$$SLQScos(\mathbf{x}, \mathbf{y}) = SLQS(\mathbf{x}, \mathbf{y}) \cdot cosine(\mathbf{x}, \mathbf{y})$$

, where E_x in the above is the median cross entropy of a term's top N contexts as $E_x = median_{i=1}^{N}[H(c_i)]$, and here N is a hyperparameter and $H(c_i)$ is the Shannon entropy of context c_i.

Pros and cons. Although distributional methods are preferable in many hypernymy related subtasks like hypernym detection, direction prediction and entailment by their simplicity in representing semantic inclusion meanings via distributional vectors, their performance are not so satisfying compared to even pattern based one in many challenging benchmarks [3]. One direction is to treat distributional models as a compensate for pattern based methods into a compound system with which one dominating the other according to the extraction situation they prefer [7]. Another direction is to resort to embeddings for hypernym discovery, which is what we choose to explore in this project.

2.4 Term Embeddings

Traditional count-based embedding models are trained on the co-occurrence of words within their contexts so that words in similar meanings tend to appear closer on the embedding plot. Basically, there's a sliding window (with target embedding word localizing at the center) to determine words that co-appear with the target word. Then co-occurrence matrix is constructed. Usually dimension reduction techniques like PCA or SVD-based ones are employed here to alleviate sparsity and scalability problems caused by large vocabulary as well as assist in embedding plotting on a 2-dimension occasion.

Modern embedding approaches like Word2Vec[5] (which we choose to use in later supervised learning models), FastText[6] and GloVe[7] are learning based, which also rely on the occurrence of contexts.

In this project, we explore another kind of embedding which encodes hypernymy relationship [8]. Such embedding based method improves extraction accuracy by supervision. In section 2.4.1, a neural network based model is introduced to encode hypernymy properties, and in section 2.4.2, a Support Vector Machine (SVM) is applied to identify positive hypernymy pairs.

2.4.1 The Embedding Trainer

Here we describe a distance-margin neural network to learn the embeddings supervised by some pre-extracted hypernymy pairs.

Notation:

Embedding as a map function from $TermSet \to R^d$:

· $O(x)$ - The hyp**O**nym embedding of term x.

· $E(x)$ - The hyp**E**rnym embedding of term x.

The goal is to learning embeddings to represent hypernymy relation. Thus the notation we used has three

Properities:

[5] https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec.

[6] https://radimrehurek.com/gensim/models/fasttext.html.

[7] https://github.com/stanfordnlp/GloVe.

i. hyponym-hypernym measure:
$$O(hypo) \sim E(hyper).$$
ii. co-hyponymy measure:
$$O(hypo1) \sim O(hypo2).$$
iii. co-hypernymy measure:
$$E(hyper1) \sim E(hyper2).$$

Then comes the learning rule as that to ensure $O(hypo)$ close to $E(hyper)$, in which case we use a distance measure function
$$f(x) = \|O(u) - E(v)\|_1$$
, where $x = (u, v, q)$, q is the frequency of its corresponding hypernymy pair.

Dynamic Distance-Margin Model.

Objective Function:
$$f(x) \leq f(x^{'}) - m(x, x^{'})$$
is one pairwise training where $m(x, x^{'})$ is the margin between positive hypernymy pair x and negative hypernymy pair $x^{'}$. Both $(v^{'}, u, 0)$ and $(v, u^{'}, 0)$ are negative hypernymy pairs.

Loss Function:
$$J = \sum_{x=(u,v,q)} \sum_{j=1}^{q} max(0, f(x) - f(x_j^{'}) + m(x, x_j^{'}))$$
is the total loss across all training data.

According to Yu et al, 2015 [8], the above loss function is minimized through stochastic gradient descent (SGD) over a neural network model. Such network architecture is simply a feedforward neural network (FNN) with parameters fixed, so we will not give details in this model section and implement it directly to train the loss function and finally give the updated embeddings.

2.4.2 Hypernymy Identification

As there's no threshold between $f(x)$ and $f(x^{'})$, an SVM classifier may be appropriate to maximum the distance between $f(x)$ and $f(x^{'})$ plus additional signals. Therefore, we input the embedding pairs, standardize the distance function f and train the classifier for hypernymy identification.

The resulting embedding transcends traditional embeddings which only tell the frequent occurrence of pairs and encodes hypernymy property. The output of the identification system gives prediction whether a pair is hypernymy positive one or not.

2.5 Projection Learning

The idea of projection learning starts from the fact that word embedding is able to capture some linguistic regularity as well as hypernymy relation. Fu et al. (2015) [9] proposed a two-stage projection learning methods with the first step to cluster and second projection. The separation of clustering and projection plus the potential inappropriateness of clustering may lead to unclear hypernymy learning results. Thus, Yamane et al. (2016) [10] proposed a joint learning model based on similarity measure in clusters. The paper we follow [11] in this project still differs from the previous two, although the common goal is to learn clustering and projection matrices in an appropriate manner. The general idea of this model is to calculate the likelihood that a hypernymy relation between a pair of phrases. Since this is a classification setting, a decision function is given based on the closeness of query and candidate hypernym. The input of projection model is the pretrained embeddings of a query q and a candidate hypernym h, namely e_q, $e_h \in R^{d\times 1}$. Then the model is given by
$$P_i = (\phi_i \cdot e_q)^T$$
in each direction $i \in 1, ..., k$, where $\phi_i \in R^{d\times d}$ is a 3-D k square projection matrics. Then the model checks how close between each of the k projections of e_q and e_h by
$$s = P \cdot e_h,$$
where column vector $s \in R^{k\times 1}$ and is then used to calculate the likelihood that q and h are hypernymy related through an affine transformation and a sigmoid activation:
$$y = \sigma(W \cdot s + b).$$
We need to compute the likelihood y for all candidates h and select the top-ranked words.

Training is done by adding negative sampling, which is a technique used to improve projection learning [12]. Here in the model, for each positive example (query, hypernym), generate a fixed number of m negative examples by replacing the hypernym with a random sample of a word in vocabulary. The likelihood is then trained to close to 1 for positive instances, and 0 for negative ones, which is accomplished by minimizing the binary cross-entropy between positive and negative instances.

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

,where q, h represents query and candidate hypernym, t is 1 for positive instances, 0 for negative ones. Likelihood y is calculated as before. The cost function is then given by summing H for every instance in training set D:

$$J = \sum_{(q,h,t)\in D} H(q,h,t),$$

which is the familiar form that can be solved by gradient descent (GD).

2.6 Deep Learning Methods

Deep learning models have the potential in relation extraction either under large datasets or through distant supervision. Traditional deep learning models tend to involve embeddings at both word and positional levels, supervised learning based on Convolutional Neural Networks (CNNs), or other learning methods through distant supervision and have shown good performance over non deep learning models [17]. Thus, in this project, we plan to try a Bi-LSTM model for hypernymy classification by means of sequence labeling.

Bidirectional long short term memory (Bi-LSTM) model is one of the most popular methods in sequence labeling [13]. Although LSTM has already been able to encode long term dependencies in sequence, Bi-LSTM goes beyond LSTM by its forward and backward architecture so that it can capture both past (left) and future (right) contexts [14]. Here we take advantage of Bi-LSTM model to do some sequence tagging tasks. Given a set of sentences as sequences, the Bi-LSTM outputs tag sequences. We then use IOB2 to encode the tags as [B-TAR, I-TAR, O, O, B-HYP, O] to be the corresponding output tagged sequence. We find hyponyms by tag 'TAR' and hypernyms by tag 'HYP'. For example, given the input tokenized sentence [Stephen, Hawking, is, a, physicist, .], the output then be [B-TAR, I-TAR, O, O, B-HYP, O], where we find the hypernym physicist and hyponym Stephen Hawking.

In addition, we also include a character embedding layer concatenating to the word embedding after data preprocessing by LSTM. The model architecture looks as follows.

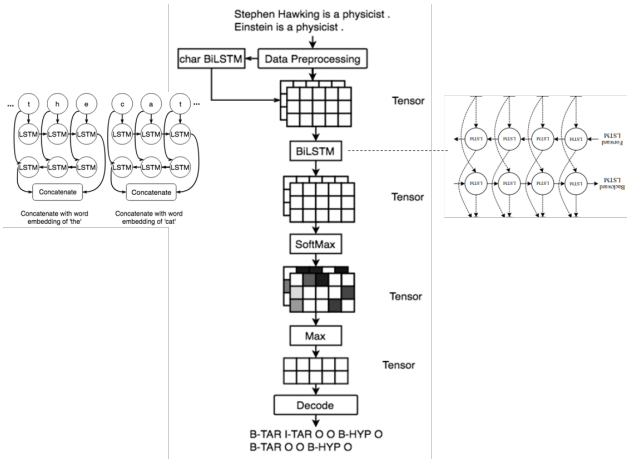

Figure 1: Sample Sequence Labeling Model Architecture.

3 Implementation Details

Setup. As there's some difference in the preprocessing stage of each part of task, we provide implementation details in each section separately.

3.1 Hearst Patterns

In this problem setting, I just extract hypernymies from raw text after necessary tokenization, POS tagging, NP chunking, which are finished by means of the nltk[8] library. This method mainly relies on hearst patterns to match candidate hypernymy pairs. Table 4 gives the list of hearst patterns applied in this project.

3.2 Pattern-based Models

Data is prepared by hypernymy pairs with co-occurrence counts across the corpora. The extraction method is still hearst patterns. Postprocessing even removes pairs that were not extracted by at least two distinct patterns. Final hypernymy pairs guarantee 10,860 MB matched pairs. We first give the PPMI value of candidate pairs, then SVD transformed value

[8] https://www.nltk.org/.

Patterns
NP such as NP, NP, ..., (and‖or) NP
such NP as NP,*(or‖and) NP
NP ,NP*, or other NP; NP ,NP*, and other NP
NP , including NP,*or‖and NP
NP , especially NP,*or‖and NP
NP which is a (example‖class‖kind‖...) of NP
NP (and‖or)(any‖some) other NP
NP which is called NP
NP is a special case of NP
.....

Table 4: Hearst Patterns Used in this Project.

where we try a range of rank k in {5, 10, 15, 20, 25, 50, 100, 150, 200, 500}.

3.3 Distributional Models.

Simulated data is prepared by that vocabulary is tokenized, POS tagged, and named entity recognized from its contexts – here to be sentences extracted from the corpora. Then words in vocabulary are labeled *EntityID*, *PreferredName*, *AllNames* (seperated by '‖'). And corresponding sampled sentences are also labeled *SentenceID* and *Sentence*. Besides, there's also a lookup between *EntityID* and *SentenceID* in order to build distribution matrix, where each row contains each word, and each col contains their occurring contexts. For computation concise, they are all represented by id in loading matrix. Then distributional inclusive models are applied.

3.4 Term Embeddings

In this part, the neural network based embedding trainer is trained on Probase and the SVM hypernymy identification classifer is trained on BLESS. Both of the dataset requires some preprocessing. For Probase, we only split the hypernym, hyponym and counts; while for BLESS, there're some trival transformation to encode the relation type.

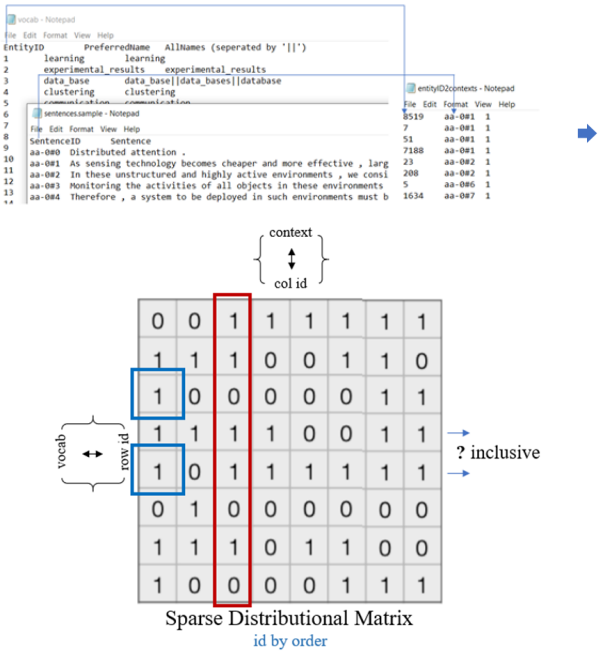

Figure 2: Distributional Models Preprocessing Workflow

3.5 Projection Learning

In this part, projection learning was applied to extract hypernyms on the SemEval2018-Task9 dataset, which is divided into training part, trial part and test part. In each part, data file contains hyponyms and gold file contains standard hypernyms with each line corresponding to the same line in their data file. Vocabulary contains the full space of candidate hypernyms up to trigrams (gold $\subset$ vocabulary). There's also corresponding corpus file downloading from the website. For the sake of memory, I considered subtask 2B – Music *Domain-Specific Hypernym Discovery* here, which only requires 500MB. Corpus preprocessing includes parsing all the queried vocabulary (including hyponyms queries and total hypernyms candidates) n-grams in corpus and replace those multi-grams with single tokens. For those out of vocabulary (OOV) words in corpus, choose to replace them with sign $\langle UNK \rangle$, so that only quried

vocabulary was targeted at embedding. Vocabulary frequency counts are also recorded. After the above preprocessing, word embeddings were trained on corpus using Word2Vec in genism. Then vocabulary were transformed into embeddings for latter model training.

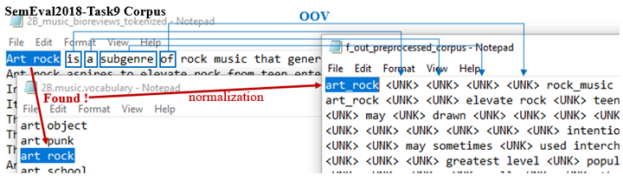

Figure 3: SemEval2018-Task9 Data Preprocess Procedure.

Some training details are provided as follows: firstly, the number of projections (k) is fixed; Secondly, word embeddings are normalized to unit length before training. Thirdly, the initialization of projection matrices are just some corrupted embeddings. Fourthly, the model is trained on random mini-batches with 32 positive instances and 32×m negative ones. Finally, techniques such as dropout and early stopping is also applied. Last but not least, positive instances sampling probability is

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

where $freq(h)$ refers to frequency of h. We use Word2Vec to perform word embedding in this part. The final prediction on test queries return 15 top candidate hypernyms for each query word.

3.6 Bi-LSTM Model

The training, dev and test set is a small dataset with IOB2 (Inside–Outside–Beginning) tags labelled with a tags file lists tag vocabulary. Each line has two columns with first to be word, along with the tag. The tokens are separated by blank. Besides, there's pretrained word embeddings (50 dimensions) for words and chars this time. Data is preprocessed into batches before feeding the model to accelerate training. $\langle PAD \rangle$ is used to pad words at the end of each sentences, and $\langle UNK\_WORD \rangle$ is used to replace out-of-vocabulary (OOV) words. In pretrained embeddings, $\langle UNK\_WORD \rangle$ is offered, while $\langle PAD \rangle$ is randomly initialized.

4 Evaluations

Due to the reason of limited corpora and labelled sequence, the dataset used in each section of model is not unified, which makes the slight difference in evaluation matrics.

F1 score. Here, we dnote

TP=True Positive predictions
TN=True Negative predictions
FP=False Positive predictions
FN=False Negative predictions

Then,

$$Precision = \frac{TP}{TP+FP}$$
$$Recall = \frac{TP}{TP+FN}$$
$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Average Precision (AP).[9] AP is an average of correctness of every example obtained achieved at each threshold.

$$AP = \sum_n (R_n - R_{n-1}) P_n,$$

where P_n and R_n are the precision and recall at the nth threshold.

AP@k. Computes AP of the model's top k predictions.

Mean Average Precision (MAP).[10]

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q),$$

where Q is a sample of experiment runs, $AP(\cdot)$ refers to average precision.

Mean Reciprocal Rank (MRR). MRR rewards correct results by rank and is defined as

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i},$$

where $rank_i$ refers to the rank of the first relevant outcome for the ith run.

In the next section, we will give specific results details.

[9] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html
[10] https://en.wikipedia.org/wiki/Mean_reciprocal_rank

	AP	AP@100
weeds_prec	0.546442	0.795885
clarkeDE	0.549216	0.801145
invCL	0.591873	0.854653
cosine	0.379949	0.494368
slqs	0.172089	0.182375
slqs_cos	0.314500	0.486050

Table 5: Results of Distributional Models.

	precision	recall	f1-score	support
0	0.96	0.95	0.96	2641
1	0.57	0.61	0.59	269
accuracy			0.92	2910
macro avg	0.76	0.78	0.77	2910
weighted avg	0.92	0.92	0.92	2910

Table 6: Results of SVM Classification on Term Embeddings.

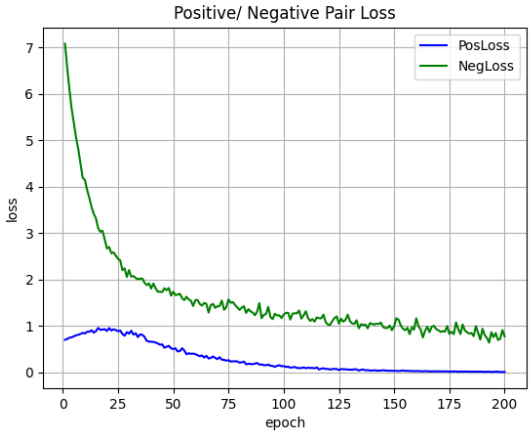

Figure 4: Projection Learning Training Loss on SemEval2018-Task9-2B Music Baseline Model

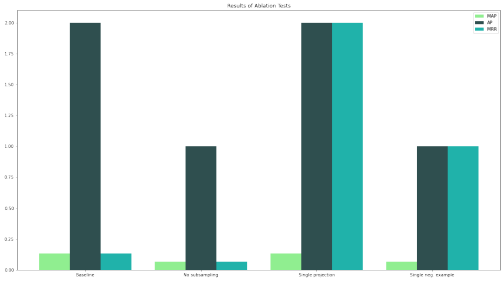

Figure 5: Results of Abalation Tests on 2B Music

5 Results

This section mainly introduces some training and exploratory results as well as ablation tests during modeling and training. See detailed extraction results under '*.Results*' folder in code.zip.

Distributional Models. We evaluate all the distributional similarity models introduced in section 2.3 by average precision (AP) and AP@100 (Table 5).

Term Embeddings. We evaluate precision, recall and f1-score on SVM classification results in Table 6.

Projection Learning. Ablation Tests (Figure 5 and Table 7):

- No subsampling. That means sampling all positive examples uniformly from train set.

- Single projection. Let k=1 rather than 24.

- Single neg. example. Set corresponding parameter to be 1.

Bi-LSTM Sequence Labeling. The f1 score on test data is around 0.75 which is satisfying. During the training procedure, it seemed that RMSprop is the best optimizer among the three (Figure 6) here. However, it is not only a tradition to use Adam [11, 12], but is also proved its outperformance in several deep learning optimization situation [15]. Therefore, we choose Adam in the following model training and prediction. The f1 score during training is given in Figure 7.

	MAP	AP	MRR
Baseline	0.1333	2.0	0.1333
No subsampling	0.0667	1.0	0.0667
Single projection	0.1333	2.0	2.0
Single neg. example	0.0667	1.0	1.0

Table 7: Results of Ablation Tests on 2B Music.

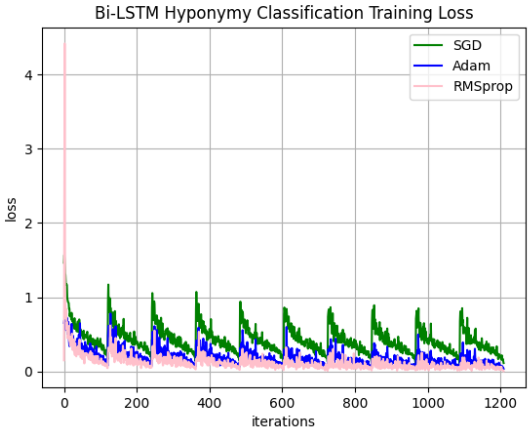

Figure 6: Comparison of Optimizers in Bi-LSTM Sequence Labeling.

6 Conclusion

We tried several methods here in this project, both supervised and unsupervised. Hearst Patterns can extract hypernymy pairs conveniently but with low precision because of the low coverage of patterns in corpora. PPMI and SVD pattern-based models are robust in hypernymy detection in literature [3]. Distributional methods gives a high average precision (AP) at around 0.5, with WeedsPrec outperforms all the rest. The average f1 score of SVM classification results on term embedding model is 0.77. In project learning, we also tried ablation tests and the results seems in alignment with the baseline model on 2B Music domain specific data. The f1 score in both training and validation set exceed 0.7 in Bi-LSTM model.

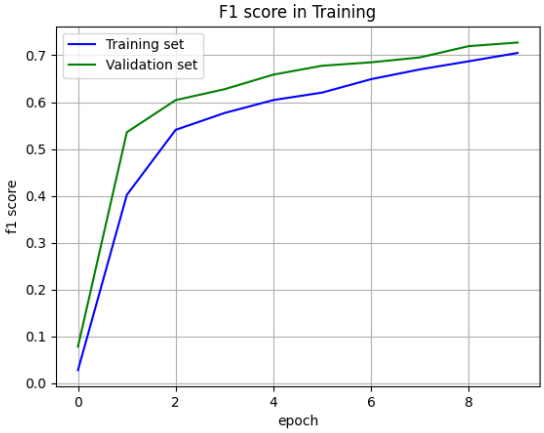

Figure 7: Bi-LSTM Model Training.

7 Future Works

There are still several directions untouched, like the bootstrap methods start from a small seed patterns and generated through bootstrap of extraction. Also we can improve current extraction methods through adversarial learning for example [18]. Domain specific study is also recommmended especially in medical field [19].

Acknowledgements

Thanks Prof. Chen and TA who recommended me various precious project materials. Thanks my classmates' help when I had questions. And thanks my family and friends who gave me unlimited supports during the term of study.

References

[1] Hearst, M A. (1992). Automatic acquisition of hyponyms from large text corpora. Proceedings of the 14th conference on Computational linguistics-Volume 2. Association for Computational Linguistics, pages 539-545.

[2] (2018). SemEval-2018 Task 9: Hypernym Discovery. https://www.aclweb.org/anthology/S18-1115.pdf

[3] (2018). Hearst Patterns Revisited: Automatic Hypernym Detection from Large Text Corpora. https://www.aclweb.org/anthology/P18-2057.pdf

[4] E Agichtein, L Gravano. (2000). Snowball: Extracting Relations from Large Plain-Text Collections. Proceedings of the fifth ACM conference.

[5] Julie Weeds, David Weir, Diana McCarthy. (2004). Characterising measures of lexical distributional similarity. https://www.aclweb.org/anthology/C04-1146.pdf

[6] Alessandro Lenci, Giulia Benotto. (2012). Identifying hypernyms in distributional semantic spaces. https://www.aclweb.org/anthology/S12-1012.pdf

[7] (2020). When Hearst Is not Enough: Improving Hypernymy Detection from Corpus with Distributional Models. https://arxiv.org/pdf/2010.04941.pdf

[8] Zheng Yu, Haixun Wang, Xuemin Lin, Min Wang. (2015). Learning Term Embeddings for Hypernymy Identification. https://www.ijcai.org/Proceedings/15/Papers/200.pdf

[9] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, Ting Liu. (2014). Learning Semantic Hierarchies via Word Embeddings. https://www.aclweb.org/anthology/P14-1113.pdf

[10] (2016). Distributional Hypernym Generation by Jointly Learning Clusters and Projections. https://www.aclweb.org/anthology/C16-1176.pdf

[11] (2018). CRIM at SemEval-2018 Task 9: A Hybrid Approach to Hypernym Discovery. https://www.aclweb.org/anthology/S18-1116.pdf

[12] (2017). Negative Sampling Improves Hypernymy Extraction Based on Projection Learning. https://www.aclweb.org/anthology/E17-2087.pdf

[13] (2020). A Survey on Recent Advances in Sequence Labeling from Deep Learning Models. https://arxiv.org/pdf/2011.06727.pdf

[14] (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. https://arxiv.org/pdf/1603.01354.pdf

[15] (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. https://arxiv.org/pdf/1412.6980.pdf

[16] Wentao Wu, Hongsong Li, Haixun Wang, Kenny Q. Zhu. (2012). Probase: A Probabilistic Taxonomy for Text Understanding. https://www.microsoft.com/en-us/research/wp-content/uploads/2012/05/paper.pdf

[17] Shantanu Kumar. (2017). A Survey of Deep Learning Methods for Relation Extraction. https://arxiv.org/pdf/1705.03645.pdf

[18] C Wang, X He, A Zhou. (2019). Improving Hypernymy Prediction via Taxonomy Enhanced Adversarial Learning. Proceedings of the AAAI Conference on Artificial Intelligence.

[19] Chenming Xu, Yangming Zhou, Qi Wang, Zhiyuan Ma, Yan Zhu. (2019). Detecting Hypernymy Relations Between Medical Compound Entities Using a Hybrid-Attention Based Bi-GRU-CapsNet Model. IEEE.