Term Definitions Help Hypernymy Detection

Wenpeng Yin and **Dan Roth**
University of Pennsylvania
{wenpeng,danroth}@seas.upenn.edu

Abstract

Existing methods of hypernymy detection mainly rely on statistics over a big corpus, either mining some co-occurring patterns like "animals such as cats" or embedding words of interest into context-aware vectors. These approaches are therefore limited by the availability of a large enough corpus that can cover all terms of interest and provide sufficient contextual information to represent their meaning. In this work, we propose a new paradigm, HYPERDEF, for **hyper**nymy detection – expressing word meaning by encoding word **def**initions, along with context driven representation. This has two main benefits: (i) Definitional sentences express (sense-specific) corpus-independent meanings of words, hence definition-driven approaches enable strong generalization – once trained, the model is expected to work well in open-domain testbeds; (ii) Global context from a large corpus and definitions provide complementary information for words. Consequently, our model, HYPERDEF, once trained on task-agnostic data, gets state-of-the-art results in multiple benchmarks[1].

1 Introduction

Language understanding applications like textual entailment (Dagan et al., 2013), question answering (Saxena et al., 2007) and relation extraction (Mintz et al., 2009), benefit from the identification of lexical entailment relations. Lexical inference encompasses several semantic relations, with hypernymy being one of the prevalent (Roller et al., 2014; Shwartz et al., 2016), an *i.e., "Is-A" relation that holds for a pair of terms*[2] *(x, y) for specific terms' senses*.

Two families of approaches have been studied for identifying term hypernymy. (i) *Pattern matching* exploits patterns such as "animals such as cats" to indicate a hypernymy relation from "cat" to "animal" (Hearst, 1992; Snow et al., 2004). However, it requires the co-occurrence of the two terms in the same sentence, which limits the recall of this method; (ii) *Term representation learning* depends on a vector embedding of each term, where each entry in the vector expresses an explicit context feature (Baroni et al., 2012a; Roller and Erk, 2016; Shwartz et al., 2017) or a latent semantic (Fu et al., 2014; Vulic and Mrksic, 2017; Glavas and Ponzetto, 2017).

Both approaches hinge on acquiring context-aware term meaning in a large corpus. The generalization of these corpus-based representation learning paradigms, however, is limited due to the domain specificity of the training data. For example, an IT corpus hardly mentions "apple" as a fruit. Furthermore, the surrounding context of a term may not convey subtle differences in term meaning – "he" and "she" have highly similar context that may not reveal the important difference between them. Moreover, rare words are poorly expressed by their sparse global context and, more generally, these methods would not generalize to the low resource language setting.

Humans can easily determine the hypernymy relation between terms even for words they have not been exposed to a lot, given a definition of it in terms of other words. For example, one can imagine a "teaching" scenario that consists of *defining* a term, potentially followed by a few examples of the term usage in text.

Motivated by these considerations and the goal of eventually develop an approach that could generalize to unseen words and even to the low resource languages scenario, we introduce the following hypernymy detection paradigm, HYPERDEF, where we augment distributional contextual models with that of learning terms representations

[1] cogcomp.org/page/publication_view/836
[2] This paper uses "term" to refer to any words or phrases.

*Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 203–213
New Orleans, June 5-6, 2018.

from their definitions. This paradigm has an important advantage in its *powerful generalization*, as definitions are agnostic to specific domains and benchmarks, and are equally available for words regardless of their frequency in a given data set. Consequently, the task of identifying the relation between two terms is enhanced by the knowledge of the terms' definitions. Our model can be applied to any new terms in any domain, given some context of the term usage and their domain-agnostic definitions. Moreover, given our learning approach – we learn also the notion of *lexical entailment* between terms – we can generalize to any lexical relation between terms.

Technically, we implement HYPERDEF by modifying the AttentiveConvNet (Yin and Schütze, 2017), a top-performing system on a textual entailment benchmark (Bowman et al., 2015), to model the input (x, d_x; y, d_y), where d_i ($i = x, y$) is the definition of term i. In contrast to earlier work which mostly built separate representations for terms x and y, HYPERDEF instead directly models the representation for each pair in $\{(x, y), (x, d_y), (d_x, y), (d_x, d_y)\}$, and then accumulates the four-way representations to form an overall representation for the input.

In our experiments, we train HYPERDEF on a task-agnostic annotated dataset, Wordnet, and test it on a broad array of open-domain hypernymy detection datasets. The results show the outstanding performance and strong generalization of the HYPERDEF model.

Overall, our contributions are as follows:

- To our knowledge, this is the first work in hypernymy detection that makes use of term definitions. Definitions provide complementary knowledge to distributional context, so that our model better tolerates unseen words, rare words and words with biased sense distribution.
- HYPERDEF accounts for word sense when inferring the hypernymy relation. This differs from much of the literature, which usually derives sense-unaware representative vectors for terms – earlier approaches would say 'yes' if the relation holds for some combination of the terms' senses.
- HYPERDEF has strong generalization capability – once trained on a task-agnostic definition dataset, it can be used in different testbeds, and shows state-of-the-art results.

2 Related Work

The main novelty of our HYPERDEF lies in the *information resource* that is employed to represent the terms. Prior work in exploring information resources can be put into two categories: understanding terms by the co-occurring context in raw text, or grounding the terms in open-domain objects.

2.1 Mining Distributional Context from Text

Window-based Context Baroni et al. (2012b) build distributional semantic vectors for terms from a concatenation of three corpora: the British National Corpus, WackyPedia and ukWac. Each entry in the vector is the PMI-formulated score from co-occurrence counts. Dimension reduction is conducted by Singular Value Decomposition (SVD) before feeding representation vectors to a classifier.

Dependency-based Context Roller and Erk (2016) compute a *syntactic* distributional space for terms by counting their *dependency neighbors* across the corpus.

Shwartz et al. (2017) further compare (i) contexts being parent and daughter nodes in the dependency tree, and (ii) contexts being the parent-sister pairs in the dependency tree.

Term Embeddings Unspecialized term embeddings are not informative signals for detecting specific lexico-semantic relations. Hence, community often explicitly build transformation functions from unspecialized embeddings to relation-specialized embeddings. Fu et al. (2014) first use the skip-gram model (Mikolov et al., 2013) to learn generic term embeddings from a large Chinese encyclopedia corpus, then learn a projection function from the generic space to hypernymy space by annotated hypernymy pairs. Other work trying to specify the generic word embeddings to hypernymy detection task include (Vulic and Mrksic, 2017; Glavas and Ponzetto, 2017).

Other advanced types of term embeddings specific to the hypernymy detection problem include Gaussian distributed embeddings (Vilnis and McCallum, 2015), non-negative embeddings (Chang et al., 2017), magnitude-oriented embeddings (Nguyen et al., 2017), and so on.

In our work, distributional context model is also applied. More specifically, we will directly use

pretrained word embeddings as initial word representations and specialize them in training. In contrast, distributional context only acts as one side of information resource to express words, we focus on making use of a second side of information from word definitions to build a more robust system.

2.2 Grounding Terms to Open-domain Objects

Do and Roth (2012) build Wikipedia representations for input terms – representing the input terms by a set of relevant Wikipedia pages.

Shwartz et al. (2015) represent each term pair as a set of paths which are extracted from different large-scale knowledge resources (DBPedia, Wikidata, Yago and WordNet), then train a classifier to determine whether the two terms satisfy a relation of interest given those path connections.

Young et al. (2014) map terms to a set of images, then determine the directional inference by conditional probability over statistic of image intersection.

Compared with mining of distributional context from text, these works switch the context from words to Wikipedia pages, KB paths or images. So, they share a similar mechanism while differing in the categories of entries in distributional vectors.

Our paradigm HYPERDEF shares the same inspiration with above distributional models. More importantly, It goes beyond the frame of distributional models by exploring a novel information resource – definitions – to derive the word semantics.

3 HYPERDEF Model

In this section, we first give a brief review of a top-performing neural network for textual entailment – AttentiveConvNet (Yin and Schütze, 2017), which acts as a base model to encode a pair of texts. Then, we elaborate on the adaptation we make towards AttentiveConvNet so that the resulting system can better serve the hypernymy detection problem.

3.1 AttentiveConvNet

AttentiveConvNet[3] (Yin and Schütze, 2017) is essentially a Siamese convolutional neural network (CNN) (LeCun et al., 1998) equipped with an attention mechanism. It predicts the relationship of two sentences by accumulating the dominant features of fine-grained alignments across sentences. The reason we base our system on this model is two-fold: (i) AttentiveConvNet is one of the top-performing systems of modeling sentence pairs in textual entailment, and (ii) AttentiveConvNet implements the fine-grained cross-sentence alignments in the granularity of local windows; this makes it appropriate to reason between a definitional sentence and a term.

We use bold uppercase, e.g., $\mathbf{H}$, for matrices; bold lowercase, e.g., $\mathbf{h}$, for vectors; bold lowercase with index, e.g., $\mathbf{h}_i$, for columns of $\mathbf{H}$; format $\mathbf{h}$[i] to denote the i^{th} entry of vector $\mathbf{h}$; and non-bold lowercase for scalars.

AttentiveConvNet, shown in Figure 1, represents a sentence S ($S \in \{S_1, S_2\}$) of n words as a sequence of hidden states $\mathbf{h}_i \in \mathbb{R}^d$ ($i = 1, 2, \ldots, n$), forming a feature map $\mathbf{H} \in \mathbb{R}^{d \times n}$, where d is the dimensionality of hidden states. Each $\mathbf{h}_i$ has a left context $\mathbf{h}_{i-1}$ and a right context $\mathbf{h}_{i+1}$. Given feature maps $\mathbf{H}_1$ and $\mathbf{H}_2$ for sentences S_1 and S_2 respectively, AttentiveConvNet derives a representation for the pair (S_1, S_2). Unlike conventional CNNs over single sentences, AttentiveConvNet develops an attention mechanism to achieve fine-grained alignments automatically, then puts convolution filters over aligned hidden states together with their context.

Overall, AttentiveConvNet derives the pair representation in three steps. (i) A matching function determines how relevant each hidden state in sentence S_2 is to the current hidden state $\mathbf{h}_i$ in sentence S_1. All hidden states in S_2 are then accumulated by weighted average to form an *aligned hidden state* $\tilde{\mathbf{h}}_i$. (ii) Convolution for position i in S_1 integrates the two aligned hidden states ($\mathbf{h}_i$, $\tilde{\mathbf{h}}_i$) with context $\mathbf{h}_{i-1}$ and $\mathbf{h}_{i+1}$. (iii) Max-pooling over the generated group of hidden states in step (ii) yields a representation for the pair (S_1, S_2). Next, we describe these processes in detail.

Generation of Aligned Hidden States. First, a matching function $f_e(\mathbf{h}_i, \mathbf{h}_j^{S_2})$ generates a score $e_{i,j}$ to evaluate how relevant the two hidden states $\mathbf{h}_i, \mathbf{h}_j^{S_2}$ are.

Given the matching scores, the aligned hidden state $\tilde{\mathbf{h}}_i$ in S_2 for hidden state $\mathbf{h}_i$ in S_1 is the

[3] https://github.com/yinwenpeng/Attentive_Convolution

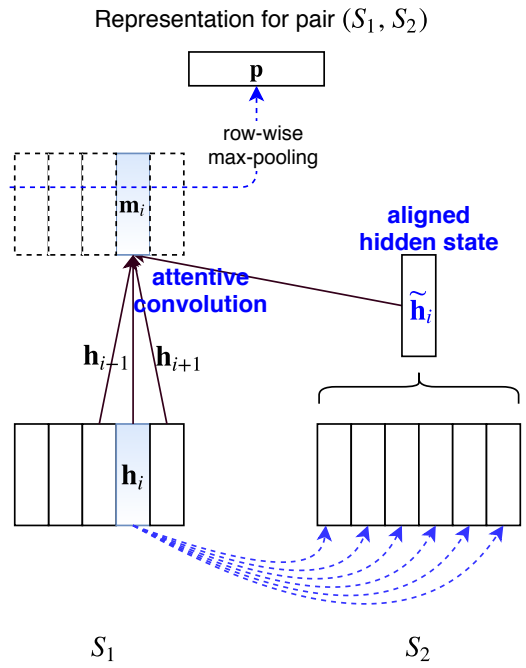

Figure 1: AttentiveConvNet models a sent. pair (S_1, S_2). In our work, S_i (i=1,2) can be the definition sentence or the term itself (we treat a term as a short sentence.)

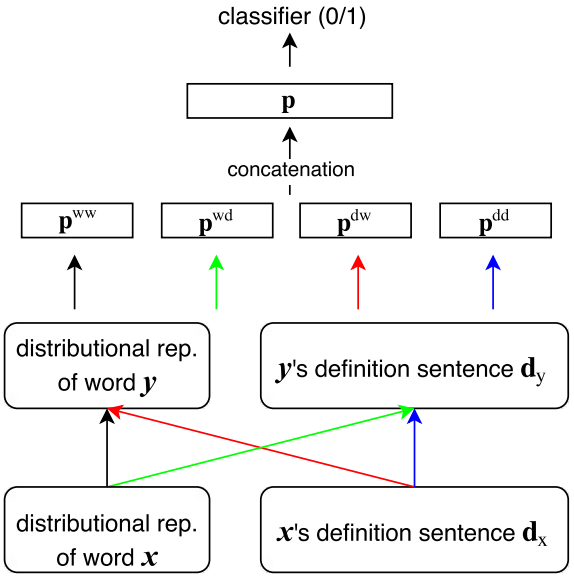

Figure 2: HYPERDEF – combining distributional model with definition encoding

weighted average of all hidden states in S_2:

$$\tilde{\mathbf{h}}_i = \sum_j \text{softmax}(\mathbf{e}_i)[j] \cdot \mathbf{h}_j^{S_2} \quad (1)$$

Attentive Convolution. A position i in S_1 has hidden state $\mathbf{h}_i$, left context $\mathbf{h}_{i-1}$, right context $\mathbf{h}_{i+1}$ and aligned hidden state $\tilde{\mathbf{h}}_i$ from S_2. Attentive convolution then generates the higher-level representation for this combination:

$$\mathbf{m}_i = \tanh(\mathbf{W} \cdot [\mathbf{h}_{i-1}, \mathbf{h}_i, \mathbf{h}_{i+1}, \tilde{\mathbf{h}}_i] + \mathbf{b}) \quad (2)$$

where parameters $\mathbf{W} \in \mathbb{R}^{d \times 4d}$, $\mathbf{b} \in \mathbb{R}^d$.

Pair Representation Generation. As Equation 2 shows, each $\mathbf{m}_i$ denotes the inference features between $\mathbf{h}_i$ and its alignment $\tilde{\mathbf{h}}_i$ in context. AttentiveConvNet uses max-pooling over $\{\mathbf{m}_i\}$ to get the overall representation $\mathbf{p}$ for the pair:

$$\mathbf{p}[i] = \max(\mathbf{m}_1[i], \mathbf{m}_2[i], \cdots, \mathbf{m}_n[i]) \quad (3)$$

Finally, the representation $\mathbf{p}$ is used in classification. The whole model is learned in an end-to-end training[4].

3.2 Four-way AttentiveConvNet

AttentiveConvNet originally works on sentence pairs. We formulate the hypernymy detection problem as $\{(x, d_x; y, d_y; 1/0)\}$. Just as in (Shwartz et al., 2016) which directly concatenates the term path representation vector with term embedding vectors as the classifier input, a simple combination of distributional models and definition encoding for us could be: separately learning the distributional model over term embedding pairs and an AttentiveConvNet model over definition pairs, then concatenate their output representations. However, the analysis over dataset $\{(x, d_x; y, d_y; 1/0)\}$ hints that HYPERDEF can obtain more indicative features by modeling (term, definition), which crosses the distributional models and definition encoding. For example, the definition of term "cat" in WordNet is: *feline mammal usually having thick soft fur and no ability to roar: domestic cats; wildcats*. Intuitively, when the system meets the pair (cat, mammal), it should be trivial to get the "hypernymy" decision since "mammal" appears in the definition sentence.

Inspired by this observation, we implement the HYPERDEF paradigm as four-way AttentiveConvNets, as Figure 2 shows, i.e., treating the two terms as word sequences as well, then do AttentiveConvNet over all four combinations: (x, y), (x, d_y), (d_x, y) and (d_x, d_y).

Assume we get four separate representations: $\mathbf{p}^{ww}$ from (x, y), $\mathbf{p}^{wd}$ from (x, d_y), $\mathbf{p}^{dw}$ from (d_x, y) and $\mathbf{p}^{dd}$ from (d_x, d_y), as Section 3.1 described. We construct the final representation for

[4]For more details, please refer to (Yin and Schütze, 2017).

	random split					lexical split				
	all	$-\mathbf{p}^{ww}$	$-\mathbf{p}^{wd}$	$-\mathbf{p}^{dw}$	$-\mathbf{p}^{dd}$	all	$-\mathbf{p}^{ww}$	$-\mathbf{p}^{wd}$	$-\mathbf{p}^{dw}$	$-\mathbf{p}^{dd}$
HYPERDEF (F_1)	.905	.874	.896	.876	.881	.887	.875	.870	.849	.862
HYPERDEF (AP)	.933	.902	.921	.905	.909	.900	.890	.883	.880	.877
w/o attention (F1)			.825					.743		
w/o definition (F1)			.734					.619		
LSTM+atten. (F1)			.757					.685		

Table 1: Tune HyperDef on wn_dev

$(x, d_x; y, d_y)$ via concatenation:

$$\mathbf{p} = [\mathbf{p}^{ww}, \mathbf{p}^{wd}, \mathbf{p}^{dw}, \mathbf{p}^{dd}] \quad (4)$$

then $\mathbf{p}$ is fed to the final classifier.

AttentiveConvNet over (x, y) resembles the conventional hypernymy classifiers which take two representation vectors (one for x, the other for y) as input and output the label. Note that AttentiveConvNet puts filter weights over (x, y) to learn more abstract representations; this actually is in common with some literature such as (Fu et al., 2014; Vulic and Mrksic, 2017; Glavas and Ponzetto, 2017), which utilize weights to project generic word representations into specified representations towards hypernymy annotations.

AttentiveConvNet over (x, d_y) and (d_x, y) compares a term with the descriptive sentence of the other term; this might provide direct clues, as we discussed in the beginning of this subsection.

AttentiveConvNet over (d_x, d_y) resembles literature (Do and Roth, 2012; Young et al., 2014). HYPERDEF provides an alternative resource for interpreting terms, resorting to definitional expressions instead of Wikipedia pages or images.

Overall, our HYPERDEF combines strengths of (i) conventional supervised classifiers over context distributions, and (ii) rich interpretation of terms in broader knowledge bases.

3.3 Analysis of HYPERDEF

Our HYPERDEF has the following properties:

- HYPERDEF combines distributional models with definition encoding, but it is not simply a concatenation of two independent subsystems. HYPERDEF enables modeling across (distributional context, definition). This is expected to generate more indicative features than a similar work (Shwartz et al., 2016), which simply concatenated distributional models with path-based models;

- HYPERDEF employs definitions to provide richer information for the terms. But it does not generate an auxiliary term representation vector from the definitive sentence as the literature (Hill et al., 2016) did. Instead, HYPERDEF formulates a pair of input elements – each can be a distributional vector or a definition representation – into a cross-sentence attention mechanism, which directly yields a compact representation to the pair rather than two separate vectors for the two input elements. This is shown more effective to model the relationship of two pieces of text (Yin and Schütze, 2017);

- Distributional models and definitive sentences in HYPERDEF provide complementary knowledge. For terms which can not retrieve a definition, HYPERDEF still works – just turning into a basic distributional model. This work uses WordNet and Wikipedia as example resources for the definition retrieval, more splendid resources will be developed gradually in released HYPERDEF models. We will also provide users the option to type into their definitions;

- WordNet provides term definitions in the sense level, so theHYPERDEF model is essentially trained in the sense level. For polysemy cases in testing, HYPERDEF can simply test on all combinations of definitions, then pick the pair with the highest probability;

- For terms that were never observed in training, we expect context distributions, such as pretrained embeddings, and definitions are available, so HYPERDEF is hardly influenced in this case. This is exactly the main advantage of HYPERDEF: generalization.

4 Experiments

4.1 Pre-training of HYPERDEF

Dataset Preparation. As we aim to build a strongly generalizing hypernymy detector, the training data we collect here is expected to be task-agnostic. Hence, extracting from structured knowledge resources, such as WordNet (Fellbaum, 1998), Wikidata (Vrandecic, 2012), DBPedia (Auer et al., 2007), and Yago (Suchanek et al., 2007), is preferred. Some literature, e.g., (Shwartz et al., 2015), claim that there is limited coverage for almost all knowledge resources. For example, WordNet does not cover many propernames (Donald Trump → president) or recent terminology (AlphaGo → computer program). Our data tends to alleviate this challenge, since in testing, descriptive sentences in the HYPERDEF paradigm can provide the precise and distinct features for terms even if these terms are OOV and in new types.

In this work, we pick one of those knowledge resources – WordNet – to *collect training data*. Specifically, our positive instances consist of (i) all direct hypernymy pairs, and (ii) switched terms from the original hyponymy pairs. Negative instances include (i) pairs with other relations such as antonym, synonym, and (ii) pairs of positive instances after exchanging the two terms. Note that each term is accompanied by its definition *in sense level*. So we get instances in form $(x, d_x; y, d_y; 1/0)$, where the binary value "1/0" indicates whether y is x's hypernymy or not. Altogether, we collect about 900K instances with roughly a 8:1 ratio between negative and positive instances.

In testing, we implement HYPERDEF to retrieve definitions and distributional context for terms automatically[5].

Random and Lexical Dataset Splits. In our primary dataset, we perform a *random* split, with 80% train, 10% dev, and 10% test.

As pointed out by Levy et al. (2015), supervised distributional lexical inference methods tend to perform "lexical memorization", i.e., instead of learning a relation between the two terms, they mostly learn an independent property of term y in the pair: whether y is a "prototypical hypernym" or not. Levy et al. (2015) suggest to splitting the train and test sets such that each will contain a distinct vocabulary ("*lexical split*"), in order to prevent the model from overfitting by lexical memorization.

In the current phase, we use notations *wn_train*, *wn_dev*, and *wn_test* to refer to the three parts. Note that *wn_train* and *wn_dev* will be used to train and tune the HYPERDEF model, while *wn_test* is set to show how well the model performs in WordNet domain – it is not expected to act as a testbed in real benchmarks. In experiments, we will compare our model in *random* and *lexical* splits.

Training Setup. Given *wn_train* in form $\{(x, d_x; y, d_y; 1/0)\}$, a binary classifier via logistic regression is trained over the pair representation $\mathbf{p}$ obtained from Equation 4, predicting *1* or *0* for the hypernymy relation. The objective function is implemented through negative log-likelihood. Terms and words in definitions are initialized by 300d Word2Vec embeddings (Mikolov et al., 2013) and kept unchanged in training. This benefits the generalization as it ensures that the words in training and the new words in test data lie in the same space. All hidden sizes are 300 as well. The whole system is trained by AdaGrad (Duchi et al., 2011) with initial learning rate 0.02.

We first run the HYPERDEF in *wn_test* to check if it is effective in the WordNet domain. Then we test it in some open-domain benchmarks. *Note that all experiments use the* HYPERDEF *models pretrained over* wn_train.

4.2 Performance within WordNet

As mentioned in Section 4.1, *wn_train*, *wn_dev* and *wn_test* have two distinct setups: "random split" and "lexical split", inspired by the "lexical memorization" observation (Levy et al., 2015).

We first tune the parameters in wn_train and search the best system layout based on wn_dev. F_1 and average precision (AP) are reported. Table 1 lists the performance records, with the first block for "random split" and the second block for "lexical split".

We first discuss three baselines: (i) "w/o definition": We discard definitions and only use distributional model, i.e., a logistic regression classifier (LR) over the concatenated (x, y) embeddings from Word2Vec. Its performance drops 11.5% from "random split" to "lexical split". This is within expectation as Levy et al. (2015) concluded that this baseline is not effective in

[5] In the released HYPERDEF model, we will provide an option for users to input definitions.

learning genuine term relations; (ii) "w/o attention": We discard the attention mechanism in AttentiveConvNet, resulting in a bi-CNN structure. It works on instances $\{(x, d_x; y, d_y; 1/0)\}$, a vanilla CNN is used to encode the definition sentence into a dense representation vector. So, each term in (x, y) will get two separate representation vectors (one is from Word2Vec, the other from the definition); finally totally four representation vectors are concatenated and fed to the LR. This baseline works much better than "w/o definition" (improvements of 9% $\sim$ 11%). Their comparison shows that incorporating term definitions in reasoning process is promising; (iii) "LSTM+attention" (Rocktäschel et al., 2016). A representative attention mechanism in LSTM (Hochreiter and Schmidhuber, 1997) for textual entailment. We apply it in the same way as our four-way AttentiveConvNet, however, found it performs poorly. We suspect that this is due to two reasons: i) Though there is entailment or hypernymy relation between a term pair, e.g., ("cat", "animal"), unfortunately there is no clear clue of that relation between their definition pair *if considering all the information contained in the definitions*. For example, "cat" – "*a small domesticated carnivorous mammal with soft fur, a short snout, and retractile claws. It is widely kept as a pet or for catching mice, and many breeds have been developed*", and "animal" – "*a living organism that feeds on organic matter, typically having specialized sense organs and nervous system and able to respond rapidly to stimuli*". Apparently, we can not infer the whole definition of "animal" by cat's definition. Instead, their help mainly comes from some key-phrases, such as "domesticated carnivorous mammal", "living organism" and so on. LSTM, encoding the whole word sequences in attention, potentially would be misled. Our approach relies on convolution filters and max-pooling, excelling in modeling keywords-driving features (Yin et al., 2017). This baseline indicates the overall strength of our system comes from the definition incorporation as well as an appropriate encoder.

Considering the whole table, we observe that: (i) HYPERDEF models have pretty close performances in "random split" and "lexical split" – mostly within 2$\sim$3%. This strongly indicates that HYPERDEF is less influenced by the "lexical memorization" problem. Our systems, equipped with definition encoding, show promising generalization (at least in WordNet domain); (ii) Though HYPERDEF models in "all" setup behave similarly in random split and lexical split, the detailed contributions of $\mathbf{p^{ww}}$, $\mathbf{p^{wd}}$, $\mathbf{p^{dw}}$ and $\mathbf{p^{dd}}$ differ in the two settings. To be specific, in "wn_dev (random split)", there is no clear winner among $\{\mathbf{p}^{ww}, \mathbf{p}^{dw}, \mathbf{p}^{dd}\}$, $\mathbf{p}^{wd}$ contributes consistently less than the other three. In "wn_dev (lexical split)", instead, $\mathbf{p}^{wd}$, $\mathbf{p}^{dw}$ and $\mathbf{p}^{dd}$ perform similarly while $\mathbf{p}^{ww}$ performs worst. This indicates that when dealing with unseen terms, definition-based components in HYPERDEF play a dominant role.

	random	lexical
F_1	.902	.881
AP	.915	.891

Table 2: Pretrained HyperDef on wn_test

Experiments on wn_dev enable to store the best HYPERDEF models – concatenation over the four representations: $\mathbf{p^{ww}}$, $\mathbf{p^{wd}}$, $\mathbf{p^{dw}}$ and $\mathbf{p^{dd}}$. Then we reload the pretrained models and report their performance on wn_test, as shown in Table 2. From Table 1 to Table 2, we observe pretty small drop in performance – mostly $\sim$ 1% . This preliminarily demonstrates the strong generalization.

Next, we test the best HYPERDEF models pretrained on "wn_train (lexical split)" in open domain benchmarks.

4.3 Performance in Open-domain Datasets

First, we use four widely-explored datasets: **BLESS** (Baroni and Lenci, 2011), **EVALution** (Santus et al., 2015), **Lenci/Benotto** (Benotto, 2015), and **Weeds** (Weeds et al., 2014). They were constructed either using knowledge resources (e.g. WordNet, Wikipedia), crowd-sourcing or both. The instance sizes of hypernymy and "other" relation types are detailed in Table 3. We also report "#OOV_pair", the proportions of unseen term pairs in above four datasets regarding the training set of HYPERDEF, i.e., *wn_train* in Section 4.1. We notice that most term pairs in BLESS and Lenci/Benotto datasets are unseen in wn_train.

First, we extract the term's *all sense definitions* from WordNet based on the term string. For a few instances which contain terms not covered by WordNet, such as proper noun "you", "everybody" etc, we set definitions the same as the term strings (this preprocessing does not influence results, just for making the system uniformed).

dataset	#hyper.	#others	#OOV_pair
BLESS	1.337	25,217	99.04%
EVALution	3,637	9,828	78.86%
Lenci/Benotto	1,933	3,077	92.50%
Weeds	1,469	1,459	71.54%

Table 3: Statistics of four benchmarks. "#OOV_pair": the proportions of unseen term pairs regarding the training set (i.e., *wn_train* in Section 4.1) of HYPERDEF.

Then, we apply the pre-trained HYPERDEF model *on the test sets of the four benchmarks*, discriminating hypernymy from "other" relations. AP and AP@100 are reported. As WordNet sorts sense definitions by sense frequency (Fellbaum, 1998), we test the term pairs in two ways: (i) Only choose the top-1 sense definition to denote a term, reported as "HYPERDEF$_{TopDef}$"; (ii) Keep all sense definitions for those terms, then test on all sense combinations and pick the highest probability as the term pair score, reported as"HYPERDEF$_{AllDef}$".

We compare HYPERDEF with baselines: (i) **Best-Unsuper**. The best unsupervised method in (Shwartz et al., 2017), implemented by similarity measurement over weighted dependency-based context vectors; (ii) **Concat-SVM** (Glavas and Ponzetto, 2017). An SVM model with RBF kernel is trained on concatenation of unspecialized concept embeddings (Baroni et al., 2012a); (iii) **DUAL-T** (Glavas and Ponzetto, 2017). Using dual tensors, DUAL-T transforms unspecialized embeddings into asymmetrically specialized representations – sets of specialized vectors – which are next used to predict whether the asymmetric relation holds between the concepts; (iv) **HyperScore** (Nguyen et al., 2017). The state-of-the-art system. It uses a large-scale hypernymy pair set to guide the learning of hierarchical word embeddings in hypernymy-structured space.

Table 4 clearly demonstrates the superiority of our HYPERDEF models over other systems. The three baselines Concat-SVM, DUAL-T and HyperScore are more in line with HYPERDEF since they did supervised learning over large numbers of annotated pairs. HYPERDEF integrates term definitions, which is shown effective in improving the performance across different testbeds.

In addition, HYPERDEF$_{AllDef}$ consistently outperforms HYPERDEF$_{TopDef}$. This makes sense as HYPERDEF$_{TopDef}$ may be misled by incorrect definitions. In addition, the superiority of HYPERDEF$_{AllDef}$ clearly supports the effectiveness of HYPERDEF in dealing with polysemy cases.

Above four benchmarks are relatively small and contain common words mostly. In real-world applications, there is a need to figure out the hypernymy relation between common nouns and proper nouns (Do and Roth, 2012). Taking "(Champlin, city)" for example, "Champlin" is not covered by WordNet vocabulary, thus uncovered by wn_train – the training data of our HYPERDEF model. Motivated, we further test HYPERDEF on the following dataset.

HypeNet Dataset. Shwartz et al. (2016) construct this dataset by extracting hypernymy relations from several resources: WordNet, DBPedia, Wikidata and Yago. Like our collected data, term pairs in other relations are considered as negative instances. It maintains a ratio of 1:4 positive to negative pairs.

Similarly, HypeNet dataset has "random split" and "lexical split" as well; their sizes are list in Table 5. HypeNet contains lots of locations, e.g., (Champlin, city), and organizations, e.g., (Telegram, company) and (Sheetz, company). We first try to extract definitions for those terms from WordNet, if fail, then we extract from Wikipedia pages, treating the first sentence as a definition.

We play HYPERDEF in two different ways, one testing its "*pre-trained*" model on HypeNet's test data, the other – "*specialized*" – training HYPERDEF on HypeNet's training data then test on HypeNet's test data like other baselines did.

Table 6 shows: (i) If trained on the specific training data of HypeNet, our system HYPERDEF can get state of the art performance. This indicates the superiority of our model over baseline systems.

(i) Our pretrained HYPERDEF model performs less satisfactorily. Only the result on "Lex. split" is relatively close to the outstanding baselines. This makes sense as baseline systems are specified by the HypeNet training set while our pretrained model comes from a different domain. We studied the dataset and found following problems.

Error Analysis. Two error sources are observed. (i) Wrong definition. For example, the system obtains the definition "*a substance or treatment with no active therapeutic effect*" for the term "Placebo" in the pair (Placebo, song); however, a successful detection requires mining an-

Model	BLESS		EVALuation		Benotto		Weeds	
	AP	AP@100	AP	AP@100	AP	AP@100	AP	AP@100
Best-Unsuper (Shwartz et al., 2017)	.051	.540	.353	.661	.382	.617	.441	.911
Concat-SVM (Glavas and Ponzetto, 2017)	.097	.235	.321	.329	.523	.586	.644	.793
DUAL-T (Glavas and Ponzetto, 2017)	.487	.823	.446	.866	.557	.847	.774	.985
HyperScore (Nguyen et al., 2017)	.454	–	.538	–	.574	–	.850	–
HYPERDEF$_{TopDef}$	**.595**	.749	.524	**.867**	.557	.825	**.872**	**.989**
HYPERDEF$_{AllDef}$	**.508**	**.872**	**.623**	**.927**	**.576**	**.909**	**.889**	**.991**

Table 4: System comparison on BLESS, EVALution, Benotto and Weeds datasets

Dataset	train	dev	test	#OOV_pair
HypeNet (rnd)	49.5K	3.5K	17.7K	95.56%
HypeNet (lex)	20.3K	1.4K	6.6K	95.33%

Table 5: Statistics of HypeNet dataset. "#OOV_pair" is for "test" regarding the "wn_train" of HYPERDEF.

Model	Lex. split			Rand. split		
	P	R	F_1	P	R	F_1
HypeNet	.809	.617	.700	.913	.890	.901
DUAL-T	.705	.785	.743	.933	.826	.876
pre-trained	.572	.717	.637	.474	.601	.530
specialized	.670	.914	**.773**	.892	.935	**.913**

Table 6: System comparison on HypeNet test

other definition – "*are an alternative rock band, formed in London, England in 1994 by singer-guitarist Brian Molko and guitarist-bassist Stefan Olsdal*" which depicts the article title "Placebo (band)". This is a common problem due to the ambiguity of entity mentions. To relieve this, we plan to refine the definition retrieval by more advanced entity linking techniques, or retrieve all highly related definitions and test as in polysemy cases (recall that in Table 4 we showed HYPERDEF has more robust performance while addressing polysemy terms); (ii) Misleading information in definitions. Our system predicts "1" for the pair (Aurangabad, India); we analyze the definition of "Aurangabad": *is a city in the Aurangabad district of Maharashtra state in India*. We intentionally removed the phrase "in India", and then the system predicts "0". This demonstrates that definitions indeed provide informative knowledge about terms, but a system must be intelligent to avoid being misled; (iii) We miss a common embedding space to initialize single words and (multi-word) entities. To generalize well to new entities, the model has to presume the new entities and the known terms lie in the same representation space. However, most pretrained embedding sets cover pretty limited entities. To learn uniformed word and entity embeddings, we may need to combine unstructured text corpus, semi-structured data (e.g., Wikipedia) and structured knowledge bases together. We will advance this data preprocessing component – the access of term definitions and term representations – in our released system.

5 Conclusion

In this work, we introduced a novel approach to detecting hypernymy relations by incorporating term definitions. We extracted a task-agnostic annotated data from WordNet, then trained a neural network to generate a universal hypernymy detector, HYPERDEF. HYPERDEF, once trained, performs competitively in diverse open-domain benchmarks, even though it was not fine-tuned on those benchmark-specific training sets. This validates the powerful generalization of our model HYPERDEF. Our hope, and one of the key future directions following this work is to generalize this approach to the low-resource language setting.

Acknowledgments

We thank all the reviewers for providing insightful comments and critiques. This research is supported in part by DARPA under agreement number FA8750-13-2-0008, and by a gift from Google.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of ISWC/ASWC*. pages 722–735.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012a. Entailment above the word level in distributional semantics. In *Proceedings of EACL*. pages 23–32.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012b. Entailment above the word level in distributional semantics. In *Proceedings of EACL*. pages 23–32.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. pages 1–10.

Giulia Benotto. 2015. Distributional models for semantic relations: A study on hyponymy and antonymy. *PhD Thesis, University of Pisa* .

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

Haw-Shiuan Chang, ZiYun Wang, Luke Vilnis, and Andrew McCallum. 2017. Unsupervised hypernym detection by distributional inclusion vector embedding. *CoRR* abs/1710.00880.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzoto. 2013. Recognizing textual entailment: Models and applications.

Quang Xuan Do and Dan Roth. 2012. Exploiting the wikipedia structure in local and global classification of taxonomic relations. *Natural Language Engineering* 18(2):235–262.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 12:2121–2159.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of ACL*. pages 1199–1209.

Goran Glavas and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Proceedings of EMNLP*. pages 1757–1767.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*. pages 539–545.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *TACL* 4:17–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*. pages 2278–2324.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of NAACL*. pages 970–976.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*. pages 1003–1011.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*. pages 233–243.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR*.

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

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

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: An evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics*. pages 64–69.

Ashish Kumar Saxena, Ganesh Viswanath Sambhu, Saroj Kaushik, and L. Venkata Subramaniam. 2007. IITD-IBMIRL system for question answering using pattern matching, semantic type and semantic category recognition. In *Proceedings of TREC*.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of ACL*.

Vered Shwartz, Omer Levy, Ido Dagan, and Jacob Goldberger. 2015. Learning to exploit structured resources for lexical inference. In *Proceedings of CoNLL*. pages 175–184.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of EACL*. pages 65–75.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of NIPS*. pages 1297–1304.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of WWW*. pages 697–706.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *Proceedings of ICLR*.

Denny Vrandecic. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of WWW*. pages 1063–1064.

Ivan Vulic and Nikola Mrksic. 2017. Specialising word vectors for lexical entailment. *CoRR* abs/1710.06371.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING*. pages 2249–2259.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of CNN and RNN for natural language processing. *CoRR* abs/1702.01923.

Wenpeng Yin and Hinrich Schütze. 2017. Attentive convolution. *CoRR* abs/1710.00519.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL* 2:67–78.