SYNET: Synonym Expansion using Transitivity

Jiale Yu, Yongliang Shen, Xinyin Ma, Chenghao Jia, Chen Chen, Weiming Lu*
College of Computer Science and Technology, Zhejiang University, China
{27121115, syl, maxinyin, chjia, chen_double, luwm}@zju.edu.cn

Abstract

In this paper, we study a new task of synonym expansion using transitivity, and propose a novel approach named SYNET, which considers both the contexts of two given synonym pairs. It introduces an auxiliary task to reduce the impact of noisy sentences, and proposes a Multi-Perspective Entity Matching Network to match entities from multiple perspectives. Extensive experiments on a real-world dataset show the effectiveness of our approach.

1 Introduction

Synonym discovery has become an important task, which can benefit many downstream applications, such as web search (Cheng et al., 2012), question answering (Zhou et al., 2013), knowledge graph construction (Boteanu et al., 2019), clinical text analysis (Wang et al., 2019b), and etc.

One straightforward approach to obtain synonyms is from public knowledge bases, such as WordNet (Fellbaum, 2000) and DBpedia (Lehmann et al., 2015). For example, WordNet groups terms into *synsets*, and DBpedia uses *Redirects to URIs* to indicate synonyms. However, these synonyms are constructed manually, which makes the coverage rather limited.

Two types of approaches are widely exploited to discover synonyms automatically from text corpora, including the distributional based approaches (Wang et al., 2019a,b; Fei et al., 2019) and the pattern based approaches (Nguyen et al., 2017). The distributional based approaches assume that if two terms appear in similar contexts, they are likely to be synonyms. For example, "*USA*" and "*the United States*" are often mentioned in similar contexts, so they both refer to the same country. The pattern based approaches lay emphasis on the local contexts, such as "commonly known as". However, they both have some limitations. The distributional based approaches suffer from low precision, while the pattern based approaches suffer from low recall. In order to address these limitations, DPE (Qu et al., 2017) integrated these two approaches for synonym discovery.

Intuitively, people believe that *synonyms possess transitivity*, that is (m_i, *synonym*, m_b) $\wedge$ (m_b, *synonym*, m_j)$\rightarrow$(m_i, *synonym*, m_j), where m_i, m_b and m_j are three different mentions, and m_b is the bridge mention of two synonym pairs (m_i, m_b) and (m_b, m_j). This transitivity can be used for synonym discovery directly from existing synonyms. For example, *the United States of America* and *the United States* are synonyms, *the United States* and *U.S.* are synonyms, so *the United States of America* and *U.S.* should also be synonyms. Distiller (Ali et al., 2019) even designed loss functions based on the synonym transitivity properties.

Baidu Baike[1] and Wikidata[2] both use "*Also known as*" to indicate synonyms, as shown in Figure 1(a) and Figure 1(b). Therefore, we can extract synonym pairs such as (金丝雀, 芙蓉) and (芙蓉, 木莲) easily. However, *synonyms possess transitivity* is not always hold. In our example, 金丝雀 and 木莲 are not synonymous, as shown in Figure 1(c). This is because 芙蓉 is polysemous, which has two meanings: 金丝雀(*canary*) and 木莲(*hibiscus*). Therefore, using transitivity between synonym pairs directly would make wrong synonym pairs.

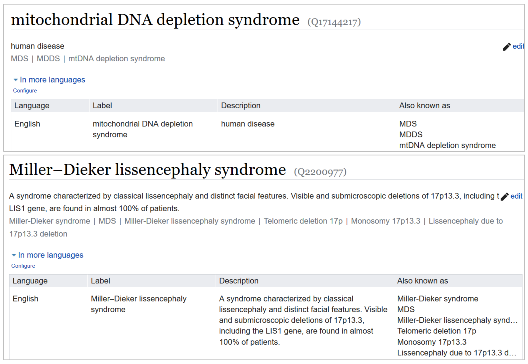
Therefore, it is hazardous to infer (m_i, synonym, m_j) directly, when (m_i, synonym, m_b) and (m_b, synonym, m_j) are given. There are several challenges to address this problem. Firstly, if we directly use distributional approaches to predict whether

*Corresponding author

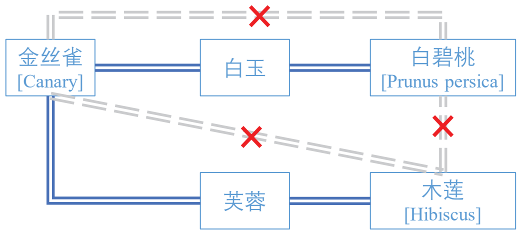
[1] https://baike.baidu.com/
[2] http://www.wikidata.org/

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1961–1970
November 16 - 20, 2020.

(a) Synonyms mentioned in infobox in Baidu Baike.



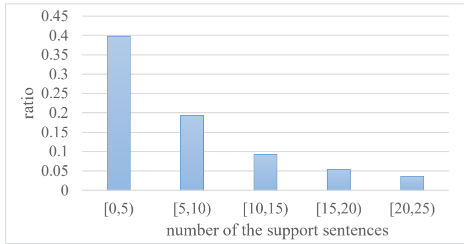
(b) Synonyms mentioned in Wikidata.



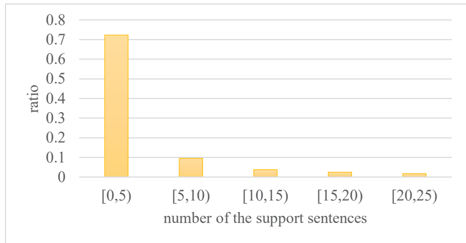
(c) Synonym transitivity is not always hold.

Figure 1: Motivation of our task: synonyms are mentions by *Also known as* in Baidu Baike (a) and Wikidata (b). However, synonym transitivity is not always hold as shown in (c), which is the transitivity graph from (a). In Figure (c), the edges with red cross indicate that the corresponding two mentions are not synonymous.

two mentions m_i and m_j are synonymous without using the information of (m_i, synonym, m_b) and (m_b, synonym, m_j), the precision would be low, since the global context of m_i (m_j) is various. Secondly, pattern based approaches can not be applied effectively, since the sentences mentioning both m_i and m_j may be fewer than the sentences mentioning both m_i and m_b (or m_b and m_j). In our paper, these sentences mentioning both two mentions are called *support sentences*. We analyze the distribution of the support sentences in our dataset, which will be elaborated in Section 4.1, and the results are shown in Figure 2. From the figure, we find that about 60% pairs of (m_i, m_b) or (m_b, m_j) have more than 5 support sentences, but only less than 30% pairs of (m_i, m_j) have more than 5 support sentences, and even 43% pairs of (m_i, m_j) have no support sentences. Thirdly, the support sentences are obtained in a distant-supervised way, which may bring in lots of noises. Although the sentences mentioning two mentions in a synonym pair may express the same meaning, which can partly reduce the noise, we still have to reduce the impact of the noisy sentences further.



(a) The distribution of mention pair (m_i, m_b) or (m_b, m_j).



(b) The distribution of mention pair (m_i, m_j).

Figure 2: The distribution of mention pairs according to the number of support sentences in the dataset.

In order to address these challenges, we propose a new synonym discovery task *Synonym Expansion using Transitivity* (Figure 3): Given two sets of synonym pairs (m_i, m_b) and (m_b, m_j) with a bridge mention m_b and their corresponding support sentences, which are obtained from text corpus through distant supervision, we aim to predict whether m_i and m_j are synonyms or not.

For the task, we propose a novel framework, named SYNET, which leverages both the contexts of two given synonym pairs. First, it introduces an auxiliary task to reduce the impact of noisy sentences further, and then proposes a Multi-Perspective Mention Matching Network (MPMM) to match mentions from multiple perspectives, including M2M (Mention-to-Mention), M2B (Mention-to-mention Bag) and B2B (mention Bag-to-mention Bag) matches.

Our contributions in this paper are as follows:

- We study a new task of synonym expansion using transitivity, and propose a novel approach named SYNET for this task. To the best of our

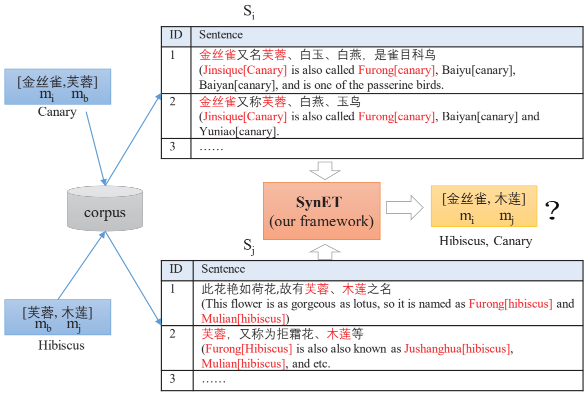


Figure 3: Task illustration: We aim to expand synonyms using Transitivity. The Chinese are translated into English at below, and we use Pinyin in English sentences to differentiate the mentions which refer to the same entity.

knowledge, it is the first to study the problem of synonym expansion using transitivity.

- We construct a dataset from encyclopedias through distant supervision, and the experiments on it show the effectiveness of our approach.

2 Task Definition

We first introduce basic concepts and their notations, and then present the task definition.

Synonym Pair. A synonym pair is a pair of strings (i.e. word or phrases) that refer to the same entity in the world. For example, ("*United States*", "*USA*") is a synonym pair, since "*United States*" and "*USA*" represent the same country. We can extract synonym pairs directly from the infobox of Baidu Baike or Wikidata as shown in Figure 1.

Synonym Pair Candidate. A synonym pair candidate can be obtained from existing synonym pairs according to the synonym transitivity properties. For example, ("*the United States of America*", "*USA*") and ("*USA*", "*America*") are two synonym pairs, so ("*the United States of America*" and "*America*") can be considered as a synonym pair candidate. Formally, if (m_i, m_b) and (m_b, m_j) are two synonym pairs, (m_i, m_j) can be considered as a synonym pair candidate. Since synonym transitivity is not always hold, (m_i, m_j) can not be treated as a synonym pair directly, as mentioned in Figure 1(c).

Support Sentence. In order to predict whether two mentions m_i and m_j in a synonym pair candidate are synonymous, we should collect some support sentences. Since the sentences contain both m_i and m_j are sparse or even nonexistent, we turn to collect sentences which contain mentions in synonym pairs (m_i, m_b) and (m_b, m_j). We denote S_i is a bag of support sentences for (m_i, m_b), and each sentence in S_i contains the two mentions m_i and m_b. Taking the synonym pair ("*the United States of America*", "*USA*") as an example, the sentence "*The United States of America, commonly known as the United States, America or USA.*" is one of its support sentences.

Task Definition. We formally define our task of synonym discovery using transitivity as: *Given two synonym pairs (m_i, m_b) and (m_b, m_j), where m_b is the bridge mention, and two sets of corresponding support sentences S_i and S_j, $s \in S_i$ (S_j) mentions both m_i and m_b $(m_j$ and $m_b)$, the task is to predict whether the two mentions m_i and m_j in a synonym pair candidate are synonymous or not.*

Figure 3 illustrates the task with an example.

3 The SYNET Approach

In this section, we introduce our proposed approach SYNET for synonym discovery using transitivity.

As shown in Figure 4, our proposed SYNET approach mainly consists of three components, including Sentence Encoder (Section 3.1), Mention Encoder (Section 3.3) and Multi-Perspective Mention Matching Network (MPMM) (Section 3.4). In the following sections, we will elaborate each component in detail.

3.1 Sentence Encoder

We can employ a BiLSTM (Hochreiter and Schmidhuber, 1997) or BERT (Devlin et al., 2019) to encode each support sentence s in S_i, where s is a sequence of words $w_1, w_2, ..., w_n$, and two mentions m_i and m_b of a synonym pair in the sentence is a subsequence of words $w_{i_s}, ..., w_{i_e}$ and $w_{b_s}, ..., w_{b_e}$ respectively. Each word w_i is mapped to a pretrained d_w-dimensional vector $\vec{w}_i$.

3.1.1 BiLSTM based Sentence Encoder

BiLSTM based sentence encoder (Figure 5) firstly encodes sentence s into hidden states $(h_1, h_2, ..., h_n)$:

$$\overrightarrow{h}_t = LSTM_{fw}(v_t, \overrightarrow{h}_{t-1})$$
$$\overleftarrow{h}_t = LSTM_{bw}(v_t, \overleftarrow{h}_{t+1})$$

where $LSTM_{fw}$ and $LSTM_{bw}$ are the forward and backward LSTMs respectively, and v_t =

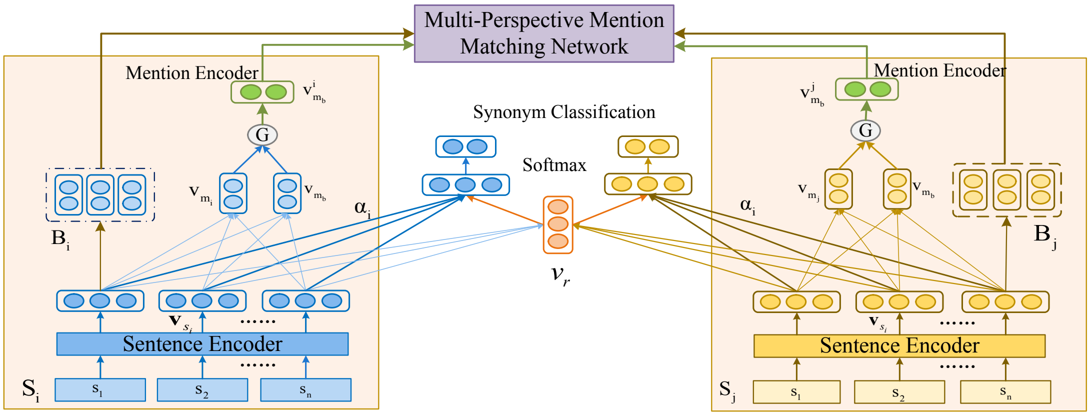


Figure 4: The SYNET framework with Sentence Encoder, Mention Encoder and Multi-Perspective Mention Matching Network.

$[\vec{w}_t \oplus p_t^1 \oplus p_t^2]$, $p_t^1, p_t^2 \in \mathbb{R}^{d_p}$ are two position embeddings (Zeng et al., 2015). We obtain $h_t = [\overrightarrow{h}_t \oplus \overleftarrow{h}_t]$ and $h_s = [\overrightarrow{h}_n \oplus \overleftarrow{h}_1]$, where $\oplus$ denotes vector concatenation.

Then, the sentence embedding is calculated by $v_s = \tanh(W_s h_s + b_s)$. In addition, the embedding of mention m_i can also be calculated by $v_{m_i} = \tanh(\frac{1}{|i_e - i_s + 1|} \sum_{t=i_s}^{i_e} W_m h_t + b_m)$. Here, $W_s, W_m \in \mathbb{R}^{d_h \times d_c}$ and $b_s, b_m \in \mathbb{R}^{d_c}$ are trainable parameters. The final sentence embedding for s is represented by $V_s = [v_s \oplus v_{m_i} \oplus v_{m_b}]$.

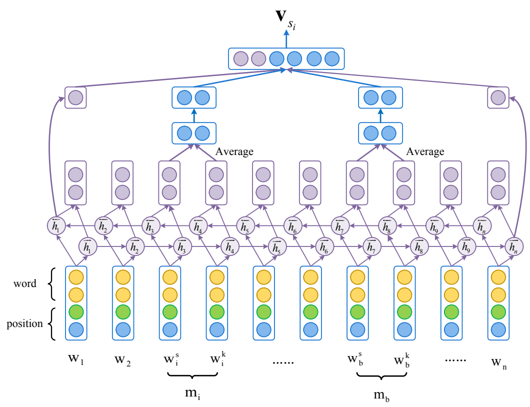


Figure 5: BiLSTM based sentence encoder

3.1.2 BERT Based Sentence Encoder

The BERT based sentence encoder is shown in Figure 6.

The input s is firstly organized as $([CLS], T_1, ..., [E_i], T_{i_s}, .., T_{i_e}, [E_i], ..., T_n, [SEP])$, where T_i is the concatenation of the word embedding, segmentation embedding and position embedding. The mention m_i is enclosed by a mark token $[E_i]$, which is trained using reserved tokens *[unused]* in BERT. Then, BERT encodes

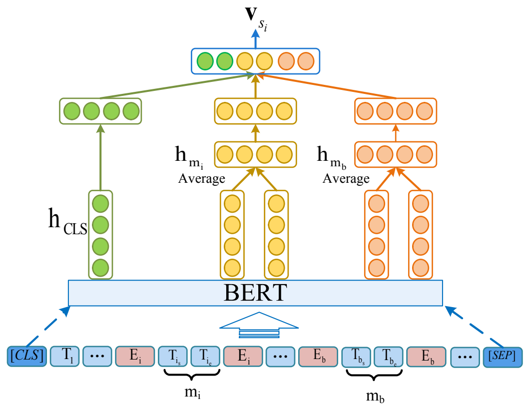


Figure 6: BERT based sentence encoder.

the input into hidden states $(h_{[CLS]}, h_1, ..., h_n)$. Thus, we obtain $v_s = \tanh(W_s h_{[CLS]} + b_s)$ and $v_{m_i} = \tanh(\frac{1}{t_{i_e} - t_{i_s} + 1} \sum_{t=t_{i_s}}^{t_{i_e}} W_m h_t + b_e)$. Similar to BiLSTM based sentence encoder, the final sentence embedding for s is $V_s = [v_s \oplus v_{m_i} \oplus v_{m_b}]$.

During training, we start from a pre-trained BERT model[3], and then fine-tune it using our training data.

3.2 Auxiliary Task for Noise Reduction

In order to reduce the impact of noise in $S_i = [s_i^1, s_i^2, ..., s_i^{l_i}]$, where l_i is the number of support sentences in S_i, we introduce an auxiliary task, which takes S_i as the input, and predicts the importance of each sentence with the attention mechanism through synonym relation classification.

Formally, a set of sentence embeddings $[V_{s_i^1}, V_{s_i^2}, ..., V_{s_i^{l_i}}]$ is obtained by the sentence encoder. Then we randomly initialize a relation vector $v_r \in \mathbb{R}^{d_c}$ to calculate the attention weight for

[3] https://github.com/google-research/bert

$s_i^j \in S_i$: $\alpha_i^j = \frac{\exp(V_{s_i^j} v_r^T)}{\sum_{k=1}^{l_i} \exp(V_{s_i^k} v_r^T)}$.

Finally, S_i can be represented by $V_{S_i} = \sum_{k=1}^{l_i} \alpha_i^k V_{s_i^k}$. Therefore, the probability of synonym prediction is $p(m_i \sim m_b|S_i) = softmax(W_o V_{S_i} + b_o)$, where $\sim$ denotes two mentions are synonymous. The loss for the synonym triple (m_i, m_b, m_j) in this auxiliary task is:

$$\mathcal{L}_{aux}^{i,j} = -\log p(m_i \sim m_b|S_i) - \log p(m_j \sim m_b|S_j)$$

3.3 Mention Encoder

During the sentence encoding for each sentence s in S_i, we can also obtain the mention embeddings v_{m_i} and v_{m_b} for m_i and m_b as in Section 3.1. Thus, two bags of mention embeddings can be obtained from S_i: $B_i = \{v_{m_i}^1, v_{m_i}^2, ..., v_{m_i}^{l_i}\}$ and $B_b = \{v_{m_b}^1, v_{m_b}^2, ..., v_{m_b}^{l_i}\}$, where l_i is the size of B_i and B_b.

Since sentences in the bag have some noise, we have calculated the attention weight α_i^j for each sentence $s_i^j \in S_i$ in Section 3.2. Therefore, the aggregated embeddings for mention m_i and m_b in S_i can also be calculated as: $V_{m_i} = \sum_{j=1}^{l_i} \alpha_i^j v_{m_i}^j$ and $V_{m_b} = \sum_{j=1}^{l_i} \alpha_i^j v_{m_b}^j$.

3.4 Multi-Perspective Mention Matching Network

In order to predict whether $m_i \in S_i$ and $m_j \in S_j$ are synonyms, an intuitional and direct idea is to measure the semantic similarity between m_i and m_j. We can fuse V_{m_i} and V_{m_b} to represent the semantic of the mention m_i with a gating mechanism:

$$V_m = g \odot V_{m_b} + (1-g) \odot V_{m_i}, g = \sigma(\tilde{g})$$

where $\tilde{g} \in \mathbb{R}^{d_c}$ is a learnable parameter, σ is a Sigmoid function, and $\odot$ is an element-wise multiplication. Thus, we can use $softmax(W(V_m^i \odot V_m^j) + b)$ to predict the synonymity between m_i and m_j, where $V_m^{i(j)}$ is the mention representation of $S_{i(j)}$, W and b are two learnable parameters.

Besides V_{m_i}, V_{m_b} and V_{m_j}, B_i, B_b and B_j are also used to represent the mentions of m_i, m_b and m_j. Thus, we propose a multi-perspective mention matching network (MPMM) to match mentions from multiple perspectives, including M2M (Mention-to-Mention), M2B (Mention-to-mention Bag) and B2B (mention Bag-to-mention Bag) matches. In order to differentiate m_b in S_i and S_j, we use $B_b^i = [v_i^1, v_i^2, ..., v_i^{l_i}]$ and $B_b^j = [v_j^1, v_j^2, ..., v_j^{l_j}]$ to denote embeddings of m_i in S_i and S_j respectively, where $l_{i(j)}$ is the size of $B_b^{i(j)}$, and $v_{i(j)}^k \in B_b^{i(j)}$ is the bridge mention embedding of $s_k \in S_{i(j)}$.

Figure 7 illustrates the MPMM in detail. In our experiment, we find that the semantic consistency of m_b between S_i and S_j is more effective to predict the synonymity between m_i and m_j. Thus, we use $B_b^{i(j)}$ instead of directly using $B_{i(j)}$. The perspectives of mention matching network are as follows.

M2M: V_m^i and V_m^j are compared directly to obtain a matching vector $V_{M2M}^{ij} = V_m^i \oplus V_m^j$.

B2B: Inspired by (Wang et al., 2019b), we use the dynamic context matching mechanism to measure to the similarity between B_b^i and B_b^j.

Given B_b^i and B_b^j, we first calculate the similarity matrix $M = (B_b^i) W_m (B_b^j)^T$, and then obtain the attention weights:

$$\beta_i^k = softmax(mean\_pooling(M_{k:}))$$

$$\beta_j^k = softmax(mean\_pooling(M_{:k}))$$

Finally, we get two matching vectors $V_{B2B}^i = \sum_{k=1}^{l_i} \beta_i^k v_i^k$ and $V_{B2B}^j = \sum_{k=1}^{l_j} \beta_j^k v_j^k$.

M2B: LSTM has achieved some success in aggregating an unordered set, such as in (Hamilton et al., 2017; Zhang et al., 2020). Here, given V_m^i and B_b^j, we also use LSTM to aggregate them as follows.

$$\begin{aligned} h'_{t+1}, c_{t+1} &= LSTM(v_j^t, [h_t \oplus V_m^i, c_t]) \\ h_{t+1} &= h'_{t+1}[: d_c] + v_j^t \end{aligned}$$

where $LSTM(x, [h, c])$ is a LSTM cell. The final output of the LSTM h_{l_j} is denoted as V_{M2B}^i. Similarly, we can also obtain $V_{M2B}^j = h_{l_i}$ when putting V_m^j and B_b^i into the LSTM.

Finally, the probability of m_i and m_j being synonymous can be calculated by $p(m_i \sim m_j|S_1, S_2) = softmax(o_{ij})$, where $o_{ij} = W[V_{M2M}^{ij} \odot V_{B2B}^i \odot V_{B2B}^j \odot V_{M2B}^i \odot V_{M2B}^j] + b$. The following loss is used for the synonym triple (m_i, m_b, m_j) with corresponding support sentences S_i and S_j:

$$\mathcal{L}_{mm}^{i,j} = -\log p(m_i \sim m_j|S_i, S_j)$$

3.5 Model Optimization and Inference

To train the SYNET, we minimize the overall objective: $\mathcal{L} = \sum_{t=1}^{T}(\mathcal{L}_{aux}^{i_t,j_t} + \mathcal{L}_{mm}^{i_t,j_t})$, where T is the number of synonym triples $\{(m_{i_t}, m_{b_t}, m_{j_t})\}_{t=1}^{T}$.

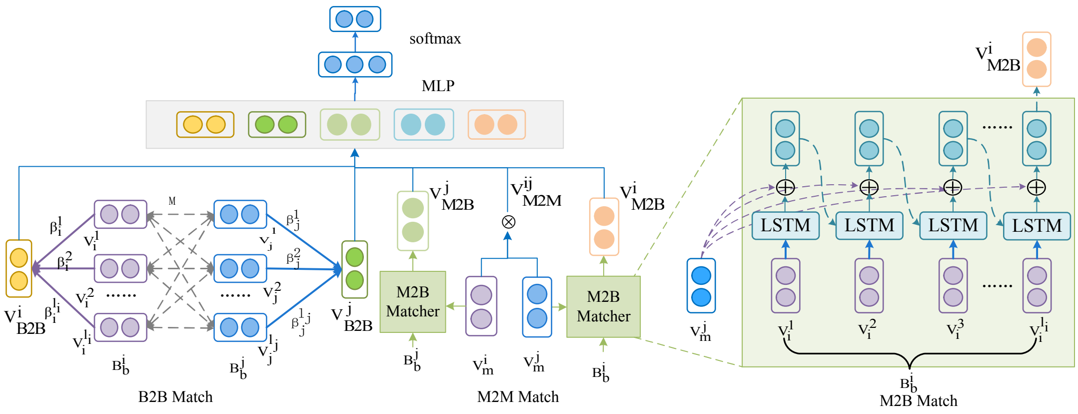


Figure 7: The Multi-Perspective Mention Matching Network.

During the inference step, we use $p(m_i \sim m_j|S_i, S_j)$ to predict whether m_i and m_j are synonyms or not.

4 Experiments

4.1 Dataset Construction

We build a dataset SYNETDATA from Baidu Baike, which is the largest Chinese encyclopedia in China, in a distant supervision way.

The instance of the dataset is a six-tuple $(m_i, m_b, m_j, S_i, S_j, l)$, where (m_i, m_b) and (m_b, m_j) are synonym pairs with a bridge mention m_b, (m_i, m_j) is a synonym pair candidate, S_i and S_j are two sets of support sentences for (m_i, m_b) and (m_b, m_j), l is the label of indicating whether (m_i, m_j) is a synonym pair or not.

Specifically, we firstly crawl articles from Baidu Baike, and extract synonyms for each article by analyzing the infobox, which forms a group of synonyms denoted as $(m_1, m_2, ..., m_n)$, such as (金丝雀, 芙蓉, 白玉, ...) from Figure 1(a). Then, we randomly selected 3 mentions from a group, which can be considered as a positive instance (m_i, m_b, m_j) with label $l = 1$.

For negative instances, we first crawl disambiguation pages from Baidu Baike and extract all senses for each mention. This mention can be considered as a bridge mention. For example, Figure 8 shows several senses for 芙蓉. Then, we randomly select two senses, such as *a plant* and *a bird*, and then extract synonyms for each sense from infoboxs of the articles. For *plant*, we can extract 木莲, while for *bird*, we can extract 金丝雀. Thus, (m_i=金丝雀, m_b=芙蓉, m_j=木莲) can be considered as a negative instance.

After (m_i, m_b, m_j) has been extracted, we search sentences with queries m_i+m_b and m_b+m_j in articles of Baidu Baike, which are indexed with Lucene[4]. Since the sentence with a longer distance between two mentions would be noisier, we sort the sentences according to the distance between two mentions, and select the top 16 sentences as S_i (S_j) in order to fit in the BERT model. All sentences in S_i and S_j are segmented by HanLP[5].

Figure 8: An example of a disambiguation page in Baidu Baike, which contains several senses.

The statistics of the dataset are presented in Table 1, and the number of support sentences in each bag is from 2 to 16.

Table 1: Dataset statistics.

	Total	Positive	Negative
Train	10201	5175	5026
Validation	470	234	236
Test	475	236	239

4.2 Experimental Settings

We compare SYNET with the following baselines.

- Word2vec. We concatenate the word embeddings of m_i and m_j, which are pre-trained using word2vec[6] with all articles in Baidu Baike,

[4] https://lucene.apache.org
[5] https://github.com/hankcs/HanLP
[6] https://code.google.com/p/word2vec

and then input it to a multi-layer perceptron for synonym prediction.

- BiLSTM. We employ a BiLSTM to encode each support sentence s and calculate the embedding of the mention m_i v_{m_i} as in Section 3.1.1. Then, we average the embeddings of the mention m_i over all support sentences to obtain the final representation of m_i: $V_i = \frac{1}{|S_i|}\sum_{s \in Si} v_{m_i}$. Finally, we concatenate the embeddings of two mentions V_i and V_j, and input it to a multi-layer perception for synonym prediction.

- BERT. We concatenate the embeddings of two mentions V_{m_i} and V_{m_j}, which are obtained from the BERT based sentence encoder as in Section 3.1.2, and then input it to a multi-layer perception for synonym prediction.

- SynonymNet (Zhang et al., 2019). SynonymNet also use BiLSTM to encode the contexts of each mention, and then use a bilateral matching schema to determine synonymity. In our experiment, we use S_i and S_j as the contexts of m_i and m_j. In addition, two architectures for training the SynonymNet are also implemented, including a siamese architecture and a triplet architecture.

- SynSetMine (Shen et al., 2019). SynSetMine learns a set-instance classifier to determine whether a synonym set S should include an instance t. In our experiment, we use SynSetMine to determine whether m_i can be added to the set (m_j, m_b) or m_j can be added to the set (m_i, m_b). We also implement its variants using different word embeddings, including word2vec, BERT and BiLSTM, and different aggregation methods, including mean pooling and sum pooling.

The *accuracy*, *precision*, *recall* and *F1* are used to evaluate the approaches.

In our implements, we set the dimension of word embeddings with $d_w = 100$, and set $d_c = 128$, $d_p = 5$ and $d_h = 768$ for hidden states in the sentence encoder and mention encoder. We optimize our model using Adam (Kingma and Ba, 2015) and apply dropout technique with rate 0.1.

4.3 Main Results

We present our main results in Table 2. From the table, we can see that our approach outperforms all other approaches and their variants. SynonymNet and SynSetMine perform better than Word2vec and BiLSTM. For SynonymNet, the Siamese architecture works better on our dataset compared against the triplet architecture. While for SynSetMine, *sum pooling* can achieve a better performance than *mean pooling*.

SYNET(BERT) and SYNET(BiLSTM) have the comparable results. However, SYNET(BERT) runs much faster than SYNET(BiLSTM) (49 min VS. 79 min per epoch with a single GeForce GTX 1080 Ti), since BERT suppots efficient parallel training.

4.4 Ablation Studies

We conduct an ablation study to evaluate the contribution of each model component, and show the results in Table 3.

From the table, we can see that (1) The auxiliary task can boost the performance both for SYNET(BERT) and SYNET(BiLSTM) by putting different weights on sentences, which can reduce the impact of noisy sentences. The benefit of the auxiliary task is statistically significant with $p < 0.05$ under t-test. (2) All perspectives of mention matching in MPMM are useful, and using only one perspective would reduce the performance greatly. The effectiveness of each perspective is M2B $>$ B2B $>$ M2M. The reason may be that L-STM can capture "deep" feature interactions and accumulate expression capability of mention embeddings. (3) When only using M2M in MPMM, our approach will degrade to a synonym prediction model using BiLSTM with attention, where BiLSTM is used to encode mention m_i and m_j, while the auxiliary task calculates the attention weights of support sentences in S_i and S_j. Our approach performs better than the baseline BiLSTM in Table 2, which also verifies the effectiveness of the auxiliary task.

Besides, we also compare two strategies, using B_b or $B_{i(j)}$, in MPMM, and the results are shown in Table 4. From the table, we can see that the semantic consistency of m_b between S_i and S_j is more effective than directly using $B_{i(j)}$ in MPMM both in SYNET(BiLSTM) and SYNET(BERT).

5 Related Work

Synonym discovery is a crucial task in NLP, and many efforts have been invested. One straightforward approach to obtain synonyms is from pub-

Table 2: The performance evaluated on the test set with different approaches. In SynonymNet, we implement siamese and triple architecture. In SynSetMine, we use different word representations such as pre-trained word embeddings, BERT and BiLSTM for synonyms, and use two pooling approaches.

Model	accuracy	precision	recall	F1
Word2vec	0.655	0.622	0.775	0.691
BiLSTM	0.682	0.646	0.797	0.714
BERT	0.741	0.733	0.754	0.743
SynonymNet (Triple)	0.722	0.717	0.723	0.723
SynonymNet (Siamese)	0.688	0.617	**0.970**	0.755
SynSetMine (Word2vec + sum pooling)	0.730	0.739	0.708	0.723
SynSetMine (Word2vec + mean pooling)	0.702	0.762	0.585	0.662
SynSetMine (BERT + sum pooling)	0.766	0.788	0.725	0.755
SynSetMine (BERT + mean pooling)	0.677	0.713	0.589	0.645
SynSetMine (BiLSTM + sum pooling)	0.764	0.758	0.771	0.765
SynSetMine (BiLSTM + mean pooling)	0.703	0.727	0.644	0.683
SYNET (BiLSTM)	**0.832**	**0.820**	0.848	0.833
SYNET (BERT)	0.830	0.802	0.873	**0.836**

Table 3: Ablation results on the dataset, where "w/o" means *without*.

Model	acc	prec	recall	F1
SYNET(BiLSTM)	**0.832**	0.820	**0.848**	**0.833**
-w/o auxiliary task	0.827	**0.824**	0.830	0.827
-only M2M	0.743	0.734	0.759	0.746
-only B2B	0.773	0.762	0.788	0.775
-only M2B	0.827	0.818	0.839	0.829
SYNET(BERT)	**0.830**	0.802	**0.873**	**0.836**
-w/o auxiliary task	0.830	**0.833**	0.822	0.827
-only M2M	0.760	0.724	0.835	0.776
-only B2B	0.785	0.779	0.792	0.786
-only M2B	0.796	0.788	0.805	0.797

Table 4: The effectiveness of two strategies in MPMM, where $B_b^i \leftrightarrow B_b^j$ and $B_i \leftrightarrow B_j$ indicate using B_b or $B_{i(j)}$ respectively.

Model			acc	prec	recall	F1
SYNET(BiLSTM)	M2B	$B_b^i \leftrightarrow B_b^j$	**0.827**	0.818	**0.839**	**0.829**
		$B_i \leftrightarrow B_j$	0.802	**0.820**	0.771	0.795
	B2B	$B_b^i \leftrightarrow B_b^j$	**0.773**	**0.762**	0.788	**0.775**
		$B_i \leftrightarrow B_j$	0.767	0.747	**0.801**	0.773
SYNET(BERT)	M2B	$B_b^i \leftrightarrow B_b^j$	**0.796**	**0.788**	0.805	**0.797**
		$B_i \leftrightarrow B_j$	0.777	0.758	**0.809**	0.783
	B2B	$B_b^i \leftrightarrow B_b^j$	**0.785**	**0.779**	0.792	**0.786**
		$B_i \leftrightarrow B_j$	0.771	0.755	**0.797**	0.775

lic knowledge bases, such as WordNet (Fellbaum, 2000), ConceptNet (Speer et al., 2017) and DBpedia (Lehmann et al., 2015). However, these synonyms are constructed manually, which makes the coverage rather limited.

Many efforts have been made to discover synonyms automatically. Some approaches discover synonyms from query logs (Chaudhuri et al., 2009; Wei et al., 2009; Chakrabarti et al., 2012; Ren and Cheng, 2015) and web table schemas (Cafarella et al., 2008; He et al., 2016). However, these approaches are limited to structured or semi-structured data.

Recently, researchers focus on mining synonyms from a raw text corpus, which is more challenging. Two types of approaches are widely exploited, including the pattern based approaches (Nguyen et al., 2017) and the distributional based approaches (Wang et al., 2019a,b; Fei et al., 2019; Zhang et al., 2019). The pattern based approaches lay emphasis on the local contexts, such as "commonly known as". While the distributional based approaches assume that if two terms appear in similar contexts, they are likely to be synonyms. For example, SynonymNet (Zhang et al., 2019) proposed a multi-context bilateral matching framework for synonym discovery from free-text corpus. SurfCon (Wang et al., 2019b) discovered synonyms on privacy-aware clinical data by utilizing the surface form information and the global context information. However, they suffer from either low precision or low recall. Thus, DPE (Qu et al., 2017) and SynMine (Yu et al., 2019) integrated these two approaches for synonym discovery. Moreover, SynSetMine (Shen et al., 2019) learned a set-instance classifier to generate entity synonym sets from a given vocabulary using example sets from external knowledge bases as distant supervision.

Our approach focuses on mining synonyms using transitivity which is not the focus of the previous works. Although He et al. (2016) also utilized transitivity, they assumed that transitivity does hold in almost all cases for attribute synonyms, so they used this transitivity property to discover cluster-based synonyms by a linear programming-based algorithm. While our approach called this property into question, and only used it to generate synonym candidates.

6 Conclusion

In this paper, we study a new task of synonym expansion using transitivity, and propose a novel approach named SYNET. To the best of our knowledge, it is the first time to study this problem. The SYNET considers both the contexts of two given synonym pairs. It introduce an auxiliary task to reduce the impact of noisy sentences, and proposes a Multi-Perspective Entity Matching Network to match entities from multiple perspectives. Extensive experiments on a real-world dataset show the effectiveness of our approach.

Acknowledgments

This work is supported by the National Key Research and Development Project of China (No. 2018AAA0101900), the Fundamental Research Funds for the Central Universities (No. 2019FZA5013), the Zhejiang Provincial Natural Science Foundation of China (No. LY17F020015), the Chinese Knowledge Center of Engineering Science and Technology (CKCEST) and MOE Engineering Research Center of Digital Library.

References

Muhammad Asif Ali, Yifang Sun, Xiaoling Zhou, Wei Wang, and Xiang Zhao. 2019. Antonym-synonym classification based on new sub-space embeddings. In *AAAI*.

Adrian Boteanu, Adam Kiezun, and Shay Artzi. 2019. Synonym expansion for large shopping taxonomies. In *AKBC*.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Kaushik Chakrabarti, Surajit Chaudhuri, Tao Cheng, and Dong Xin. 2012. A framework for robust discovery of entity synonyms. In *KDD*.

Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. 2009. Exploiting web search to generate synonyms for entities. In *WWW*.

Tao Cheng, Hady Wirawan Lauw, and Stelios Paparizos. 2012. Entity synonyms for structured web search. *IEEE Transactions on Knowledge and Data Engineering*, 24:1862–1875.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Hongliang Fei, Shulong Tan, and Ping Li. 2019. Hierarchical multi-task word embedding learning for synonym prediction. In *KDD*.

Christiane Fellbaum. 2000. Wordnet : an electronic lexical database.

William L. Hamilton, Zhitao Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

Yeye He, Kaushik Chakrabarti, Tao Cheng, and Tomasz Tylenda. 2016. Automatic discovery of attribute synonyms using query logs and table corpora. In *WWW*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Distinguishing antonyms and synonyms in a pattern-based neural network. In *EACL*, pages 76–85.

Meng Qu, Xiang Ren, and Jiawei Han. 2017. Automatic synonym discovery with knowledge bases. In *KDD*.

Xiang Ren and Tao Cheng. 2015. Synonym discovery for structured entities on heterogeneous graphs. In *WWW*.

Jiaming Shen, Ruiliang Lyu, Xiang Ren, Michelle Vanni, Brian M. Sadler, and Jiawei Han. 2019. Mining entity synonyms with efficient neural set generation. In *AAAI*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Jin Wang, Chunbin Lin, Mingda Li, and Carlo Zaniolo. 2019a. An efficient sliding window approach for approximate entity extraction with synonyms. In *EDBT*.

Zhen Wang, Xiang An Yue, Soheil Moosavinasab, Yungui Huang, Simon Lin, and Huan Sun. 2019b. Surfcon: Synonym discovery on privacy-aware clinical data. In *KDD*.

Xing Wei, Fuchun Peng, Huihsin Tseng, Yumao Lu, and Benoît Dumoulin. 2009. Context sensitive synonym discovery for web search queries. In *CIKM*, pages 1585–1588. ACM.

Jiale Yu, Weiming Lu, Wei Xu, and Zeyun Tang. 2019. Entity synonym discovery via multiple attentions. In *JIST*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jian Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S. Yu. 2019. Synonymnet: Multi-context bilateral matching for entity synonyms. *ArXiv*, abs/1901.00056.

Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Z. Li, and Nitesh V. Chawla. 2020. Few-shot knowledge graph completion. In *AAAI*.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jian Zhao. 2013. Improving question retrieval in community question answering using world knowledge. In *IJCAI*.