CKG: Dynamic Representation Based on Context and Knowledge Graph

Xunzhu Tang*
National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology
Wuhan, China
tangxz@hust.edu.cn

Tiezhu Sun
Momenta
Suzhou, China
suntiezhu@momenta.ai

Rujie Zhu
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL, USA
rujie.zhu@ucf.edu

Shi Wang*
Institute of Computing Technology, Chinese Academy
beijing, China
wangshi@ict.ac.cn

Abstract—Recently, neural language representation models pre-trained on large corpus can capture rich co-occurrence information and be fine-tuned in downstream tasks to improve the performance. As a result, they have achieved state-of-the-art results in a large range of language tasks. However, there exists other valuable semantic information such as similar, opposite, or other possible meanings in external knowledge graphs (KGs). We argue that entities in KGs could be used to enhance the correct semantic meaning of language sentences. In this paper, we propose a new method CKG: Dynamic Representation Based on Context and Knowledge Graph. On the one side, CKG can extract rich semantic information of large corpus. On the other side, it can make full use of inside information such as co-occurrence in large corpus and outside information such as similar entities in KGs. We conduct extensive experiments on a wide range of tasks, including QQP, MRPC, SST-5, SQuAD, CoNLL 2003, and SNLI. The experiment results show that CKG achieves SOTA 89.2 on SQuAD compared with SAN (84.4), ELMo (85.8), and BERT$_{Base}$ (88.5).

I. INTRODUCTION

The language representation models, such as ELMo [1], OpenAI-GPT [2] and BERT [3] have achieved good performance on a large range of NLP tasks including sentence classification, question answering, and sentence tagging. Existing language models mainly focused on pre-training on unstructured corpora with limited available information. For example, polysemy is a pervasive phenomenon. To mitigate it, current models require pre-training embedding on large corpus and then fine-tune them on downstream tasks. It is time-consuming and also difficult to capture more senses of a word by pre-training and fine-tuning [1], [3]. For instance, word apple has different meanings, including a fruit, a brand or IT products. The corpora for training embedding should be collected carefully to make sure different kinds of senses of a word in question are included. After pre-training, it still may be unclear how to interpret the embedding in downstream tasks. Considering sentence "People in cities usually buy ap-

ples and other fruits in local markets", downstream tasks need much cost to fine-tune it before they learn the exact meaning of word 'apple'. Therefore, performance improvement may be hindered by using unstructured training corpus only.

Existing methods predict the masked word based only on its left or/and right context which are local and can not predict the exact meaning of the word in the question. It will restrict the power of the trained embedding if local context is not large enough. The major limitation is that the context from few surrounded words is local. Now, a wealth of knowledge is available in different forms (*i.e.,* wikipedia, Freebase) on the web due to the advent of open data initiative. And it is growing rapidly. For example, the volume of data connected by Linked Data [1], a practice for sharing and connecting data using RDF specification [2], increases from 93 datasets (March 2009) to 1,239 datasets (as of March 2019). It is thus imperative to advance pre-training.

We propose model CKG to refine semantic of words by knowledge graph, which can make use of global knowledge. The proposed language models are not pre-trained on unstructured texts but on refined ones using knowledge graph, which can strengthen the meaning of each polysemous word in its context. We use BiLSTM to pre-train our model for reason that it uses both left-to-right and right-to-left LSTMs to generate features, and long input will worsen the pre-training performance. Naturally, we introduce 1D-CNN layer before BiLSTM for extracting features so that long input is shortened. We compare CKG with ELMo, BERT, and other models on wide variety of tasks. CKG achieves significant performance improvement compared to existing methods.

The contributions of our work are as follows:

- We propose CKG to mitigate polysemy by extending the words using knowledge graph and then fusing the semantic of words.

*Joint first authors

[1]http://linkeddata.org/home
[2]https://www.w3.org/2001/sw/wiki/RDF

- We optimize the training process by introducing 1D-CNN layer before BiLSTM to extract features so that dimension reduction can reduce computational costs further.
- We compare our method with ELMo, BERT, word2vec, GloVe in multiple tasks (*i.e.,* QQP, MRPC, and SST-5). Further, CKG+ELMo get SOTA 89.2 on SQuAD task compared with SAN(84.4), pure ELMo(85.8), and $BERT_{BASE}$(88.5).

II. RELATED WORK

Word representation has been the most important foundation for NLP tasks. Context-independent representations [4], [5] provide pre-trained word vectors from unlabeled text. They achieved good performance on several major NLP benchmarks including SNLI and SQuAD. These approaches have been generalized to sentence embedding and even paragraph embedding. However, these approaches for learning word vectors from large corpus only allow one context-independent representation for each word, making it difficult to understand text containing polysemous words. This problem is called unsupervised word sense disambiguation.

Some previous works have focused on enriching embedding with subword information [6], learning separate vectors for each word sense [7] and splitting word into several word senses based on graph clustering [8]. In addition, [9] uses world knowledge in the form of KGs to enhance learning models. Furthermore, [10], [11] employs general KGs information to improve BERT. Inspired by [8]–[10], our approach extends senses of central words by traversing its closest neighbors in knowledge graph. Such an approach can enrich embedding with different senses.

Other recent works have paid attention on learning context-dependent representations. Sentence or document encoders have been pre-trained on unlabeled texts and fine-tuned for a supervised downstream task to generate contextual token representations [2]. The advantage of these models is that few parameters need to be learned from scratch and they can mitigate polysemy problem to some extent. For example, ELMo [1] uses two BiLSTM layers to encode the context from large corpus on character level for semantic scalability, and fine-tunes word embedding on downstream tasks. Therefore, the model do not only captures complex characteristics of a word use but also mitigates polysemous problems by its dynamic word representations in different contexts. ELMo is a feature-based approach, which means that the pre-trained word representations can be added to downstream models as additional features. Different from ELMo, BERT [3] takes bidirectional encoder representations from transformers, which is designed to pre-train on large unlabeled corpus with masked language model on both left and right context in all layers. BERT created state-of-the-art performance in a large range of NLP tasks, but it costs too much time to do pre-training and fine-tuning. And similar to ELMo, when it comes to complex text with polysemous words, these approaches become inefficient.

In this paper, we will take full advantage BiLSTM for text encoding and benefit from knowledge graph to mitigate polysemous problems on NLP tasks. We call the approach in this paper as CKG. CKG is divided into two main parts, of which one is used to extend and refine semantic of the polysemous entities in a given text and the other extracts feature for dimension before pre-training. These two parts will be introduced as follows, and at same time we will show that CKG work well across a broad range of diverse NLP tasks.

III. DETAILS OF CKG MODEL

In this section, we introduce CKG and its detailed implementation, including the model architecture in Section III-A, the semantic extension designed for capturing more information from KGs in Section III-B and semantic fusion designed for extracting the correct information out of the extensions in Section III-C, and the details of the pre-training procedure in Section III-D.
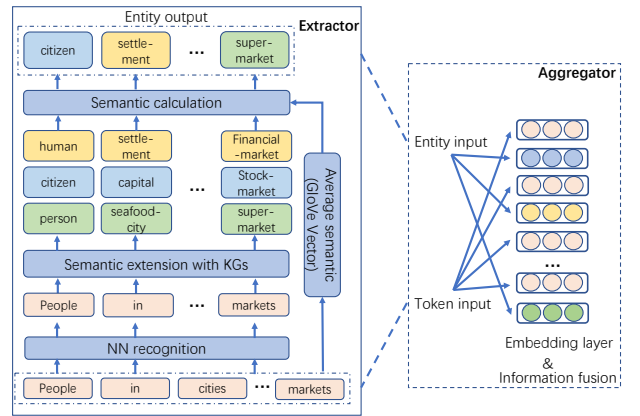

Fig. 1: Architecture of CKG, which consists of two main parts (*i.e.,* Extractor and Aggregator).

A. Model Architecture

As shown in Figure 1, the whole architecture of CKG consists of two main modules: (1) the semantic extension (**Extractor**) responsible to capture multiple concrete meanings of some entities they could have, and (2) semantic fusion (**Aggregator**) responsible to figure out the most possible ones of those extended meanings in a concrete sentence or context and fusing the results with central entities in footnote style.

To make the description more specific, we denote the input sequence as $\{w_1, w_1, ..., w_n\}$ [3], the tokens of extended entities as $\{e_1, e_2, ..., e_m\}$ [4]. We employ two kinds of initial ways of embedding to represent the words [5] in corpus and their extended entities. One is GloVe vector responsible to do calculation in Aggregator, and another one is embedding function from Pytorch responsible to initialize fused input tokens before pre-training. Given a token sequence

---

[3] w_i means the *i*-th token and in this paper, tokens are at the word level.
[4] We take the name style from [10]
[5] In this paper, we use nouns to represent the central entities to be extended.

$\{w_1, w_1, ..., w_n\}$, we figure out its corresponding entity sequence $\{e_{1,1}, e_{1,2}, e_{1,3}, e_{2,2}, ..., e_{m,1}, e_{m,2}, e_{m,3}\}$ by **Extractor**.

By similarity calculation, we finally get only one extension for each central entity. Input tokens and extended entities can get features $\{\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n}\}$, $\{\boldsymbol{e_{1,1}}, ..., \boldsymbol{e_{m,3}}\}$ as follows:

$$\{\boldsymbol{w_1}, \boldsymbol{w_1}, ..., \boldsymbol{w_n}\} = GloVe(\{w_1, w_2, ..., w_n\}) \quad (1)$$

$$\{\boldsymbol{e_{1,1}}, ..., \boldsymbol{e_{m,3}}\} = GloVe(\{e_{1,1}, ..., e_{m,3}\}) \quad (2)$$

where $GloVe(\cdot)$ is a initial word-vector function which can map a word to GloVe vector [5], and $e_{m,1}, e_{m,2}, e_{m,3}$ [6] are top 3 extensions of central entity m.

Firstly, CKG sums GloVe vectors of $w_1, w_2, ..., w_n$ and figure the cosine value between each of $\{e_{m,1}, e_{m,2}, e_{m,3}\}$ ($e_{m,i}$ is the i-th hop extension of entity m) and the average vector of $w_1, w_2, ..., w_n$. Then, it employ the one with minimum angle to represent the meaning of entity m in sequence $\{w_1, w_2, ..., w_n\}$.

After computing $\{e_{1,1}, ..., e_{m,3}\}$ with average vector of $\{w_1, w_2, ..., w_n\}$, CKG gets the most suitable extensions to represent the origin entities in the given sequence. Aggregator fuses these extensions with origin entities in the given sequence to inject the meanings of extensions out of KGs. We can get fused sequence as follows,

$$Aggregator(\{w_1, w_2, ..., w_n\}, \{e_{1,1}, ..., e_{m,3}\})$$
$$= \{\boldsymbol{w_1}, \boldsymbol{e_{1,2}}, ..., \boldsymbol{w_n}, \boldsymbol{e_{m,3}}\} \quad (3)$$

where $Aggregator(\cdot)$ is a function responsible for fusing extensions with their corresponding central entities. More details of **Extractor** and **Aggregator** will be introduced in Section III-B and III-C.

B. Semantic Extension Using Knowledge Graph (Extractor)

To understand the true meaning or sense of a word with multi meanings in a sequence, we should know potential meanings of the word. Both context-independent [4], [5] and contextual [1]–[3] representation models can not capture all the meanings of entities in a sequence. Therefore, we extend meanings of the entities in a sequence through KGs as more as possible [7]. However, it remains a problem which n-th-hop of traversed entity should be taken. In fact, it would be quite time-consuming if we take more than two-hop entities into account.

Given an extended route $({}_re_{m,1}, {}_re_{m,2}, {}_re_{m,3}, ..., {}_re_{m,k})$ where ${}_re_{m,k}$ represents the k-th hop extension in route r of central entity m, ${}_re_{m,k}$ has influence on ${}_re_{m,k-1}$, and ${}_re_{m,k-1}$ has influence on ${}_re_{m,k-2}, ..., {}_re_{m,2}$ has influence on ${}_re_{m,1}$. We donate the influence as $\Leftarrow$ and elements a set, and the smaller index means more influence. So, we can get that, ${}_re_{m,k-1} \Leftarrow ({}_re_{m,k})$, ${}_re_{m,k-2} \Leftarrow ({}_re_{m,k-1}, {}_re_{m,k}), ..., {}_re_{m,1} \Leftarrow ({}_re_{m,2}, {}_re_{m,3}, ..., {}_re_{m,k})$. After simple computing, we get that:

$$_re_{m,1} \Leftarrow ({}_re_{m,2}, {}_re_{m,3}, ..., {}_re_{m,k-1}, {}_re_{m,k})$$
$$_re_{m,1} \Leftarrow (({}_re_{m,3}, ..., {}_re_{m,k}), ..., ({}_re_{m,k}), 0) \quad (4)$$

[6] We only choose the top 3 extensions of each central entity.
[7] In this work, the one of KGs we take is Dbpedia.

In Equation 4, every former element in the set cover the information of the later ones, which satisfies the Markov property (sometimes characterized as "memory lessness") [12]. Therefore, we can convert the problem mentioned above into a Markov chain. In other words, we only take the first hop extensions into account while traversing entities in KGs of the central entities.

To be specific and easy-understanding, we give an example to describe the process of extension. Given a sentence "People in cities usually buy apples in the local markets.", we employ tool postag [13] of package nltk in python to extract properties of words in the sentence and we get that [('People', 'NNS'), ('in', 'IN'), ('cities', 'NNS'), ('usually', 'RB'), ('buy', 'VBP'), ('apples', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('local', 'JJ'), ('markets', 'NNS')]. CKG can easily capture the entities ['people', 'city', 'apple', 'market'] because they all are nouns. Then we traverse the central entities in knowledge graph and extend the first hop extensions of them which is described as Figure 2. Every entity may have many extensions in KGs, but for the simplicity and low computing cost, we only choose the top 3 extensions of each central entity by the similarity calculation of their GloVe vectors. For the convenience, we donate the extensions of entity e as set dt['e']. So, dt['apple'] = ['tree', 'culture', 'fruit']. However, in general, each word in a specific sequence has only one meaning. Therefore, it turns to be a problem that which extension will be used to emphasize the meaning of the central entity. This will be discussed in Section III-C.
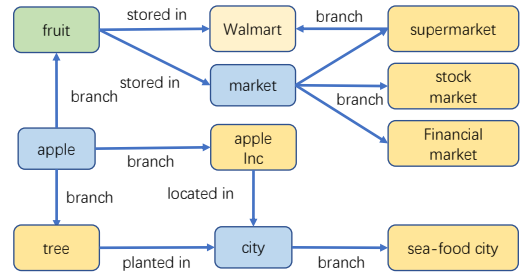

Fig. 2: Partial Knowledge Graph from Dbpedia. The figure shows that 'apple' could be a kind of food, as well as a company. We can get the precise meaning of 'apple' by the neighbours around it.

C. Fusing Extensions and Central Entities (Aggregator)

After we extracted top 3 possible meanings for each central entities, it turns to be a problem that which one extension will be chosen to represent the true meaning of the central entity in a specific sequence. In this section, we present **Aggregator** to mitigate this problem.

In sequence $\{w_1, w_2, ..., w_n\}$, we can get the initial vectors $\{\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n}\}$ by the Equation 1. The extensions of entities in the sequence can get initial vectors by the same way and we get that $\{\boldsymbol{e_{1,1}}, ..., \boldsymbol{e_{m,3}}\}$. CKG sums $\{\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n}\}$ and then figure out the average of them, which is shown in Equation 5.

$$V_{avg} = \frac{\sum_i \boldsymbol{w_i}}{\{len(w_1, w_2, ..., w_n)\}} \quad (5)$$

$$e_m \to \underset{k\epsilon\{e_{m,1}, e_{m,2}, e_{m,3}\}}{\arg\max} \left(\cos\left(GloVe(k), V_{avg}\right)\right) \quad (6)$$

Then, CKG figures out the cosine value between each of $\{e_{m,1}, e_{m,2}, e_{m,3}\}$ with the average of the $\{w_1, w_2, ..., w_n\}$ and employ the one which the minimum angle to represent the meaning of entity m in the sequence, which is shown in Equation 6. In Equation 6, we define a function $\cos(a, b)$ which is used to figure out the cosine value of vector a and b. Besides, we donate the relation "with the meaning of " as symbol '$\to$', and Function $\underset{k\epsilon\{e_{m,1}, e_{m,2}, e_{m,3}\}}{\arg\max} \left(\cos\left(GloVe(k), V_{avg}\right)\right)$ means that we take the element k which makes $\cos\left(GloVe(k), V_{avg}\right)$ maximum.

In the example of Section III-B, e_m= 'apple', and top 3 extensions of apple in KGs is ['tree', 'culture', 'fruit'], which means dt$_{'apple'}$ = ['tree', 'culture', 'fruit'] (*i.e.,* for 'apple', e_1 = 'tree',e_2 = 'culture', e_3 = 'fruit'). After computing e_i with V_{avg} with Equation 6, we can easily get that 'apple' $\to$ 'fruit'. Similar to 'apple', 'people' $\to$ 'citizen', and 'city' $\to$ 'settlement', and 'market' $\to$ 'supermarket'.

Finally, we add these precise meanings to the origin sequence with the form of note. As a result, we get a new sentence "People (citizen) in cities (settlement) usually buy apples (fruit) in the local markets (supermarket)". And use this new sentence to replace the original input before training in Figure 3 in the next section.

D. Semantic Feature Extraction for Dimension Reduction

After semantic extension and fusing, we need to train a context-dependent representation model and test the performance of CKG. It proved that BiLSTM is a great sequence-to-sequence model which was widely used to train language models and has a good performance on most NLP benchmarks. Therefore, in this section, we take bidirectional LSTM to train our language model.

However, BiLSTM has a step size limit to the length of input sequences because it is a sequential model. The general step size limit ranges from 250 to 500. If the length of the sequential step of BiLSTM is oversize, the gradient will vanish soon. Moreover, it is time-consuming for BiLSTM because the outputs of prior cells will be used as inputs of the next cell. While if we reduce the dimension of the input sequence of BiLSTM or the input is full features, the time costs for training would be greatly saved and the gradient vanishing would also be avoided.

Before BiLSTM layer, we introduce a 1D-CNN layer for identifying patterns within data. The patterns will then be used to form more complex patterns in higher layers. So, after 1D-CNN extraction processing, the information of origin sequence $(t_1, t_2, \ldots, t_n)$ can be represented as a middle state embedding sequential $(c_1, c_2, \ldots, c_m)$ where $m < n$. Then we put the middle sequential into BiLSTM layer in CKG. A forward language model computes the probability of the
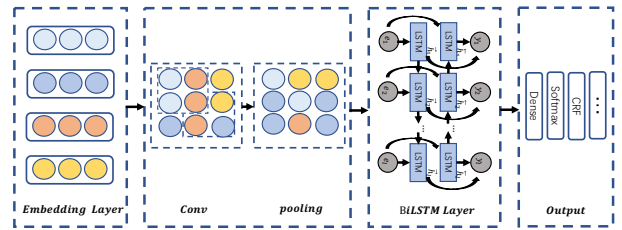

Fig. 3: Semantic feature extraction from input embedding so that we can not only save the training time, but also avoid vanishing gradient.

sequence by modeling the probability of c_k given the history $(c_1, c_2, \ldots c_{k-1})$.Each position k of forward LSTM will output a context-dependent representation $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$ (where $j = 1, 2, \ldots, L$). Similar to forward LSTM, the backward LSTM will get $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$. We connect results of forward and backward LSTMs as $\mathbf{h}_{k,j}^{LM} = \overrightarrow{\mathbf{h}}_{k,j}^{LM} + \overleftarrow{\mathbf{h}}_{k,j}^{LM}$. The probability of the sequence is shown as Equation 8.

$$p(c_1, c_2, \ldots, c_m) = \prod_{k=1}^{m} p(c_k | c_1, c_2, \ldots, c_{k-1}) \quad (7)$$

$$p(c_1, c_2, \ldots, c_m) = \prod_{k=1}^{m} p(c_k | c_{k+1}, c_{k+2}, \ldots, c_m) \quad (8)$$

We design a loss function to adjust CKG parameters, which is shown as the Equation 9. The goal in our work is to make the Equation 9 maximum. It needs to modify the weights and values of our network until we make the result of Equation 9 maximize. And then we use output layer like Softmax layer, crf layer to do the downstream tasks.

$$\log p(c_k | c_1, \ldots, c_{k-1}) + \log p(c_k | c_{k+1}, \ldots, c_m)) \quad (9)$$

IV. Experiments

In this section, we present the details of pre-training CKG and the fine-tuning results on nine NLP datasets, which contain knowledge-driven tasks and common NLP tasks. In this paper, we train BiLSTM on a corpus with approximately 30 million sentences [14] which is mentioned in [1]. The pre-trained biLSTM here is similar to the architectures in [1].

A. Qualitative Argument

In order to demonstrate the performance of enhancing semantic in specific context, we conduct CKG on an entity typing task with a well-established dataset FIGER [15]. The training set of FIGER is labeled with distant supervision, and its test set is annotated by human. Following the evaluation method in [16], we compare NFGEC, ERNIE, CKG on FIGER, and adopt strict accuracy, loose macro, loose micro scores for evaluation standard. The experimental results on FIGER is shown in Table I, the baseline methods for entity typing we compare our models with are as follows:

- **NFGEC** [16]. NFGEC is the SOAT model on FIGER, which combines the representations of three-granularity (i.e., entity mention, context, and extra hand-craft) features as input.

- **ERNIE** [10]. ERNIE is a pre-trained method considering both large-scale textual corpora and KGs, which outperforms on FIGER.

TABLE I: Results of various models on FIGER(%)

Model	NFGEC (Attentive)	NFGEC (LSTM)	ERNIE (tsinghua)	CKG
Acc.	54.53	55.60	57.19	58.84
Macro	74.76	75.15	76.51	76.23
Micro	71.58	71.73	73.39	75.24

Table I shows the results in FIGER, and we can find that: (1) ERNIE achieves higer accuracy than NFGEC method, indicating the external knowledge regularizes ERNIE to avoid fitting the noisy labels and accordingly benefits entity typing. (2) CKG has lower macro than the best ERNIE method, but it significantly improves the strict accuracy and micro, which achieves 58.87% in accuracy (2.89% higher than ERNIE) and 76.23% (2.52% higher than the second score). Obviously, CKG can be used to reduce the noisy label challenge in FIGER by injecting the external information from KGs.

B. Benchmarks

To make a objectively comparison with baseline models, we conduct experiments separately on the following datasets.

- Word analogy task [4]. The dataset is divided into 14 subsets and made up of 19,544 question like "a is to b like c is to d".
- Quora Question Pairs (QQP) [17]. QQP is a binary classification task including 400,000 question pairs, where we need to value if two questions in each pair are equal or not.
- Microsoft Research Paraphrase Corpus (MRPC) [18]. MRPC contains sentence pairs which are extracted from online news sources, in which there are annotations for whether the sentences are equal in semantic from human.
- Stanford Sentiment Treebank (SST-5) [19]. SST-5 is a binary single-sentence classification task from the Stanford Sentiment Tree bank, including five levels of comment(from very negative to very positive), used to describe a sentence out of a movie review.
- Stanford Question Answering Dataset (SQuAD v1.1) [20].SQuAD consists of questions raised by a crowd of workers on a set of Wikipedia articles, each of which is a piece of text or span from the corresponding reading paragraph.Each pair contains a question and a passage where we can find the answer.
- Named Entity Extraction (NER) [21]. NER includes 1,393 English and 909 German news articles. To build the English-language corpus we need the RCV1 Reuters corpus where entities are annotated with LOC (location), ORG (organisation), PER (person) and MISC (miscellaneous).
- Stanford Natural Language Inference (SNLI) [22]. SNLI takes a pair of sentences and predicts if the former one

cover the meaning of the later one, which collects 570k English sentence pairs written by human and labeled with entailment, contradiction and neutral as the judgment.

C. Results

TABLE II: Comparison among CKG+ELMo and other models in QQP, SST-5, and MRPC tasks

Baseline	Pre-OpenAI SOTA	BiLSTM+ ELMo+Attn	ERNIE (tsinghua)
QQP	93.2	90.4	93.5
MRPC	86.0	84.9	88.9
SST-5	66.1	64.8	71.2
	ERNIE2.0 (baidu)	CKG	CKG+ELMo
QQP	90.4	92.7	93.2
MRPC	88.9	88.2	86.3
SST-5	70.4	70.2	72.3

Word analogy task. We combine the part one of CKG shown as Figure 1 with GloVe to retrain wikipedia2014 and get a new word vectors which improve the score on word analogy task to 79.37. Main details are shown in the Table III. Table III indicates that we get a higher score based spearman algorithm, comparing with different models like CBOW, SG, and pure Glove. Table III shows CKG+GloVe gets 10.44 score more than pure GloVe on word analogy task.

TABLE III: Comparison of models in word similarity with rank of sepearman. And the comparison of word analogy in semantic, syntactic, average.

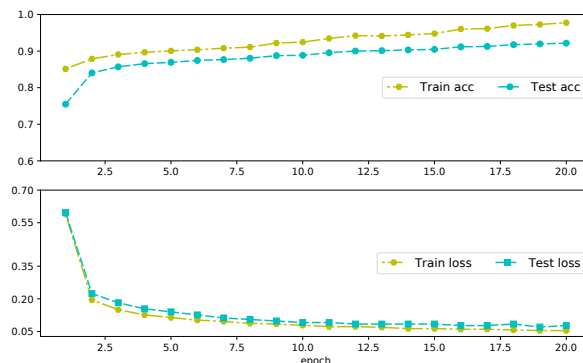| Baseline	Semantic	Syntactic	Average	Rank of Spearman
CBOW	73.58	65.95	69.5	73.25
SG	65.62	56.61	60.64	68.69
GloVe	71.39	53.72	61.57	68.93
CKG+GloVe	78.34	69.32	73.83	79.37

Fig. 4: The detailed information on the dataset of SNLI. The upper figure represents the accuracy results of training and test in different epochs, and the lower figure indicates the losses of them.

QQP, MRPC, and SST-5. In QQP, CKG+ELMo gets score 93.2 equals Pre-OpenAI SOTA and better than pure ELMo (90.4) (shown as Table II), but ERNIE performs better than our CKG. In both MRPC and SST-5, CKG+ELMo gets score 86.3/72.3 which is better than the other Pre-OpenAI SOTA and

pure ELMo. In addition, CKG+ELMo performs better than other models on SST-5.

SQuAD (v1.1). Our baseline [1] is an improved version of an unsupervised LM in [23]. It is deep, and in the sense, ELMo representations are a function dealing with all of the internal layers of the biLM. We combine CKG and BiDAF [24] to conduct experiment on SQuAD. Table V shows results on the [20] with combination of CKG and ELMo. In [20], there will give a question and a paragraph containing the answer to the former question. ELMo gets score 85.8 based on the baseline of SAN with BiLSTM. $BERT_{BASE}$ gets 88.5 with its outstanding pre-training parameters and fine-tuning part. Above all, CKG+ELMo get a higher score of 89.2, which proves the dynamic adjustment model in CKG effective in natural language processing task. As a fine-tuning tool, the CKG always performs well considering the global factor.

TABLE IV: Analysis for SQuAD, SNLI, and NER, comparing different choices. We compares systems with pure GloVe, ELMo, the combination of GloVe and CKG shown as Figure 1 and only CKG. The set of comparison of the three models. ELMo has achieved a great performance in NER task, while CKG performs better.

Task	GloVe	ELMo	CKG+GloVe	CKG
SQuAD	80.8	85.8	85.6	88.7
SNLI	88.1	89.1	90.2	91.1
NER	87.7	91.9	-	92.56

TABLE V: The set of comparison of the five models. We compare SAN, ELMo, $BERT_{BASE}$, with CKG and CKG+ELMo across SQuAD task. The "INCREASE" column lists improvement over our baseline.

Model	SAN	ELMo	BERT (base)	CKG	CKG + ELMo
SOTA	84.4	85.8	88.5	88.7	89.2
INCREASE	baseline	1.66%	4.86%	5.09%	5.69%

SNLI. Our baseline is [1], and we combine CKG with ESIM [25] to deal with semantic entailment task. ESIM is a sequence model which uses a biLSTM to encode the premise and hypothesis. And ESIM has three layers including a matrix attention layer, a local inference layer, and biLSTM inference composition layer, followed by a pooling operation before the output layer. In general, the combination of CKG and ESIM improves accuracy by an average of 2.7%. CKG pushes the overall accuracy to 91.1%, exceeding [1].

NER. Our baseline [1], [23] uses pre-trained word embeddings trained on a big corpus, and includes a character-based CNN representation for word-level scalability, two biLSTM layers for hidden states and a conditional random field (CRF) loss. As shown in Table IV, CKG with CRF achieves the score 92.56. The main difference between CKG and the previous great work from [1] is that we took the semantic refining before pre-training into consideration.

V. CONCLUSION

This paper presents a new model called CKG, including extending and refusing semantic of entities by KGs. To obtain more meanings of a polysemous entity, we traverse its neighbours in Dbpedia, and get neighbours within one-hop as its extensions. Then we choose one most similar to the average meaning of the rest entities to represent the true meaning of the central entity. As a result, we can mitigate polysemy problem in NLP. In addition, CKG can improve the semantic in entity type classification, and outperforms in multiple NLP tasks.

REFERENCES

[1] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL*, 2018.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *Technical report, OpenAI*, 2018.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013, pp. 3111–3119.

[5] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceddings of EMNLP*, 2014, pp. 1532–1543.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *TACL*, vol. 5, pp. 135–146, 2017.

[7] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, "Efficient non-parametric estimation of multiple embeddings per word in vector space," in *Proceedings of EMNLP*, 2014, pp. 1059–1069.

[8] H.-S. Chang, A. Agrawal, A. Ganesh, A. Desai, V. Mathur, A. Hough, and A. McCallum, "Efficient graph-based word sense induction by distributional inclusion vector embeddings," in *Proceedings of NAACL-HLT*, 2018, p. 38.

[9] K. Annervaz, S. B. R. Chowdhury, and A. Dukkipati, "Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing," in *Proceedings of NAACL-HLT*, 2018, pp. 313–322.

[10] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," in *Proceedings of ACL*, 2019, pp. 1441–1451.

[11] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, "Ernie: Enhanced representation through knowledge integration," *arXiv preprint arXiv:1904.09223*, 2019.

[12] F. SPITZER, "Interaction of markov processes," *ADVANCES IN MATHEMATICS*, vol. 5, pp. 246–290, 1970.

[13] M.-C. De Marneffe and C. D. Manning, "Stanford typed dependencies manual," Stanford University, Tech. Rep., 2008.

[14] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005*, 2013.

[15] X. Ling, S. Singh, and D. S. Weld, "Design challenges for entity linking," *TACL*, pp. 315–328, 2015.

[16] S. Shimaoka, P. Stenetorp, K. Inui, and S. Riedel, "An attentive neural architecture for fine-grained entity type classification," in *Proceedings of AKBC (WS)*, 2016, pp. 69–74.

[17] Z. Chen, H. Zhang, X. Zhang, and L. Zhao, "Quora question pairs," 2018.

[18] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of IWP*, 2005.

[19] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of EMNLP*, 2013, pp. 1631–1642.

[20] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of EMNLP*, 2016, pp. 2383–2392.

[21] E. F. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," *Development*, 2003.

[22] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[23] M. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," in *Proceedings of ACL*, 2017, pp. 1756–1765.

[24] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[25] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," in *Proceedings of ACL*, 2017, pp. 1657–1668.