# Semantic Definition Ranking

Zehui Hao[1], Zhongyuan Wang[2], Xiaofeng Meng[1(✉)], Jun Yan[2], and Qiuyue Wang[1]

[1] School of Information, Renmin University of China, Beijing, China
{jane0331,xfmeng,qiuyuew}@ruc.edu.cn
[2] Microsoft Research Asia, Beijing, China
wzhy@outlook.com, junyan@microsoft.com

**Abstract.** Question answering has been a focus of much attention from academia and industry. Search engines have already tried to provide direct answers for question-like queries. Among these queries, "What" is one of the biggest segments. Since results excerpted from Wikipedia often have a coverage problem, some models begin to rank definitions that are extracted from web documents, including Ranking SVM and Maximum Entropy Context Model. But they only adopt syntactic features and cannot understand definitions semantically. In this paper, we propose a language model incorporating knowledge bases to learn the regularities behind good definitions. It combines recurrent neural network based language model with a process of mapping words to context-appropriate concepts. Using the knowledge learnt from neural networks, we define two semantic features to evaluate definitions, one of which is confirmed to be effective by experiments. Results show that our model improves precision a lot. Our approach has been applied in production.

**Keywords:** Definition ranking · Question answering · Recurrent neural network · Conceptualization

## 1 Introduction

Question Answering (QA), which provides direct answers instead of ten blue links for users' queries, has become a hot trend in web searching. Definition questions, like "What is bandy?", occupy more than 20% of query logs in QA systems [4]. In this paper, we focus on this type of queries.

Figure 1 shows that search engines have already offered definitions for some definienda (the terms being defined). However, these answers are a little coarse and narrow in coverage. Some are just excerpted from *Wikipedia*.

This research was partially supported by the grants from the National Key Research and Development Program of China (No. 2016YFB1000603, 2016YFB1000602); the Natural Science Foundation of China (No. 61532010, 61379050, 91646203, 61532016); Specialized Research Fund for the Doctoral Program of Higher Education (No. 20130004130001), and the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University (No. 11XNL010).


S. Candan et al. (Eds.): DASFAA 2017, Part II, LNCS 10178, pp. 153–168, 2017.
DOI: 10.1007/978-3-319-55699-4_10

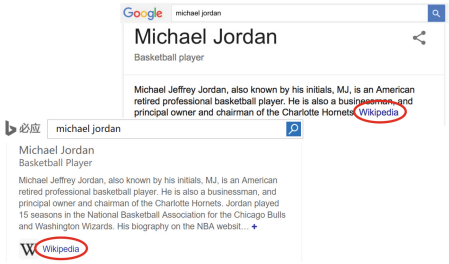



**Fig. 1.** Definitions excerpted from *Wikipedia*

To break the bottleneck of *Wikipedia*, some approaches begin to extract answers from unstructured texts in the web. A general pipeline consists of two steps. First, use human-defined rules to collect definition candidates automatically from web documents. Then, rank the candidates through a scoring system. Typically, the second step is based on co-occurrence frequency [8,23], scores learnt from Support Vector Machine (SVM) or Language Models (LMs) [4,24], and discriminant functions of Maximum Entropy Model [5]. These methods, even if some simple LMs, cannot analyze sentences thoroughly in semantics. The candidates they prefer are the ones that look like definitions, but may not give a precise description. For the second step here, which we call "definition ranking", unsatisfying results mainly come from the following challenges:

- **Semantic analysis:** The semantics in unstructured texts are complicated. It is hard to define semantic features for machines to tell good definitions from bad ones like human beings.
- **Language polysemy:** Most words denote more than one sense in natural language. Different senses often result in misunderstanding and increase the difficulty of ranking.
- **Text particularity:** Definitions are a special kind of texts. Traditional language models cannot be applied directly to analyze them.

With the effective practice of neural networks, recent studies on text understanding and information retrieval have begun to take advantage of implicit knowledge representation models. Based on the co-occurrence relation, the most successful method [9] maps words to real-number vectors in a low dimensional space. But these models pay little attention to other relations, such as the is-a relation, which plays an important role when defining a term.

To make up for above drawbacks, we take an explicit knowledge representation model [6,18,21] into consideration. It infers the most likely concept of a word in a specific context. In other words, it understands texts with the help of the is-a relation among words. Because each concept is a clear category name shared by a group of hyponyms in KBs, this model captures explicit semantics.

Good definitions usually contain the notion and major attributes of definienda. In general, the concepts of these words are quite related to the concepts of definienda, while the concepts of irrelevant words or trivial matters are

not. **Our starting point is that good definitions should be sentences not only full of related words with definienda, but keeping the concepts of these words coherent.** Take the following definition candidates of "emphysema" as an example:

**Candidate1.** Emphysema is a disease associated with smoking, and usually manifests itself in patients after 50 years of age, affecting about two million Americans each year.

**Candidate2.** Emphysema is a progressive lung disease that primarily affects smokers and causes shortness of breath and difficulty breathing.

Candidate1 does not mention any symptom of "emphysema". Impacts, like "affecting about two million Americans each year", are less important when defining a disease. Candidate2 contains the notion and major symptoms without too much trivial details.

If we only rely on syntactic features or word-vector similarity, Candidate1 can still rank high. It has a definition format and seems longer than Candidate2. Moreover, "emphysema" is indeed related to some words in it, and is likely to appear with "Americans" frequently in web documents. But looking at a concept level, we will find that the concepts of some words in Candidate1 (like ⟨Americans, *nation*⟩[1]) do not relate to the concept of "emphysema" (*disease*) as closely as Candidate2. Section 5.4 explains how our model analyzes at a concept level in details.

In this paper, we propose a combined model to rank answers semantically for definition questions. Our contributions are in two aspects:

- We combine the neural network with the is-a relation through conceptualization and make them complement each other, so that our model is more suitable to handle definition sentences.
- We provide a semantic feature to judge definitions, which is learnt from both unstructured texts and KBs.

The rest of this paper is organized as follows. Section 2 introduces related works, including definition ranking and conceptualization. Section 3 gives some preliminaries about Recurrent Neural Network based Language Model and analyzes its drawbacks. Section 4 describes our model and defines semantic features. Section 5 explains the experiment framework and shows the results. We conclude the paper in Sect. 6.

## 2 Related Work

Many approaches have been proposed to rank definition candidates collected from the web. Xu, Licuanan, and Weischedel [23] learn the centroid vector (i.e., vector of word frequency) from *Wikipedia*. Definitions are ranked according to

[1] The angle brackets mean a word and its concept.

how closely the distribution of their descriptive words correlates with the centroid vector of definienda. Kaisser, Scheible, and Webber [8] employ a similar approach and use the frequency of descriptive words as their weights. These statistical methods only grasp the shallow co-occurrence relation among words. The performance falls into decrease for definienda not covered by *Wikipedia*.

Xu et al. [24] rank definitions through Ranking SVM [7,20]. The features they adopt are syntactic. In practice, sentence length often dominates the preference and reduces accuracy.

Language models are also used for definition ranking. Chen, Zhou, and Wang [4] compare the influence of three kinds of features, including unigrams, bigrams and biterms. However, these LMs ignore the problem of language polysemy. Only one representation for one word may lead to misunderstanding.

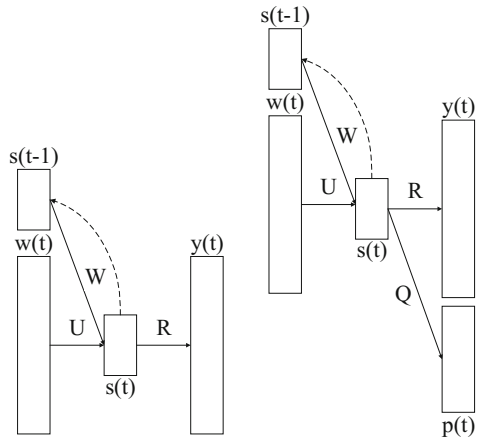
Figueroa and Atkinson [5] propose a Maximum Entropy Context (MEC) Model which performs better than centroid vector models and biterm LMs. It accounts for regularities across both positive and negative examples. Although some context indicators are merged to classify definienda, they still only adopt syntactic features finally.

A related work introduces an unsupervised reasoning process which determines context-appropriate concepts for words, called conceptualization [18,21]. For example, seeing "product of apple", we know that "apple" is a *company* or *brand* rather than *food* or *fruit*. Under the hypothesis that concepts of adjacent words should be coherent, Hua et al. [6] develop an efficient framework to perform human-like conceptualization. Given a word in a sentence, it first obtains a concept set $\{c_1, \ldots, c_n\}$ from the is-a relation among words. Then, it uses a weighted-vote algorithm to pick out the most appropriate concepts based on the co-occurrence relation among concepts. The two kinds of relation both come from KBs. Probabilistic knowledge in KBs enables probabilistic reasoning and helps understand sentences precisely. However, compared to the is-a relation, the offline co-occurrence relation is so large that the whole framework becomes too heavy.

## 3 Preliminary

Recurrent Neural Network based Language Model (RNNLM) [12] is a robust model in natural language processing. Although humans can use long contexts to understand sentences, classic N-gram models barely rely on several preceding words to predict the next word. To break the limitation, RNNLM maintains the hidden layer to make contextual information loop inside the network. It has an advantage of learning arbitrary-length history itself over other LMs.

Considering the task of definition ranking, our target is to let the LM learn the regularities behind good definitions. Despite semantic coherence, words in good definitions are usually organized in a regular way (which syntactic features also try to learn). A LM ignoring the order of words is not competent to handle this special kind of texts. Fortunately, RNNLM often shows an outstanding ability to comprehend sequential data [10,11,19]. This is why we choose it instead of




(a) Original RNNLM (b) CRNNLM

**Fig. 2.** Model structure of RNNLM

other LMs which perform better on training word vectors or computing word similarity.

The structure of original RNNLM is shown in Fig. 2(a). The input vector $w(t)$ and the output vector $y(t)$ are both $V$-dimensional, where $V$ is the vocabulary size. $w(t)$ represents the input word in time $t$ and uses 1-of-$V$ coding. $y(t)$ represents probability distribution over all the words. Its each dimension corresponds to the probability of a word to appear next, namely $P(w_i|w(t), s(t-1))$, where $w_i$ is a word in vocabulary. The other vector in the input layer, $s(t-1)$, which represents contextual information, is copied from the hidden layer of last iteration. Vector $s(t)$ and $y(t)$ are computed as follows:

$$s(t) = F(Uw(t) + Ws(t-1)) \tag{1}$$

$$y(t) = G(Rs(t)) \tag{2}$$

where $U$, $W$ and $R$ on arrows are transition matrices. $F(z)$ is sigmoid activation function, and $G(z_m)$ is softmax function:

$$F(z) = \frac{1}{1+e^{-z}}, \quad G(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

The objective of training RNNLM is to maximize the probability of words in the training corpus:

$$\sum_{t=1}^{T} \log P(w(t+1)|w(t), s(t-1)) \tag{3}$$

Despite its outstanding performance, RNNLM is often criticized for high computational complexity, especially the calculation between $s(t)$ and $y(t)$. One way to speedup is to factorize $y(t)$ as shown in Fig. 2(b), called CRNNLM (RNNLM

with classes) [13]. Words in vocabulary are classified in advance according to their frequency. Vector $p(t)$ is probability distribution over all the classes, and is computed similarly to Eq. 2:

$$p(t) = G(Qs(t)) \tag{4}$$

where $Q$ is also a transition matrix. Probability estimation and backpropagation are done only on part of $y(t)$, that is the words sharing the same class with $w(t+1)$ (the real next word in the corpus). Other words are put aside for the moment. Therefore, computational complexity between the hidden and output layer is reduced. Word probability turns to:

$$P(w_i|w(t), s(t-1)) = P(w_i|p_i, s(t))P(p_i|s(t)) \tag{5}$$

where $w_i$ is a word which must belong to the same class $p_i$ with $w(t+1)$.

We illustrate the training process of CRNNLM through the sentence "She is eating apple pies". We hypothesizes that "eating" is the current input word, and is the 100th word in the vocabulary. Hence, the 100th dimension of $w(t)$ is 1, and other dimensions are 0. Vector $s(t-1)$ is copied from the hidden layer of last iteration, namely when "is" is the input word. Vector $s(t)$ and $p(t)$ are obtained according to Eqs. 1 and 4. Since the next word is "apple", we check which words belong to the same class with it. Only the corresponding dimensions of these words, including "apple" itself, in $y(t)$ are computed. Each dimension of $p(t)$ is the probability of a class to appear next, namely $P(p_i|s(t))$ in Eq. 5. Each dimension of $y(t)$ is the conditional probability of a word, namely $P(w_i|p_i, s(t))$. After that, we do backpropagation based on the objective Function 3 to update the parameters in all transition matrices, namely $U$, $W$, $R$ and $Q$. After reading all the sentences in the corpus, the training of these matrices is finished. Then, we can use the whole network to predict other sentences.

## 4 RNNLM Combined with Knowledge Bases

Despite the effectiveness of CRNNLM, there are some drawbacks in its structure. For the output layer, whether a word to be considered or not just depends on its frequency proximity to $w(t+1)$. The factorization method is a little arbitrary, making RNNLM lose much semantics.

For the input layer, since vector $w(t)$ uses 1-of-$V$ coding, every column in matrix $U$ is actually the implicit representation of a word in the vector space. $Uw(t)$ in Eq. 1 is adding the vector of the input word to the context. But words are polysemous. For sentences "product of apple" and "eating apple pie", the same vector of two "apple" is used. Different senses may disturb the training and predicting process.

Therefore, we propose two variants complementing CRNNLM with the is-a relation among words, making RNNLM adapt to the definition-ranking task naturally. Since the is-a relation comes from KBs, we call our model KRNNLM. Both of our variants conduct a reasonable method to factorize the output layer

and implement learning-based conceptualization in the input layer. On the basis of KRNNLM-1, KRNNLM-2 adds a direct connection between the input and output layer, which strengthens the effect of conceptualization.

### 4.1 The First Variant (KRNNLM-1)

**Factorization Based on Is-A Relation.** As shown in Fig. 3(a), KRNNLM-1 replaces the class vector $p(t)$ with a concept vector $cout(t)$. Its each dimension equals to the probability of a concept given the context, namely $P(cout_j|s(t))$, where $cout_j$ is a concept in KBs. In the output layer, we estimate probability distribution first over all concepts ($cout(t)$), and then over the words sharing the same concepts with the next word (part of $y(t)$). Vector $cout(t)$ is obtained similarly to Eq. 2:

$$cout(t) = G(Qs(t)) \tag{6}$$

Word probability turns to:

$$\begin{aligned} P(w_i|w(t), s(t-1)) &= P(w_i|C_{w_i}, s(t))P(C_{w_i}|s(t)) \\ &= P(w_i|C_{w_i}, s(t)) \sum_{j \in C_{w_i}} P(cout_j|s(t)) \end{aligned} \tag{7}$$

where $C_{w_i}$ denotes the concept set of $w_i$ in KBs.

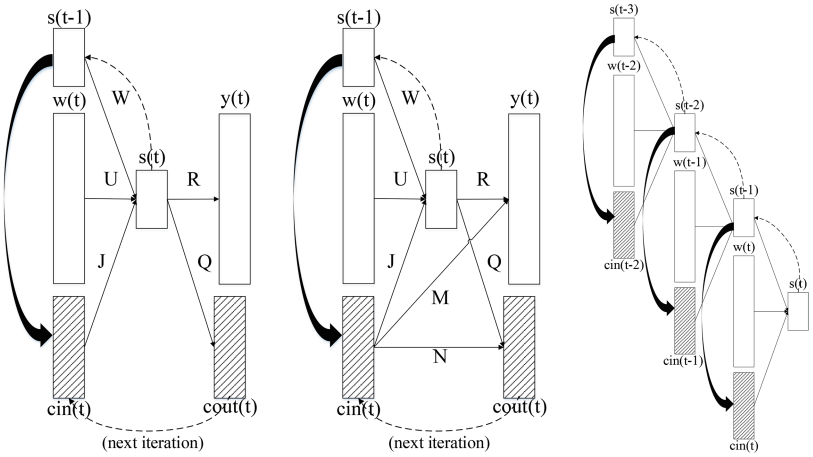
Vector $cout(t)$ here seems like $p(t)$ in CRNNLM. The difference between them does not lie in themselves, but how they influence the factorization of $y(t)$. In CRNNLM, words are classified according to frequency. When we update the network, it makes no sense to discard those words just because their frequency is not proximate to the next word.

In KRNNLM, words are classified according to the is-a relation in KBs. We think that words are more likely to appear in the next position if they belong to the same concepts with $w(t+1)$. These words are what we are really interested in when $time = t$. Besides, words can belong to more than one concept in KBs because of polysemy, so there is a summation in Eq. 7.

To understand our factorization method more clearly, we take "apple" as the next word $w(t+1)$ for example, and go through the computation process from the hidden layer to the output layer. It consists of two parts. For the concept part $cout(t)$, we use Eq. 6 to get the probability distribution of all concepts, including the concepts of "apple". For the word part $y(t)$, according to KBs, "apple" have four concepts, namely *company*, *brand*, *food* and *fruit*. Therefore, we only select words belonging to the four concepts to update (like "microsoft" and "orange") and discard others. This is how our model combines KBs to reduce computational complexity of original RNNLM without losing too much semantics.

**Conceptualization in the Input Layer.** In Fig. 3(a), KRNNLM-1 also adds a concept vector $cin(t)$ to the input layer. The objective function turns to:

$$\sum_{t=1}^{T} \log P(w(t+1)|w(t), cin(t), s(t-1)) \tag{8}$$




(a) KRNNLM-1 (b) KRNNLM-2 (c) BPTT of KRNNLM

**Fig. 3.** Structure of KRNNLM (The bold arrow means that there exists a conceptualization process.)

The hidden layer is computed as:

$$s(t) = F(Uw(t) + Jcin(t) + Ws(t-1)) \tag{9}$$

Similar to vector $cout(t)$, each dimension of $cin(t)$ equals to the probability of a concept given the context, namely $P(cin_j|s(t-1))$. Note that the context here is not $s(t)$ but $s(t-1)$. In fact, this value has already been estimated in the output layer when $time = t-1$, so we obtain $cin(t)$ from $cout(t-1)$ in last iteration. This operation is similar to the loop of $s(t)$. In Fig. 3(a), the dashed arrow means copying the concept vector in the output layer to the input layer of next iteration.

In practice, we do not directly copy $cin(t-1)$ to $cout(t)$. In order to decrease noise, we only retain concepts of the input word. The values of other concepts in $cin(t)$ are set to zero. In this way, $cin(t)$ represents the context-specified probability of linking the input word to each of its concepts. Specifically, each dimension of $cin(t)$ is computed as follows:

$$P(cin_j|s(t-1)) = \begin{cases} \frac{P(cout_j|s(t-1))}{\sum_{k \in C_w} P(cout_k|s(t-1))}, & j \in C_w \\ 0, & \text{else} \end{cases}$$

where $cin_j$ is the $j$-th concept in $cint(t)$, and $cout_j$ corresponds to the same dimension in $cout(t-1)$. $C_w$ denotes the concept set of the input word.

Vector $cin(t)$ here is intended to distinguish different senses of the input word in given contexts. Take "apple" as the input word $w(t)$ in sentence "eating apple pies" for example. Through Eq. 9, concept *food* and *fruit* are added to the hidden layer with larger probability than concept *company* and *brand*.

$Uw(t) + Jcin(t)$ represents not only the word "apple", but an "apple" with specific semantics. We do not assign a vector to each sense of a word like [3,16], but achieve disambiguation through incorporating concepts to the vector space.

The process of determining $cin(t)$ is doing human-like conceptualization. Since concepts are added to the hidden layer with different probability, the whole model is embedded with semantics from a concept level. Meanwhile, compared to [6], we acquire the co-occurrence relation among concepts from the vector space rather than large offline data. That is why we call it learning-based conceptualization.

### 4.2 The Second Variant (KRNNLM-2)

The second variant is based on KRNNLM-1. As shown in Fig. 3(b), we connect $cin(t)$ to both the concept and word parts of the output layer. Even if the context vector $s(t)$ loses some information about concepts, the direct connection between $cin(t)$ and the output layer can compensate for it. Thus, the output layer becomes:

$$y(t) = G(Rs(t) + Mcin(t))$$
$$cout(t) = G(Qs(t) + Ncin(t))$$

### 4.3 Backpropagation Through Time

To ensure that the neural network can learn what information to store in the hidden layer itself, RNNLM usually conducts a different backpropagation, called Backpropagation Through Time (BPTT) [1,17]. As illustrated in Fig. 3(c), our two variants both implement BPTT between the input and hidden layer. The concept vector $cin(t)$ is recorded together with $w(t)$. Thus, the is-a relation from KBs loops with historical information in the neural network.

Through Sects. 4.2 and 4.3, our conceptualization in Sect. 4.1 strengthens its influence on the whole neural network.

### 4.4 Semantic Features

Here, we define two semantic features: the vector similarity between definienda and definitions (f-sim); the log-probability sum of words in definitions (f-obj). They can be derived from any variant of RNNLM, including CRNNLM and KRNNLM. We obtain the two features from trained models and compare their effects in Sect. 5.3.

**f-sim.** As analyzed in Sect. 3, every column in matrix $U$ can be regarded as the vector of a word. Given a definiendum, we use its corresponding column $u$ in $U$ to represent it. Every definition is also represented by a vector $s$. We construct it through the traditional bag-of-words (BOW) approach, namely adding up the vectors of all the words in the sentence. After that, f-sim could be computed through any vector similarity between $u$ and $s$. We use cosine similarity in the experiments.

**f-obj.** The log-probability sum of words is exactly the objective function of RNNLM, namely Function 3 for CRNNLM and Function 8 for KRNNLM. It tells how likely a series of words is to be a sentence. Moreover, since sentences in our corpus are all positive examples, our f-obj measures how much regularities of good definitions a given sentence correlates with. The larger f-obj is, the better a definition is.

### 4.5 Complexity

The complexity of a neural network is measured by its multiplications between transition matrices and vectors from each layer to the next one. For original RNNLM, the complexity of calculation from $s(t-1)$ to $s(t)$, namely $Ws(t-1)$, is $H \times H$. Similarly, the complexity of calculation from $s(t)$ to $y(t)$, namely $Rs(t)$, is $H \times V$. Because $w(t)$ uses 1-of-$V$ coding, the complexity of from $w(t)$ to $s(t)$, namely $Uw(t)$, is $O(1)$. Hence, the complexity of original RNNLM is:

$$T(n) \propto O(H \times H + H \times V) \propto O(H \times V)$$

where $H$ is the predefined dimensionality of $s(t)$ usually within 1000. As the vocabulary size, $V$ is also the dimensionality of $y(t)$, which can reach several hundreds of thousands. Therefore, the complexity of RNNLM is dominated by the calculation from the hidden layer to the output layer. For CRNNLM:

$$T(n) \propto O(H \times P + H \times V')$$

where $P$ is the predefined class size, and also the dimensionality of $p(t)$, usually within 1000. $V'$ is the number of words sharing the same class with the next word. The value of $V'$ depends on $V$ and $P$, but often ranges from tens to tens of thousands. For both KRNNLM-1 and KRNNLM-2:

$$T(n) \propto O(H \times C + H \times V'')$$

where $C$ is the concept size, and $V''$ is the number of words sharing the same concepts with the next word. The values of $C$ and $V''$ are much smaller than $V$, although they depend on the knowledge bases and may be little larger than $P$ and $V'$. Thus, our model also reduces the complexity of original RNNLM.

## 5 Experiments and Results

### 5.1 Framework

Our experiments consist of three steps in general:

**Extract.** The experiments are evaluated on a set of open-domain definienda randomly selected from Freebase [2]. Since our work focuses on ranking, we simply send these definienda to a search engine and extract their definition candidates from top three web documents according to the patterns: ... $\langle term \rangle$ is a|is the|is one of|is known as|stands for|, called|means ...

**Label.** All the candidates are annotated by three annotators with bad, indifferent, or good[2]. Bad means definitions do not refer to any general notion or attribute of definienda. Good means definitions contain the notion and major attributes of definienda. Indifferent means definitions mention some notions or attributes, but contain trivial matters. Definienda with less than five definitions are removed. As a result, we obtain 2,936 definienda with 417 definitions (bad: 1,386, indifferent: 927, good: 623). Note that these definitions are labeled for testing, not for training LMs. As an unsupervised model, RNNLM trains over corpus without any annotation, so do CRNNLM and KRNNLM.

**Evaluate.** The baseline method is [24] which only adopts syntactic features in Ranking SVM [7]. Besides, our model is compared with CRNNLM [13], MEC [5] and Skip-gram [9] model. MEC is designed for the definition-ranking task, while the other two are not. Skip-gram is a state-of-the-art LM to train word vectors and compute word similarity. We directly use the Word2Vec toolkit[3] to implement Skip-gram, and extend the RNNLM toolkit[4] to implement KRNNLM-1 and KRNNLM-2. We also add our two variants to the baseline method as a feature respectively. Table 1 explains our evaluation metrics.

**Table 1.** Evaluation metrics for definition ranking

| Metrics | Explanation |
|---|---|
| Error rate | The ratio of mistaken ranked definitions. A definition is regarded as mistaken ranked when there is another definition with better label ranked behind it |
| Recall | The average ratio of good definitions in top-K candidates. K varies with each term, as it is the real number of good definitions |
| P@N | The ratio of definienda whose top-N candidates contain good definitions |

### 5.2 Training Language Models

The corpus for training LMs (namely CRNNLM, Skip-gram, and our model) is composed of good definitions. In *Wikipedia*, the first sentence in every document is often a concise and accurate definition. We collect these sentences from Wikipedia Dumps[5] and filter out those obeying the patterns in Sect. 5.1. Finally, we construct a corpus having 1,343,249 good definitions (11M words, 237K vocabulary, words appearing less than 3 times is removed from vocabulary).

Any knowledge base providing the is-a relation among words can be adopted in our model. We use an open-domain KB, called Probase[6] [22], and aggregate

[2] If there is a contradiction among annotators, they will be asked to re-annotate the definition. If different opinions still exist, another two annotators will take part, and we will adopt the label given by most annotators.

[3] https://code.google.com/archive/p/word2vec/.

[4] http://www.rnnlm.org.

[5] https://dumps.wikimedia.org/.

[6] Probase data is available at http://probase.msra.cn/dataset.aspx.

all the concepts into 500 clusters. Besides, the size of the hidden layer in the four LMs is all set to 200. The BPTT step for CRNNLM and our model is 4.

### 5.3 Results

First, we examine whether f-sim and f-obj are capable to evaluate definitions. For the testing definitions in Sect. 5.1, we rank them according to their two features from trained CRNNLM and Skip-gram respectively. As a state-of-the-art model to train word vectors, Skip-gram has a similar matrix to matrix $U$ of RNNLM. It can also provide f-sim for each definition. CRNNLM(f-sim), Skip-gram(f-sim) and CRNNLM(f-obj) in Table 2 show the results.

It is obvious that f-obj is more effective than f-sim, no matter f-sim from CRNNLM or Skip-gram. f-sim indeed measures the relevance between definienda and definitions, but does not reflect word sequence. Even if Skip-gram often performs best on finding related words or documents [9,14,15], it cannot be applied to the definition-ranking task directly. f-obj is derived from the whole neural network and thus takes advantage of the characteristic of RNNLM. It picks out definitions which not only contain relevant words but organize them as how positive examples in the training corpus do. The results confirm that definition ranking is not a task only requiring to find semantically related sentences. Therefore, for our two variants, we merely use them to calculate f-obj.

**Table 2.** Definition ranking results(%)

| | Error rate | Recall | P@1 | P@3 |
|---|---|---|---|---|
| SVM | 35.7 | 48.3 | 49.9 | 88.2 |
| CRNNLM(f-sim) | 55.1 | 31.5 | 30.4 | 75.0 |
| Skip-gram(f-sim) | 50.4 | 41.6 | 41.4 | 79.6 |
| CRNNLM(f-obj) | 33.8 | 53.3 | 51.8 | 89.8 |
| MEC | 32.7 | 58.1 | 54.7 | 92.2 |
| KRNNLM-1(f-obj) | 28.5 | 68.3 | 66.0 | 92.5 |
| KRNNLM-2(f-obj) | 26.2 | 70.9 | 68.1 | 92.5 |
| KRNNLM-1(f-obj)+SVM | 17.9 | **74.6** | 73.4 | 92.8 |
| KRNNLM-2(f-obj)+SVM | **16.6** | 74.2 | **75.8** | **93.1** |

We can see that only relying on one feature f-obj, CRNNLM and our two variants all perform better than the baseline method on P@1. Our model is also more accurate than MEC. After adding f-obj to the baseline method (KRNNLM-1(obj)+SVM and KRNNLM-2(obj)+SVM in Table 2), we get the best results. It indicates that no matter how many syntactic features the baseline method and MEC adopt, semantic analysis is the key to improve the precession. For MEC, the semantics complemented by context indicators are still not thorough enough. In contrast, our model provides more sufficient semantics for ranking.

In general, KRNNLM-1 and KRNNLM-2 surpass all compared models. Both CRNNLM and Skip-gram sidestep the problem of language polysemy. Since they do not distinguish different word senses, their training and prediction process is disturbed. To reduce this kind of disturbance, our model figures out word senses though determining their context-appropriate concepts with the help of the is-a relation from Probase. The coherence of these concepts meanwhile is used by our model to evaluate definitions, which exactly achieves our starting point described in the introduction.

Moreover, in KRNNLM-2, the direct connection between the input and output layer consolidates the effect of conceptualization and helps it get higher precision than KRNNLM-1. Though, KRNNLM-1 has less transition matrices, namely less parameters, than KRNNLM-2. For a neural network, less parameters mean smaller training corpus to reach a well-trained convergence. If there is not enough corpus, KRNNLM-1 can be a better choice.

Additionally, it is reasonable that the differences among all models are not that much on P@3. It is a quite loose standard compared to P@1. For terms having five definition candidates, a model can easily achieve 100% on P@3, as long as it puts one good definition in the top three. For real QA systems, P@1 is the most important metric since they usually offer only one answer to users.

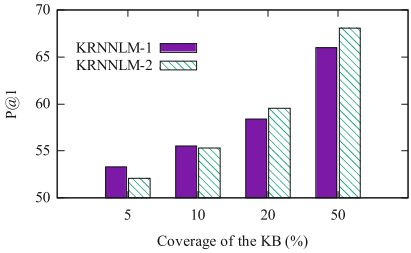



**Fig. 4.** P@1 with different coverage of the KB

In the above experiments of KRNNLM-1 and KRNNLM-2, about 50% words can find their concepts from Probase. To examine the influence of the KB, we change its coverage. Figure 4 shows that the larger the coverage is, the higher precision our model achieves. When only 5% words have concepts, P@1 on our two variants is very close to CRNNLM. Besides, we can see that the superiority of KRNNLM-2 becomes obvious with the increasing coverage.

### 5.4 Case Studies

We take "emphysema" in Sect. 1 for example to explain how our model analyzes definitions at a concept level. When we put Candidate1 into trained KRNNLM, four words can find their most appropriate concepts: ⟨disease, *disease*⟩, ⟨patients, *patient*⟩, ⟨Americans, *nation*⟩ and ⟨year, *time*⟩. These words, including "Americans" and "year", are likely to appear frequently with "emphysema" in corpus.

But concept *nation* and *time* are not coherent to the concept of "emphysema" (*disease*) as highly as the concepts of Candidate2, including ⟨lung disease, *disease*⟩, ⟨smokers, *patient*⟩ and ⟨breath, *physiological process*⟩.

Even though words are close to "emphysema" in the vector space, their context-appropriate concepts may not locate near *disease*. If users search for relevant sentences or relevant titles of web pages, Candidate1 is good enough. However, according to the analysis of our starting point in the introduction, Candidate2 should rank higher. Our model achieves this from a concept level.

Table 3 gives some other examples that KRNNLM-2 makes right judgements but CRNNLM does not. Inside one model, higher scores mean better definitions. Scores from different models are not comparable.

For the first term "bandy", Candidate1 has more relevant concepts (⟨team, *activity*⟩, ⟨sport, *activity*⟩, ⟨skaters, *athlete*⟩, ⟨sticks, *device*⟩, ⟨ball, *device*⟩, ⟨goal, *device*⟩) with it than Candidate2 (⟨sport, *activity*⟩, ⟨athletes, *athlete*⟩), even if the words themselves of Candidate1 may not appear as frequently as the words of Candidate2 in corpus. In other words, Candidate1 has more concepts close to the concepts of "bandy" (*sport*) in the vector space. Definition candidates of the other two terms in Table 3 are in a similar situation.

**Table 3.** Examples

| Term | Definition | CRNNLM (f-obj) | KRNNLM-2 (f-obj) |
|---|---|---|---|
| Bandy | 1. Bandy is a team winter sport played on ice, in which skaters use sticks to direct a ball into the opposing team's goal | 0.3182 | 0.1214 |
| | 2. Bandy is the second most popular winter sport in the world based on the number of participating athletes | 0.4938 | 0.0759 |
| Honey | 1. Honey is a sweet food made by bees foraging nectar from flowers | −0.1985 | −0.1293 |
| | 2. Honey is adulterated if it has the addition of other sugars, syrups or compounds, making it cheaper to produce, or has many fructose contents in order to stave off | −0.3266 | −0.1082 |
| Diving | 1. Diving is the sport of jumping or falling into water from a platform or springboard | 0.0304 | 0.1182 |
| | 2. Diving is a separate sport in Olympic | 0.3097 | −0.0836 |

## 6 Conclusion

In this paper, we introduce a language model to do semantic definition ranking. It combines RNNLM with learning-based conceptualization and captures semantics from both unstructured texts and the is-a relation in knowledge bases. We define two semantic features derived from neural networks and confirm that the log-probability sum can evaluate definitions effectively. We compare our model with

other approaches which only adopt syntactic features or sidestep the problem of language polysemy. Results indicate that semantic analysis is the key to improve the precision, and the incorporated knowledge base makes our model suit for the definition-ranking task better. Our approach has been applied in production.

## References


1. Boden, M.: A guide to recurrent neural networks and backpropagation. Dallas Project Sics Technical report T Sics (2001)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. ACM (2008)
3. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: EMNLP, pp. 1025–1035. Citeseer (2014)
4. Chen, Y., Zhou, M., Wang, S.: Reranking answers for definitional QA using language modeling. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pp. 1081–1088. Association for Computational Linguistics (2006)
5. Figueroa, A., Atkinson, J.: Maximum entropy context models for ranking biographical answers to open-domain definition questions. In: AAAI 2011, San Francisco, California, USA, August (2011)
6. Hua, W., Wang, Z., Wang, H., Zheng, K., Zhou, X.: Short text understanding through lexical-semantic analysis. In: International Conference on Data Engineering (ICDE) (2015)
7. Joachims, T.: Training linear SVMS in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–226. ACM (2006)
8. Kaisser, M., Scheible, S., Webber, B.L.: Experiments at the University of Edinburgh for the TREC 2006 QA track. In: TREC (2006)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
10. Mikolov, T., Deoras, A., Kombrink, S., Burget, L., Cernockỳ, J.: Empirical evaluation and combination of advanced language modeling techniques. In: INTERSPEECH, pp. 605–608, no. s1 (2011)
11. Mikolov, T., Deoras, A., Povey, D., Burget, L., Černockỳ, J.: Strategies for training large scale neural network language models. In: 2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 196–201. IEEE (2011)
12. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: INTERSPEECH 2010, Makuhari, Chiba, Japan, 26–30 September 2010, pp. 1045–1048 (2010)
13. Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J.H., Khudanpur, S.: Extensions of recurrent neural network language model. In: 2011 IEEE International Conference on ICASSP, pp. 5528–5531. IEEE (2011)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)

15. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: HLT-NAACL, vol. 13, pp. 746–751 (2013)
16. Neelakantan, A., Shankar, J., Passos, A., McCallum, A.: Efficient non-parametric estimation of multiple embeddings per word in vector space. arXiv preprint arXiv:1504.06654 (2015)
17. Rumelhart, D.E.: Leaning internal representations by back-propagating errors. Nature **323**, 318–362 (1986)
18. Song, Y., Wang, H., Wang, Z., Li, H., Chen, W.: Short text conceptualization using a probabilistic knowledgebase. In: Proceedings of the Twenty-Second IJCAI-Volume Three, pp. 2330–2336. AAAI Press (2011)
19. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of ICML-11, pp. 1017–1024 (2011)
20. Vapnik, V.: The Nature of Statistical Learning Theory. Springer Science & Business Media, New York (2013)
21. Wang, Z., Zhao, K., Wang, H., Meng, X., Wen, J.R.: Query understanding through knowledge-based conceptualization. In: Proceedings of the Twenty-Fourth IJCAI (2015)
22. Wu, W., Li, H., Wang, H., Zhu, K.Q.: Probase: a probabilistic taxonomy for text understanding. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 481–492. ACM (2012)
23. Xu, J., Licuanan, A., Weischedel, R.M.: TREC 2003 QA at BBN: answering definitional questions. In: TREC, pp. 98–106 (2003)
24. Xu, J., Cao, Y., Li, H., Zhao, M.: Ranking definitions with supervised learning methods. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, pp. 811–819. ACM (2005)