Definition Extraction with LSTM Recurrent Neural Networks

SiLiang Li, Bin Xu, and Tong Lee Chung

Tsinghua University, knowledge Engineering Group,
Beijing, China

Abstract. Definition extraction is the task to identify definitional sentences automatically from unstructured text. The task can be used in the aspects of ontology generation, relation extraction and question answering. Previous methods use handcraft features generated from the dependency structure of a sentence. During this process, only part of the dependency structure is used to extract features, thus causing information loss. We model definition extraction as a supervised sequence classification task and propose a new way to automatically generate sentence features using a Long Short-Term Memory neural network model. Our method directly learns features from raw sentences and corresponding part-of-speech sequence, which makes full use of the whole sentence. We experiment on the Wikipedia benchmark dataset and obtain 91.2% on F_1 score which outperforms the current state-of-the-art methods by 5.8%. We also show the effectiveness of our method in dealing with other languages by testing on a Chinese dataset and obtaining 85.7% on F_1 score.

Keywords: Definition extraction, LSTM Recurrent Neural Networks

1 Introduction

Definitions play an important role in creating and enriching ontology concepts from unstructured text [1]. Definitions are also put to use in Question Answering to solve ”what is” problems [2]. A big challenge is how to collect definitions from emerging mass text, since manually extracting definitions from unstructured text can be costly and slow. Therefore automatic definition extraction has drawn much attention in Natural Language Processing.

Current state-of-the-art methods treat definition extraction as a supervised classification task where a sentence is classified as definitional or not. The key point of this task is to generate features which are usually manually selected in previous approaches. For example, DefMiner system [3] specifies 12 features (8 word level features,3 sentence level features and 1 document level feature). In a weakly supervised method [4], they use 14 features to describe a sentence. Both methods use many manually selected features (e.g. dependency path distance: distance from the current word to the root of the sentence in the dependency tree) to approximately describe a sentence structure from its dependency tree. However there are two shortcomings in these features. First of all, they can only reflect part of the dependency tree which may lose hidden structure feature of sentences. Moreover they rely on the output of dependency parsing which involves error propagation. Therefore we focus on studying a method which can generate features directly from raw sentence.

In this paper, we propose a supervised learning method where features for definition extraction are automatically learned from raw sentences. Instead of relying on dependency parsing result, we regard a sentence as a word sequence and directly learn from the whole sequence. We generate sentence feature using recurrent neural network with Long Short-Term Memory (LSTM) due to the reason that LSTM has shown great ability in capturing long-term and short-term dependencies in a sequence. In our method, a sentence will first be transformed into a sequence consisted of word feature vectors. A LSTM encoder will be trained to encode the sequence into a vector representation. Finally a logistic regression classifier will be used to predict the sentence label with its feature vector. Instead of manually selecting features from dependency parsing, our sentence feature vector is directly generated by our LSTM encoder automatically, which can reduce the work of sentence feature engineering.

We evaluate the performance of our method on a Wikipedia benchmark corpus [5] and a Chinese dataset[1] originated from Baidu Baike[2]. The result shows that our method can significantly outperforms current state-of-the-art approach in F1 measure by 5.8%, and improve the recall to 92%. The main contributions of this work can be summarized as follows:

– We propose a new method utilizing LSTM to do definition extraction.
– Our method does not rely on handcraft patterns nor features manually specified from sentence dependency parsing. It can reduce the work of feature engineering for supervised learning based methods by automatically learning structure features from sentence sequence.
– Our method relies little on linguistic features and can be used for definition extraction in multiple languages. Our experiments prove its effectiveness in both English and Chinese.

The rest of the paper is organized as follows. We briefly introduce related work in section 2. Then section 3 describes the method we use to build our definitional classification model. We describe our experiment in section 4, followed by results and analysis in section 5. Finally, our paper is concluded in section 6.

2 Related Work

The task of definition extraction has attracted many attention in the past few years. Previous researches can be divided into three kinds: pattern based [8, 9], semi-supervised learning [5, 4] and supervised approaches [10, 3].

[1] You can download it from http://166.111.7.170:28090/zh.zip
[2] http://baike.baidu.com

2.1 Pattern based methods

The method focusing on the use of lexico-syntactic patterns, is first put forward by Hearst [11]. However relying on simple definitional patterns (such as is-a, is called) matching can be noisy, since short patterns may not fully represent the sentences‘ structure features. Some systems [8, 12] use patterns to extract candidates which will be further checked by grammar analysis. In order to introduce more complex patterns, a method [9] uses part-of-speech (POS) tag patterns rather than simple sequences of words.

However methods which use patterns manually selected usually suffer from low recall. The expression of definitional sentences can vary greatly in different datasets, so it is difficult to craft complete patterns which can identify all definitional sentences. Due to lack of generating ability caused by the simple strategy of pattern matching, a fully automated way using genetic programming is proposed to learn individual weights of features [13]. The genetic programming based method takes the combination of features into consideration but it ignores the importance of the order of feature occurrence.

2.2 Semi-supervised methods

Semi-supervised approaches [14, 4] show that bootstraping is efficient for definition extraction. The basic idea of these methods is to start from gold sentences and extract more from articles in other datasets such as ACL Anthology Reference Corpus (ACL ARC) [15].

Word-Class Lattices (WCLs) [5] use a directed acyclic graph to represent definitional sentences. WCLs method uses ”star patterns” to make sentence clustering and then learns the sentences’ structure in each cluster. Unlike patterns consisted of short phrases, their ”star patterns” are drawn from sentences where all infrequent words have been replaced with a wildcard (*), so they can capture long distance dependency of sentences with these star patterns. Although they form generalized sentences with WCL, their result for WCL (F_1 of 75.23%) is close to the performance of method only using star patterns (F_1 of 75.05%). It shows that even only using sequence structure of frequent words, we can also predict whether a sentence is definitional or not well .

2.3 Supervised methods

This line of research area is to regard definition extraction task as a supervised sequence classification work. A method [10] proves to work well on a corpus of Dutch Wikipedia articles. They generate three kinds of features to describe a sentence: sentence-level features (e.g. the position of the sentence in a document); word-level statistic features (e.g. bigrams, bag-of-words); word-level syntax features (e.g. determiner type). With these features, they use naive Bayes, maximumu entropy (MaxEnt) and the support vector machine (SVM) as different learners and find MaxEnt has the best performance. Sentences‘ linguistical structural features are taken into consideration in an approach proposed by [16]. A random forest classifier is used to capture more deep features of a sentence structure.

DefMiner [3] uses Conditional Random Fields (CRF) to predict the function of a word in a sentence as term, definitional part or others. One of their key contributions is the analysis of a real world corpus: W00 (a manually annotated subset of ACL ARC). They use a combination of lexical, orthography, dictionary lookup and corpus statistics (e.g. sentence position, idf) as features. Meanwhile, they also use features manually selected from dependency tree to describe sentence long distance structures.

Another approach relying on syntactic dependencies is proposed by [7]. They focus on extracting features from the dependency parsing of a sentence. Each noun in a sentence will be represented into a numeric vector according to its syntactic dependencies with other nouns in the same sentence. The vector representation will be fed to two classifiers to determine whether the noun is hyponym, hypernym or neither of two. If a sentence has both hyponym and hypernym, it will be labeled as definitional. Their work highlights the importance of learning from relations of words.

Supervised approaches face the challenge that manually specified features involve many feature engineering work. Meanwhile, structure features are chosen from part of the dependency tree, which may not be able to fully describe the whole sentence structure.

Our method focus on capturing the whole sentence structure feature so as to automatically generating sentence features for supervised classification.

3 Our method

In our method, we design our procedure based on the assumption that we can determine whether a sentence is definitional or not by certain structures. We call these sentence structures as definitional structures. Definitional structure can be a short phrase or a long discontinuous word sequence.

Previous pattern based approaches use patterns to describe definitional structures. In order to increase the generalization ability of our model, we use feature vector to reflect the usage of definitional structure and classify sentence according to its feature vector instead of simple pattern matching.

Our method considers definition extraction as a supervised classification work where a typical challenge is how to generate feature from long-term definitional structure. Previous supervised based approaches learn definitional structure using n-grams or conditional random field (CRF). These algorithms have strong ability in learning short-term dependency features but are weak in capturing long-term definitional structure. Therefore, we generate sentence feature using LSTM which is fit for both short-term and long-term structure learning.

Our method consists of the following three steps: (See Fig 1 for graphical structure.)

- **Token transformation:** each word in a sentence will be transformed into a token according to its frequency in training set. (Section 3.1)
- **Word feature generation:** each word will be represented as a word vector by capturing features of the word‘s context. (Section 3.2)
- **Sentence feature generation:** A LSTM encoder will be used to automatically transform a sentence into a vector representation by learning sentence hidden struc-

ture feature. A classifier will also be trained to predict a sentence label by its feature. (Section 3.3)

3.1 Token transformation

In definition extraction, for most of the words, we care more about its POS tag rather than the word itself. For example, We do not care the difference between ”Cat is an animal” and ”Dog is an animal”. Thus these two sentences can be transformed to one sentence ”NN is an NN” by replace some words with their POS tags.

Due to the above reason, a word is transformed into a token to maintain its most significant form for definition extraction. In this step, we pick top N frequent words from our training set. The reason we do not choose words’ tf-idf is that some words (e.g. is, a, etc.) are important to form a definitional pattern in spite of their low idf. These chosen words form a set F which can be regarded as a cluster of words. These words are potential constituent part of definitional patterns. More precisely, given a sentence S of length n, for the i-th word w_i of s, we produce token t_i by [3]:

$$t_i = \begin{cases} w_i & w_i \in F \\ POS(w_i) & w_i \notin F \end{cases}$$

3.2 Word feature generation

After transforming each word into a corresponding token, our next question is how to generate the features of these tokens. In supervised classification based methods, they use words bigrams [10] and POS tag bigrams [17] as sentence features and use information gain method to reduce the count of features. However, they make their features completely independent with each other. For example, if we only use a bigram (Num, N) as feature, then other bigrams (e.g. (Adj, N)) will never be taken into consideration.

In order to describe the similarity between tokens, we use a vector to represent a token so that we can measure their similarity by calculating their distance. The more similar two tokens are, the less influence will be caused when one changes to another. We use the context of a token (i.e. tokens near the token) to describe the token as the context can reflect the usage of the token.

Word2vec [18] is a useful method to generate a word vector representation by maximum the possibility of the occurrence of a word according to its context. The method works on the assumption that similar words have similar context which is also fit for our situation. Thus we use word2vec to encode tokens in our training set.

3.3 Sentence feature generation

Sentence features can be divided into two kinds. One is to generate features only within a single sentence (e.g. bag-of-words, n-grams [10], the presence of determiners in the defines, etc.). The other kind is using features beyond a sentence such as the position of a sentence in the document [3].

[3] Our pos tagger tool is from http://nlp.stanford.edu/software/tagger.shtml

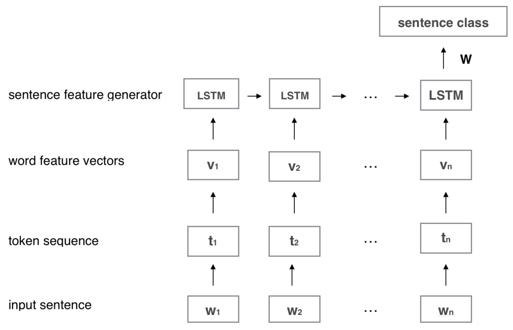

Fig. 1. Structure of our model.

However features beyond sentences are often corpus dependent. Take sentence position feature for example, sentences appearing at the beginning of documents are likely to be definitional in wikipedia documents, while things may not be the same with other corpus.

Due to this reason, we focus on extracting features from a sentence only by learning its definitional structure. Previous methods generate features from sentence dependency parsing, which can only reflect part of structure feature. In our method, we will treat a sentence as a sequence of tokens and use LSTM to capture the whole structure of the sequence.

Feature generator Our feature generator consists of a single layer of Long short-term memory (LSTM) unit [19] which is designed to overcome the gradients vanishing problem in recurrent neural network (RNN) [20]. LSTM uses special gates to control the flow of data in repeating module, which can help them be capable of learning long-term dependencies. LSTM uses three gates to implement the repeating module: a forget gate **f** to control of the flow of cell state, an input gate **i** to control the inputing data and an output gate **o** to control the output of the module. Each LSTM unit maintains a memory cell **c**.

Given a sentence s of length n, we get a sequence of word feature vectors **x** in the word feature generation step. For the t-th token in **x**, the output **h** of LSTM is given by

$$\mathbf{h}_t = o_t tanh(\mathbf{c}_t)$$

where the output gate is given by

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_t)$$

The cell state is calculated by

$$\mathbf{c}_t = \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t\tilde{\mathbf{c}}_t$$

where $\tilde{\mathbf{c}}_t$ denotes a new memory content computed by

$$\tilde{\mathbf{c}}_t = tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1})$$

Meanwhile, the forget and input gates are computed by

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1})$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1})$$

We use the n-th output $\mathbf{h}_n$ of LSTM as the feature vector of the sentence.

Joint learning of feature generator and classifier Given a sentence feature vector **x**, we use a logistic regression classifier $\sigma(\mathbf{w}^T\mathbf{x})$ to predict the sentence label. The final result is influenced by both parameter **w** of classifier and the sentence feature vector **x**. Therefore we train our classifier and sentence feature generator together.

In the word feature generation step, a sentence s of length n is generalized to a sequence of word feature vectors s'. Let T' denote the cluster of s'. In the sentence feature generation step, we use $\mathbf{h}(s')$ to represent the n-th step output of LSTM. We train our model by mining the cost function:

$$L = \sum_{s' \in T'} crossentropy(y_{s'}, \sigma(\mathbf{w}^T\mathbf{h}(s'))$$

where $y_{s'}$ stands for the label of s', 1 for definitional and 0 for the other. **w** is a parameter vector. Here crossentropy is given by

$$crossentropy(a, b) = -(a \log(b) + (1 - a) \log(1 - b))$$

We use gradient descent to optimize **w** and the parameters of our LSTM encoder together. An open source implementation, Theano [21] is used in our work[4]. We calculate gradients for all the parameters (i.e. **w** and parameters of LSTM) and update them using RMSprop optimizer.

Here we want to emphasize that as pointed by [17], highly imbalanced datasets can greatly influenced supervised classification based methods. As the training procedure of our method uses gradient descent, we overcome this problem by setting different learning rate to positive and negative samples according to their count ratio.

4 Experiments

We prepare two datasets for the evaluation of our method. One is an English dataset [5] consisted of 1,908 definitional sentences and 2,711 non-definitional sentences manually annotated from Wikipedia. The other one is a Chinese dataset containing 2,161 definitional sentences and 2,161 non-definitional sentences manually annotated from Baidu Baike[5]. In this part, we will describe detail experiment settings and the way we evaluate our result.

[4] http://deeplearning.net/software/theano/

[5] You can download it from http://166.111.7.170:28090/zh.zip

4.1 Experiment settings

In the word feature generation step, we use Top-1,000 frequent words set for English dataset and Top-500 frequent words set for Chinese dataset. Every token will be encoded into a 50-dimension vector. The output dimension of the LSTM encoder used in our sentence feature generation step is 50. The parameters of LSTM encoder are initialized with the uniform distribution with the scale introduced by [22].

For the evaluation of our definition extraction on English dataset, we compare with the following implements.

- **Star patterns**: A pattern based method where a sentence is classified as a definition if it matches any of the star patterns [5].
- **Bigrams**: A pattern based method which uses bigram classifier for soft pattern matching [2].
- **WCL**: A semi-supervised method which uses WCL-3 model to learn lattices separately for each sentence field (i.e. DEFINIENDUM, DEFINITOR and DEFINIENS) [5].
- **DefMiner**:a supervised method which uses long distance features [3].
- **SVM**: a supervised method which uses SVM classifier and syntactic dependencies. [7].
- **DR system**: a supervised method which only uses syntactic features derived from dependency relations [6].
- **our method+LSTM**: Our method using LSTM as sentence feature generator.
- **our method+RNN**: Our method using RNN as sentence feature generator.
  Experiment on Chinese dataset is used as a comparison with the experiment on English dataset so as to show our method‘s ability in dealing with different languages.

Our training step can be regarded as optimizing a logistic regression classifier and our sentence encoding layer at the same time. Therefore we need to make sure that whether our LSTM encoder plays an important role in our method or the performance is simply influenced by optimizing the logistic regression classifier. Sentences in our dataset have rather long length thus RNN cannot generate a good sentence representation due to gradients vanishing problem. Therefore We use RNN as a control experiment to check whether our LSTM implementation can learn a good sentence representation vector.

4.2 Measures

- **Precision** - The number of definitional sentences correctly labeled by our model divided by the number of sentences marked by our model as definitional.
- **Recall** - The number of definitional sentences correctly labeled by our model divided by the number of definitional sentences in test dataset.
- **F_1-measure** - Calculated by $\frac{2PR}{P+R}$ with precision(P) and recall(R).

We calculate the above three measures to judge the performance of our method. Experiments are performed with 10-fold cross validation. Dataset is separated into 10 parts and we use one part as testing set and nine parts as training set in each fold.

5 Results and Analysis

In this section, we will evaluate the overall performance of our method in comparison with other systems. Additionally, we will show how frequent set influences the performance and explain the reason.

5.1 Overall performance

Table 1. Performance on the English dataset

Algorithm	**P(%)**	**R(%)**	**F_1(%)**
our method+RNN	55.0	41.9	47.6
our method+LSTM	90.4	**92.0**	**91.2**
Star patterns	86.7	66.1	75.1
Bigrams	66.7	82.7	73.8
WCL	98.8	60.7	75.2
DR system	85.9	85.3	85.4
DefMiner	92.0	79.0	85.0
SVM	88.0	76.0	81.6

In table 1, we display the results of different definition extraction systems on the English dataset. Results of other systems are obtained from aforementioned literatures [2, 3, 5, 7, 6].

Compared with RNN, LSTM completely outperforms in both precession and recall, which indicates that our sentence feature generation step does play an important role and LSTM can preferably encode a sentence according to its structure.

Compared with the other previous systems, the performance of LSTM is satisfying. In our LSTM implementation, we make near 6% progress in the F_1 measure, which proves that our method have a good generating ability in extracting sentence features for definition extraction. Although precision of our method is lower than WCL and DefMiner, we make a significant improvement in recall performance. We contribute the improvement to the following reasons:

- Compared with pattern based method, our method does not use any manual specific star patterns to preprocess sentences. Every possible pattern will be learned in our LSTM encoder.
- In our method, We use dense feature vectors to encode tokens (i.e. words or POS tags) in sentences. Tokens are not treated independently but can be calculated similarity according to their distance. In other words, we smooth the impact caused when a token of a sentence change to another, which can improve the generating ability of our method.
- We do not manually pick sentence structure features but let our LSTM encoder automatically learn how to generate features. Manually specific dependency features in previous work use only a small part of sentence dependency parsing tree, which

may lose hidden information. However our LSTM encoder can directly learn from sentence sequences which ensures the full use of sentence structures.

Our experiment on Chinese dataset reaches a result of **86.7%** in precision, **84.7%** in recall and **85.7%** in F1 score. Compared with English dataset, sentences in Chinese dataset have more complicated definitional structures. Considering the possible error caused in word segmentation and the structure complexity, we think it is a satisfying result, which shows that our method‘s effectiveness in dealing with different languages.

5.2 Result with different frequent words sets

Table 2. Performance with different Top-N frequent words sets on English and Chinese datasets

Top-N	**English F_1(%)**	**English Frequency count(%)**	**Chinese F_1(%)**	**Chinese Frequency count(%)**
Top-0	76.0	0	76.7	0
Top-50	87.9	44	84.0	30
Top-100	89.6	48	85.1	35
Top-500	90.9	61	85.7	52
Top-1,000	91.2	68	84.3	60
Top-8,000	90.9	90	81.4	88

In order to find out how our word feature generation step affect the performance of our system, we perform another experiment. In table 2, we present the performance with different top-N frequent words sets. Here the frequency count refers to the count of frequency of all the words used in different datasets.

The performance with Top-0 frequent words set is the worst as expected due to the reason that it replace all the words with their POS tags. The treatment will cause information loss since original forms of some words (e.g. is, a, etc.) are more important than their POS tags in definition extraction.

Until Top-500, the F_1 measure rises quickly when we enlarge our frequent words set with more words which are often essential parts of star patterns in previous approaches. The result shows that frequent words are critical in determining a sentence is definitional or not.

Performance on English dataset becomes rather stable when we choose more words as frequent words. Even with Top-8,000 frequent words set where POS tags are hardly used, our method still obtains a satisfying performance. Compared with Chinese, POS tag feature does not make significant effect on definitional extraction in English. We believe the main reason lies on the language itself. English definitional sentences have rather common structures which are usually consisted of frequent words. In our sentence feature generation step, our LSTM encoder gradually learns to reduce the impact of infrequent words since they are rarely involved in definitional structures. Therefore it does not matter whether we use words‘ original forms or their POS tags.

Result is quite different in the Chinese dataset. F_1 score falls when the size of frequent words set becomes larger than 500, which indicates that POS tagging replacing step becomes important when dealing with Chinese. The reason lies on the fact that Chinese sentences do not have common frequent words based patterns which can be used to classify definitional sentences. POS tags of infrequent words will participate in classification so replacing these words with their POS tags can help to improve the performance of our method.

In general, considering both English and Chinese task, we think Top-500 frequent words set is the best set for our method. Although with this words set, our method cannot reach its best in English dataset, its F_1 score (90.9 %) is actually very close to the highest score (91.2%).

6 Conclusion

A method for the task of definition extraction based on LSTM Recurrent Neural Network has been described in this paper. In order to directly learn from the whole raw sentence, we propose a new way to generate sentence representation.

From the initial idea that LSTM has the ability of learning sequence structures from sentences, we use a LSTM generator to capture definitional structures in a sentence. Next, the dataset of our experiment has been presented to the reader, followed by our detail experiment settings and our results.

We are encouraged by the performance of LSTM implementation, which proves that our method can learn certain structures from sentences well and is competent for definition extraction task. Our method can reduce the work of feature engineering for supervised learning based definition extraction by automatically learning structure features from sentence sequence. Besides our method is also proved to be effective in multiple languages.

Acknowledgments

This work is supported by China National High-Tech Project (863) under grant No. 2015AA015401, and Tsinghua University Initiative Scientific Research Program (No. 20131089190). Beijing Key Lab of Networked Multimedia also supports our research work.

References

1. Aldo Gangemi, Roberto Navigli, and Paola Velardi.: The ontowordnet project: extension and axiomatization of conceptual relations in wordnet. In *On the move to meaningful internet systems 2003: CoopIS, DOA, and ODBASE*, pages 820–838. Springer (2003)
2. Hang Cui, Min-Yen Kan, and Tat-Seng Chua.: Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems (TOIS)*, 25(2):8 (2007)
3. Yiping Jin, Min Yen Kan, Jun Ping Ng, and Xiangnan He.: Mining scientific terms and their definitions: A study of the acl anthology. *Newdesign.aclweb.org* (2013)
4. Luis Espinosa-Anke, Francesco Ronzano, and Horacio Saggion.: Weakly supervised definition extraction. *RECENT ADVANCES IN*, page 176 (2015)

5. Roberto Navigli and Paola Velardi.: Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327. Association for Computational Linguistics (2010)
6. Luis Espinosa-Anke and Horacio Saggion.: Applying dependency relations to definition extraction.: In *Natural Language Processing and Information Systems*, pages 63–74. Springer (2014)
7. Guido Boella and Luigi Di Caro.: Extracting definitions and hypernym relations relying on syntactic dependencies and support vector machines. In *ACL (2)*, pages 532–537 (2013)
8. Judith L Klavans and Smaranda Muresan.: Evaluation of the definder system for fully automatic glossary construction. In *Proceedings of the AMIA Symposium*, page 324. American Medical Informatics Association (2001)
9. E. N. Westerhout and E. N. Westerhout.: Extraction of dutch definitory contexts for elearning purposes. *Lot Occasional*, 7 (2007)
10. Ismail Fahmi and Gosse Bouma.: Learning to identify definitions using syntactic features. In *Proceedings of the EACL 2006 workshop on learning structured information in natural language applications*, pages 64–71. Citeseer (2006)
11. Marti A Hearst.: Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics (1992)
12. Smaranda Muresan and Judith Klavans.: A method for automatically building and evaluating dictionary resources. In *Proceedings of the Language Resources and Evaluation Conference (LREC* (2002)
13. Claudia Borg, Mike Rosner, and Gordon Pace.: Evolutionary algorithms for definition extraction. In *Proceedings of the 1st Workshop on Definition Extraction*, pages 26–32. Association for Computational Linguistics (2009)
14. Melanie Reiplinger, Ulrich Schäfer, and Magdalena Wolska.: Extracting glossary sentences from scholarly articles: A comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 55–65. Association for Computational Linguistics (2012)
15. Steven Bird, Robert Dale, Bonnie J Dorr, Bryan R Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R Radev, and Yee Fan Tan.: The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *LREC* (2008)
16. Eline Westerhout.: Definition extraction using linguistic and structural features. In *Proceedings of the 1st Workshop on Definition Extraction*, pages 61–67 (2009)
17. Rosa Del Gaudio, Gustavo Batista, and António Branco.: Coping with highly imbalanced datasets: A case study with definition extraction in a multilingual setting. *Natural Language Engineering*, 20(03):327–359 (2014)
18. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.: Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013)
19. Sepp Hochreiter and Jürgen Schmidhuber.: Long short-term memory. *Neural computation*, 9(8):1735–1780 (1997)
20. Yoshua Bengio, Patrice Simard, and Paolo Frasconi.: Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166 (1994)
21. Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio.: Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590* (2012)
22. Xavier Glorot and Yoshua Bengio.: Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256 (2010)