Entity Disambiguation Based on LSTM-CNN and Sparse Features

Yuxuan CHENG† and Mizuho IWAIHARA‡

†早稲田大学大学院情報生産システム研究科

〒808-0135 福岡県北九州市若松区ひびきの２－７

E-mail: †lingualcyx@toki.waseda.jp, ‡iwaihara@waseda.jp

Abstract Entity linking is the task to link a named entity retrieved in the web text with a corresponding entity in a knowledge base such as Wikipedia, which can be uti-lized in various NLP tasks, such as text understanding, question answer and in-formation retrieval. The main challenge of entity linking is name ambiguity and context dependencies. In this paper, we propose a long short-term memory recur-rent neural network (LSTM-CNN) model for entity disambiguation, which inte-grates LSTM-CNN and sparse features such as link-popularity and PageRank values extracted from the whole Wikipedia. The CNN part extracts high-level phrase representations from words, which can reduce noise caused by noisy mentions. The LSTM part can obtain sentence representation from phrase repre-sentations. We train our model with the dump of Wikipedia, and demonstrate pre-cision, recall and F1 scores of our model on several datasets, showing significant performance improvement over conventional LSTM-CNN models.

Keyword Entity linking, Entity disambiguation, LSTM, CNN

1. Introduction

1.1. Entity disambiguation

With the development of Internet and the rapid increase of Internet information resource, entities linking and disambiguation have become a fundamental task for information retrieval. An entity is an objective thing in the real world, which includes not only specific things such as names, place names, organization names, but also abstract things such as concepts and relationships. However, a named entity may have multiple names, and names can represent several different entities. Entity dis-ambiguation refers to the process of linking entities in the document to entries in a specific knowledge base, such as Wikipedia.

With millions of active online users, hundreds of millions of new web postings are published every day, which contain billions of mentions of entities. Connecting web data with a knowledge base is beneficial to annotate a large number of raw and often noisy texts on the web, which helps readers to deepen their understanding of the entity, thus helping people or computers to better understand and deal with the texts.

The main challenge of the entity disambiguation lies in two aspects, namely multiple names of entities and word polysemy. Multi-name refers to an entity that has multiple alias. Word polysemy means that a mention can refer to more than one entity. For example, as we can see in Figure 1, given a sentence: "*Eclipse is written mostly in Java and it may also be used to develop applications in other programming languages via plug-ins, including C++, C#, JavaScript, Python, Ruby*." How can a computer distinguish what the *Eclipse*, *Python* and *Ruby* means? How does it know the *Ruby* is a programming language or a kind of jewel?

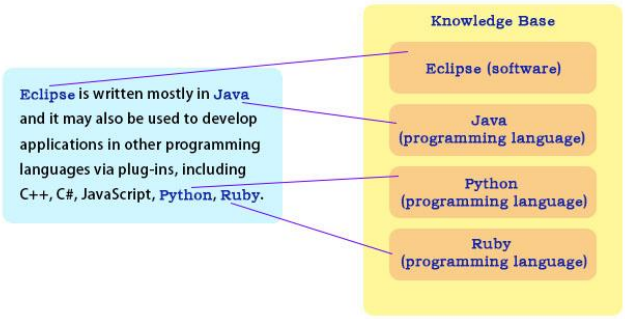

Figure 1 Illustration of entity disambiguation task

The aim of entity disambiguation is to discover mentions of entities within a text and to link them to the most suitable entries in a reference knowledge base.

1.2. RNN and LSTM

The Recurrent Neural Network (RNN) [2] sequence model learns mapping of an input sequence to an output sequence with intermediate hidden states estimated by a continuous vector. It analyzes a text word-by-word and stores the semantics of all the previous texts in a fixed-sized hidden layer. The advantage of RNN is the ability to better capture the contextual information, which could be beneficial to capture semantics of long texts. The basic RNN is difficult to train because of the problem of the so-called vanishing gradient. The gradient signals used to train the network are often close to Zero (“vanishing”) or divergent (“explosion”) because it propagated back

through time steps during learning [18]. LSTM is widely used in NLP since it can deal with arbitrary-length sequences of input. It introduces a memory cell and solves the problem of exploding and vanishing gradients in RNN. The LSTM model and its different variants have achieved impressive performance in different sequence learning problems in speech, image, music and text analysis, where it is useful in capturing long-range dependencies in sequences [19].

1.3. CNN

Convolutional Neural Network (CNN) [8] have been accredited for major breakthroughs in image classification and are the core of most of recent computer vision systems. But we can also use CNNs in NLP tasks. Instead of image pixels, the input to most of NLP tasks are sentences or documents represented as a matrix by word embeddings like word2vec. The convolutional layers apply one-dimensional filters across each row of features in the sentence matrix to obtain high-level representations [1]. With a long sentence, computation will be slow and expensive. Convolutional filters can learn good representations automatically, without representing the whole vocabulary, which can speed up the training process.

1.4. Task

Given a text which contains several mentions of entities which are identified in advance and a knowledge base containing a set of entities, the goal of entity linking is to map each textual entity mention to its corresponding entity in the knowledge base [20]. The process can be divided into two steps:

- **Candidate generation**: The entity linking system tries to find all possible entities which a mention may refer to in the entity set of the knowledge base.
- **Candidate ranking**: For candidate entities of a mention, the entity linking system needs to incorporate different kinds of evidence to rank the candidate entities and find the entity which is mentioned. This is the most important part of entity linking.

1.5. Our contribution

In this paper, we propose a deep learning model which combines LSTM-CNN and sparse features such as PageRank and link-popularity extracted from the whole Wikipedia dump.

The rest of the paper is organized as follows. Section 2 presents a brief review of related work. Section 3 discusses the architecture of our framework of integrating LSTM-CNN model and sparse features. Section 4 presents experimental results on performance comparison. Section 5 concludes this paper.

2. Related Work

Existing entity disambiguation methods are mainly based on machine learning methods, graph-based methods, probabilistic methods, unsupervised learning method, and integration method. Ganea et al. proposed a probabilistic bag-of-hyperlinks model which improves bag-of-words model with arising probabilities of entities in hyperlinks in context, but they considered only pairwise potentials, which will be weak when a text has many entities. Han et al. [5] proposed a graph-based method which can model and exploit the global interdependence between different entity linking decisions, which only considered relation between entities, not the meaning of context. Shen et al. [19] resented a novel framework which leverages the rich semantic information derived from Wikipedia and the taxonomy of the knowledge base to deal with the entity linking task.

Deep learning models have recently made remarkable progress in various NLP tasks. For example, word embedding [12], question answering [21], sentiment analysis [22] and so on. Lai et al. [9] proposed a Recurrent Convolutional Neural Network (RCNN) and applied it to the task of text classification, which shows a good result for different types of datasets in English and Chinese. Ouyang, Liubo, et al. [15] presented a hybrid model of B-LSTM and showed reasonable results on labeled People's Daily in January 1998 dataset in Chinese.

From the above description, we can see that the traditional way for entity linking is mainly using probabilistic or graph model, where the meaning of context is not eagerly captured. On the other hand, with the pre-trained word embeddings, neural networks demonstrate their great performance in many NLP tasks and show great results in context comprehension. In the following sections, we describe how we apply an LSTM-CNN to entity linking task, and augment with sparse features.

3. Proposed Method

In this section, we first describe our input formation and introduce the full architecture of our LSTM-CNN. Then we describe each layer in detail. Finally, we introduce a techniques to avoid overfitting in our model.

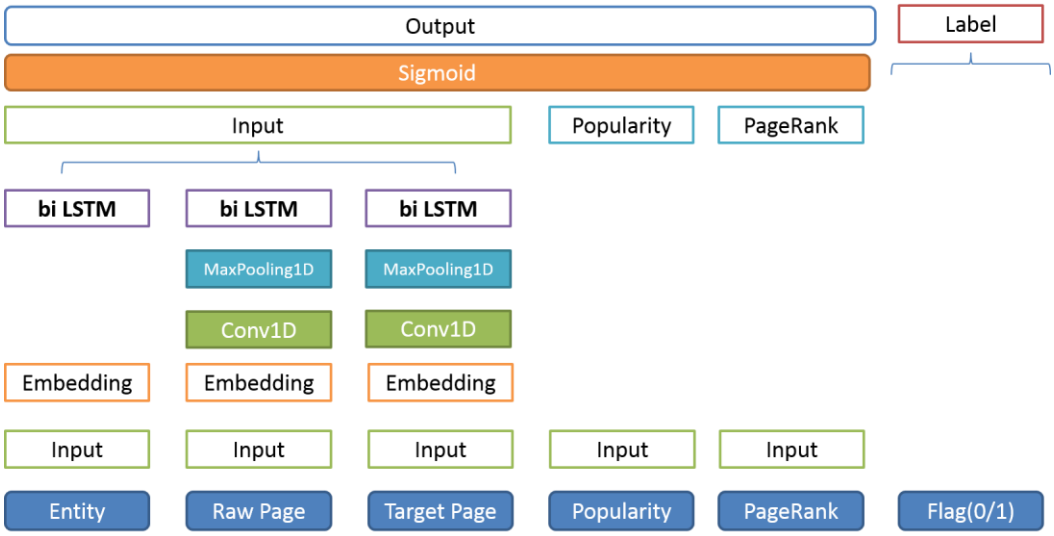

Figure 2 full architecture

An overall illustration of our architecture is shown in Figure 2. The full model is composed of several sub-models, which are trained on a sub-dataset extracted from Wikipedia-dump. We divide the dataset into three parts: mention part, original text part, target text part. As the original text and target text are long, we add the LSTM-CNN layer in our text sub-model. Figure 3 shows the function of the three kinds of neural network layers. CNN extracts high-level phrase representations from words, which can reduce noise caused by noisy mentions. LSTM can obtain sentence repre-sentation from phrase representations. The last sigmoid layer makes the entity dis-ambiguation task into a binary classification task.

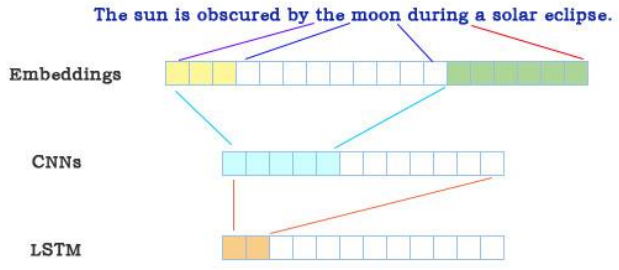

Figure 3 The structure of text sub-model

3.1. Pre-training Word Embeddings

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. Distributed representations of words in a vector space help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. Traditional representations, such as one-hot representation, will lead to the curse of dimensionality. Recent research shows that neural networks like word2vec or GloVe can converge to a better local minima with suitable unsupervised pre-training vectors [13].

In this paper, we use the publicly available GloVe vectors that were provided by Pennington et al.[17]. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

3.2. CNN

As we introduced early in this paper, in order to generate a feature map, the convolution layer will take a filter (an array of weights) and slide it over the sentence matrix. For this task, we use a 1D convolutional layer. The size of convolutional filters and the number of kernel cores are important parameters.

3.2.1. Activation units

The purpose of the activation unit is to introduce non-linear function to a system that just has been computing linear operations in the convolutional neural layers, to learn non-linear decision boundaries. The choice of the activation unit has been shown to affect the convergence rate and the quality of the obtained solution. The rectified linear unit (ReLu) has become very popular in the last few years. It was found to work far better because the network is able to train much faster without making a significant difference to the accuracy. The purpose of the activation unit is to introduce non-linear function to a system that just has been computing linear operations in the convolutional neural layers, to learn non-linear decision boundaries. The

choice of the activation unit has been shown to affect the convergence rate and the quality of the obtained solution. The rectified linear unit (ReLu) has become very popular in the last few years. It was found to work far better because the network is able to train much faster without making a significant difference to the accuracy [14]. In this paper, we will also use ReLu as our activation unit for CNN. It computes the function $f(x) = max(0, x)$ to ensure that feature maps are always positive.

3.2.2. Pooling

The output from the convolutional layer is then passed to the pooling layer, whose function is to progressively reduce the spatial size of the representation, and to reduce the number of parameters and amount of computation in the network, hence controlling overfitting. The max pooling we are using is the most commonly used, applying a max filter to non-overlapping sub-regions of the initial representation. For each of the regions represented by the filter, max pooling will take the maximum of that region and create a new output matrix where each element is the maximum of a region in the original input.

3.3. LSTM

As we mentioned earlier, RNN suffer from the problem of vanishing gradients. It can be resolved by using LSTM blocks instead of sigmoid cells in the hidden layers. The gates of LSTM blocks are a way to optionally let information through, which can choose to retain their memory over arbitrary periods of time or forget if necessary, which is shown in Figure 4.

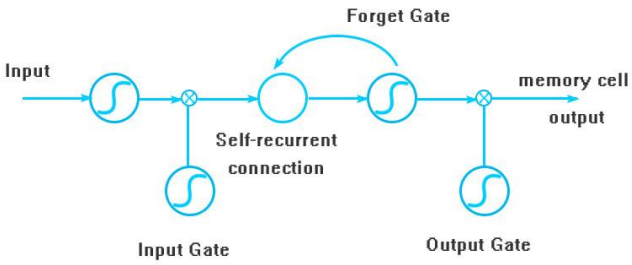

Figure 4 Structure of LSTM

We use the B-LSTM network [3] to acquire bidirectional word sequence for making predictions. The bidirectional LSTM can present each training sequence forwards and backwards to two separate recurrent nets, both of which are connected to the same output layer. This means that for every point in a given sequence, the network has complete, sequential information about all points before and after it. There is no information flow between the forward and backward hidden layers, which ensures that the expansion graph is non cyclic. The structure of BLSTM is shown in Figure 5. We add 0.2 dropout to BLSTM layer to prevent it from overfitting.

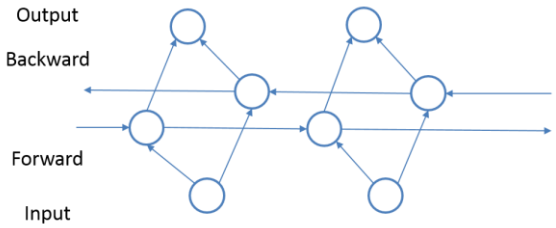

Figure 5 Structure of BLSTM

3.4. Sparse Feature

3.4.1. PageRank

PageRank is proposed by Lawrence Page and Sergey Brin in 1999, which is a method for ranking nodes in a graph according to their relative structural importance [16]. The main idea of PageRank is that whenever a link from V_i to V_j exists in a network graph, a vote from node *i* to node *j* is produced, and the rank of node j increases, which makes pages important when it is linked by other important pages. The PageRank score of a page *A* is given as follows:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)})$$

Here, pages $T_1 \ldots T_n$ point to page *A*. *C(A)* is defined as the number of links going out from page *A*. In entity linking, important entities are considered to be more frequently mentioned. Thus we introduce PageRank scores computed from links on entity pages in Wikipedia as a sparse feature.

3.4.2. Link-popularity

Entity popularity found to be an effective feature in entity linking, which is the popularity of a candidate entity with regard to its entity mention, defined as the prior probability of the appearance of a candidate entity given the entity mention. Each mention has a number of candidates, while each candidate has different popularity. For example, for the anchor *Walt Disney* we might consider as candidates both the person *Walt Disney* and the film *Walt Disney*. While both are valid candidates worthy of consideration, the person *Walt Disney* appears more frequently, so we can say it is more likely to be mentioned in documents.

3.5. Optimization Algorithm

In this paper, we use Adam [9] which is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights in iteration on training data. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name Adam is derived from adaptive moment

estimation.

3.5.1. Early Stopping

When training neural networks, numerous decisions need to be made regarding the settings (hyper-parameters) used, in order to obtain good performance. But we often do not know how many full passes of the data set (epochs) should be used. If we use too many epochs, the model might become overfit. With early stopping we need not to set this value manually. Early stopping rules have been employed in various machine learning methods, with varying amounts of theoretical foundation. In this paper we set it to supervise the accuracy of the validation set.

3.5.2. Sigmoid

The sigmoid function is used for the two-class logistic regression, defined by the formula:

$$S(x) = \frac{e^x}{e^x + 1}$$

The sigmoid classifier's output is a value between 0 and 1, having a characteristic "S"-shaped curve. In the entity linking task, we just need to classify our result into true or false. So we just need to use the sigmoid function as our classifier.

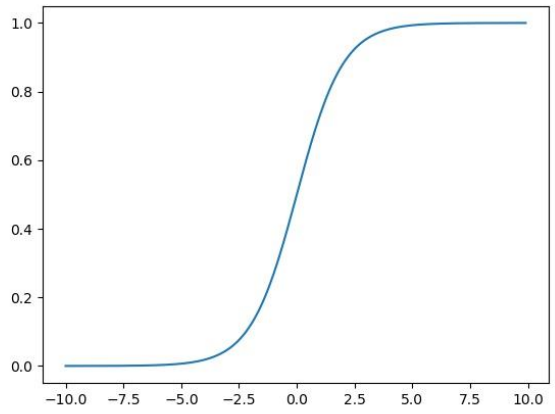

Figure 6 "S"-shaped curve

4. Experiment and Evaluation

4.1. Dataset

We train our model on a sub-dataset of Wikipedia dump exported on Nov. 3rd, 2017. We extracted all inner-links from the dump. Then, we select 6000 pairs of inner-links randomly as our positive examples. Then we select top n (n = 5, 13, 20, 50) ambiguous pages as negative examples by searching the mention in Wikipedia's page titles with Apache Lucene, which returns the most similar entity by its algorithm, consisting of a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model. When the original text part and target text part are too long, we extract the first 200 words of long text as our model's input. We train our model in a 128 batch size and 30 epochs with early stopping. We evaluate our model on the following datasets:

- **AQUAINT** This dataset, introduced by Milne and Witten [11], contains documents from a news corpus from the Xinhua News Service, the New York Times and the Associated Press. It has 50 articles with over 700 mentions. Each article has 14 mentions in average.
- **KORE50** This dataset is a subset of the larger AIDA dataset, which is introduced by Hoffart et al. [7]. It has 50 hand-crafted, difficult sentences containing a large number of very ambiguous mentions.

4.2. Experiment Settings

For the AQUAINT dataset, we also use Lucene to obtain top n (n = 5, 13, 20, 50) candidates to do our experiment. When the correct entities are not retrieved, we add them manually. However, we find that the top-10 candidates can already retrieve 84% correct entities from Wikipedia. So we choose our 1:10 result for comparing with other models.

We use the Gerbil testing platform [24] version 1.2.5 with the Disambiguate to Knowledge Base (D2KB) task, which is focused on the input an annotator obtains and the output that is expected. We run additional experiments that allow us to com-pare against more recent approaches, such as:

- **The AGDISTIS** is a novel NED approach and framework, achieved by combining the HITS algorithm with label expansion and string similarity measures [23].
- **The AIDA** is a performant graph-based model, placing emphasis on state-of-the-art ranking of candidate entity sets [6].
- **DBpedia Spotlight** is one of the first semantic approaches [12], published in 2011. This framework combines NER and NED approaches, relying on DBpedia.

We evaluate the quality of entity linking systems by measuring standard metrics precision, recall and F1 scores.

$$precision = \frac{|relevant\ entity| \cap |retrieved\ entity|}{|retrieved\ entiy|}$$

$$recall = \frac{|relevant\ entity| \cap |retrieved\ entity|}{|relevant\ entity|}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

To evaluate micro-averaged F1@MI, we sampled four

training datasets from the Wikipedia dump. The proportion of positive samples to negative samples is changed as 1:5, 1:10, 1:20, 1:50. We also test on the AQUAINT dataset with a mention to 5, 10, 20, and 50 candidates. The average result of our experiment is our F1@MI.

First, we compare (1) our model LSTM-CNN combined with sparse features (we call LCS), (2) the LSTM-CNN model which contains only the convolutional and BLSTM layers (LSTM-CNN), and (3) the model which contains only BLSTM (LSTM). The AQUAINT dataset is used here. The results are shown in Table 1, in which our proposing model LCS is showing the highest F1-score, 0.28-0.35 points higher than LSTM-CNN and LSTM. We note that LSTM, the LSTM-CNN model that only contains BLSTM, is quite time-consuming than the other models.

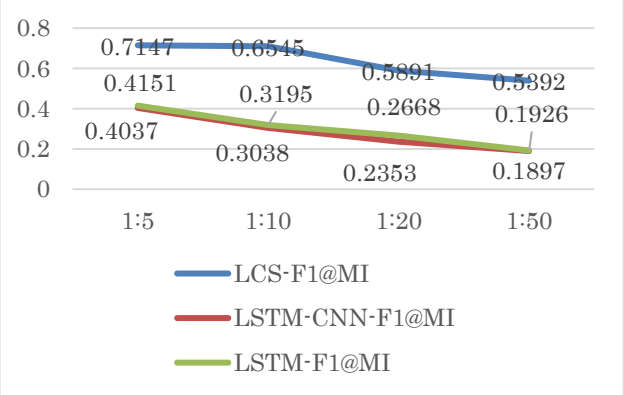

Table 1 Comparing F1@MI of LCS ,LSTM-CNN, and LSTM models

In the next experiment, we compare our model LCS with AGDISTIS, AIDA, and DBpedia Spotlight, by the 1:10 dataset of AQUAINT. The results are shown in Table 1, where the results of the models except LCS are provided by the Gerbil testing plat-form [23].

AQUAINT	Micro F1	Micro Precision	Micro Recall
AGDISTIS	0.5213	0.5213	0.5213
AIDA	0.5543	0.6068	0.5089
DBpedia Spotlight	0.5252	**0.7584**	0.4017
LCS	**0.7090**	0.7090	**0.7090**

Table 2 Micro Precision, F1, recall scores reported by Gerbil for AQUAINT dataset compared with LSTM-CNN combining Sparse Feature model

Finally, we compare our model LCS with AGDISTIS, AIDA, and DBpedia Spot-light, on the 1:10 dataset of KORE50. The results are shown in Table 3, where the results other than our LCS model are again provided by the Gerbil testing platform.

KORE50	Micro F1	Micro Precision	Micro Recall
AGDISTIS	0.3264	0.3264	0.3264
AIDA	0.6884	**0.7197**	0.6597
DBpedia Spotlight	0.5214	0.5929	0.4653
LCS	**0.6923**	0.6923	**0.6923**

Table 3 Micro Precision, F1, recall scores reported by Gerbil for KORE50 dataset compared with LSTM-CNN combining Sparse Feature model

4.3. Results and Discussion

From the results, we can clearly see that our LCS model has a significant improvement by incorporating the sparse features to the LSTM-CNN model. On the AQUAINT dataset, DBpedia Spotlight shows the highest precision, but our LCS mod-el outperforms on recall and F1-score. For the KORE50 dataset, we can see that AIDA has the highest precision, but it has a drawback on recall, so its F1-score is falling behind of our model. For a practical example, there is a sentence *"While Apple is an electronics company,* ***Mango*** *is a clothing one and Orange is a communication one."* in KORE50. We need to link the ***Mango*** to the entity in knowledge base. The result of these method are shown in table 4. We can see the AGDISTIS all link to the fruit mango, the AIDA link it to a company but not mango. Our method link the mention to the Mango Company.

Method	**Result**
AGDISTIS	Mango
AIDA	Island_Records
DBpedia Spotlight	Mango
LCS	Mango_(clothing)

Table 4 Result of four methods on the mention "Mango"

Overall, our model LCS which combines LSTM-CNN with sparse features model showed the best performance among the compared models, in both AQUAINT and KORE50 datasets.

5. Conclusion and Future Work

In this paper, we proposed an entity disambiguation model LCS, which integrates the neural network model LSTM-CNN and the sparse features of PageRank and popularity in a novel way. Our evaluation results show that our model realized significant superiority in accuracy over conventional LSTM-CNN models, as well as traditional approaches.

In future, we plan to enrich the model by incorporating more feature s based various contexts, and evaluate our model on more datasets.

Reference

[1] Blunsom, Phil, E.Grefenstette, and N. Kalchbrenner. "A convolutional neural network for modelling sentences." Proc. 52nd Annual Meeting of the Association for Computational Linguistics, 2014.

[2] Graves, A. (2012). Supervised sequence labelling. In Supervised sequence labelling with recurrent neural networks (pp. 5-13). Springer, Berlin, Heidelberg.

[3] Graves, A., Jaitly, N., & Mohamed, A. R. (2013, December). Hybrid speech recognition with deep bidirectional LSTM. In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on (pp. 273-278). IEEE.

[4] Ganea, Octavian-Eugen, et al. "Probabilistic bag-of-hyperlinks model for entity linking." Proc. 25th Int. Conf. World Wide Web, 2016.

[5] Han, Xianpei, Le Sun, and Jun Zhao. "Collective entity linking in web text: a graph-based method." Proc. 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2011.

[6] Hoffart, Johannes, et al. "Robust disambiguation of named entities in text." Proc. Conf. on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011.

[7] Hoffart, J., Seufert, S., Nguyen, D. B., Theobald, M., & Weikum, G. “KORE: keyphrase overlap relatedness for entity disambiguation.” Proc. 21st ACM Int. Conf. Information and Knowledge Management (CIKM), pp. 545-554, 2012.

[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[9] Kingma, Diederik, and Jimmy Ba. “Adam: A method for stochastic optimization,” in Proceedings of International Conference on Learning Representations Workshop, pp. 1–15, 2015

[10] Lai, Siwei, et al. "Recurrent Convolutional Neural Networks for Text Classification." AAAI. Vol. 333. 2015.

[11] Milne, David, and Ian H. Witten. "Learning to link with wikipedia." Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008.

[12] Mendes, P. N., M. Jakob, A. Garcia-Silva, and C. Bizer. “DBpedia Spotlight: Shedding Light on the Web of Documents.” Proc. 7th Int. Conf. Semantic Systems (I-Semantics), 2011.

[13] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in Neural Information Processing Systems. 2013.

[14] Nair, V., & Hinton, G. E “Rectified linear units improve restricted boltzmann machines.” Proc. 27th Int. Conf. Machine Learning (ICML-10), pp. 807-814, 2010.

[15] Ouyang, Liubo, et al. "Chinese Named Entity Recognition Based on B-LSTM Neural Network with Additional Features." International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, Cham, 2017.

[16] Page, Lawrence, et al. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab, 1999.

[17] Pennington, J., R. Socher, and C. Manning. "Glove: Global vectors for word representation." Proc. 2014 Conference on Empirical Mmethods in Natural Language Processing (EMNLP). 2014.

[18] Pichotta, Karl, and Raymond J. Mooney. "Learning Statistical Scripts with LSTM Recurrent Neural Networks." AAAI. 2016.

[19] Shen, Wei, et al. "Linden: linking named entities with knowledge base via semantic knowledge." Proc. 21st Int. Conf. World Wide Web, ACM, 2012.

[20] Shen, Wei, Jianyong Wang, and Jiawei Han. "Entity linking with a knowledge base: Issues, techniques, and solutions." IEEE Trans. Knowledge and Data Engineering 27.2 (2015): 443-460.

[21] Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." Advances in Neural Information Processing Systems. 2015.

[22] Tang, Duyu, Bing Qin, and Ting Liu. "Learning Semantic Representations of Users and Products for Document Level Sentiment Classification." ACL (1). 2015.

[23] Usbeck, Ricardo, et al. “AGDISTIS-graph-based disambiguation of named entities using linked data.” Int. Semantic Web Conf., Springer LNCS, 2014

[24] Usbeck, Ricardo, et al. “GERBIL: general entity annotator benchmarking framework." Proc. 24th Int. l Conf. World Wide Web”, 2015.

[25] Yosef, Mohamed Amir, et al. "Aida: An online tool for accurate disambiguation of named entities in text and tables." Proceedings of the VLDB Endowment 4.12, 2011.

[26] Zhou, Qianrong, et al. "A hierarchical lstm model for joint tasks." China National Conference on Chinese Computational Linguistics. Springer International Publishing, 2016.