# CogIE: An Information Extraction Toolkit for Bridging Texts and CogNet

**Zhuoran Jin**[1,2], **Yubo Chen**[1,2], **Dianbo Sui**[1,2],
**Chenhao Wang**[1,2] , **Zhipeng Xue**[1] , **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences,
Beijing, 100049, China
{zhuoran.jin, yubo.chen, dianbo.sui}@nlpr.ia.ac.cn
{chenhao.wang, zhipeng.xue, jzhao}@nlpr.ia.ac.cn

## Abstract

CogNet is a knowledge base that integrates three types of knowledge: linguistic knowledge, world knowledge and commonsense knowledge. In this paper, we propose an information extraction toolkit, called CogIE, which is a bridge connecting raw texts and CogNet. CogIE has three features: **versatile**, **knowledge-grounded** and **extensible**. First, CogIE is a versatile toolkit with a rich set of functional modules, including named entity recognition, entity typing, entity linking, relation extraction, event extraction and frame-semantic parsing. Second, as a knowledge-grounded toolkit, CogIE can ground the extracted facts to CogNet and leverage different types of knowledge to enrich extracted results. Third, for extensibility, owing to the design of three-tier architecture, CogIE is not only a plug-and-play toolkit for developers but also an extensible programming framework for researchers. We release an open-access online system [1] to visually extract information from texts. Source code, datasets and pre-trained models are publicly available at GitHub [2], with a short instruction video [3].

## 1 Introduction

Knowledge bases (KBs) such as FrameNet (Baker et al., 1998), DBpedia (Lehmann et al., 2015), Wikidata (Vrandečić and Krötzsch, 2014), and ConceptNet (Liu and Singh, 2004) are becoming popular for a variety of downstream tasks including information retrieval, recommender system and dialog system. Wang et al. (2021) divide KBs into three categories according to the type of knowledge, respectively linguistic KBs (e.g., FrameNet), world KBs (e.g., DBpedia, Wikidata) and commonsense KBs (e.g., ConceptNet). Unlike most of the above KBs which focus on a single type of knowledge, CogNet (Wang et al., 2021) models linguistic, world and commonsense knowledge using a unified representation architecture for better knowledge integration.

To apply CogNet to downstream tasks, it is challenging to expand CogNet and ground raw texts to CogNet automatically. For this target, information extraction (IE) is an effective method, which aims to extract entity, relation, event, and other factual information from raw texts and link them to KBs.

With the rapid development of IE area, a few remarkable open-source toolkits have been developed in recent years. The mainstream toolkits can be classified into two categories: task-specific toolkits and task-agnostic toolkits. Task-specific toolkits focus on one or a few specific tasks, such as FLAIR (Akbik et al., 2019) for named entity recognition (NER), BLINK (Ledell Wu, 2020) for entity linking (EL), OpenNRE (Han et al., 2019) for relation extraction (RE) and Open-SESAME (Swayamdipta et al., 2017) for frame-semantic parsing. On the other end of the spectrum, AllenNLP (Gardner et al., 2017), OpenNMT (Klein et al., 2017) and other task-agnostic toolkits are designed to provide programming framework without the implementation of specific tasks.

As mentioned above, various toolkits have been widely used, but they also suffer from several limitations. First, most of the existing NLP toolkits only support one or a few IE functions, and there is a lack of an integrated and efficient IE toolkit. Second, very few IE toolkits can align the extracted facts to KBs, which may cause the extracted facts not to be applied directly to downstream tasks. Third, for an efficient and effective toolkit, providing application program interfaces (APIs) is as important as supporting the secondary development. Still, only a few toolkits can do both at the same time.

[1] http://cognet.top/cogie
[2] https://github.com/jinzhuoran/CogIE
[3] https://youtu.be/csgnjU_F3Qs



*Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 92–98, August 1st - August 6th, 2021.

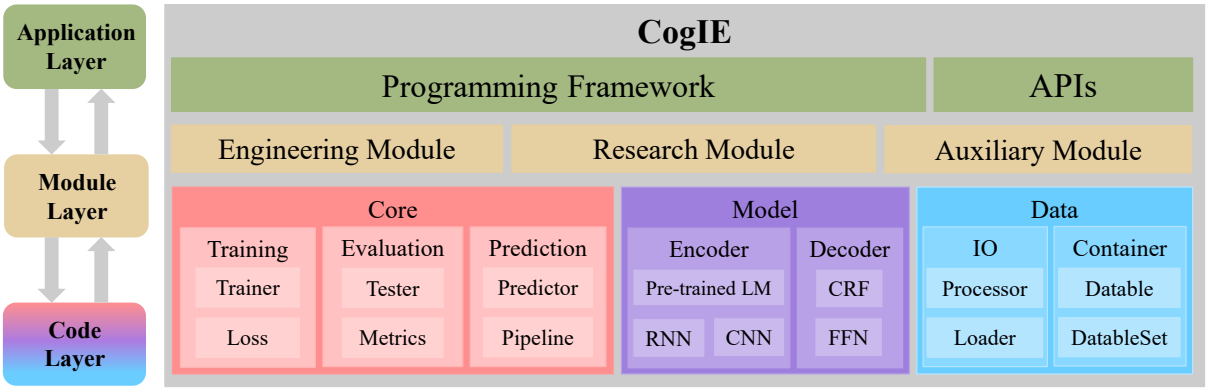


Figure 1: **Left:** The three-tier architecture of CogIE. **Right:** The internal structure of each layer.

Therefore, it is highly desirable to have an open-source toolkit that can implement and integrate various IE tasks and can take advantage of the knowledge resources in KBs to enrich the extracted facts. Such a toolkit should achieve the equilibrium among usability, extensibility and efficiency.

To this end, we propose **CogIE**, an **IE** toolkit that bridges raw texts and **Cog**Net, making it easy to extract facts from texts as well as ground the extracted facts to CogNet. The toolkit supports both English and Chinese, building upon PyTorch with the same uniform design. Moreover, CogIE can meet the requirements of function customizability and model extensibility for researchers. CogIE also provides APIs for developers to build applications rapidly. We release an online CogIE system to extract information from input texts with friendly interactive interfaces and fast response speed.

In summary, the main features and contributions are as follows:

- **Versatile.** We develop a professional and integrated IE toolkit. CogIE can support high-performance named entity recognition, entity typing, entity linking, relation extraction, event extraction and frame-semantic parsing.
- **Knowledge-grounded.** We build a bridge between raw texts and CogNet. CogIE can ground the extracted facts to CogNet and leverage different types of knowledge to enrich results.
- **Extensible.** We contribute not just user-friendly APIs, but an extensible programming framework. Our goal in designing CogIE is to provide a universal toolkit for all sorts of users.

## 2 System Design and Architecture

In this section, we introduce the design choice and system architecture of CogIE. Designing a powerful toolkit is challenging due to different types of IE tasks and fast-growing new models. As illustrated in Figure 1, we tackle the challenges by dividing the main modules and components of CogIE into three layers. Each layer in CogIE plays a unique role separately.

### 2.1 Application Layer

The application layer acts as a mediator between CogIE and users, including researchers and developers. Researchers pay more attention to internal details and prefer a programming framework to support function customization and model construction. On the contrary, developers are more likely to use the high-level functions provided by the toolkit directly without knowing too many low-level details. Considering the different requirements of both sides, we divide CogIE into two parts at the application layer: (1) a programming framework supporting NLP research; (2) APIs providing IE functions.

**NLP Programming Framework.** The primary design goal of CogIE is to make it easy to meet some individual requirements with our experiment paradigm. Specifically, we decouple NLP experiments, forming three consecutive parts of **Training - Evaluation - Prediction**. Thus, users can use the programming framework to train new models, validate performances and make predictions based on the trained models.

**IE APIs.** We also implement a series of typical models by the unified framework of CogIE. CogIE provides APIs with multilingual support (En-

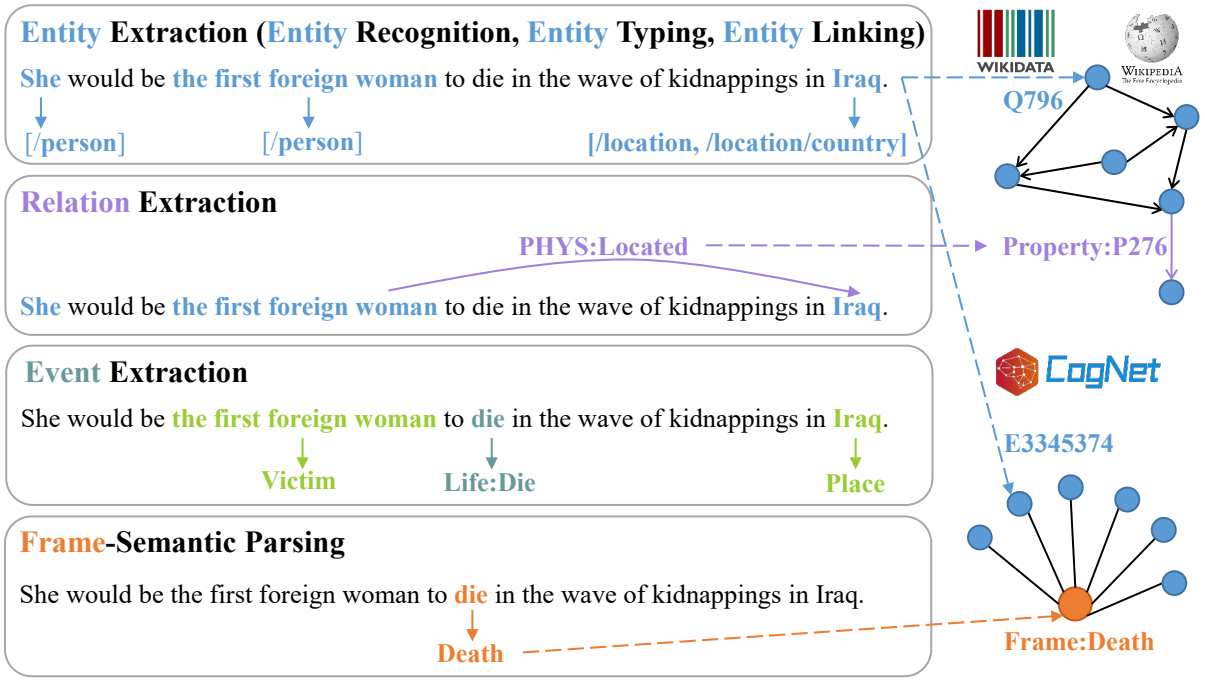


Figure 2: The examples of main functions in CogIE.

glish and Chinese), including word segmentation, named entity recognition, entity typing, entity linking, relation extraction, event extraction and frame-semantic parsing, etc. APIs take raw texts as input and produce structured extraction results accurately and quickly.

### 2.2 Module Layer

The module layer is based on the principle that each module has a single function and contacts with as few modules as possible. In this layer, CogIE consists of three independent modules, namely engineering module, research module and auxiliary module. In this way, users only need to focus on core neural network models without writing repetitive and complex engineering code, allowing experiments easier and faster. The following is the detailed design philosophy of each module.

**Engineering Module.** This module mainly integrates the code with high repeatability in different task scenarios. In this way, CogIE is less error-prone and more time-saving by automating most of the training loop and tricky engineering.

**Research Module.** To make code more concise and extensible, we decouple the research module from the engineering module. The research module mainly includes the user-defined neural network models, loss functions, etc., which are the core of research.

**Auxiliary Module.** The auxiliary module is designed to assist experiments by accelerating training, saving checkpoints, recording logs, and visualizing results. For example, CogIE can support 16-bit precision to cut memory footprint by half and use TensorBoard [4] to visualize experimental parameters.

### 2.3 Code Layer

The code layer relates to the underlying design of CogIE. This layer consists of three interdependent parts: core code, model code and data code.

**Core Code.** In the core code, we develop a variety of ready-to-use components for users. Because of the special **Training - Evaluation - Prediction** experiment paradigm, `Trainer`, `Tester` and `Predictor` class are the key components of core code. In the case of `Trainer` class, users just need to feed the expected components (e.g., model, dataset, loss function, evaluation metric, configuration file, etc.) into it, everything else is automatically done.

**Model Code.** `BaseModel` class is the base class of all models in CogIE. `BaseModel` class organizes code into four sections: (1) `forward`

[4] https://github.com/lanpa/tensorboardX

function for computation, (2) `loss` function for train, (3) `evaluate` function for validation, and (4) `predict` function for prediction. Model code consists of two parts: encoder module (e.g., Pre-trained Language Model, RNN, CNN, etc.) and decoder module (e.g., CRF, FFN, etc.). By designing model code this way, it is convenient to change from one model to another by simply plugging in and swapping out a single or few modules.

**Data Code.** The data code is built around the notion of `Datable` which stores data in table form. CogIE includes built-in `Loader` and `Processor` class for lots of popular datasets and provides easy-to-use data containers to encapsulate all the steps needed to process data.

## 3 Core Functions

CogIE is designed for a series of IE functions, including named entity recognition, entity typing, entity linking, relation extraction, event extraction, and frame-semantic parsing, etc. CogIE can also align the extracted facts to CogNet via entity linking, relation matching and frame matching. As shown in Figure 2, we give some examples to illustrate these functions.

### 3.1 Named Entity Recognition

Named entity recognition (NER) is a task for locating and classifying certain occurrences of words or expressions in unstructured texts into predefined semantic categories. To achieve the function of entity recognition, we adopt BERT as the textual encoder and use CRF as the decoder. Up to now, CogIE can not only recognize the common four entity types: locations, persons, organizations, and miscellaneous entities, but also support the recognition of 54 entity types.

### 3.2 Entity Typing

Fine-grained entity typing aims to assign one or more types to each entity mention given a certain context and can provide valuable prior knowledge for a wide range of NLP tasks, such as relation extraction and question answering. To achieve the function of entity typing, we adopt a two-step mention-aware attention mechanism to enable the model to focus on important words like Lin and Ji (2019). Compared with NER, ET has finer and richer entity labels with internal correlations (e.g., /person, /person/artist, /person/artist/actor), there are 87 fine-grained entity lables in CogIE.

### 3.3 Entity Linking

Entity linking is the task to link entity mentions in texts with their corresponding entities in a knowledge base. To achieve the function of entity linking, we use BLINK which adopts a two-stage approach for entity linking based on fine-tuned BERT architectures. CogIE supports link entities to CogNet and Wikidata, users can leverage multiple types of knowledge obtained through EL to implement knowledge base population (KBP) and knowledge based question answering (KBQA).

### 3.4 Relation Extraction

Relation extraction aims at predicting semantic relations between pairs of entities. More specifically, after identifying entity mentions in texts, the main goal of RE is to classify relations. To achieve the function of relation extraction, we adopt BERT as the textual encoder and use FFN as the decoder. As CogIE implements relation extraction simultaneously, it also matches extracted relations to Wikidata in the form as shown in Figure 2. We train relation matching on T-REx (Elsahar et al., 2018), which is a large-scale alignment dataset between free text documents and KB triples, there are currently 500 relation classes in CogIE.

### 3.5 Event Extraction

Events are classified as things that happen or occur, and usually involve entities as their properties. Event extraction need to identify events that are composed of an event trigger, an event type, and a set of arguments with different roles. To achieve the function of event extraction, we realize DM-CNN (Chen et al., 2015) and a joint model based on BERT.

### 3.6 Frame-Semantic Parsing

Frame semantic parsing is the task of automatically extracting semantic structures in plain texts according the framework of FrameNet. Each frame represents a kind of event, situation, or relationship, and consists of a frame name, a list of lexical units (LUs), and a set of frame elements (FEs). LU is a word that plays the role of evoking the corresponding frame. FE indicates different semantic roles associated with the frame.

Frame-semantic parsing is usually performed as a pipeline of tasks: target identification, frame identification and argument identification. To achieve the function of argument identification, we add tar-

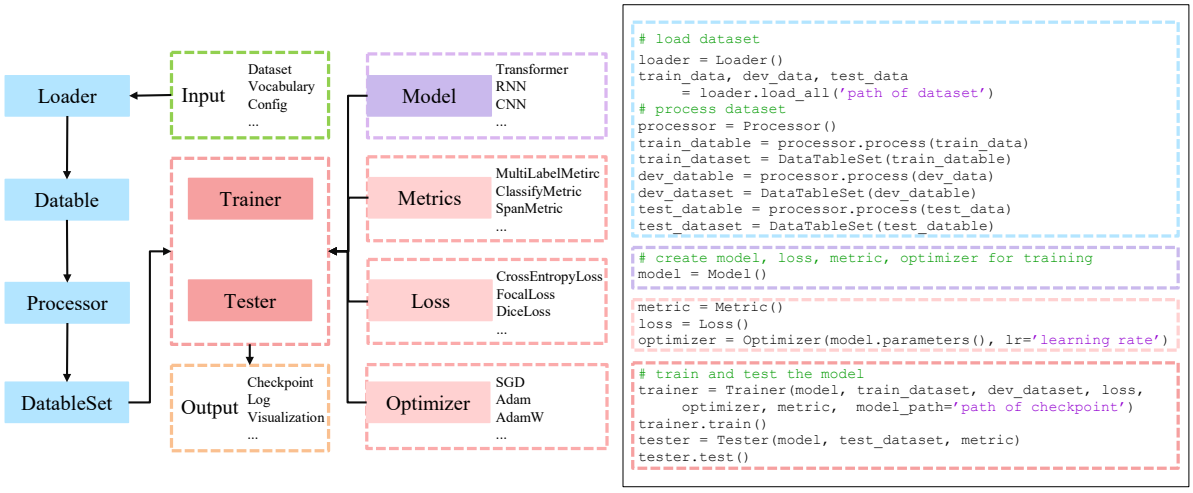


Figure 3: The sample code of model training in CogIE.

get representation and position representation to BERT encoder. CogIE currently supports to identify 749 frames and 816 FEs in FrameNet.

## 4 System Usage

Our goal of designing CogIE is to provide a user-friendly toolkit for users by achieving the equilibrium among usability, extensibility and efficiency.

### 4.1 Interface Calls

CogIE's APIs can be directly called by `Toolkit` class, where the previous output is pipelined to the following input. Considering APIs' flexibility, users need to specify different tasks, languages and datasets, while pre-trained models can be downloaded and loaded to `Toolkit` class automatically.

As shown in Figure 4, the code snippet shows a pipelined usage of CogIE for tokenizing a sentence into words, recognizing entities, and extracting relations between entities:

```
import cogie
# tokenize the text into words
token_toolkit =
    cogie.TokenizeToolkit(language='english')
words = token_toolkit.run('Ontario is the most
    populous province in Canada.')
# recognize the entities in the texts
ner_toolkit = cogie.NerToolkit(language='english')
ner_result = ner_toolkit.run(words)
# extract the relations between entities
re_toolkit = cogie.ReToolkit(language='english')
re_result = re_toolkit.run(words, ner_result)
print(re_result)
```

Figure 4: The sample code of interface calls in CogIE.

### 4.2 Model Training

The hallmark of any good toolkit is its extensibility. As a programming framework, CogIE supports users to train their customized models without modifying the CogIE codebase. Figure 3 shows the sample code of model training, and one can use only a tiny amount of code for data processing, component initializing, model training, and model evaluating.

To do this, users need to use `Loader` class to load the dataset and process it into `DatableSet` class by `Processor` class. Then, `Model`, `Loss`, `Metric`, `Optimizer` class should be initialized before added to `Trainer` class. And finally, `Trainer` and `Tester` class can train and validate the model while generating checkpoints, logs and visualization results.

### 4.3 Online System

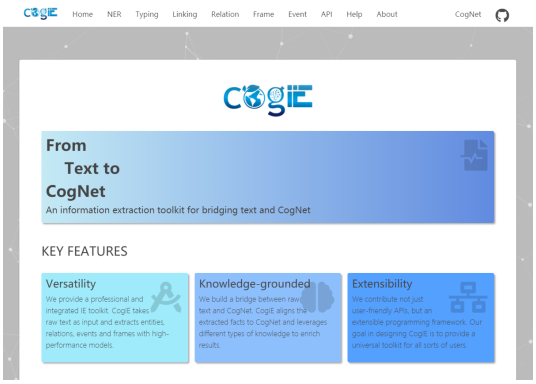


Figure 5: An example of the online system.

In addition to this toolkit, we also release an open-access online system as shown in Figure 5.

| Task | | Corpus | Language | Types | Metric | Score |
|---|---|---|---|---|---|---|
| Word Segmentation | | MSRA | Chinese | - | $F_1$ | 91.2 |
| Named Entity Recognition | | CoNLL2003 | English | 4 | $F_1$ | 91.4 |
| | | OntoNotes5.0 | English | 18 | $F_1$ | 85.6 |
| | | OntoNotes4.0 | Chinese | 4 | $F_1$ | 80.0 |
| Entity Typing | | BBN | English | 47 | $F_1$ | 75.5 |
| Relation Extraction | | KBP37 | English | 37 | $F_1$ | 69.9 |
| | | DuIE | Chinese | 48 | $F_1$ | 93.0 |
| Event Extraction | Trigger | ACE2005 | English | 33 | $F_1$ | 68.9 |
| | Argument | | | 35 | $F_1$ | 46.4 |
| | Trigger | ACE2005 | Chinese | 33 | $F_1$ | 58.8 |
| | Argument | | | 35 | $F_1$ | 52.8 |
| Frame-Semantic Parsing | Frame | Frame 1.5 | English | 749 | Acc | 91.0 |
| | Element | | | 816 | $F_1$ | 56.4 |

Table 1: Performance of each task. The datasets references are: MSRA (Emerson, 2005), CoNLL2003 (Sang and De Meulder, 2003), OntoNotes5.0 (Pradhan et al., 2013), OntoNotes4.0 (Weischedel et al., 2011), BBN (Weischedel and Brunstein, 2005), KBP37 (Zhang and Wang, 2015), DuIE (Li et al., 2019), ACE2005 (Walker et al., 2006), and Frame 1.5 (Kabbach et al., 2018).

We train models for different tasks and deploy pre-trained models for online access. The online system can be directly used for extracting entities, relations, events and frames from plain texts. Besides, the extracted results can be linked to CogNet, so users can further acquire external knowledge through CogNet. We also visualize the extracted results in the form of knowledge graphs to improve the availability of the online system. Meanwhile, open online APIs [5] can be called directly .

## 5 Experiment and Evaluation

In this section, we train and evaluate CogIE on several datasets in different tasks. Each task's performance is shown in Table 1, all pre-trained models are publicly downloadable.

For the NER component, we compare CogIE against Stanza (v1.0), FLAIR (v0.4.5) and spaCy (v2.2), we find that CogIE can achieve either higher or close F1 scores when compared against other toolkits. For the RE component, we compare CogIE with two baselines: RNN+PI (Zhang and Wang, 2015) and BERT$_{EM}$ (Soares et al., 2019), we observe that CogIE can achieve comparable or even better performance than them. For the frame-semantic parsing component, we compare CogIE against SimpleFrameId (Hartmann et al., 2017), we find that CogIE can have better performance than SimpleFrameId. For the other components, we also compare CogIE with a series of baselines and toolkits, and the evaluation results show that CogIE can provide powerful IE functions.

[5] http://cognet.top/cogie/api.html

## 6 Conclusion and Future Work

In this paper, we propose CogIE, an information extraction toolkit for bridging texts and CogNet. We have shown that CogIE is a plug-and-play toolkit and an extensible programming framework due to its **Application - Module - Code** three-tier architecture design. Moreover, as an integrated and professional IE toolkit, CogIE can extract information from texts while aligning the extracted facts to CogNet and other KBs. We conduct experiments on several datasets in different tasks, and the evaluation results demonstrate the models implemented by CogIE are efficient.

In the future, we consider the following points during improvement: (1) To use CogIE on any device, we will further optimize model sizes and speed up computation in CogIE while striking a balance between accuracy and efficiency; (2) For making models robust to the texts of different domains and styles, we plan to utilize various sources of consistent data to train a universal model; (3) We will build an open-source community for CogIE so that all researchers can contribute their models and participate in long-term maintenance.

## Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2020AAA0106400), the National Natural Science Foundation of China (No.61806201).

## References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proc. of NAACL-HLT*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proc. of ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proc. of ACL*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proc. of LREC*.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proc. of SIGHAN*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. *ArXiv:1803.07640*.

Xu Han, Tianyu Gao, Yuan Yao, Demin Ye, Zhiyuan Liu, and Maosong Sun. 2019. Opennre: An open and extensible toolkit for neural relation extraction. In *Proc. of EMNLP*.

Silvana Hartmann, Ilia Kuznetsov, M Teresa Martín-Valdivia, and Iryna Gurevych. 2017. Out-of-domain framenet semantic role labeling. In *Proc. of ECAL*.

Alexandre Kabbach, Corentin Ribeyre, and Aurélie Herbelot. 2018. Butterfly effects in frame semantic parsing: impact of data processing on model ranking. In *Proc. of COLING*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. of ACL*.

Martin Josifoski Sebastian Riedel Luke Zettlemoyer Ledell Wu, Fabio Petroni. 2020. Zero-shot entity linking with dense entity retrieval. In *Proc. of EMNLP*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*.

Shuangjie Li, Wei He, Yabing Shi, Wenbin Jiang, Haijin Liang, Ye Jiang, Yang Zhang, Yajuan Lyu, and Yong Zhu. 2019. Duie: A large-scale chinese dataset for information extraction. In *Proc. of NLPCC*.

Ying Lin and Heng Ji. 2019. An attentive fine-grained entity typing model with latent type representation. In *Proc. of EMNLP*.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. *Proc. of CoNLL*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proc. of ACL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. *ArXiv:1706.09528*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium*.

Chenhao Wang, Yubo Chen, Zhipeng Xue, Yang Zhou, and Jun Zhao. 2021. Cognet: Bridging linguistic knowledge, world knowledge and commonsense knowledge. In *Proc. of AAAI*.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *Linguistic Data Consortium*.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *ArXiv:1508.01006*.