A Predicate-Function-Argument Annotation of Natural Language for Open-Domain Information eXpression

Mingming Sun, Wenyue Hua, Zoey Liu, Kangjie Zheng, Xin Wang, Ping Li
Cognitive Computing Lab
Baidu Research
No.10 Xibeiwang East Road, Beijing 100193, China
10900 NE 8th St. Bellevue, Washington 98004, USA
{sunmingming01, wangxin60, liping11}@baidu.com
{norahua1996, zoeyliu0108, kangjie.zheng}@gmail.com

Abstract

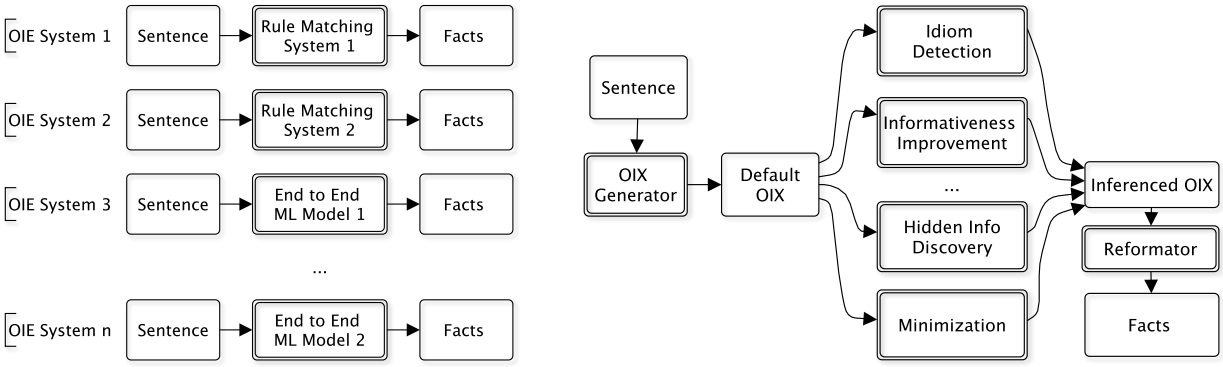
Existing OIE (Open Information Extraction) algorithms are independent of each other such that there exist lots of redundant works; the featured strategies are not reusable and not adaptive to new tasks. This paper proposes a new pipeline to build OIE systems, where an Open-domain Information eXpression (OIX) task is proposed to provide a platform for all OIE strategies. The OIX is an OIE friendly expression of a sentence without information loss. The generation procedure of OIX contains shared works of OIE algorithms so that OIE strategies can be developed on the platform of OIX as inference operations focusing on more critical problems. Based on the same platform of OIX, the OIE strategies are reusable, and people can select a set of strategies to assemble their algorithm for a specific task so that the adaptability may be significantly increased. This paper focuses on the task of OIX and propose a solution – Open Information Annotation (OIA). OIA is a predicate-function-argument annotation for sentences. We label a data set of sentence-OIA pairs and propose a dependency-based rule system to generate OIA annotations from sentences. The evaluation results reveal that learning the OIA from a sentence is a challenge owing to the complexity of natural language sentences, and it is worthy of attracting more attention from the research community.

1 Introduction

In the past decades, various OIE (Open Information Extraction) systems (Banko et al., 2007; Yates et al., 2007; Wu and Weld, 2010; Etzioni et al., 2011; Fader et al., 2011; Mausam et al., 2012) have been developed to extract various types of facts. Earlier OIE systems extract verbal relations between entities, while more recent systems enlarge the types of relations. For example, RelNOUN (Pal and Mausam, 2016) extract nominal properties. Sun et al. (2018a; 2018b) can extract four types of facts: verbal, prepositional, nominal, and conceptional. OLLIE (Mausam et al., 2012) and ClauseIE (Corro and Gemulla, 2013) extract relations between clauses. In addition to extracting the fact tuples, NestIE (Bhutani et al., 2016) and StuffIE (Prasojo et al., 2018) extract nested facts. Furthermore, MinIE (Gashteovski et al., 2017) add factuality annotations to the facts.

Currently, existing OIE systems were typically developed from scratch, generally independent from each other. Each of them has their own concerned problem and builds its own pipeline from a sentence to the final set of facts (See Figure 1a). Generally, each OIE system is a complex composition of several extraction strategies (for rule-based systems) or data labeling strategies (for end-to-end supervised learning). It is rather straightforward for specific problems. However, this practice has several major drawbacks outlined as follows:

- *Redundant works*. Some common works are implemented again and again in different ways in each OIE system, such as converting simple sentences with clear *subj* and *obj* dependencies into a predicate-argument structure.
- *Strategies are not reusable*. During the years of OIE practice, several sub-problems are believed valuable, e.g., nested structure identification (Bhutani et al., 2016), informative predicate construction (Gashteovski et al., 2017), attribute annotation (Corro and Gemulla, 2013; Gashteovski et al., 2017), etc. Each sub-problem is worthy of being standardized and continually studied given a well defined objective and data sets so that the performance could be fairly evaluated and the progress can be continually made. However, it is not easy in the current methodology, since each pipeline's strategies are closely bonded to own implementation.



(a) Traditional OIE systems.

(b) OIX based OIE system.

Figure 1: Methodologies to construct OIE systems

- *Unable to adapt*. Because of the above two factors, there is no platform to implement the shared requirement to provide unified data set, and the strategies are not reusable. Furthermore, each OIE system extracts the interested facts in the desired form at the time of development and omits the uninterested facts. Consequently, they are not adaptable to new requirements. If the interests or the requested form of facts change, one may need to write an entire new OIE pipeline.

As the OIE task has attracted more and more interest (Christensen et al., 2013, 2014; Fader et al., 2014; Mausam, 2016; Stanovsky et al., 2015; Khot et al., 2017), the above mentioned drawbacks have delayed the progress of OIE techniques. The key to conquering those obstacles is to provide a shared platform for all OIE algorithms, which express all the information in sentences in the form of OIE facts (that is, predicate-arguments tuples) without losing information. OIE strategies can focus on inferring new facts from existing ones without knowing the existence of the sentence. With this platform, the strategies are reusable and can be fairly compared. When confronting a specific task, one can select a set of strategies or develop new strategies and run the strategies on the platform to build a new OIE pipeline. In this manner, the adaptability is much improved. This new methodology of OIE is shown in Figure 1b.

We name the task of implementing such a platform as Open Information eXpression (OIX), where *eXpression* is used to distinguish from *Extraction* to emphasize that it focuses on expressing all the information in the sentence rather than extracting the interested part of the information. This methodology potentially results in a multi-task learning scenario where many agents (each one is interested in a part of information) compete with each other for words. This competition may result in more robust expressions than those who only extract part of the information. This paper focuses on investigating the OIX task requirements and finding a solution for this task.

In Section 2, we discuss the principle of design solution for OIX and propose a solution – the Open Information Annotation (OIA) – to fulfill those principles. The OIA of a sentence is a single-rooted directed-acyclic graph (DAG) with nodes representing phrases and edges connecting the predicate nodes to their argument nodes. We describe the detailed annotation strategies of OIA in Section 3. Based on the OIA, several featured strategies from existing OIE algorithms can be ported to work on the OIA. Section 4 discusses the possible implementation of those strategies on the OIA. We label a data set of OIA graphs, build a rule-based pipeline for automatically generating OIA graphs from sentences, and evaluate the pipeline's performance on the labeled data set. All these work are stated in Section 5. We discuss the connection from OIA to Universal Dependency, Abstract Meaning Representation (Banarescu et al., 2013), and SAOKE (Sun et al., 2018b) in Section 6. We conclude the paper in Section 7.

2 Open Information eXpression

2.1 Design Principles of the Expression Form

We consider the following factors in designing the expression form for the OIX task:

- **Information Lossless** As the OIX task is to provide a platform for following OIE strategies, the loss of any information is unacceptable. A simple constraint can guarantee this: any word in the

sentence must appear in the target form of OIX.

- **Validity** It must implement the information structure of OIE tasks, that is, the predicate-argument structure. It builds a boundary for the OIE pipeline: after the OIX task, followed strategies all work on open-domain facts, without knowing the original sentences.
- **Capacity** The form should be able to express all kinds of information involved in the sentences, including 1) relation between entities; 2) the nested facts, that is, fact as an argument of another fact; 3) the relationships between facts, including the logical connections such as "if-else" and discourse relations such as "because", "although"; 4) information in the natural language other than declarative sentences, such as questions that ask to return one or a list of possible answers (Karttunen, 1977).
- **Atomicity** Since the form is a common expression of facts to serve different OIE strategies, we have no bias in the form of predicate and perform atomic expression so that followed strategies can assemble them according to their preference. For example (Gashteovski et al., 2017), for the sentence "Faust made a deal with the Devil", ClausIE produces (Faust, made, a deal with the Devil), while the MinIE extracts (Faust, made a deal with, the Devil). Instead, we would like a nested structure ((Faust, made, a deal), with, Devil) so that followed strategies can assemble the predicate according to the favor of either ClauseIE or MinIE. Notice that the atomicity does not means it is in word-level. We still need a phrase-level expression of facts, following the traditional OIE system's preference for *simple* phrase (detailed in later sections).

2.2 Information in Natural Languages

Natural languages talk about entities, the factual/logical relationship among them, and describe the status/attributes of them. When talking about entities, the human may talk about some explicit entity or refer a delegate of some unknown entities. When talking about relationships, the relationship may be among entities and can be among entities and relationships; that is, the relationship can be nested. So, from the logical view, we need the following components to express the information in languages:

- Constants: express entities, such as "the solar system", "the Baidu company"; or status of entities/events/relationships, such as "expensive", "hardly".
- Functions: $\mathbf{f}(arg_1, \cdots, arg_n) \rightarrow \{e\}$, express query of entities or delegation of entities, such as "the CEO of X", "when Y", where X and Y denote the arguments of the functions;
- Predicates: $\mathbf{p}(arg_1, \cdots, arg_n) \rightarrow \{0, 1\}$, express factual relationships and logical connections among entities, predicates, and functions, such as "X buy Y", "X says Y ", "Y, because Z".

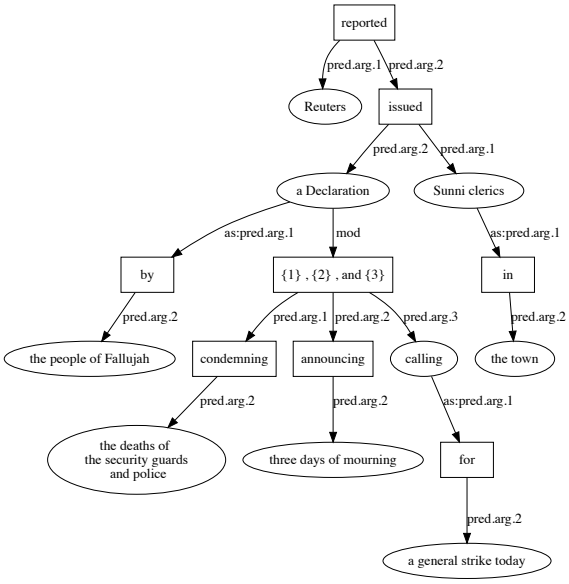
where arg_i could be a constant, predicate or function, and $\{e\}$ is some unknown set of entities returned by the function. With these components, the constants and the instantiated functions become terms, the instantiated factual predicates become atom formulas, the instantiated logical predicates become general formulas, and finally, a sentence can be expressed as a formula. Through this kind of expression, the gap between the language and the knowledge is narrowed. We propose Open Information Annotation to implement this methodology.

2.3 Open Information Annotation

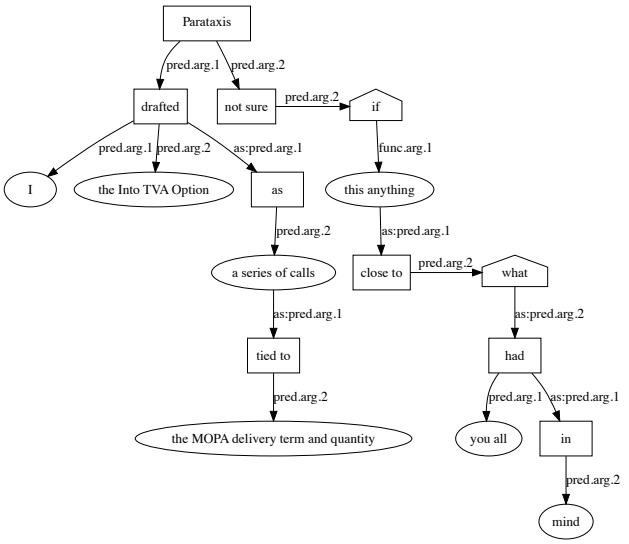
Open Information Annotation (OIA) annotation of a sentence is a single-rooted directed-acyclic dependency graph (DAG), where nodes are predicates/functions/arguments and edges connect the predicates or functions to their arguments. OIA minimizes the information loss by requiring all the words (except the punctuation) in source sentences to appear in the graph. It is single-rooted, which meets the sentence's hierarchical semantic structure, and is for better visualization, understanding, and annotation. Figure 2 gives two sample sentences and their corresponding OIA annotations for intuitive understanding. We give a formal description of the OIA graph as follows:

Nodes. The OIA takes the simple phrases as the basic information units and build nodes based on these simple phrases. By *simple* phrase, we mean a fixed expression, or a phrase with a headword together with its auxiliary, determiner dependents, or adjacent *ADJ/ADV* modifiers. There are three types of nodes: constant, predicate, and function:

- Constant Nodes: simple nominal phrases, representing entities in a knowledge base, or simple description phrases, representing a description



(a) Case I – Reuters reported "Sunni clerics in the town issued a 'Declaration by the people of Fallujah' condemning the deaths of the security guards and police, announcing three days of mourning, and calling for a general strike today."



(b) Case II – I drafted the Into TVA Option as a series of calls tied to the MOPA delivery term and quantity - not sure if this anything close to what you all had in mind.

Figure 2: Two example cases of Open Information Annotations

for an event. They are visualized as the ellipse shapes;

- Function Nodes: the question phrases (what, where) since they are desired to return a set of entities in a knowledge base, or the "of" phrase that delegates an unknown entity. They are visualized as the house shapes;
- Predicate Nodes: predicate phrases, including the simple verbal phrase, simple prepositional phrase, simple conjunction phrases, simple modification phrases, etc. They are visualized as the box shapes;

The principles of OIX require that each word (except punctuation) in the sentences must belong to one and only one of the nodes. However, there is some information hidden in natural language that is not expressed by words. To honestly express the information, we introduce predefined functions and predicates, as shown in Table 1. Many predefined predicates are borrowed from the Universal Dependency (Nivre et al., 2020).

Edges. Edges in OIA are connecting each predicate node or function node to its argument, which can be any constant node, predicate node or function node. There are only two basic types of connecting edges: *pred.arg.*$\{n\}$ for predicates and *func.arg.*$\{n\}$ for functions, where n is the index of the argument.

Function	Meaning
Whether	whether-or-not function
2-ary Predicate	**Meaning**
Modification	modification
Reference	reference
Discourse	discourse element
Vocative	the dialogue participant
Appos	apposition
Reparandum	speech repair
***n*-ary Predicate**	**Meaning**
Parataxis	parataxis of *args*
List	*args* are elements of a list

Table 1: Predefined Functions and Predicates, where for 2-ary predicates, their meanings are "*arg1* has a {Meaning} *arg2*".

When a term is modified by a relative clause, the term is acting as an argument of the predicate expressed by the relative clause, but the predicate is used to modify the term. To express such relation, we reverse the edge and add a prefix *as:* to the argument edge, such as *as:pred.arg.1* or *as:func.arg.2*.

For those predefined predicates with two arguments, to reduce the graph's complexity, we al-

Edge	Meaning
$\mathbf{p} \xrightarrow{pred.arg.i} arg_i$	predicate to its i-th *arg*
$\mathbf{f} \xrightarrow{func.arg.i} arg_i$	function to its i-th *arg*
$arg_i \xrightarrow{as:+} \mathbf{p/f}$	i-th *arg* to its predicate/function
$arg_1 \xrightarrow{\mathbf{P}} arg_2$	$\mathbf{P}(arg_1, arg_2)$
$arg_1 \xrightarrow{as:\mathbf{P}} arg_2$	$is\_\mathbf{P}\_of(arg_1, arg_2)$

Table 2: Edges in OIA. "as:+" means add prefix "as:" to the previous listed predicates, and $\mathbf{P}$ denotes any predefined predicate with two arguments.

low the use of an edge connecting two arguments with the label of that predicates (lowercased) to express the relationship (just as the UD annotation). That is, the predicate *Appos(arg1, arg2)* would be expressed by an edge $arg1 \xrightarrow{appos} arg2$ in the OIA graph. The *as:* prefix applies these shortcut edges too, expressing the meaning of "*arg1* is the {Meaning} of *arg2*". We also give abbreviated names for most frequently used edges: *mod* for *modification*, and *ref* for *reference*.

3 Information Expression Using OIA

In this section, we show how to express information involved in various language phenomenons with our OIA. We can only brief the basic framework in the limited content of this paper. More details can be found on the online website for OIX [1].

3.1 Events

Eventive facts (Davidson and Harman, 2012; Kratzer and Heim, 1998) are facts about entities' actions or status, which is generally expressed by the *subj*, *obj* and **comp* dependencies. In OIA, the *pred.arg.1* always points to the subject of the event, and *pred.arg.2* to *pred.arg.N* refer to the (multiple) objects. A simple example is illustrated by Figure 3a. Events themselves can be arguments of predicates as well, as illustrated by Figure 3d.

3.2 Modification

Adjective/Adverbial Modification. Simple modifiers for nouns, verbs, and prepositions are directly merged into the corresponding phrase. For a complex or remote modifier, we use the predicate "Modification" with two arguments B and A (or an edge from B to A with label *mod*) to express the relation of A modifies B. The "today" in Figure 3a is an example.

Modification by Preposition. For preposition phrases such as "A in B" or "A for B", we take the prepositions as the predicates and A, B as the arguments. If A is an argument of another predicate, to preserve the single-root property, we reverse the edge from the preposition to A and add a *as:* prefix to the label, that is, a new edge from A to the preposition with the label *as:pred.arg.1*. Figure 3e is such an example.

Modification by Relative clause. When the relative clause B modifies an argument a of some other predicate/function, that is, B itself conveys a predicate/function with argument a, we reverse the related edge in B and add the *as:* prefix as we do for "Modification" by Preposition. Figure 3f illustrates this case. If B does not involve a as argument but an argument b referencing a, like "which", "who", we do the same thing to b, and add an edge from a to b with label *ref*.

3.3 Cross-Fact Relations

Cross-sentential Connectives. Sentential connectives are ignored in many OIE systems, but they are the "first-class citizen" in our scheme. Sentential connectives such as "therefore", "so", "if" and "because" can represent logical and temporal relations between sentences. We treat them as predicates and facts/propositions as arguments. An example is shown in Figure 3c.

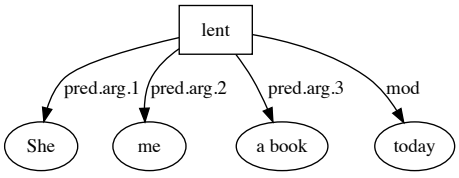
Conjunction/Disjunction. The conjunction and disjunction are expressed by "and" and "or" among a list of parallel components. OIA annotation adds a connecting predicate node delegating the components such as "and" for two components and "{1} and {2} or {3}" for three components, and then link to the arguments with *pred.arg.*{n}. This is illustrated by Figure 3c. More complex situations like Figure 3e are detailed in the online document.

Adverbial Clause. We first build the OIA subgraph for the adverbial clause, and then connect the modified predicate to the root of the sub-graph with edge *mod*.

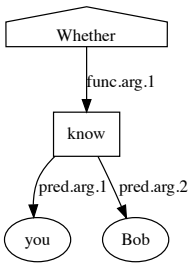
3.4 Questions and Wh-Clauses

We treat question phrases and wh-phrases as functions (Hamblin, 1976; Groenendijk and Stokhof, 1984; Groenendijk and Roelofsen, 2009) and as the root of the OIA graph/sub-graph for sen-

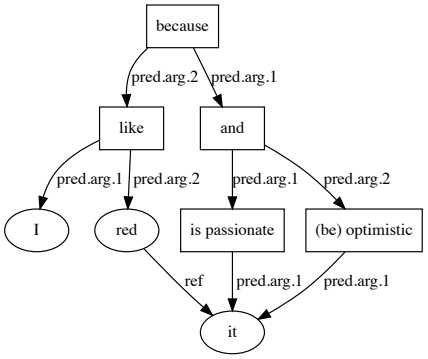
[1] https://sunbelbd.github.io/Open-Information-eXpression/



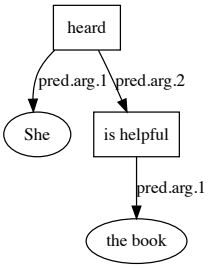
(a) She lent me a book today.



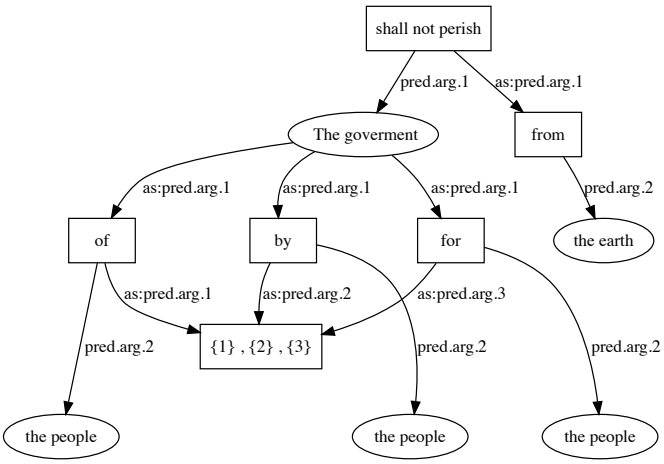
(b) Do you know Bob?



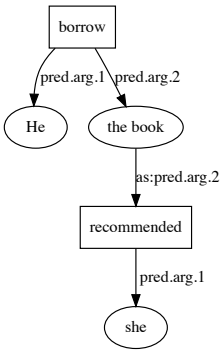
(c) I like red because it is passionate and optimistic.



(d) She heard the book is helpful



(e) The government of the people, by the people, for the people, shall not perish from the earth.



(f) He borrow the book she recommended.

Figure 3: Illustration of Information Expression in Open FPA Graph

tence/clauses. If the phrase ("what", "who", etc.) is an argument of the head predicate of the sentence/clause , the connecting edge is reversed and the *as:* prefix is added to the label; otherwise ("when", "where", etc.), we connect the phrase to the head predicate of the sentence/clause with the label *func.arg.1*. For polarity questions such as "Do you know Bob?", we introduce a predefined function "Whether" (see Table 1) to avoid the confusion caused by taking "Do" as the function phrase. See Figure 2b and Figure 3b.

3.5 Reference

In natural language sentences, words like "it, that, which" refer to an entity mentioned earlier. We express this knowledge by adding an edge *ref* from the entity to the reference word. Again, if this edge violates the single-root rule, the edge will be reversed as *as:ref*. Figure 3c shows the annotation for reference.

4 Inference Operations on OIA Graph

After the OIA graph is constructed, inference operations can be applied to generate a new graph. In this way, strategies from existing OIE algorithms can be ported to the OIA pipeline. We describe several possible operations as follows:

Constant Merging and Expansion. Noun phrases with conjunction/dis-conjunction and preposition involved (such as "the deaths of the security guards and police") may correspond to many nodes in the default OIA graph, which raise the costs of reading and annotation of the OIA graph. We can merge those nodes as one constant node to reduce the cost and expand it back when necessary. Figure 2 shows the merged versions of the OIA graphs.

Nested Facts. Nested fact extraction is a feature of NestIE, which is naturally supported by the OIA graph.

Idiom Discovery. Idioms like "in order to", "as soon ... as", "be proud of" have specific meanings and should be taken as one predicate. One can apply graph pattern mining on a set of OIA graphs and learn the pattern for idioms, or directly use the patterns discovered by previous OIE algorithms such as OLLIE or ClauseIE. Once an idiom is discovered and matched, we merge the relevant nodes to form one single predicate.

Informativeness Improvement. MinIE proposed this strategy to select informative expression of predicates, that is, in favor of (Faust, made a deal with, the Devil) instead of (Faust, made, a deal with the Devil). The informativeness measurement can be ported to OIA, and the target predicate can be obtained by merging relevant nodes.

Factuality. We can extract factuality annotations (negation, certainty/possibility) as in MinIE and add property edges to OIA linking the predicate node to the value node.

Condition and Attribution. The conditional relation considered in OLLIE is naturally supported by the OIA by taking the conditional word as the predicate. Attributions that mark facts by their contexts, such as "Some people say", can be done by examining the nested structure in OIA.

Hidden Information in Nouns. OLLIE, RelNOUN, MinIE and Logician can extract relations hidden in noun phrases. We can apply these algorithms to extract the hidden facts and attach them to the OIA graph for future usage.

Minimization. The minimization strategies proposed by MinIE can be ported as a prune operation on the OIA graph to drop words useless to the current task.

5 Parsing Sentence into OIA Graph

This section introduces the automatic pipeline for parsing sentences in English into OIA graphs, which is illustrated in Figure 4. We first introduce each component of the pipeline, and then evaluate the proposed OIA parser's performance.

5.1 Components of Pipeline

Universal Dependency Parser. The first step is to convert the sentence into Universal Dependency (UD) (Nivre et al., 2020) graph using a Universal Dependency Parser. Among various types of dependencies, we choose the Universal Dependency because 1) UD is designed cross-linguistically, which makes our pipeline potentially possible to port to languages other than English. 2) UD is one of the biggest data sets for dependency grammar. In this paper, we adopt the UD 2.0 standard as the target form of UD graphs and employed the neural network-based StanfordNLP toolkit [2] (Qi et al., 2018) to generate the Universal Dependency graphs for sentences.

Enhanced++ Universal Dependencies. The second step is to convert the original UD graph into an Enhanced++ UD graph. The Enhanced++ Universal Dependencies (Schuster and Manning, 2016) provide richer information about the relationships between the components in sentences, and some of them greatly help the construction of OIA graphs. Since there is no UD 2.x compatible Enhanced++ annotator available (while UD 1.x compatible version is available in the CoreNLP toolkit), we develop a UD 2.x compatible Enhanced++ annotator in Python by ourselves. Our Enhanced++ annotator's accuracy on the set of changed edges of the UD English test data is 95.05%.

OIA Graph Annotator. The OIA Graph annotator works in three steps: 1) Simplifying the UD graph: Identify the simple phrases and merge the relevant word nodes in Enhanced++ UD graph into one node. Conjunction/dis-conjunction relationships are processed by adding an extra predicate node to the graph, connecting to all parallel components as arguments. Thirty-nine heuristic rules are developed to fulfill these procedures. 2) Mapping to the OIA graph: Map the dependencies in the simplified UD graph into the relationship between the OIA nodes, according to the conversion described in Section 3. In total, 37 heuristic rules are involved in this step. 3) Making the DAG: Select the root of the OIA graph (usually the predicate corresponding to the root of the UD graph or a connection word to that root) and then convert the graph to a DAG by reversing conflicting arcs and changing labels as described at Section 3.

5.2 Building the Pipeline and the Data Set

We used the Universal Dependencies project version 2.4 for English data set [3] as the source to build our pipeline. The data set contains about

[2] https://stanfordnlp.github.io/stanfordnlp

[3] http://hdl.handle.net/11234/1-2988

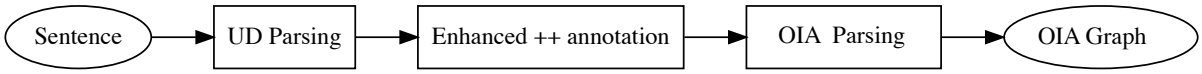


Figure 4: Pipeline to converting sentence into OIA graph

16,000 human-labeled pairs of the sentence and its Enhanced++ UD annotation, split into the train, develop, and test sets. With the existence of the ground-truth UD graph, we can investigate how the UD parser's accuracy influences the accuracy of the OIA pipeline.

We first implemented an initial version of the pipeline and then ran the pipeline over all the samples from the UD training set. All the samples that resulted in parsing errors like unexpected situations, disconnected components, missing words were collected and examined to improve our pipeline. The procedure continued until the pipeline could successfully run through almost all training samples. Then we labeled 100 samples from the development set of the UD data set and a small number of sentences from the UD training set. We tested and improved the pipeline on the labeled training data by examining the detailed correctness and evaluated the performance on the development data set. If there was a large gap between the development performance and train performance, we labeled more data until the gap tended to vanish. (The evaluation metrics are introduced in the next section.)

Finally, 500 sentences from the UD training set were labeled to obtain a converged pipeline. Furthermore, we labeled all (about 2,000) sentences from the UD testing set for performance evaluation. All the data were labeled by two annotators, with each labeling a half and then double-checking another half. We make all our labeled data public on the online website of OIX.

5.3 Evaluation

There are two configurations of OIA pipelines. One uses the ground-truth Enhanced++ UD annotation as input; the other uses the raw sentence as input and uses UD parser and our Enhanced++ annotator to generate the enhanced UD graph.

Evaluation on Generated OIA Graph. We measure how well the predicted OIA graphs match the ground truth OIA graph at three levels: Node Level, Edge Level, and Graph Level. The set of representations is collected at each level, and the precision, recall and F1 scores are evaluated. For node level, the representation is the node label; for edge level, the representation is a triplet <*starting node label*, *edge label*, *end node label*>; for graph level, the representation is the set of all edge triples. At all levels, we find the matched representations by exact match. The results of the pipeline with Enhanced++ input are shown in Table 3, and the results of the pipeline with raw sentence input are shown in Table 4.

Level	Precision	Recall	F1
Node	0.930	0.913	0.921
Edge	0.763	0.764	0.763
Graph	0.565	0.565	0.565

Table 3: Performance of our OIA converter given the ground-truth Enhance++ annotations.

Level	Precision	Recall	F1
Node	0.853	0.871	0.862
Edge	0.629	0.628	0.628
Graph	0.450	0.450	0.450

Table 4: Performance of the OIA pipeline given the raw sentences.

Evaluation on Facts Extracted from OIA. Extracting open-domain facts from an OIA graph is rather straightforward. First, we recover all the short-cut edges back into its original predicate form. Then, for each predicate node, we collect all its arguments and produce the OIE fact tuples. The sets of facts from predicted OIA graphs are compared to those from the ground-truth OIA graphs to compute the evaluation results. Exact match is used in evaluation and the precision, recall and F1 scores are computed as shown in Table 5.

Input	Precision	Recall	F1
UD Graph	0.696	0.708	0.702
Sentence	0.479	0.484	0.481

Table 5: Fact level performances of the OIA pipeline.

5.4 Error Analysis

From the above results, we can see that without the input of ground-truth Enhanced++ annotation, there are a roughly 10% increase in error for the OIA graph and even 20% for facts. The error in dependency parsing and Enhanced++ annotation is the major part of the error for the pipeline without ground-truth Enhanced++ annotation input.

We reviewed the error cases of predicted results with Enhanced++ annotation input and found several major sources of error: 1) the complexity of natural language sentences that our convert rules do not cover, especially in inversion sentences; 2) mistaken or incomplete annotations in Enhanced++ while a human can correctly annotate; 3) the ambiguity of human-labeled OIA samples since various inferences over the graph (see Section 4) are allowed while all preserve the validity.

A possible way to cope with the above errors is to formalize a standardized form of OIA graphs (see online website for details) and learn the mapping from sentence to the standard form in an end-to-end way. Recent advances in neural graph learning (You et al., 2018; Li et al., 2018; Sun and Li, 2019; Rahmani and Li, 2020) are suitable for generating the OIA graphs. Together with the recent advances on pre-trained language model (Devlin et al., 2019; Radford et al., 2019), the results are worth to be expected. These directions could be in the pipeline of our future work.

6 Discussion

Dependency Graph. One may wonder whether it is necessary to propose a new OIX or OIA learning task since the information in OIA can also be expressed by the dependency graph, especially Enhanced ++. However, the above experiments reveal that even with our very carefully written rule system, the error rate is still high. Due to the complexity of the natural language and the error in the dependency pipeline, it is very difficult to improve the rule-based pipeline. On the contrary, based on phrases with much fewer types of edge, the OIA is much simpler than the dependency graph, so end-to-end learning may avoid the error introduced by the dependency parser and obtain better results, which belongs to our future work. Defining the task and building a rule-based pipeline as the baseline is the first step to learn a good OIA annotator.

AMR. Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a symbolic representation of the sentence. Same as our OIA, information lossless is also a principle of AMR. AMR contains approximately 100 relations and selects symbolized concepts from PropBank (Palmer et al., 2005). It is also very abstract that sentences with the same meaning but in very different expressions will share the same AMR annotation. As a result, AMR is difficult to label (cost about 10 min to label a sample[4]) and is very difficult to learn. OIA can be viewed as an open-domain approximation of AMR and maybe a valuable step for AMR learning.

SAOKE. SAOKE(Symbol Aided Open Knowledge Expression) (Sun et al., 2018b) is our previous attempt to express various types of knowledge uniformly. It is designed following four requirements: *Completeness*, *Accurateness*, *Atomicity*, and *Compactness*, which are the predecessors of the principles of OIX. However, due to the limitation of the annotation form (a list of tuples), the expression capability of SAOKE is restricted, while the OIA greatly extends the expression capability. Several end-to-end learning strategies, such as dual learning (Sun et al., 2018a) and reinforcement learning (Sun et al., 2018a; Liu et al., 2020b,a) are developed to learn the SAOKE annotation, which can be ported to the learning of OIA graphs.

7 Conclusions and Future Work

This paper proposes a reusable and adaptive pipeline to construct OIE systems. As the core of the pipeline, the Open-domain Information eXpression (OIX) task is thoroughly studied, and an Open Information Annotation (OIA) is proposed as a solution to the OIX task. We discuss how to port the strategies of various existing OIE algorithms to the OIA graph. We label data for OIA annotation and build a rule-based baseline method to convert sentences into OIA graphs.

There are many potential directions for future work on OIA, including 1) more labeled data; 2) better learning algorithm; 3) becoming cross-lingual by adding support for more natural languages; 4) porting existing OIE strategies on OIA and evaluating the performance compared with the original ones.

[4] https://amr.isi.edu/editor.html

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse (LAW-ID@ACL)*, pages 178–186, Sofia, Bulgaria.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676, Hyderabad, India.

Nikita Bhutani, H. V. Jagadish, and Dragomir R. Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 55–64, Austin, TX.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*, pages 1163–1173, Atlanta, GA.

Janara Christensen, Stephen Soderland, Gagan Bansal, and Mausam. 2014. Hierarchical summarization: Scaling up multi-document summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 902–912, Baltimore, MD.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd International World Wide Web Conference (WWW)*, pages 355–366, Rio de Janeiro, Brazil.

Donald Davidson and Gilbert Harman. 2012. *Semantics of natural language*, volume 40. Springer Science & Business Media.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, Minneapolis, MN.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3–10, Barcelona, Spain.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, Edinburgh, UK.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1156–1165, New York, NY.

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. Minie: Minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2630–2640, Copenhagen, Denmark.

Jeroen Groenendijk and Floris Roelofsen. 2009. Inquisitive semantics and pragmatics.

Jeroen Antonius Gerardus Groenendijk and Martin Johan Bastiaan Stokhof. 1984. *Studies on the Semantics of Questions and the Pragmatics of Answers*. Ph.D. thesis, Univ. Amsterdam.

Charles L Hamblin. 1976. Questions in montague english. In *Montague grammar*, pages 247–259. Elsevier.

Lauri Karttunen. 1977. Syntax and semantics of questions. *Linguistics and philosophy*, 1(1):3–44.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–316, Vancouver, Canada.

Angelika Kratzer and Irene Heim. 1998. *Semantics in generative grammar*, volume 1185. Blackwell Oxford.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Guiliang Liu, Xu Li, Miningming Sun, and Ping Li. 2020a. An advantage actor-critic algorithm with confidence exploration for open information extraction. In *Proceedings of the 2020 SIAM International Conference on Data Mining (SDM)*, pages 217–225.

Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, and Ping Li. 2020b. Large scale semantic indexing with deep level-wise extreme multi-label learning. In *Proceedings of the World Wide Web Conference (WWW)*, pages 2585—-2591, Taipei.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4074–4077, New York, NY.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 523–534, Jeju Island, Korea.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajic, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis M. Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, pages 4034–4043, Marseille, France.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction (AKBC@NAACL-HLT)*, pages 35–39, San Diego, CA.

Martha Palmer, Paul R. Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguistics*, 31(1):71–106.

Radityo Eko Prasojo, Mouna Kacimi, and Werner Nutt. 2018. Stuffie: Semantic tagging of unlabeled facets using fine-grained information extraction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 467–476, Torino, Italy.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (CoNLL)*, pages 160–170, Brussels, Belgium.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Mostafa Rahmani and Ping Li. 2020. The necessity of geometrical representation for deep graph analysis. In *Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM)*.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 303–308, Beijing, China.

Mingming Sun and Ping Li. 2019. Graph to graph: a topology aware approach for graph structures learning and generation. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2946–2955, Naha, Okinawa, Japan.

Mingming Sun, Xu Li, and Ping Li. 2018a. Logician and Orator: Learning from the duality between language and knowledge in open domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2119–2130, Brussels, Belgium.

Mingming Sun, Xu Li, Xin Wang, Miao Fan, Yue Feng, and Ping Li. 2018b. Logician: a unified end-to-end neural approach for open-domain information extraction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, pages 556–564, Marina Del Rey, CA.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 118–127, Uppsala, Sweden.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael J. Cafarella, Oren Etzioni, and Stephen Soderland. 2007. Textrunner: Open information extraction on the web. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT)*, pages 25–26, Rochester, NY.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning, (ICML)*, pages 5694–5703, Stockholmsmässan, Sweden.