ASER: A Large-scale Eventuality Knowledge Graph

Hongming Zhang* HZHANGAL@CSE.UST.HK
Xin Liu* XLIUCR@CSE.UST.HK
Haojie Pan* HPANAD@CSE.UST.HK
Yangqiu Song YQSONG@CSE.UST.HK
CSE, HKUST, Hong Kong

Cane Wing-Ki Leung CANELEUNG@WISERS.COM
Wisers AI Lab, Hong Kong

Abstract

Understanding human's language requires complex world knowledge. However, existing large-scale knowledge graphs mainly focus on knowledge about entities while ignoring knowledge about activities, states, or events, which are used to describe how entities or things act in the real world. To fill this gap, we develop ASER (activities, states, events, and their relations), a large-scale eventuality knowledge graph extracted from more than 11-billion-token unstructured textual data. ASER contains 15 relation types belonging to five categories, 194-million unique eventualities, and 64-million unique edges among them. Both human and extrinsic evaluations demonstrate the quality and effectiveness of ASER.

1. Introduction

In his conceptual semantics theory, Ray Jackendoff, a Rumelhart Prize[1] winner, describes semantic meaning as 'a finite set of mental primitives and a finite set of principles of mental combination (Jackendoff, 1990)'. The primitive units of semantic meanings include *Thing* (or *Object*), *Activity*[2], *State*, *Event*, *Place*, *Path*, *Property*, *Amount*, etc. Understanding the semantics related to the world requires the understanding of these units and their relations. Traditionally, linguists and domain experts built knowledge graphs (KGs)[3] to formalize these units and enumerate categories (or senses) and relations of them. Typical KGs include WordNet (Fellbaum, 1998) for words, FrameNet (Baker, Fillmore, & Lowe, 1998) for events, and Cyc (Lenat & Guha, 1989) and ConceptNet (Liu & Singh, 2004) for commonsense knowledge. However, their small scales restricted their usage in real-world applications.

Nowadays, with the growth of Web contents, computational power, and the availability of crowdsourcing platforms, many modern and large-scale KGs, such as Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), KnowItAll (Etzioni, Cafarella, & Downey, 2004),

∗. Equal contribution.

1. The David E. Rumelhart Prize is funded for contributions to the theoretical foundations of human cognition.
2. In his original book, he called it *Action*. But given the other definitions and terminologies we adopted (P. D. Mourelatos, 1978; Bach, 1986), it means *Activity*.
3. Traditionally, people used the term 'knowledge base' to describe the database containing human knowledge. In 2012, Google released its knowledge graph where vertices and edges in a knowledge base are emphasized. We discuss in the context of the knowledge graph, as our knowledge is also constructed as a complex graph. For more information about terminologies, please refer to (Ehrlinger & Wöß, 2016).

TextRunner (Banko, Cafarella, Soderland, Broadhead, & Etzioni, 2007), YAGO (Suchanek, Kasneci, & Weikum, 2007), DBpedia (Auer, Bizer, Kobilarov, Lehmann, Cyganiak, & Ives, 2007), NELL (Carlson, Betteridge, Kisiel, Settles, Jr., & Mitchell, 2010), Probase (Wu, Li, Wang, & Zhu, 2012), and Google Knowledge Vault (Dong, Gabrilovich, Heitz, Horn, Lao, Murphy, Strohmann, Sun, & Zhang, 2014), have been built based on semi-automatic mechanisms. Most of these KGs are designed and constructed based on *Things* or *Objects*, such as instances and their concepts, named entities and their categories, as well as their properties and relations. On top of them, a lot of semantic understanding problems such as question answering (Berant, Chou, Frostig, & Liang, 2013) can be supported by grounding natural language texts on knowledge graphs, e.g., asking a bot for the nearest restaurants for lunch. Nevertheless, these KGs may fall short in circumstances that require not only knowledge about *Things* or *Objects*, but also those about *Activities*, *States*, and *Events*. Consider the following utterance that a human would talk to the bot: 'I am hungry', which may also imply one's need for restaurant recommendation. This, however, will not be possible unless the bot is able to identify that the consequence of being hungry would be 'having lunch' at noon.

In this paper, we propose an approach to discovering useful real-world knowledge about *Activities* (or process, e.g., 'I sleep'), *States* (e.g., 'I am hungry'), *Events* (e.g., 'I make a call'), and their *Relations* (e.g., 'I am hungry' may result in 'I have lunch'), for which we call ASER. In fact, *Activities*, *States*, and *Events*, which are expressed by verb-related clauses, are all eventualities following the commonly adopted terminology and categorization proposed by Mourelatos (P. D. Mourelatos, 1978) and Bach (Bach, 1986). While both activity and event are occurrences (actions) described by active verbs, a state is usually described by a stative verb and cannot be qualified as actions. The difference between an activity and an event is that an event is defined as an occurrence that is inherently countable (P. D. Mourelatos, 1978). For example, 'The coffee machine brews a cup of coffee once more' is an event because it admits a countable noun 'a cup' and cardinal count adverbials 'once', while 'The coffee machine brews coffee' is not an event with an imperfective aspect and it is not countable. Thus, ASER is essentially an eventuality-centric knowledge graph.

For eventualities, traditional extraction approaches used in natural language processing based on FrameNet (Baker et al., 1998) or ACE (NIST, 2005) first define complex structures of events by enumerating triggers with senses and arguments with roles. They then learn from limited annotated examples and try to generalize to other text contents. However, detecting trigger senses and argument roles suffers from the ambiguity and variability of the semantic meanings of words. For example, using the ACE training data, the current state-of-the-art system can only achieve about 40% overall F1 score with 33 event types (Li, Ji, & Huang, 2013). Different from them, we use patterns to extract eventuality-centric knowledge based on dependency grammar since the English language's syntax is relatively fixed and consistent across domains and topics. Instead of defining complex triggers and role structures of events, we simply use syntactic patterns to extract all possible eventualities. We do not distinguish between semantic senses or categories of particular triggers or arguments in eventualities but treat all extracted words with their dependency relations as hyperedge in a graph to define an eventuality as a primitive semantic unit in our knowledge graph.

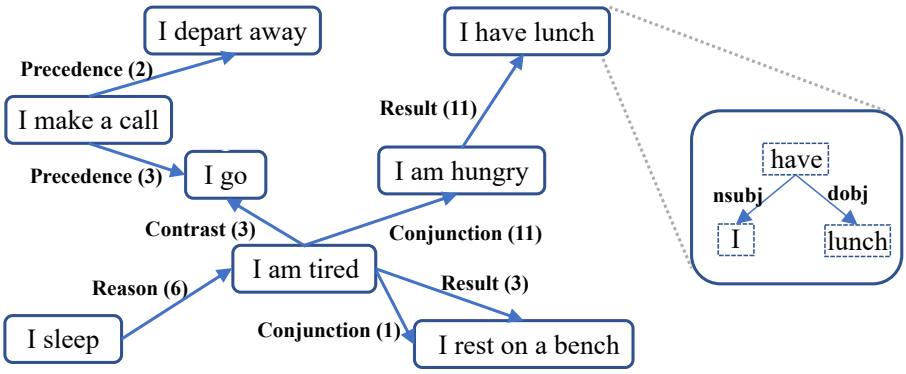


Figure 1: ASER Demonstration. Eventualities are connected with weighted directed edges. Each eventuality is a dependency graph.

For eventuality relations, we use the definition linguistic shallow discourse relations used in Penn Discourse Treebank (PDTB) (Prasad, Miltsakaki, Dinesh, Lee, Joshi, Robaldo, & Webber, 2007). In PDTB, the relations are defined between two sentences or clauses. Simplified from PDTB, we focus on relations between two eventualities, which are defined with simple but semantically complete patterns. Moreover, as shown in PDTB, some connectives, e.g., 'and' and 'but', are less ambiguous than others, e.g., 'while'. Thus, we use less ambiguous connectives as seed connectives to find initial relations and then bootstrap the eventuality relation extraction using large corpora. Although relations are extracted based on linguistic knowledge, we will show that they have correlations with previously defined commonsense knowledge in ConceptNet (Liu & Singh, 2004).

In ASER, we have extracted 194 million unique eventualities. After bootstrapping, ASER contains 64 million edges among eventualities. One example of ASER is shown in Figure 1. Table 1 provides a size comparison between ASER and existing eventuality-related (or simply verb-centric) knowledge bases. Essentially, they are not large enough as modern knowledges graph and inadequate for capturing the richness and complexity of eventualities and their relations. FrameNet (Baker et al., 1998) is considered the earliest knowledge base defining events and their relations. It provides annotations about relations among about 1,000 human defined eventuality frames, which contain 27,691 eventualities. However, given the fine-grained definition of frames, the scale of the annotations is limited. ACE (NIST, 2005) (and its follow-up evaluation TAC-KBP (Aguilar, Beller, McNamee, Van Durme, Strassel, Song, & Ellis, 2014)) reduces the number of event types and annotates more examples in each of event types. PropBank (Palmer, Gildea, & Kingsbury, 2005) and NomBank (Meyers, Reeves, Macleod, Szekely, Zielinska, Young, & Grishman, 2004) build frames over syntactic parse trees, and focus on annotating popular verbs and nouns. TimeBank focuses only on temporal relations between verbs (Pustejovsky, Hanks, Sauri, See, Gaizauskas, Setzer, Radev, Sundheim, Day, Ferro, et al., 2003). While the aforementioned knowledge bases are annotated by domain experts, ConceptNet[4] (Liu &

4. Following the original definition, we only select the four relations ('HasPrerequisite', 'HasFirstSubevent', 'HasSubEvent', and 'HasLastSubEvent') that involve eventualities.

Table 1: Size comparison of ASER and existing eventuality-related resources. # Eventuality, # Relation, and # R types are the number of eventualities, relations between these eventualities, and relation types. For KGs containing knowledge about both entity and eventualities, we report the statistics about the eventualities subset. ASER (core) filters out eventualities that appear only once and thus has better accuracy while ASER (full) can cover more knowledge.

	# Eventuality	# Relation	# R Types
FrameNet (Baker et al., 1998)	27,691	1,709	7
ACE (Aguilar et al., 2014)	3,290	0	0
PropBank (Palmer et al., 2005)	112,917	0	0
NomBank (Meyers et al., 2004)	114,576	0	0
TimeBank (Pustejovsky et al., 2003)	7,571	8,242	1
ConceptNet (Liu & Singh, 2004)	74,989	116,097	4
Event2Mind (Smith et al., 2018)	24,716	57,097	3
ProPora (Dalvi et al., 2018)	2,406	16,269	1
ATOMIC (Sap et al., 2018)	309,515	877,108	9
Knowlywood (Tandon, de Melo, De, & Weikum, 2015)	964,758	2,644,415	4
ASER (core)	27,565,673	10,361,178	15
ASER (full)	194,000,677	64,351,959	15

Singh, 2004), Event2Mind (Smith, Choi, Sap, Rashkin, & Allaway, 2018), ProPora (Dalvi, Huang, Tandon, tau Yih, & Clark, 2018), and ATOMIC (Sap, LeBras, Allaway, Bhagavatula, Lourie, Rashkin, Roof, Smith, & Choi, 2018) leveraged crowdsourcing platforms or the general public to annotate commonsense knowledge about eventualities, in particular the relations among them. Furthermore, KnowlyWood uses semantic parsing to extract activities (verb+object) from movie/TV scenes and novels to build four types of relations (parent, previous, next, similarity) between activities using inference rules. Compared with all these eventuality-related KGs, ASER is larger by one or more orders of magnitude in terms of the numbers of eventualities[5] and relations it contains.

In summary, our contributions are as follows.

• **Definition of ASER.** We define a brand new KG where the primitive units of semantics are eventualities. We organize our KG as a relational graph of hyperedges. Each eventuality instance is a hyperedge connecting several vertices, which are words. A relation between two eventualities in our KG represents one of the 14 relation types defined in PDTB (Prasad et al., 2007) or a co-occurrence relation.

• **Scalable Extraction of ASER.** We perform eventuality extraction over large-scale corpora. We designed several high-quality patterns based on dependency parsing results and extract all eventualities that match these patterns. We use unambiguous connectives

5. Some of the eventualities are not connected with others, but the frequency of an eventuality is also valuable for downstream tasks. One example is the coreference resolution task. Given one sentence 'The dog is chasing the cat, it barks loudly', we can correctly resolve 'it' to 'dog' rather than 'cat' because 'dog barks' appears 12,247 times in ASER, while 'cat barks' never appears. This is usually called selectional preference (Wilks, 1975), which has recently been evaluated in a larger scale in (Zhang, Ding, & Song, 2019). ASER naturally reflects human's selectional preference for many kinds of syntactic patterns.

obtained from PDTB to find seed relations among eventualities. Then we leverage a neural bootstrapping framework to extract more relations from the unstructured textual data.

• **Inference over ASER.** We also provide several ways of inference over ASER. We show that both eventuality and relation retrieval over one-hop or multi-hop relations can be modeled as conditional probability inference problems.

• **Evaluation and Applications of ASER.** We conduct both intrinsic and extrinsic evaluations to validate the quality and effectiveness of ASER. For intrinsic evaluation, we sample instances of extracted knowledge in ASER over iterations, and submitted them to the Amazon Mechanical Turk (AMT) for human workers to verify. We also study the correlation of knowledge in ASER and the widely accepted commonsense knowledge in ConceptNet (Liu & Singh, 2004). For extrinsic evaluation, we use the Winograd Schema Challenge (Levesque, Davis, & Morgenstern, 2011) to test whether ASER can effectively address the language understanding problem and a dialogue generation task to demonstrate the effect of using ASER for the language generation problem. The results of both evaluations show that ASER is a promising large-scale KG with great potentials. The proposed ASER and supporting packages are available at: `https://github.com/HKUST-KnowComp/ASER`.

2. Overview of ASER

Each eventuality in ASER is represented by a set of words, where the number of words varies from one eventuality to another. Thus, we cannot use a traditional graph representation such as triplets to represent knowledge in ASER. We devise the formal definition of our ASER KG as below.

Definition 1 ASER KG *is a hybrid graph $\mathcal{H}$ of eventualities E's. Each* **eventuality** *E is a hyperedge linking to a set of vertices v's. Each vertex v is a* **word** *in the vocabulary. We define $v \in \mathcal{V}$ in the vertex set and $E \in \mathcal{E}$ in the hyperedge set. $\mathcal{E} \subseteq \mathcal{P}(\mathcal{V}) \setminus \{\emptyset\}$ is a subset of the power set of $\mathcal{V}$. We also define a* **relation** *$R_{i,j} \in \mathcal{R}$ between two eventualities E_i and E_j, where $\mathcal{R}$ is the relation set. Each relation has a* **type** *$T \in \mathcal{T}$ where $\mathcal{T}$ is the type set. Overall, we have ASER KG $\mathcal{H} = \{\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{T}\}$.*

ASER KG is a hybrid graph combining a hypergraph $\{\mathcal{V}, \mathcal{E}\}$ where each hyperedge is constructed over vertices, and a traditional graph $\{\mathcal{E}, \mathcal{R}\}$ where each edge is built among eventualities. For example, E_1=`(I, am, hungry)` and E_2=`(I, eat, anything)` are eventualities, where we omit the internal dependency structures for brevity. They have a relation $R_{1,2}$=`Result`, where `Result` is the relation type.

2.1 Eventuality

Different from named entities or concepts, which are noun phrases, eventualities are usually expressed as verb phrases, which are more complicated in structure. Our definition of eventualities is built upon the following two assumptions: (1) syntactic patterns of English are relatively fixed and consistent; (2) the eventuality's semantic meaning is determined by the words it contains. To avoid the extracted eventualities being too sparse, we use words fitting certain patterns rather than a whole sentence to represent an eventuality. In addition, to make sure the extracted eventualities have complete semantics, we retain all

Table 2: Selected eventuality patterns ('v' stands for normal verbs other than 'be', 'be' stands for 'be' verbs, 'n' stands for nouns, 'a' stands for adjectives, and 'p' stands for prepositions.), Code (to save space, we create a unique code for each pattern and will use that in the rest of this paper), and the corresponding examples.

Pattern	Code	Example
n_1-nsubj-v_1	s-v	'The dog barks'
n_1-nsubj-v_1-dobj-n_2	s-v-o	'I love you'
n_1-nsubj-v_1-xcomp-a	s-v-a	'He felt ill'
n_1-nsubj-(v_1-iobj-n_2)-dobj-n_3	s-v-o-o	'You give me the book'
n_1-nsubj-a_1-cop-be	s-be-a	'The dog is cute'
n_1-nsubj-v_1-xcomp-a_1-cop-be	s-v-be-a	'I want to be slim'
n_1-nsubj-v_1-xcomp-n_2-cop-be	s-v-be-o	'I want to be a hero'
n_1-nsubj-v_1-xcomp-v_2-dobj-n_2	s-v-v-o	'I want to eat the apple'
n_1-nsubj-v_1-xcomp-v_2	s-v-v	'I want to go'
(n_1-nsubj-a_1-cop-be)-nmod-n_2-case-p_1	s-be-a-p-o	'It' cheap for the quality'
n_1-nsubj-v_1-nmod-n_2-case-p_1	s-v-p-o	'He walks into the room'
(n_1-nsubj-v_1-dobj-n_2)-nmod-n_3-case-p_1	s-v-o-p-o	'He plays football with me'
n_1-nsubjpass-v_1	spass-v	'The bill is paid'
n_1-nsubjpass-v_1-nmod-n_2-case-p_1	spass-v-p-o	'The bill is paid by me'

necessary words extracted by patterns rather than those simple verbs or verb-object pairs in sentences. The selected patterns are shown in Table 2. For example, for the eventuality `(dog, bark)`, we have a relation `nsubj` between the two words to indicate that there is a subject-of-a-verb relation in between. We now formally define an eventuality as follows.

Definition 2 *An eventuality E_i is a hyperedge linking multiple words $\{v_{i,1}, \ldots, v_{i,N_i}\}$, where N_i is the number of words in eventuality E_i. Here, $v_{i,1}, \ldots, v_{i,N_i} \in \mathcal{V}$ are all in the vocabulary. A pair of words in E_i $(v_{i,j}, v_{i,k})$ may follow a syntactic relation $e_{i,j,k}$.*

We use patterns from dependency parsing to extract eventualities E's from unstructured large-scale corpora. Here $e_{i,j,k}$ is one of the relations that dependency parsing may return. Although in this way the recall is sacrificed, our patterns are of high precision and we use very large corpora to extract as many eventualities as possible. This strategy is also shared with many other modern KGs (Etzioni et al., 2004; Banko et al., 2007; Carlson et al., 2010; Wu et al., 2012).

2.2 Eventuality Relation

For relations among eventualities, as introduced in Section 1, we follow PDTB's (Prasad et al., 2007) definition of relations between sentences or clauses but simplify them to eventualities. Following the CoNLL 2015 discourse parsing shared task (Xue, Ng, Pradhan, Prasad, Bryant, & Rutherford, 2015), we select 14 discourse relation types and an additional co-occurrence relation to build our knowledge graph.

Table 3: Eventuality relation types between two eventualities E_1 and E_2 and explanations.

Relation	Explanation
$<E_1$, 'Precedence', $E_2>$	E_1 happens before E_2.
$<E_1$, 'Succession', $E_2>$	E_1 happens after E_2.
$<E_1$, 'Synchronous', $E_2>$	E_1 happens at the same time as E_2.
$<E_1$, 'Reason', $E_2>$	E_1 happens because E_2 happens.
$<E_1$, 'Result', $E_2>$	If E_1 happens, it will result in the happening of E_2.
$<E_1$, 'Condition', $E_2>$	Only when E_2 happens, E_1 can happen.
$<E_1$, 'Contrast', $E_2>$	E_1 and E_2 share a predicate or property and have significant difference on that property.
$<E_1$, 'Concession', $E_2>$	E_1 should result in the happening of E_3, but E_2 indicates the opposite of E_3 happens.
$<E_1$, 'Conjunction', $E_2>$	E_1 and E_2 both happen.
$<E_1$, 'Instantiation, $E_2>$	E_2 is a more detailed description of E_1.
$<E_1$, 'Restatement', $E_2>$	E_2 restates the semantics meaning of E_1.
$<E_1$, 'Alternative', $E_2>$	E_1 and E_2 are alternative situations of each other.
$<E_1$, 'ChosenAlternative', $E_2>$	E_1 and E_2 are alternative situations of each other, but the subject prefers E_1.
$<E_1$, 'Exception', $E_2>$	E_2 is an exception of E_1.
$<E_1$, 'Co-Occurrence', $E_2>$	E_1 and E_2 appear in the same sentence.

Definition 3 *A relation R between a pair of eventualities E_1 and E_2 has one of the following types $T \in \mathcal{T}$ and all types can be grouped into five categories:* **Temporal** *(including Precedence, Succession, and Synchronous),* **Contingency** *(including Reason, Result, and Condition),* **Comparison** *(including Contrast and Concession),* **Expansion** *(including Conjunction, Instantiation, Restatement, Alternative, ChosenAlternative, and Exception), and* **Co-Occurrence**. *The detailed definitions of these relation types are shown in Table 3. The weight of R is defined by the number of tuple $<E_1$, R, $E_2>$ appears in the whole corpora.*

2.3 KG Storage

All eventualities in ASER are small-dependency graphs, where vertices are the words and edges are the internal dependency relations between these words. We store the information about eventualities and relations among them separately in two tables with a SQL database. In the eventuality table, we record information about event ids, all the words, dependencies edges between words, and frequencies. In the relation table, we record ids of head and tail eventualities and relations between them.

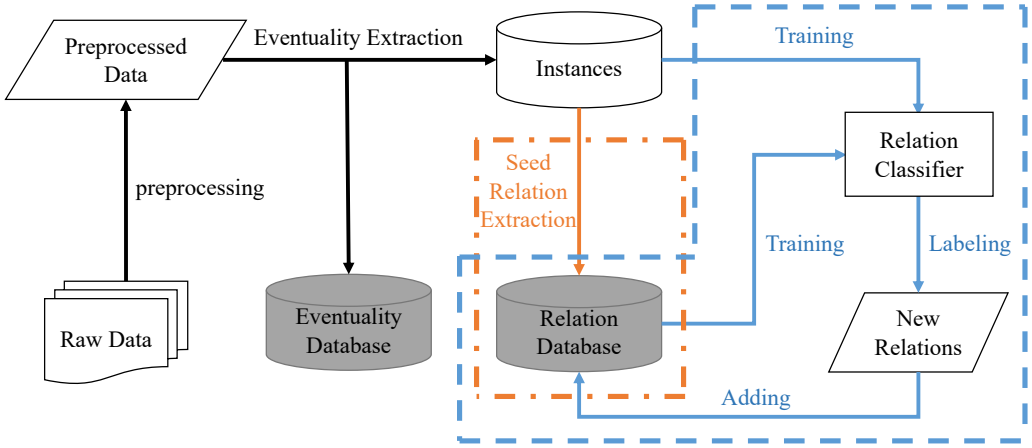


Figure 2: ASER extraction framework. The seed relation selection and the bootstrapping process are shown in the orange dash-dotted and blue dashed box respectively. Two gray databases are the resulted ASER.

3. Knowledge Extraction

In this section, we introduce the knowledge extraction methodologies for building ASER.

3.1 System Overview

We first introduce the overall framework of our knowledge extraction system. The framework is shown in Figure 2. After textual data collection, we first preprocess the texts with the dependency parser. Then we perform eventuality extracting using pattern matching. For each sentence, if we find more than two eventualities, we first group these eventualities into pairs. For each pair, we generate one training instance, where each training instance contains two eventualities and their original sentence. After that, we extract seed relations from these training instances based on the less ambiguous connectives obtained from PDTB (Prasad et al., 2007). Finally, a bootstrapping process is conducted to learn more relations and train the new classifier repeatedly. In the following sub-sections, we will introduce each part of the system separately.

3.2 Corpora

To make sure the broad coverage of ASER, we select corpora from different resources (reviews, news, forums, social media, movie subtitles, e-books) as the raw data. The details of these datasets are as follows.

• Yelp: Yelp is a social media platform where users can write reviews for businesses, e.g., restaurants, hotels. The latest release of the Yelp dataset[6] contains over five million reviews.

6. https://www.yelp.com/dataset/challenge

Table 4: Statistics of used corpora. (M means millions.)

Name	# Sentences	# Tokens	# Instances	# Unique Eventualities
Yelp	48.9M	758.3M	54.2M	20.5M
NYT	56.8M	1,196.9M	41.6M	23.9M
Wiki	105.1M	2,347.3M	38.9M	38.4M
Reddit	235.9M	3,373.2M	185.7M	82.6M
Subtitles	445.0M	3,164.1M	137.6M	27.0M
E-books	27.6M	618.6M	22.1M	11.1M
Overall	919.2M	11,458.4M	480.1M	194.0M

• New York Times (NYT): The NYT (Sandhaus & Evan, 2008) corpus contains over 1.8 million news articles from the NYT throughout 20 years (1987 - 2007).

• Wiki: Wikipedia is one of the largest free knowledge dataset. To build ASER, we select the English version of Wikipedia[7].

• Reddit: Reddit is one of the largest online forums. In this work, we select the anonymized post records[8] over one period month.

• Movie Subtitles: The movie subtitles corpus was collected by (Lison & Tiedemann, 2016) and we select the English subset, which contains subtitles for more than 310K movies.

• E-books: The last resource we include is the free English electronic books from Project Gutenberg[9].

We merge these resources as a whole to perform knowledge extraction. The statistics of different corpora are shown in Table 4.

3.3 Preprocessing and Eventuality Extraction

For each sentence s, we first parse it with the Stanford Dependency Parser[10]. We then filter out all the sentences that contain clauses. As each sentence may contain multiple eventualities and verbs are the centers of them, we first extract all verbs. To make sure that all the extracted eventualities are semantically complete without being too complicated, we design 14 patterns to extract the eventualities via pattern matching. Each of the patterns contains three kinds of dependency edges: positive dependency edges, optional dependency edges, and negative dependency edges. All the positives edges are shown in Table 2. Six more dependency relations (`advmod`, `amod`, `nummod`, `aux`, `compound`, and `neg`) are optional dependency edges that can associate with any of the selected patterns. We omit all optional edges in the table because they are the same for all patterns. All other dependency edges are considered are negative dependency edges, which are designed to make sure all the extracted eventualities are semantically complete and all the patterns are exclusive with each other. Take sentence 'I have a book' as an example, we will only select <'I', 'have', 'book'> rather than <'I', 'have'> as the valid eventuality, because 'have'-dobj-'book' is a

7. `https://dumps.wikimedia.org/enwiki/`
8. `https://www.reddit.com/r/datasets/comments/3bxlg7`
9. `https://www.gutenberg.org/`
10. `https://nlp.stanford.edu/software/stanford-dependencies.html`

Algorithm 1 Eventuality Extraction with One Pattern P_i

INPUT: Parsed dependency graph D, center verb v. Positive dependency edges P_i^p, optional edges P_i^o, and negative edges P_i^n. **OUTPUT:** Extracted eventuality E.

```
Initialize eventuality E.
for Each connection d (a relation and the associated word) in positive dependency edges P_i^p do
    if Find d in D then
        Append d in E.
    else
        Return NULL.
    end if
end for
for Each connection d in optional dependency edges P_i^o do
    if Find d in D then
        Append d in E.
    end if
end for
for Each connection d in negative dependency edges P_i^n do
    if Find d in D then
        Return NULL.
    end if
end for
Return E
```

negative dependency edge for pattern 's-v'. For each verb v and each pattern, we first put it in the position of v_1 and then try to find all the positive dependency edges. If we can find all the positive dependency edges around the center verb we consider it as one potential valid eventuality and then add all the words connected via those optional dependency edges. In the end, we will check if any negative dependency edge can be found in the dependency graph. If not, we will keep it as one valid eventuality. Otherwise, we will disqualify it. The pseudo-code of our extraction algorithm is shown in Algorithm 1. The time complexity of eventuality extraction is $\mathcal{O}(|S| \cdot |D| \cdot |v|)$ where $|S|$ is the number of sentences, $|D|$ is the average number of dependency edges in a dependency parse tree, and $|v|$ is the average number of verbs in a sentence.

3.4 Eventuality Relation Extraction

For each training instance, we use a two-step approach to decide the relations between the two eventualities.

We first extract seed relations from the corpora by using the unambiguous connectives obtained from PDTB (Prasad et al., 2007). According to PDTB's annotation manual, we found that some of the connectives are more unambiguous than the others. For example, in the PDTB annotations, the connective 'so that' is annotated 31 times and is only with the `Result` relation. On the other hand, the connective 'while' is annotated as `Conjunction` 39

Table 5: Selected seed connectives. Here relations are directed relation from E_1 to E_2. Each relation can have multiple seed connectives, where the corresponding connectives are highlighted as boldface.

Relation Type	Seed Patterns
Precedence	E_1 **before** E_2; E_1 **, then** E_2; E_1 **till** E_2; E_1 **until** E_2
Succession	E_1 **after** E_2; E_1 **once** E_2
Synchronous	E_1, **meanwhile** E_2; E_1 **meantime** E_2; E_1, **at the same time** E_2
Reason	E_1, **because** E_2
Result	E_1, **so** E_2; E_1, **thus** E_2; E_1, **therefore** E_2; E_1, **so that** E_2
Condition	E_1, **if** E_2; E_1, **as long as** E_2
Contrast	E_1, **but** E_2; E_1, **however** E_2; E_1, **, by contrast** E_2; E_1, **, in contrast** E_2; E_1, **, on the other hand,** E_2; E_1, **, on the contrary,** E_2
Concession	E_1, **although** E_2
Conjunction	E_1 **and** E_2; E_1, **also** E_2;
Instantiation	E_1, **for example** E_2; E_1, **for instance** E_2
Restatement	E_1, **in other words** E_2
Alternative	E_1 **or** E_2; E_1, **unless** E_2; E_1, **as an alternative** E_2; E_1, **otherwise** E_2
ChosenAlternative	E_1, E_2 **instead**
Exception	E_1, **except** E_2

times, `Contrast` 111 times, `expectation` 79 times, and `Concession` 85 times, etc. When we identify connectives like 'while', we can not determine the relation between the two eventualities related to it. Thus, we choose connectives that are less ambiguous, where more than 90% annotations of each are indicating the same relation, to extract seed relations. The selected connectives are listed in Table 5. Formally, we denote one informative connective word(s) and its corresponding relation type as c and T. Given a training instance x=(E_1, E_2, s), if we can find a connective c such that E_1 and E_2 are connected by c according to the dependency parse, we will select this instance as an instance for relation type T.

Since the seed relations extracted with selected connectives can only cover the limited number of the knowledge, we use a bootstrapping framework to incrementally extract more eventuality relations. Bootstrapping (Agichtein & Gravano, 2000) is a commonly used technique in information extraction. Here we use a neural network based approach to bootstrap. The general steps of bootstrapping are as follows.

• Step 1: Use the extracted seed training instances as the initial labeled training instances.

• Step 2: Train a classifier based on labeled training instances.

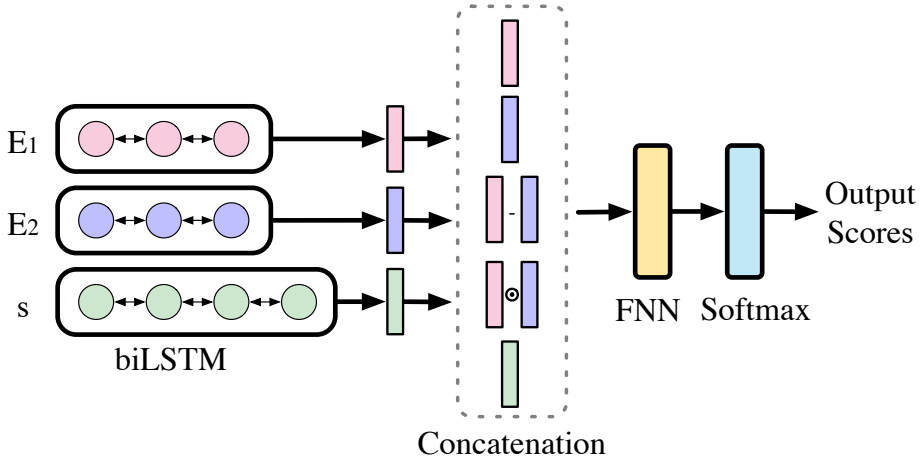


Figure 3: The overview of the neural classifier. For each instance $x = (E_1, E_2, s)$, we first encode the information of two eventualities E_1, E_2 and the original sentence s with three bidirectional LSTMs (Hochreiter & Schmidhuber, 1997) module and the output representations are h_{E_1}, h_{E_2} and h_s respectively. We then concatenate h_{E_1}, h_{E_2}, $h_{E_1} - h_{E_2}$, $h_{E_1} \circ h_{E_2}$ and h_s together, where $\circ$ indicates the element-wise multiplication, and feed them to a two-layer feed forward network. In the end, we use a softmax function to generate scores for different relation types.

• Step 3: Use the classifier to predict relations of each training instance. If the prediction confidence of certain relation type T is higher than the selected threshold, we will label this instance with T and add it to the labeled training instances. Then go to Step 2.

The neural classifier architecture is shown in Figure 3. In the training process, we randomly select labelled training instances as the positive examples and unlabelled training instances as negative examples. The cross-entropy is used as the loss and the whole model is updated via Adam (Kingma & Ba, 2015). In the labeling process, for each training instance x, the classifier can predict a score for each relation type. For any relation type, if the output score is larger than a threshold τ_k, where k is the number of bootstrapping iteration, we will label x with that relation type. To avoid error accumulation, we also use the annealing strategy to increase the threshold $\tau_k = \tau_0 + (1 - \tau_0)/(1 + \exp(-(k - K/2)))$, where K is the total iteration number. The complexities of both training and labeling processes in k^{th} iteration are linear to the number of parameters in LSTM cell $|L|$, the number of training examples $|I_{train_k}|$, and the number of instances to predict $|I_{predict_k}|$ in k^{th} iteration. So the overall complexity in k^{th} iteration is $\mathcal{O}(|L| \cdot (|I_{train_k}| + |I_{predict_k}|))$.

Used hyper-parameters and other implementation details are as follows: For preprocessing, we first parse all the raw corpora with the Stanford Dependency parser, which costs eight days with two 12-core Intel Xeon Gold 5118 CPUs. After that, We extract eventualities, build the training instance set, and extract seed relations, which costs two days with the same CPUs. For bootstrapping, Adam optimizer (Kingma & Ba, 2015) is used and the initial learning rate is 0.001. The batch size is 512. We use GloVe as the pre-trained word embeddings. The dropout rate is 0.2 to prevent overfitting. The hidden sizes of LSTMs

are 256 and the hidden size of the two-layer feed forward network with ReLU is 512. As relation types belonging to different categories could both exist in one training instance, in each bootstrapping iteration, four different classifiers are trained corresponding to four categories (**Temporal**, **Contingency**, **Comparison**, **Temporal**). Each classifier predicts the types belong to that category or 'None' of each instance. Therefore, classifiers do not influence each other so that they can be processed in parallel. Each iteration using ASER (core) takes around one hour with the same CPUs and four TITAN X GPUs. We spend around eight hours predicting ASER (full) with the learned classifier in the 10th iteration.

4. Inference over ASER

In this section, we provide two kinds of inferences (eventuality retrieval and relation retrieval) based on ASER. For each of them, inferences over both one-hop and multi-hops are provided. Complexities of these two retrieval algorithms are both $\mathcal{O}(A^k)$, where A is the number of average adjacent eventualities per eventuality and k is the number of hops. In this section, we show how to conduct these inferences over one-hop and two-hop as the demonstration.

4.1 Eventuality Retrieval

The eventuality retrieval inference is defined as follows. Given a head eventuality[11] E_h and a relation list $\mathcal{L} = (R_1, R_2, ..., R_k)$, find related eventualities and their associated probabilities such that for each eventuality E_t we can find a path, which contains all the relations in $\mathcal{L}$ in order from E_h to E_t.

4.1.1 One-hop Inference

For the one-hop inference, we assume the target relation is R_1. We then define the probability of any potential tail eventuality E_t as:

$$P(E_t|E_h, R_1) = \frac{f(E_h, R_1, E_t)}{\sum_{E'_t, s.t., (E_t, R_1) \in ASER} f(E_h, R_1, E'_t)}, \tag{1}$$

where $f(E_h, R_1, E_t)$ is the relation weight, which is defined in Definition 3. If no eventuality is connected with E_h via R_1, $P(E'|E_h, R)$ will be 0 for any $E' \in \mathcal{E}$.

4.1.2 Two-hop Inference

On top of Eq. (1), it is easy for us to define the probability of E_t on two-hop setting. Assume the two relations are R_1 and R_2 in order. We can define the probability as follows:

$$P(E_t|E_h, R_1, R_2) = \sum_{E_m \in \mathcal{E}_m} P(E_m|E_h, R_1) P(E_t|E_m, R_2), \tag{2}$$

where $\mathcal{E}_m$ is the set of intermediate eventuality E_m such that (E_h, R_1, E_m) and $(E_m, R_2, E_t) \in ASER$.

11. ASER also supports the prediction of head eventualities given tail eventuality and relations. We omit it in this section for the clear presentation.

	Eventuality Retrieval	Relation Retrieval
One-hop	P('I have lunch' | 'I am hungry', *Result*) = 1	P(*Result* | 'I am hungry', 'I have lunch') = 1
	P('I go' | 'I make a call', *Precedence*) = 0.6	P(*Result* | 'I am tired', 'I rest on a bench') = 0.75
	P('I depart away' | 'I make a call', *Precedence*) = 0.4	P(*Conjunction* | 'I am tired', 'I rest on a bench') = 0.25
Two-hop	P(' I rest one a bench' | 'I sleep', *Reason, Result*) = 1	P(*Reason*, *Conjunction* | 'I sleep', 'I am hungry') = 1
	P('I am hungry' | 'I sleep', *Reason, Conjunction*) = 0.91	P(*Reason*, *Result* | 'I sleep', 'I rest on a bench') = 0.75
	P('I rest one a bench' | 'I sleep', *Reason, Conjunction*) = 0.09	P(*Reason*, *Conjunction* | 'I sleep', 'I rest on a bench') = 0.25

Figure 4: Examples of inference over ASER.

4.2 Relation Retrieval

The relation retrieval inference is defined as follows. Given two eventualities E_h and E_t, find all relation lists and their probabilities such that for each relation list $\mathcal{L} = (R_1, R_2, ..., R_k)$, we can find a path from E_h to E_t, which contains all the relations in $\mathcal{L}$ in order.

4.2.1 One-hop Inference

Assuming that the path length is one, we define the probability of one relation R exist from E_h to E_t as:

$$P(R|E_h, E_t) = \frac{f(E_h, R, E_t)}{\sum_{R' \in \mathcal{R}} f(E_h, R', E_t)}, \tag{3}$$

where $\mathcal{R}$ is the relation set.

4.2.2 Two-hop Inference

Similarly, given two eventualities E_h and E_t, we define the probability of a two-hop connection (R_1, R_2) between them as follows:

$$\begin{aligned} P(R_1, R_2|E_h, E_t) &= \sum_{E_m \in \mathcal{E}_m} P(R_1, R_2, E_m|E_h, E_t) \\ &= \sum_{E_m \in \mathcal{E}_m} P(R_1|E_h) P(E_m|R_1, E_h) P(R_2|E_m, E_t), \end{aligned} \tag{4}$$

where $P(R|E_h)$ is the probability of relation R, given head eventuality E_h, and is defined as follows:

$$P(R|E_h) = \frac{\sum_{E_t, s.t., (E_t, R) \in ASER} f(E_h, R, E_t)}{\sum_{R' \in \mathcal{R}} \sum_{E_t, s.t., (E_t, R) \in ASER} f(E_h, R', E_t)}. \tag{5}$$

4.3 Case Study

In this section, we showcase several interesting inference examples with ASER in Figure 4, which is conducted over the extracted sub-graph of ASER shown in Figure 1. By doing

Table 6: Statistics and annotations of the eventuality extraction. # Eventuality and # Unique means the total number and the unique number of extracted eventualities using corresponding patterns ('M' stands for millions). # Agreed means the number of agreed eventualities among five annotators. # Valid means the number valid eventualities labeled by annotators. Accuracy=# Valid/# Agrees. The Overall accuracy is calculated based on the pattern distribution.

Pattern Code	# Eventuality	# Unique	# Agreed	# Valid	Accuracy
s-v	109.0M	22.1M	171	158	92.4%
s-v-o	129.0M	60.0M	181	173	95.6%
s-v-a	5.2M	2.1M	195	192	98.5%
s-v-o-o	3.5M	1.7M	194	187	96.4%
s-be-a	89.9M	29.0M	189	188	99.5%
s-v-be-a	1.2M	0.5M	190	187	98.4%
s-v-be-o	1.2M	0.7M	186	171	91.9%
s-v-v-o	12.4M	6.6M	193	185	95.9%
s-v-v	8.7M	2.7M	185	155	83.8%
s-be-a-p-o	13.2M	8.7M	189	185	97.9%
s-v-p-o	39.0M	23.5M	178	161	90.4%
s-v-o-p-o	27.2M	19.7M	181	167	92.2%
spass-v	15.1M	6.2M	177	155	87.6%
spass-v-p-o	13.5M	10.3M	188	177	94.1%
Overall	468.1M	194.0M	—	—	94.5%

inference over eventuality retrieval, we can easily find out that 'I am hungry' usually results in having lunch and the eventuality 'I make a call' often happens before someone goes or departs. More interestingly, leveraging the two-hop inference, given the eventuality 'I sleep', we can find out an eventuality 'I rest on a bench' such that both of them are caused by the same reason, which is 'I am tired' in this example. From another angle, we can also retrieve possible relations between eventualities. For example, we can know that 'I am hungry' is most likely the reason for 'I have lunch' rather than the other way around. Similarly, over the 2-hop inference, we can find out that even though 'I am hungry' has no direct relation with 'I sleep', 'I am hungry' often appears at the same time with 'I am tired', which is one plausible reason for 'I sleep'.

5. Intrinsic Evaluation

In this section, we present intrinsic evaluation to assess the quantity and quality of extracted eventualities.

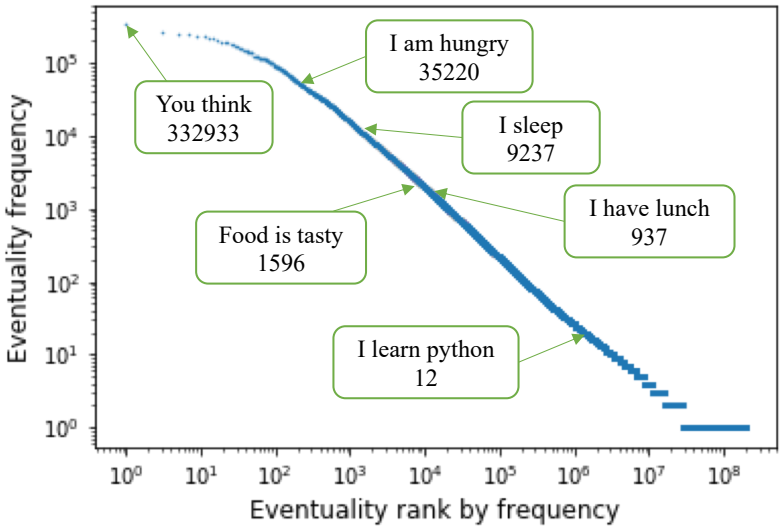


Figure 5: Distribution of eventualities by their frequencies. Sampled eventualities are shown along with their frequencies.

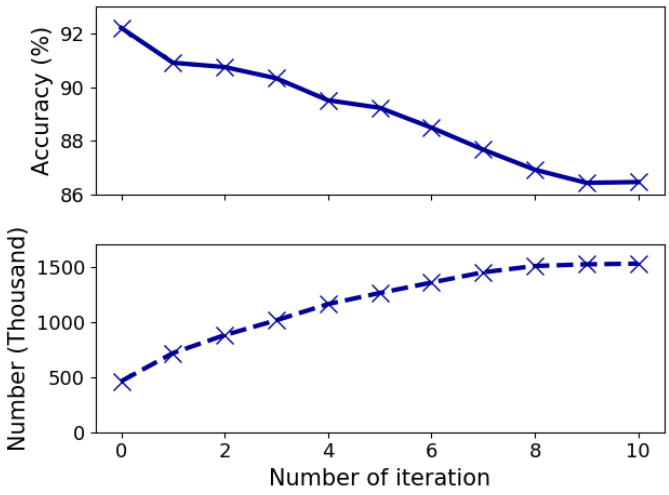
5.1 Eventualities Extraction

We first present the statistics of the extracted eventualities in Table 6, which shows that simpler patterns like 's-v-o' appear more frequently than the complicated patterns like 's-v-be-a'.

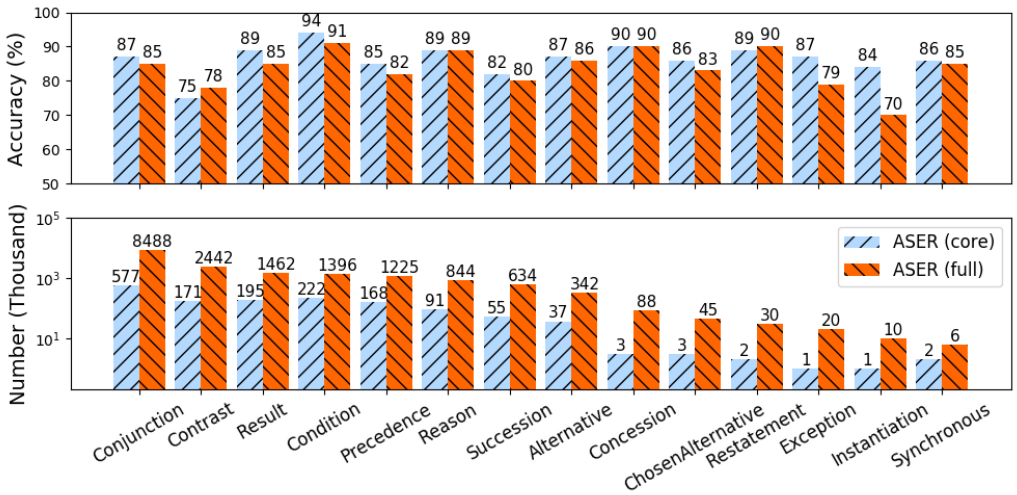
The distribution of extracted eventualities is shown in Figure 5. In general, the distribution of eventualities follows the Zipf's law, where only a few eventualities appear many times while the majority of eventualities appear only a few times. To better illustrate the distribution of eventualities, we also show several representative eventualities along with their frequencies and we have two observations. First, eventualities which can be used in general cases, like 'You think', appear much more times than other eventualities. Second, eventualities contained in ASER are more related to our daily life like 'Food is tasty' or 'I sleep' rather than domain-specific ones such as 'I learn python'.

After extracting the eventualities, we employ the Amazon Mechanical Turk platform (MTurk)[12] for annotations. For each eventuality pattern, we randomly select 200 extracted eventualities and then provide these extracted eventualities along with their original sentences to the annotators. In the annotation task, we ask them to label whether one auto-extracted eventuality phrase can fully and precisely represent the semantic meaning of the original sentence. If so, they should label them with 'Valid'. Otherwise, they should label it with 'Not Valid'. For each eventuality, we invite 4 workers to label and if at least 3 of them give the same annotation result, we consider it to be one agreed annotation. Otherwise, this extraction is considered as disagreed. In total, we spent $201.6. The detailed result is shown in Table 6. We got 2,597 agreed annotations out of 2,800 randomly selected eventualities, and the overall agreement rate is 92.8%, which indicates that annotators can

12. `https://www.mturk.com/`



(a) Statistics and evaluation of bootstrapping.



(b) Distribution and accuracy of different relation types.

Figure 6: Human Evaluation of the bootstrapping process. Relation *Co_Occurrence* is not included in the figures since it is not influenced by the bootstrapping.

easily understand our task and provide consistent annotations. Besides that, as the overall accuracy is 94.5%, the result proves the effectiveness of the proposed eventuality extraction method.

5.2 Relations Extraction

In this section, we evaluate the quantity and quality of extracted relations in ASER. Here, to make sure the quality of the learned bootstrapping model, we filter out eventuality and eventuality pairs that appear once and use the resulting training instances to train the

bootstrapping model. The KG extracted from the selected data is called the core part of ASER. Besides that, after the bootstrapping, we directly apply the final bootstrapping model on all training instances and get the full ASER. In this section, we will first evaluate the bootstrapping process and then evaluate relations in two versions of ASER (core and full).

For the bootstrapping process, similar to the evaluation of eventuality extraction, we invite annotators from Amazon Turk to annotate the extracted edges. For each iteration, we randomly select 100 edges for each relation type. For each edge, we generate a question by asking the annotators if they think certain relation exists between the two eventualities. If so, they should label as 'Valid'. Otherwise, they should label it as 'Not Valid'. Similarly, if at least 3 of the 4 annotators give the same annotation result, we consider it to be an agreed one and the overall agreement rate is 82.8 %. For simplicity, we report the average accuracy, which is calculated based on the distribution of different relation types, as well as the total number of edges in Figure 6(a). The number of edges grows very fast at the beginning and slows down later. After ten iterations of bootstrapping, the number of edges grows four times with the decrease of less than 6% accuracy (from 92.3% to 86.5%).

Finally, we evaluate the core and full versions of ASER. For both versions of ASER, we randomly select 100 edges per relation type and invite annotators to annotate them using the same way as we annotating the bootstrapping process. Together with the evaluation on bootstrapping, we spent $1698.4. The accuracy along with the distribution of different relation types are shown in Figure 6(b). We also compute the overall accuracy for the core and full versions of ASER by computing the weighted average of these accuracy scores based on the frequency. The overall accuracies of the core and full versions are 86.5% and 84.3% respectively, which is comparable with KnowlyWood (Tandon et al., 2015) (85%), even though Knowlywood only relies on human designed patterns and ASER involves bootstrapping. From the result, we observe that, in general, the core version of ASER has a better accuracy than the full version, which fits our understanding that the quality of those rare eventualities might not be good. But from another perspective, the full version of ASER can cover much more relations than the core version with acceptable accuracy.

5.3 Comparison with Commonsense Knowledge

We study the relationship between ASER and the commonsense knowledge in ConceptNet (Liu & Singh, 2004), or previously called Open Mind Common Sense (OMCS) (Singh, Lin, Mueller, Lim, Perkins, & Zhu, 2002). The ConceptNet contains 600K crowdsourced commonsense triplets and 75K among them involve eventualities, such as (`sleep`, `HasSubevent`, `dream`) and (`wind`, `CapableOf`, `blow to east`). All relations in ConceptNet are human-defined. We select all four commonsense relations (`HasPrerequisite`, `Causes`, `MotivatedByGoal`, and `HasSubevent`) that involve eventualities to examine how many relations are covered in ASER. Here by covered, we mean that for a given ConceptNet pair (E_{o1}, E_{o2}), we can find an edge $x = (E_{a1}, E_{a2}, c)$ in ASER such that $E_{o1} = E_{a1}$, $E_{o1} = E_{a1}$. The detailed statistic of coverages are shown in Table 7.

Table 7: Statistics of selected OMCS data. we only select ConceptNet pairs that involve eventualities.

Relation	# Examples	# Covered	Coverage
HasPrerequisite	22,389	21,515	96.10%
Causes	14,065	12,605	89.62%
MotivatedByGoal	11,911	10,692	89.77%
HasSubevent	30,074	28,856	95.95%
Overall	78,439	73,668	93.92%

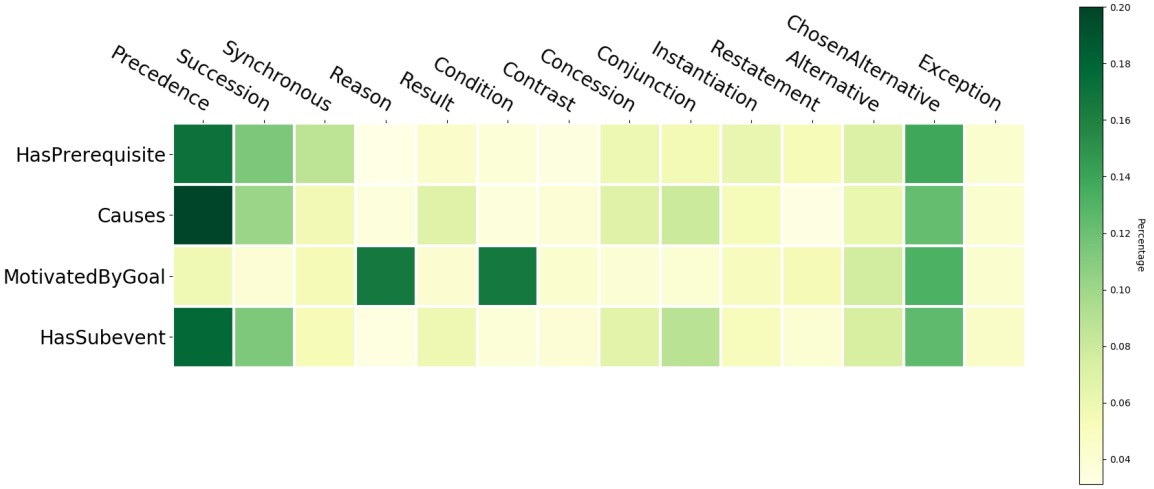


Figure 7: Heatmap of overlapping between OMCS and ASER relations. For each OMCS relation, the distribution of matched ASER edges is computed. Darker color indicates more overlaps for two relations.

Moreover, to show the connection between the ConceptNet and ASER, we use a heat map to show the distribution of their relation pairs.[13] The result is shown in Figure 7, where darker color indicates more coverage, and we observe many interesting findings. First, we can find a strong connection between `Causes` and `Precedence`, which fits our understanding that eventualities happen first is probably the reason for the eventualities happen later. For example, I eat and then I am full, where 'we eat' is the reason of 'I am full'. Such correlation between temporal and causal relations is also observed in (Ning, Feng, Wu, & Roth, 2018). Second, most of the `MotivatedByGoal` pairs appear in the `Reason` or `Condition` relation in ASER, which makes sense because the motivation can be both the reason or the condition. For example, 'I am hungry' can be viewed as the motivation of 'I eat' and both 'I eat because I am hungry' and 'I eat if I am hungry' are valid statements. These observations

13. As all different relations are not evenly distributed in ASER, we normalize the co-occurrence number with the total number of ASER relations and then show it with the heatmap.

demonstrate that the knowledge contained in ConceptNet can be effectively covered by ASER. Considering that ConceptNet is often criticized for its scale and ASER is 100 times larger than ConceptNet, even though ASER may not be as accurate as ConceptNet, it could be a good supplement.

6. Extrinsic Evaluations

In this section, we use two extrinsic experiments to demonstrate the importance of ASER. All the experiments are conducted with the support of the core version of ASER.

6.1 Winograd Schema Challenge

Winograd Schema Challenge (WSC) is known as related to commonsense knowledge and argued as a replacement of the Turing test (Levesque et al., 2011). Given two sentences s_1 and s_2, both of them contain two candidate noun phrases n_1 and n_2, and one targeting pronoun p. The goal is to detect the correct noun phrase p refers to. Here is an example (Levesque et al., 2011).

(1) The fish ate the worm. `It` was `hungry`. Which was `hungry`?
Answer: `the fish`.

(2) The fish ate the worm. `It` was `tasty`. Which was `tasty`?
Answer: `the worm`.

This task is challenging because s_1 and s_2 are quite similar to each other (only one-word difference), but the result is totally reversed. Besides that, all the widely used features such as gender/number are removed, and thus all the conventional rule-based resolution system failed on this task. For example, in the above example, both fish and worm can be hungry or tasty by themselves. We can solve the problem because fish is subject of 'eat' while the worm is the object, which requires understanding eventualities related to 'eat'. Moreover, due to the small size of the Winograd schema challenge, supervised learning based methods are not practical.

To demonstrate the effectiveness of ASER, we try to solve Winograd questions using simple inference based on ASER. For each question sentence s, we first extract eventualities with the same method introduced in Section 3.3 and then select eventualities E_{n_1}, E_{n_2}, and E_p that contain candidates nouns n_1/n_2 and the target pronoun p respectively. We then replace n_1, n_2, and p with placeholder X, Y, and P, and hence generate the pseudo-eventualities E'_{n_1}, E'_{n_2}, and E'_p. After that, if we can find the seed connectives in Table 5 between any two eventualities, we use the corresponding relation type as relation type T. Otherwise, we use *Co_Occurrence* as the relation type. To evaluate the candidate, we first replace the placeholder P in E'_p with the corresponding placeholders X or Y and then use the following equation to define its overall plausibility score:

$$F(n, p) = ASER_R(E'_n, E'_p), \tag{6}$$

where $ASER_R(E_n, E_p)$ indicates the number of edges in ASER that can support that there exist one typed T relation between the eventuality pairs E'_n and E'_p. For each edge (E_h, T, E_t) in ASER, if it can fit the following three requirements:

1. $E_h = E'_n$ other than the words in the place holder positions.

2. $E_t = E'_p$ other than the words in the place holder positions.

3. Assume the word in the placeholder positions of E_h and E_t are w_h and w_t respectively, w_h has to be same as w_t.

we consider that edge as a valid edge to support the observed eventuality pair. If any of E_n and E_p cannot be extracted with our patterns, we will assign 0 to $F(n, p)$. We then predict the candidate with the higher score to be the correct reference. If both of them have the same score (including 0), we will make no prediction. At current stage, We only use one-hop relations in ASER to perform inference for Winograd questions.

6.1.1 Baseline Methods.

To demonstrate the difficulty of the WSC, we first compare ASER with the state-of-the-art general co-reference resolutions:

- **Deterministic** model (Raghunathan, Lee, Rangarajan, Chambers, Surdeanu, Jurafsky, & Manning, 2010), which proposes one multi-pass seive model with human designed rules for the coreference resolution task.

- **Statistical** model (Clark & Manning, 2015) uses human-designed entity-level features between clusters and mentions for coreference resolution.

- **Deep-RL** model (Clark & Manning, 2016) is a reinforcement learning method to directly optimize the coreference matrix instead of the traditional loss function.

- **End2end** model (Lee, He, & Zettlemoyer, 2018) is the current state-of-the-art coreference model, which performs in an end-to-end manner and leverages both the contextual information and a pre-trained language model (Peters, Neumann, Iyyer, Gardner, Clark, Lee, & Zettlemoyer, 2018).

Besides these general co-reference models, we also compare ASER with the following models that designed specifically for the WSC task:

- **Knowledge Hunting** (Emami, Cruz, Trischler, Suleman, & Cheung, 2018) first search commonsense knowledge on search engines (e.g., Google) for the Winograd questions and then leverages rule-based methods to make the final predictions based on the collected knowledge.

- **LM** model (Trinh & Le, 2018) is the language model trained with very large-scale corpus and tuned specifically for the WSC task.

In the end, we also compare with the selectional preference (SP) based method (Zhang et al., 2019). Following the original setting, two resources (human annotation and Posterior Probability) of SP knowledge are considered and we denote them as SP (human) and SP (PP) respectively[14].

14. In their original paper, they only consider two-hop SP knowledge for the WSC task, but we consider both one-hop and two-hop SP knowledge in our experiment.

Table 8: Experimental results on Winograd Schema Challenge. $\surd$ indicates the number of correct answers, $\times$ indicates the number of wrong answers, and NA means that the model cannot give a prediction. A_p means the prediction accuracy without NA examples, and A_o means the overall accuracy.

Methods	$\surd$	$\times$	NA	A_p	A_o
Random Guess	83	82	0	50.3%	50.3%
Deterministic (Raghunathan et al., 2010)	75	71	19	51.4%	51.2%
Statistical (Clark & Manning, 2015)	75	78	12	49.0%	49.1%
Deep-RL (Clark & Manning, 2016)	80	76	9	51.3%	51.2%
End2end (Lee et al., 2018)	79	84	2	48.5%	48.5%
Knowledge Hunting (Emami et al., 2018)	94	71	0	56.9%	56.9%
LM (single) (Trinh & Le, 2018)	90	75	0	54.5%	54.5%
SP (human) (Zhang et al., 2019)	15	0	150	**100%**	54.5%
SP (PP) (Zhang et al., 2019)	50	26	89	65.8%	57.3%
ASER	63	27	75	70.0%	**60.9%**

As the Deterministic, Statistical, and Deep-RL model are included in the Stanford CoreNLP toolkit[15], we use their released model as baselines. For the end-to-end[16], knowledge hunting[17], and LM[18] models, we use their released code as baselines[19].

6.1.2 Experiment Setting.

We select all Winograd questions satisfying two criteria to form the dataset: (1) They should have no subordinate clause; (2) The targeting pronoun is covered by an eventuality detected from the questions. As a result, we get 165 out of the total 273 questions[20].

15. `https://stanfordnlp.github.io/CoreNLP/coref.html`
16. `https://github.com/kentonl/e2e-coref`
17. `https://github.com/aemami1/Wino-Knowledge-Hunter`
18. `https://github.com/tensorflow/models/tree/master/research/lm_commonsense`
19. Besides the aforementioned models, the ensemble version of the LM models (Trinh & Le, 2018) and a more recent study (Kocijan, Cretu, Camburu, Yordanov, & Lukasiewicz, 2019), which is based on BERT (Devlin, Chang, Lee, & Toutanova, 2019) and GPT (Radford, Narasimhan, Salimans, & Sutskever, 2018) contextualized word embeddings, report better performance on the WSC task. However, the ensemble LM has the problem of capturing the statistical associativity of the test data rather than understanding the questions (Trichelair, Emami, Cheung, Trischler, Suleman, & Diaz, 2018). BERT and GPT require additional similar datasets to fine-tune the models. (Kocijan et al., 2019) showed that without fine-tuning, results of BERT and GPT are 60.1% and 55.3% on the overall 273 questions.
20. The latest winograd schema challenge contains 285 questions, but to be consistent with the baseline methods, we select the widely used 273 questions version.

6.1.3 Result Analysis.

As shown in Table 8, the Winograd schema challenge is challenging for the current co-reference system as all the general co-reference models and contextual representations cannot achieve better performance than the random guess. This is because all the Winograd questions are specifically designed to test model's ability to do inference over commonsense knowledge.

Different from them, the knowledge hunting and language model approaches can thus achieve better performance because they inject commonsense knowledge into the model via search engine and language model respectively. The experiments on SP and ASER demonstrate the importance of the eventuality knowledge. For the SP knowledge, which can also be viewed as the internal structure of eventualities, human annotation can provide 100% accuracy but can only cover very limited questions due to the annotation size and the PP approach can provide larger coverage but may contain more noise. On top of the SP knowledge, ASER adds information about relations between eventualities so it can answer more questions with high precision. But still, we notice that a large percentage of questions remain unsolved with the proposed model. This is because (1) some of the questions, although eventualities can be found using the same patterns, cannot be covered by ASER; and (2) they may require more complicated reasoning methods. Some advanced knowledge graph analysis techniques like long-distance reasoning and graph embedding might be helpful for the two mentioned problem. But as the main purpose of this section is demonstrating the value of ASER rather than designing a complex method to solve the Winograd schema challenge task with ASER, we chose to use simple match and count based approach and leave other potential usage of ASER for the future exploration.

6.1.4 Case Study.

One example is shown in Figure 8, our model can correctly resolve 'it' to 'fish' in question 97, because 18 edges in ASER support that the subject of 'eat' should be 'hungry', while only one edge supports the object of 'eat' should be 'hungry'. Similarly, our model can correctly resolve 'it' to 'the worm' in question 98, because seven edges in ASER support that the object of 'eat' should be 'tasty' while no edge supports that the subject of 'eat' should be 'tasty'.

6.2 Eventuality knowledge enhanced dialogue system

As one of the most direct way for machines to interact with human, the dialogue system has been a hot research topic. We conduct experiments to demonstrate that the knowledge contained in ASER can help generate better dialogue response.

6.2.1 Experiment Details.

To test the effectiveness of ASER in daily life rather than a specific domain, we select Dailydialog (Li, Su, Shen, Li, Cao, & Niu, 2017) as the experimental dataset and use the widely used BLEU (Papineni, Roukos, Ward, & Zhu, 2002) score (%) as the evaluation metrics. We use the sequence-to-sequence with attention mechanism model (Luong, Pham, & Manning, 2015) as the base model and leverage the memory module to incorporate knowledge

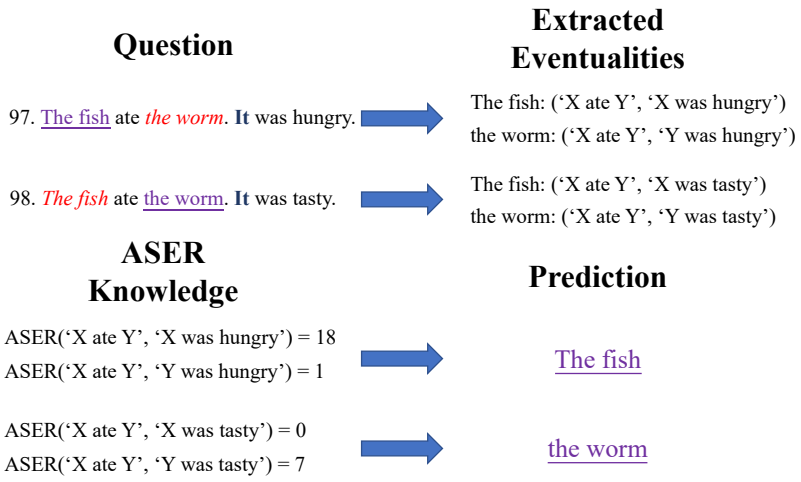


Figure 8: Example of using ASER to solve Winograd questions. The number before questions are the original question ID. Correct answer and the other candidate are labeled with purple underline and red italic font respectively.

about eventuality into the dialogue generation model inspired by (Ghazvininejad, Brockett, Chang, Dolan, Gao, Yih, & Galley, 2018)[21]. Two existing eventuality-related resources ConceptNet (Liu & Singh, 2004) and KnowlyWood (Tandon et al., 2015) are selected as the baseline KGs. Originally, Dailydialog contains 13,118 conversations and 49,188 post-response pairs. We first count the number of conversation pairs whose eventuality can be covered by the three KGs. ConceptNet, Knowlywood, and ASER can cover 7,246, 17,183, and 20,494 pairs respectively. For each conversation pair, if it contains an eventuality that can be found in any of the three KGs, we select it as a valid experiment dialogue conversation pair. As a result, we have 30,145 pairs. These pairs are divided into training, validation, and test data following the original setting.

6.2.2 Coverage Statistics.

The detailed statistics about the coverages of different KGs are shown in Table 9. The number of covered conversation pairs, the percentage of such pairs, and the number of unique covered eventualities of each KG are reported. The statistics show that ConceptNet can only cover a very small portion of the questions due to its relatively small size and ASER covers the most conversation pairs. We also notice that compared with ASER, Knowlywood can cover more eventualities in fewer conversation pairs. The reason behind is that the definition of eventuality is different. In Knowlywood, each eventuality is represented with two words (verb+object), which may not be semantically complete but can be more easily found in the text. In ASER, we require the matched eventualities to be semantically complete, each of which typically contains 3-5 words. This makes them more difficult to

21. The model design and implementation details are included in Appendix A.

Table 9: Statistics of the dialogue dataset. '# Covered pairs' means the number of conversation pairs, whose eventualities can be covered by the corresponding KG. 'Coverage rate' means the percentage of such pairs. '# Unique matched events' means the number of unique matched eventualities in the KG.

KG	# Covered pairs	Coverage rate	# Unique matched events
ConceptNet	7,246	24.04%	1,195
KnowlyWood	17,183	57.00%	30,036
ASER	20,494	67.98%	9,511

Table 10: Experimental results on the dialogue task. BLEU scores with standard deviations in the brackets are reported. The highest BLEU scores are in boldface. 'Base' represents the seq2seq model with the attention mechanism.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Base	30.16 (0.44)	5.75 (0.37)	2.28 (0.24)	0.98 (0.16)
+ConceptNet	30.89 (0.40)	6.14 (0.15)	2.60 (0.13)	1.21 (0.12)
+KnowlyWood	30.72 (0.19)	6.26 (0.24)	2.68 (0.17)	1.29 (0.11)
+ASER	**32.10** (0.42)	**7.14** (0.17)	**3.54** (0.10)	**2.07** (0.08)

be matched. Nonetheless, as ASER is extracted from different resources, it can cover the topics in more conversation pairs.

6.2.3 Result and Analysis.

For each KG, we repeat the experiment five times and report the average performance as well as the standard deviation. From the result shown in Table 10 we can observe that the effect of ConceptNet is not obvious due to its small coverage. KnowlyWood can cover much more examples but its effect is also limited due to its semantically incomplete definition of eventualities. Last but not least, ASER achieves the best performance on all of the four BLEU metrics, especially on BLEU-3 and BLEU-4. The reason behind is that the knowledge about eventuality can help the system generate the response with a more suitable eventuality rather than a single word and thus the metrics take more words into consideration can benefit more from using eventuality-related knowledge.

One example is shown in Table 11. After getting the post 'I should eat some food', we extract the contained eventuality 'eat food', 'eat food', and 'I eat food' for the three KGs respectively, and then find the related eventualities in KGs to generate the response. By retrieving from ConceptNet, we know that 'eat food' can be motivated by 'you are hungry' and has the prerequisite that we have to open our mouth. Similarly, by retrieving from KnowlyWood, we know that we often 'keep eating', 'enjoy taste', or 'stick swap' after 'eat

Table 11: Eventuality matching example.

Post	I should eat some food .
Response	Yeah, you must be hungry. Do you like to eat some beaf?
ConceptNet	'eat food', MotivatedByGoal, 'you are hungry' 'eat food', HasPrerequisite, 'open your mouth'
KnowlyWood	(eat,food), next, (keep, eating) (eat,food), next, (enjoy, taste) (eat,food), next, (stick, wasp) ...
ASER	i eat food [s-v-o], Conjunction, beef is good [s-be-a] i eat food [s-v-o], Condition, i am hungry [s-be-a] i eat food [s-v-o], Concession, i take picture [s-v-o] ...

food'. By retrieving from ASER, we know that 'I eat food' and 'beef is good' can happen at the same time, and eating food often has the condition of being hungry.

In general, the ConceptNet is accurate and correct, because they are generated by humans. However, their small scale limits their usage. KnowlyWood has a better scale, but its semantically incomplete definition of eventualities also limits the usage. As a comparison, ASER leverages carefully designed patterns to make sure the semantic completeness of extracted eventualities and uses a neural bootstrapping model to automatically learn relations between eventualities from large unlabeled corpus. Thus, it can provide a larger scale and higher quality eventuality knowledge.

7. Conclusions

In this paper, we introduce ASER, a large-scale eventuality knowledge graph. We extract eventualities from texts based the dependency graphs. Then we build seed relations among eventualities using unambiguous connectives found from PDTB and use a neural bootstrapping framework to extract more relations. ASER is the first large-scale eventuality KG using the above strategy. We conduct systematic experiments to evaluate the quality and applications of the extracted knowledge. Both human and extrinsic evaluations show that ASER is a promising large-scale eventuality knowledge graph with great potential in many downstream tasks.

Acknowledgements

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong. Hongming Zhang has been supported by the Hong Kong Ph.D. Fellowship. We thank Dan Roth and Daniel Khashabi for their insightful comments on this work.

Appendix A. Dialog System with ASER Implementation Details

In this section, we introduce the details about how we leverage the eventuality knowledge in ASER to help the dialog generation task.

A.1 The Task

We first formally introduce the task of eventuality-enhanced dialog system. Given an input post P, which contains multiple words $w_{p,1}, w_{p,2}, ..., w_{p,n_p}$ and multiple eventualities[22] $E_1, E_2, ..., E_m$, our goal is to generate a corresponding response A, which contains multiple words $w_{a,1}, w_{a,2}, ..., w_{a,n_a}$.

A.2 The Model

The overall structure of the proposed model in shown in Figure 9. In general, we adopt an encoder-decoder model to incorporate the eventuality knowledge for better response generation. Both the original post sentence and the retrieved related eventualities are encoded with vector representations, which are used to generate the response in the decoder. The details about the sentence encoding, eventuality encoding, and the response decoding are as follows:

• **Utterance Encoding**: Following conventional approach (Luong et al., 2015), we adopt the standard bidirectional LSTM (biLSTM) (Hochreiter & Schmidhuber, 1997) to encode the semantic meaning of the original post. Let initial word embeddings of an input post P be denoted as $\mathbf{x}_{p,1}, ..., \mathbf{x}_{p,n_p}$ and their encoded representation after the LSTM be $\mathbf{h}_{p,1}, ..., \mathbf{h}_{p,n_p}$, which are treated as the utterance memory $\widetilde{P}$.

• **Eventuality Encoding**: To leverage the eventuality knowledge for better response generation, we first need to extract eventualities from the original post P. Assume the extracted eventuality set is $\mathcal{E}_p$, which contains m eventualities $\{E_1, E_2, ..., E_m\}$. For each $E_i \in \mathcal{E}_p$, we search it in the KG (ConceptNet, KnowlyWood, or ASER) and retrieval all related edges, which are represented as triplets. For each triplet (E_{src}, R, E_{tgt}), where E_{src} is the eventuality we extract from the post, E_{tgt} is the retreivaled related eventuality, and R is the relation type between them, we represent it as a concatenation of four vectors $\widetilde{v}_k = [v_{tri}|v_{src}|v_{rel}|v_{tgt}]$, where v_{tri}, v_{src}, v_{rel}, and v_{tgt} are the embeddings of the triplets, R, E_{src}, and E_{tgt} respectively. All of them are set to be trainable. We group the representations of all triplets as the eventuality memory $\widetilde{E}$.

• **Response Decoding**: Assume the generated response is A_{t-1}, which contains $t-1$ words $w_{a,1}, w_{a,2}, ..., w_{a,t-1}$, we introduce how the decoder generates the t^{th} word in detail. We first encode R_{t-1} with a single-directed LSTM and denote the resulted state as s_t. We then use s_t as attention to extract most related information from $\widetilde{P}$ and $\widetilde{E}$. Assume the attended representation of contextual information and the eventuality knowledge are denoted as C_P and C_E, the distribution of $w_{a,t}$ is $softmax(FFN([s_t||c_P||c_E]))$, where $softmax$ is used to compute the probability for all the words in the vocabulary, FFN represents the

22. Different KGs have different definitions of eventuality. Hence, we use different formats to extract eventuality based on their original settings: ConceptNet uses strings to represent eventualities, KnowlyWood uses a verb-object pair to define eventualities, and ASER uses dependency graphs to represent eventualities.

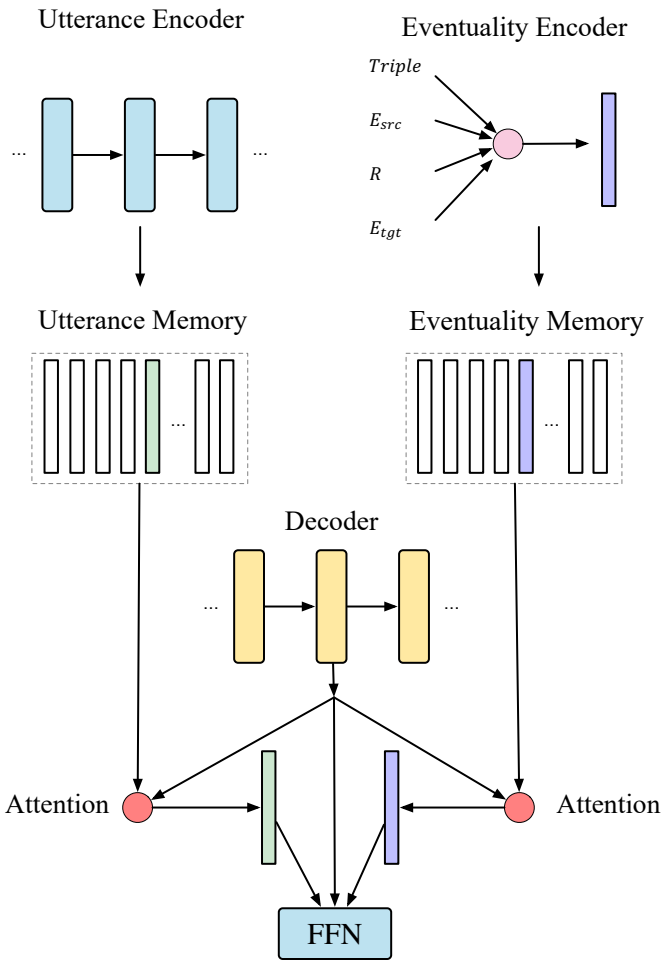


Figure 9: Overall structure of the dialogue model. The predicted sequence is used as attention to retrieve informative sentence level and eventuality level information, which are further used to generate the distribution of next word.

two-layer feed forward neural network , and || means the countenance. In the end, we select the word with the highest probability score the the t^{th} word.

A.3 The Implementation.

We use cross-entropy as the loss and Adam (Kingma & Ba, 2015) with the initial learning rate of 0.005 as the learning method. All the parameters are initialized randomly. We use the 256-dimension two-layer Bi-LSTM as the encoder and the 512-dimension two-layer single-layer LSTM as the decoder. The word embedding size is set to 300 and the embedding sizes of v_{tri}, v_{src}, v_{rel}, and v_{tgt} are all 128. All the models are trained up to 20 epochs and the best models are selected based on the dev set. Dropout is set to be 0.1. In the inference stage, the beam search size is set to be five.

References

Agichtein, E., & Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *ACM DL*, pp. 85–94.

Aguilar, J., Beller, C., McNamee, P., Van Durme, B., Strassel, S., Song, Z., & Ellis, J. (2014). A comparison of the events and relations across ace, ere, tac-kbp, and framenet annotation standards. In *Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pp. 45–53.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). *DBpedia: A nucleus for a web of open data*. Springer.

Bach, E. (1986). The algebra of events. *Linguistics and philosophy*, *9*(1), 5–16.

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The berkeley framenet project. In *COLING-ACL*, pp. 86–90.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. (2007). Open information extraction from the web. In *IJCAI*, pp. 2670–2676.

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pp. 1533–1544.

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pp. 1247–1250.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., & Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI*, pp. 1306–1313.

Clark, K., & Manning, C. D. (2015). Entity-centric coreference resolution with model stacking. In *ACL-IJCNLP, 2015*, Vol. 1, pp. 1405–1415.

Clark, K., & Manning, C. D. (2016). Deep reinforcement learning for mention-ranking coreference models. In *EMNLP*, pp. 2256–2262.

Dalvi, B., Huang, L., Tandon, N., tau Yih, W., & Clark, P. (2018). Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. *NAACL*.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pp. 4171–4186. Association for Computational Linguistics.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., & Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, pp. 601–610.

Ehrlinger, L., & Wöß, W. (2016). Towards a definition of knowledge graphs. In *SEMANTiCS (Posters, Demos, SuCCESS)*, Vol. 1695 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Emami, A., Cruz, N. D. L., Trischler, A., Suleman, K., & Cheung, J. C. K. (2018). A knowledge hunting framework for common sense reasoning. In *EMNLP*, pp. 1949–1958.

Etzioni, O., Cafarella, M., & Downey, D. (2004). Webscale information extraction in knowitall (preliminary results). In *WWW*, pp. 100–110.

Fellbaum, C. (Ed.). (1998). *WordNet: an electronic lexical database*. MIT Press.

Ghazvininejad, M., Brockett, C., Chang, M., Dolan, B., Gao, J., Yih, W., & Galley, M. (2018). A knowledge-grounded neural conversation model. In *AAAI-IAAI-EAAI*, pp. 5110–5117.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Jackendoff, R. (Ed.). (1990). *Semantic Structures*. Cambridge, Massachusetts: MIT Press.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Kocijan, V., Cretu, A.-M., Camburu, O.-M., Yordanov, Y., & Lukasiewicz, T. (2019). A surprisingly robust trick for the winograd schema challenge. In *ACL*.

Lee, K., He, L., & Zettlemoyer, L. (2018). Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*, pp. 687–692.

Lenat, D. B., & Guha, R. V. (1989). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

Levesque, H. J., Davis, E., & Morgenstern, L. (2011). The winograd schema challenge.. In *AAAI Spring Symposium: Logical formalizations of commonsense reasoning*, Vol. 46, p. 47.

Li, Q., Ji, H., & Huang, L. (2013). Joint event extraction via structured prediction with global features. In *ACL*, Vol. 1, pp. 73–82.

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., & Niu, S. (2017). Dailydialog: A manually labelled multi-turn dialogue dataset. In *IJCNLP*, pp. 986–995.

Lison, P., & Tiedemann, J. (2016). Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles..

Liu, H., & Singh, P. (2004). Conceptneta practical commonsense reasoning tool-kit. *BT technology journal*, *22*(4), 211–226.

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *EMNLP*, pp. 1412–1421.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., & Grishman, R. (2004). The nombank project: An interim report. In *Workshop Frontiers in Corpus Annotation at HLT-NAACL*.

Ning, Q., Feng, Z., Wu, H., & Roth, D. (2018). Joint reasoning for temporal and causal relations. In *ACL*, pp. 2278–2288.

NIST (2005). The ACE evaluation plan...

P. D. Mourelatos, A. (1978). Events, processes, and states. *Linguistics and Philosophy*, *2*, 415–434.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, *31*(1), 71–106.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018*, pp. 2227–2237.

Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., & Webber, B. L. (2007). The penn discourse treebank 2.0 annotation manual..

Pustejovsky, J., Hanks, P., Sauri, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., et al. (2003). The timebank corpus. In *Corpus linguistics*, Vol. 2003, p. 40.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D., & Manning, C. (2010). A multi-pass sieve for coreference resolution. In *EMNLP*, pp. 492–501.

Sandhaus, & Evan (2008). The new york times annotated corpus ldc2008t19..

Sap, M., LeBras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., & Choi, Y. (2018). ATOMIC: an atlas of machine commonsense for if-then reasoning. In *AAAI*.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., & Zhu, W. L. (2002). Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 1223–1237.

Smith, N. A., Choi, Y., Sap, M., Rashkin, H., & Allaway, E. (2018). Event2mind: Commonsense inference on events, intents, and reactions. In *ACL*, pp. 463–473.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago: a core of semantic knowledge. In *WWW*, pp. 697–706.

Tandon, N., de Melo, G., De, A., & Weikum, G. (2015). Knowlywood: Mining activity knowledge from hollywood narratives. In *CIKM*, pp. 223–232.

Trichelair, P., Emami, A., Cheung, J. C. K., Trischler, A., Suleman, K., & Diaz, F. (2018). On the evaluation of common-sense reasoning in natural language understanding. *arXiv preprint arXiv:1811.01778*.

Trinh, T. H., & Le, Q. V. (2018). A simple method for commonsense reasoning. *CoRR*, *abs/1806.02847*.

Wilks, Y. (1975). A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, *6*(1), 53–74.

Wu, W., Li, H., Wang, H., & Zhu, K. Q. (2012). Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, pp. 481–492.

Xue, N., Ng, H. T., Pradhan, S., Prasad, R., Bryant, C., & Rutherford, A. (2015). The conll-2015 shared task on shallow discourse parsing. In *CoNLL Shared Task*, pp. 1–16.

Zhang, H., Ding, H., & Song, Y. (2019). Sp-10k: A large-scale evaluation set for selectional preference acquisition. In *ACL*.