Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Review:

A survey of script learning*

Yi HAN[1], Linbo QIAO‡[1], Jianming ZHENG[2], Hefeng WU[3], Dongsheng LI[1], Xiangke LIAO[1]

[1]*Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha 410000, China*

[2]*Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410000, China*

[3]*School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China*

E-mail: hanyi12@nudt.edu.cn; qiao.linbo@nudt.edu.cn; zhengjianming12@nudt.edu.cn; wuhefeng@mail.sysu.edu.cn; dsli@nudt.edu.cn; xkliao@nudt.edu.cn

Received July 13, 2020; Revision accepted Nov. 11, 2020; Crosschecked Jan. 8, 2021

Abstract: Script is the structured knowledge representation of prototypical real-life event sequences. Learning the commonsense knowledge inside the script can be helpful for machines in understanding natural language and drawing commonsensible inferences. Script learning is an interesting and promising research direction, in which a trained script learning system can process narrative texts to capture script knowledge and draw inferences. However, there are currently no survey articles on script learning, so we are providing this comprehensive survey to deeply investigate the standard framework and the major research topics on script learning. This research field contains three main topics: event representations, script learning models, and evaluation approaches. For each topic, we systematically summarize and categorize the existing script learning systems, and carefully analyze and compare the advantages and disadvantages of the representative systems. We also discuss the current state of the research and possible future directions.

Key words: Script learning; Natural language processing; Commonsense knowledge modeling; Event reasoning
https://doi.org/10.1631/FITEE.2000347 **CLC number:** TP391

1 Introduction

Understanding is one of the most important components of a human-level artificial intelligence (AI) system. However, developing such a system is far from trivial, because natural language comes with its own complexity and inherent ambiguities. To understand natural language, machines need to comprehend not only literal meaning but also commonsense knowledge about the real world.

Commonsense knowledge includes certain internal logic and objective laws based on the development of events, which can be implicitly learned and comprehended by humans in long-term daily-life activities. After learning such knowledge, they can use it to help understand the implicit information and address the ambiguities of natural language. However, machines cannot perform such daily-life activities, and therefore lack commonsense knowledge, which is one of the major obstacles to understanding natural language and drawing human-like inferences.

Some researchers attempt to alleviate such bottlenecks from the perspective of how humans remember and understand various real-life events. According to many psychological experiments, script knowledge is an essential component of human cognition and memory systems (Bower et al., 1979; Schank, 1983; Tulving, 1983; Terry, 2006). From the

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (No. 61806216)
ORCID: Yi HAN, https://orcid.org/0000-0001-9176-8178; Linbo QIAO, https://orcid.org/0000-0002-8285-2738
© Zhejiang University Press 2021

perspective of cognitive psychology, humans use schemas to organize and understand the world. A schema is a mental framework that consists of a general template for the knowledge shared by all instances, and a number of slots that can take on different values for a specific instance. A script is a special case of a schema that describes stereotypical activities with a sequence of events organized in temporal order (Rumelhart, 1980). When a person first meets a prototypical activity, his/her episodic memory will store the events and the scenarios in which it occurred in the form of a script (Tulving, 1983). The psychological research has indicated that the activity stored in episodic memory can be activated by some particular external event (Terry, 2006). So, the next time a similar or relevant event happens, the person's episodic memory will activate the old corresponding script knowledge. Based on the script knowledge, the person will soon understand the actions and the participants of the scenario, and subsequently form the expectations for possible upcoming events. The expectations can deeply affect the ability of humans to comprehend specific scenarios and draw reasonable inferences.

Since script knowledge is critical for humans to remember and comprehend different scenarios, machines may also gain benefits by learning such knowledge. It is this intention that motivates researchers to introduce script learning to encode script knowledge for machines. A trained script learning system can process texts that describe daily-life events and capture commonsense knowledge involving the actions and the entities that participate in it. Based on this, it can also draw event-related inferences that are reasonable but not explicitly stated in the texts.

1.1 Definition of script

A script is a structural knowledge representation that captures the relationships between prototypical event sequences and their participants in a given scenario (Schank and Abelson, 1977). In other words, the script is a sequence of prototypical events organized in temporal order. These events contain stereotypical human activities, such as eating in a restaurant, cooking dinner, and making coffee. For example, Fig. 1 shows two toy examples of the "visiting restaurant" script and the "making coffee" script.

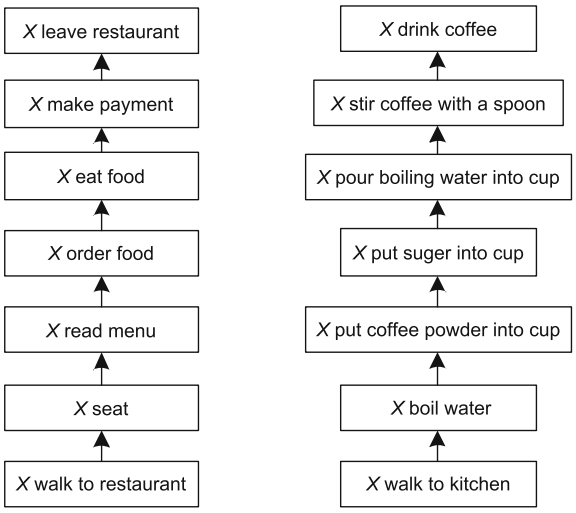

Fig. 1 Scripts for visiting a restaurant and making coffee. Each script includes a sequence of events for a prototypical scenario

1.2 Objective of script learning

The objective of script learning is encoding the commonsense knowledge contained in prototypical events (also called "script knowledge") in a machine-readable format. Then the machine can use the format to draw event-related inferences, such as inferring missing events, predicting subsequent events, determining the order of the event chain, and distinguishing similar events.

The commonsense knowledge provided by the script can help machines understand the practical meanings of natural language and draw human-like inferences. It will also play a crucial role in a wide range of AI tasks, especially many natural language processing (NLP) tasks, such as event prediction, event extraction, understanding ambiguity resolution discourse, intention recognition, question answering, and coreference resolution.

1.3 Framework of script learning

Note that this survey concentrates mainly on the broad-domain script knowledge learned from narrative event chains, which can be used to compare and infer possible upcoming events in a specific context, rather than those works attempting to construct scenario-specific event schemas. Therefore, the framework of script learning systems with which we are concerned can be illustrated by the basic process and a simple example in Fig. 2.

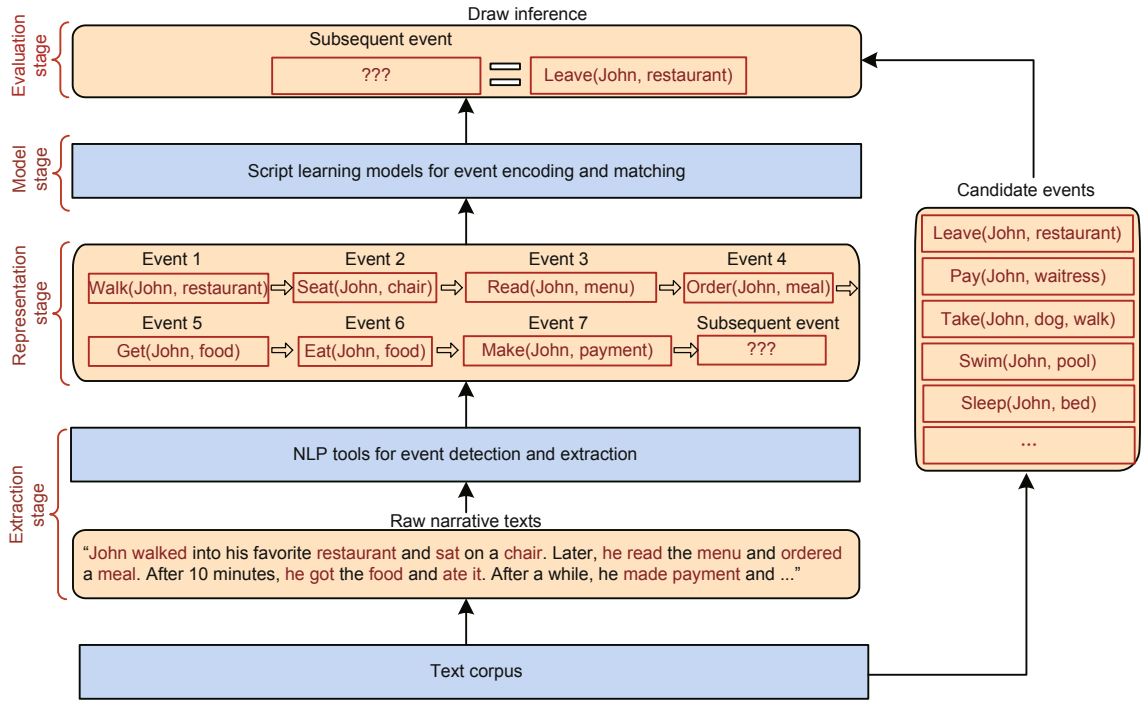

Fig. 2 Framework of script learning which contains four stages. In the extraction stage, events are extracted from raw texts. In the representation stage, events are represented by a structured form. In the model stage, the structured representations are encoded and matched by script learning models, and the matching result is outputted. In the evaluation stage, some evaluation approaches with specific tasks and corresponding metrics are employed to test the performance of the models

According to the standard script learning settings, our framework can be roughly divided into four stages: extraction stage, representation stage, model stage, and evaluation stage.

Script knowledge is often mentioned implicitly in the unstructured narrative texts that describe daily-life activities. The first task of script learning is employing NLP tools to detect and extract events from the raw text (i.e., the extraction stage in Fig. 2).

The output from the previous stage is then represented in a specific structure or neural networks to form representations of events (i.e., the representation stage in Fig. 2). In this stage, events extracted by NLP tools are regarded as the input, and structured representations of events as the output.

After that, the script learning models will encode the collected event representations, and measure the relationship between them using event matching algorithms (i.e., the model stage in Fig. 2). For this stage, the input is the structured event representations, and the output is the matching results. Notably, similarity or coherence scores are usually adopted as the matching metrics.

Finally, some evaluation approaches with specific tasks and corresponding metrics are employed to test the quality of event representations and the performance of script learning models (i.e., the evaluation stage in Fig. 2). For this stage, the input is event matching results, and the output is performance, based on comparisons between prediction results and golden results.

Specifically, in Fig. 2, the extracted events are represented by the structure predicate(subject, object), while the evaluation is to judge whether the model can correctly predict the subsequent event from a candidate set, given the event chain that occurred.

1.4 Organization of the survey

We can conclude from the above framework that script learning primarily focuses on five aspects: the text corpus used as a training dataset, the NLP tools for detecting and extracting events, the representation form of script events, the models for encoding

and matching events, and the evaluation approaches for estimating the performance of the models.

There is no standard corpus for script learning, and a majority of related works extract events from news articles by themselves. Therefore, we do not discuss a corpus in detail in the main part of the survey, but introduce such literature in future research directions described in Section 6.2. In addition, event detection and extraction belong to other independent NLP research directions, and there are many off-the-shelf open-source NLP tools that can be used conveniently. Consequently, in this survey, we do not consider event detection and extraction either. In summary, we focus mainly on three aspects, i.e., event representations, script learning models, and evaluation approaches.

The rest of the survey is also organized primarily based on these research topics. Specifically, Section 2 briefly reviews the development history of script learning; Section 3 introduces several typical event representation structures; Section 4 introduces some representative script learning models from seven categories; Section 5 introduces some common evaluation approaches and corresponding metrics; Section 6 concludes the survey and some possible future research directions are discussed. The organization of these sections and some related specific items are shown in Fig. 3.

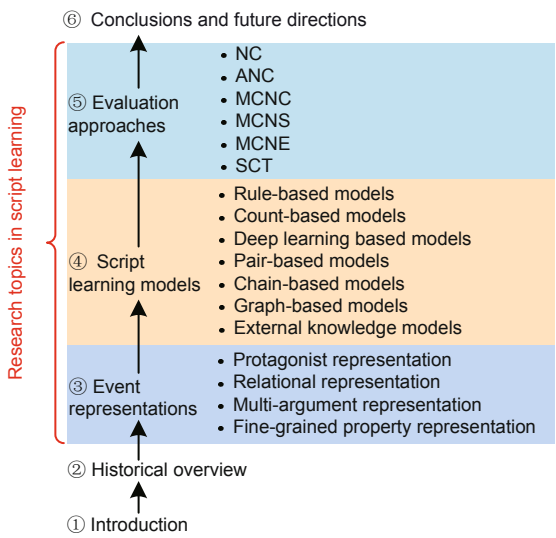

Fig. 3 Organization of the survey. We discuss mainly three research topics, i.e., event representations, script learning models, and evaluation approaches. These topics and their related specific items are introduced in Sections 3–5

Although script learning is quite a potential and significant research direction, there are currently no survey articles on the subject, to the best of our knowledge. The contributions of this survey can be summarized as follows:

1. We are the first to provide a comprehensive survey that deeply investigates the standard framework and major research topics in script learning.

2. We thoroughly summarize the development process and the technique taxonomy of the script learning system.

3. We carefully analyze and compare the most representative works on script learning and discuss some promising research directions.

2 Historical overview

In this section, we will provide a chronological research script learning timeline. Based on modeling methods, the script learning development timeline can be roughly divided into three stages: incipient rule-based approaches (1975–2008), early count-based approaches (2008–2014), and recent deep learning based approaches (2014–). Although there are different ways of dividing this timeline, we will follow the above division to briefly review the script learning development process, because it can better fit the chronological timeline.

2.1 Rule-based approaches (1975–2008)

Although script learning has attracted wide attention in recent years, the use of script-related concepts in AI research dates back to the 1970s, in particular, the seminal works on schema by Rumelhart (1980), on frame by Minsky (1975), on semantic frame by Fillmore (1976), and on scripts by Schank and Abelson (1977). These works established the core concepts of script learning.

The incipient script theory was primarily introduced to construct story understanding and generation systems. The works in this stage often employed quite complex rules (mostly handwritten) to process events and used a set of separate models to treat different scenarios. Although these works showed some abilities in script understanding and inferring, their rule-based methods do not scale to complex domains because they are characteristically domain-specific and time-consuming to construct manually.

2.2 Count-based approaches (2008–2014)

The trend of script learning was revived by count-based methods, which can automatically learn broad-domain script knowledge using statistical approaches to obtain co-occurrence counts from a large text corpus.

Count-based script learning stems from the unsupervised framework proposed by Chambers and Jurafsky (2008). They ran a co-reference system and a dependency parser to automatically extract event sequences from raw text. Pairwise mutual information (PMI) scores were employed to measure the relationship between event pairs. They also pioneered the use of <verb, dependency> pairs to represent events and proposed the narrative cloze (NC) test to evaluate script learning models.

Several researchers followed their model framework and enhanced it in different ways, such as using a skip bi-gram probability model to replace PMI (Jans et al., 2012), or proposing multi-argument event representation $v(e_{\mathrm{s}}, e_{\mathrm{o}}, e_{\mathrm{p}})$ to replace the <verb, dependency> pair (Pichotta and Mooney, 2014).

In this period, there is still a set of works that attempted to encode script knowledge by constructing an event graph, which can express denser and broader connections among events. As a general rule, the nodes of the graph refer to the events, and the edges refer to the relationships between events. Different models build script graphs differently: Regneri et al. (2010) constructed a temporal script graph for a specific scenario by learning from crowd-sourced data; Orr et al. (2014) employed the hidden Markov model (HMM) to construct transition script graphs; Glavaš and Šnajder (2015) constructed event graphs using a three-stage pipeline to extract anchor, argument, and the relationship between event pairs from the text.

However, count-based methods determine the probability of an event by calculating co-occurrence counts, and learn only shallow statistical information rather than deep semantic properties. There are two main shortcomings: impoverished representations and sparsity issues. They treat events as one fixed unit, consequently failing to consider the compositional nature of an event and suffering from sparsity issues, which seriously restricts the ability of script learning systems to infer.

2.3 Deep learning based approaches (2014–)

The emergence of deep learning technology has brought NLP to a new era; modern script learning systems widely introduce neural networks and embeddings to counter the above shortcomings. The embedding mitigates the sparsity issue and provides a useful approach to compose predicates with their arguments. Deep learning based models use word embeddings to represent event components (i.e., a predicate and its arguments) and obtain the embedding of the entire event by composing the embeddings of its components, via a composition neural network. The relationship between events can also be measured based on vector distances in the embedding space.

As for specific initial experimental works, Modi and Titov (2014a, 2014b) first introduced embedding methods to represent events; Rudinger et al. (2015) tentatively trained a log-bilinear neural language model (Mnih and Hinton, 2007) and attained a remarkable improvement in NC tests; Granroth-Wilding and Clark (2016) employed word2vec, the vector-learning system of Mikolov et al. (2013), to learn embeddings.

The aforementioned works showed that deep learning based models were certainly more effective for script learning tasks. Then, how to design a capable neural network structure became the main challenge at this stage. More advanced mechanisms and more complex structures were introduced by the follow-up work to make further improvements.

Particularly, as for more advanced mechanisms, Pichotta and Mooney (2016a) first used a recurrent neural network (RNN) based method, long short-term memory (LSTM), to capture order information of an event chain and express interactions between long-term sequences. Hu et al. (2017) applied a contextual hierarchical LSTM (CH-LSTM) that can capture both the word sequence of a single event and the temporal order of the event chain.

As for more complex structures, Wang ZQ et al. (2017) incorporated the advantage of both LSTM temporal order learning and traditional event pair coherence learning, using LSTM with a dynamic memory network (Weston et al., 2015). Lv et al. (2019) employed self-attention mechanism (Lin ZH et al., 2017) to extract event segments which have richer semantic information than individual events

and introduce less noise than event chains.

Following the recent trend of using the pre-trained language model BERT (Devlin et al., 2019), a few novel works have introduced BERT as the basic word embeddings in script learning and obtained significant improvements. For instance, Zheng et al. (2020) replaced the traditional GloVe (Pennington et al., 2014) with BERT, and proposed a unified framework to integrate raw-text training, intra-event training, and inter-event training. Li ZY et al. (2019) introduced the transferable BERT (Trans-BERT) training framework, which can transfer not only general knowledge from large-scale unlabeled data but also specific knowledge from various semantically related supervised tasks.

Graph-based works also introduced neural networks and embeddings into their models during this period. For example, Zhao et al. (2017) embedded an abstract causality network into the vector space and applied it into a neural network to make stock market movement predictions; Li ZY et al. (2018) constructed the scaled graph neural network (SGNN) to model event interactions and learn event representations.

Recently, some notable works have attempted to encode richer script knowledge by introducing external information, such as injecting fine-grained event properties (Lee IT and Goldwasser, 2018), injecting nuanced multi-relations (Lee IT and Goldwasser, 2019), and injecting additional commonsense knowledge (Ding et al., 2019b).

3 Event representations

At the linguistic level, an event and its components can be verbalized by free-form narrative texts. We refer to the lexical realization of a script event as an event description, which consists of verb predicates and some noun phrases. The verb predicates describe the actions, while the noun phrases describe the related entities. We refer to the verb predicate and its associated noun phrases as event components, which can be extracted from event descriptions by NLP tools. For example, the text "Tom brought the book to Mary" is a description of the bringing book event, containing four components: a predicate "brought" and three noun phrases "Tom," "book," and "Mary." These components can be combined with a specific structure to form the representation of the event, such as bring(Tom, book, Mary).

The main aim of event representation is to represent the event with an appropriate structure, which can include essential components of the event and capture the implicit commonsense knowledge hiding behind the text description. Event representation is very important for script learning, because it specifies what we mean by an "event," and because it is the basic operational element of the subsequent script modeling process.

In addition to the differences in representation structure, different models have great differences in representation forms. Some models use discrete representations (i.e., treating the whole structured event representation as an unbreakable atomic unit), while others use distributed representations (i.e., treating the event as a dense vector of real values). In this section, however, we will focus mainly on the various representation structures; the relevant features of different representation forms will be discussed in detail in Section 4.2 onward.

3.1 Protagonist representation

Chambers and Jurafsky (2008) first represented a free-form event description as a structured <verb, dependency> pair, which we refer to as a protagonist representation. They assumed that although a script has several participants, there is a central actor, called the protagonist, who characterizes mainly the whole event chain. They extracted the narrative chain from texts, which involves a sequence of events that share a common protagonist.

They represented events as <verb, dependency> pairs, where "verb" refers to the predicate verb describing the event, and "dependency" refers to the typed grammatical dependency relationship between the verb and the protagonist, such as "subject," "object," or "prepositional." A narrative chain contains a series of events, so it can be represented as a chain of <verb, dependency> pairs, all related to a single protagonist.

For example, the narrative text "Jessie killed a man, she ran away and got arrested by the police in the street" produces a chain for the protagonist Jessie: (kill, subject), (run, subject), (arrest, object).

Chambers and Jurafsky (2008) made a preliminary attempt to use a structured event representation, and their protagonist representation has been adopted by various subsequent works (Chambers and

Jurafsky, 2009; Jans et al., 2012; Rudinger et al., 2015). However, it still has some flaws.

First, it solely concentrates on organizing event chains around a central participator, the protagonist. Thus, a script that contains multiple participants will redundantly produce many chains. For example, the narrative text "Mary emailed Jim and he responded to her immediately" yields two chains, a chain for Mary, (email, subject) (respond, object), and a chain for Jim, (email, object) (respond, subject).

Second, it treats subject-verb and verb-object separately. Thus, it lacks coherence between subject and object; in other words, it does not express interactions between different <verb, dependency> pairs produced from the same event.

Take the above narrative as an example. There is no connection between (email, subject) of Mary and (email, object) of Jim, even though they are both yielded from the same event "Mary emailed Jim."

Third, it considers only the <verb, dependency> pairs rather than other arguments; however, sometimes the meaning of the verb changes drastically with different arguments.

For example, consider the following events: "Jim performed a music" and "Jim performed a surgery." The same verb, "perform," has quite different meanings in these two events, but if using the protagonist representation, they will obtain the same representation, namely <perform, subject>. Furthermore, even if the meanings of the verbs are the same, the meanings of the events will be changed with different arguments. For example, if using the protagonist representation, the event "go to the church" will be the same as the event "go to sleep," because they both yield (go, subject).

3.2 Relational representation

Balasubramanian et al. (2013) addressed these flaws by representing an event as a relational <Arg1, Relation, Arg2> triple, inspired by the information extraction system OLLIE (Mausam et al., 2012). We refer to <Arg1, Relation, Arg2> as a relational representation, where Arg1 and Arg2 are the subject and object of the event description, respectively, and Relation is the relation phrase between Arg1 and Arg2. For instance, the event "He cited a new study that was released by UCLA in 2008" produces three tuples:

(1) (He, cited, a new study);

(2) (A new study, was released by, UCLA);

(3) (A new study, was released in, 2008).

A relational triple provides a more specific representation method, which aids in maintaining coherence between subject and object. Balasubramanian et al. (2013) used it to induce open-domain event schemas, which effectively overcame the coherence-lacking issue of the event schemas proposed by Chambers and Jurafsky (2009).

3.3 Multi-argument representation

Pichotta and Mooney (2014) proposed a similar approach, called multi-argument representation. They represented an event as a relational atom $v(e_{\mathrm{s}}, e_{\mathrm{o}}, e_{\mathrm{p}})$, where v is the verb, and e_{s}, e_{o}, and e_{p} are arguments of v, standing for subject relation, object relation, and prepositional relation to the verb, respectively. Any argument except v may be null (represented by "-"), denoting that no word stands in relation to the verb. For example, the text "Tom brought the book to Mary" yields the representation bring(Tom, book, Mary).

Multi-argument representation produces only a single sequence for a document. Given the same text as above, "Mary emailed Jim and he responded to her immediately," it yields only one chain: mail(Mary, Jim, -), respond(Jim, Mary, -).

A multi-argument representation can involve multiple entities and express a more specific meaning of an event. It can also capture pair-wise entity relations between events. In the example above, it can directly model the fact that Tom responded to Mary after Mary emailed him into a sequence chain, while the <verb, dependency> solely captures two discrete events.

Many subsequent works, such as Pichotta and Mooney (2016a), Granroth-Wilding and Clark (2016), Modi (2016), Wang ZQ et al. (2017), and Lv et al. (2019), adopted this representation form. Pichotta and Mooney (2016a) made slight improvements in their multi-argument representation by adding a preposition dimension, representing an event as a 5-tuple $(v, e_{\mathrm{s}}, e_{\mathrm{o}}, e_{\mathrm{p}}, p)$, where p is the preposition. So, the above text "Tom brought the book to Mary" generates the representation (bring, Tom, book, Mary, to).

3.4 Fine-grained property representation

Lee IT and Goldwasser (2018) held the view that fine-grained event properties, such as argument, sentiment, animacy, time, and location, can potentially enhance the event representation, so they suggested a fine-grained property representation. They represented an event as a 6-tuple <tok(e), subj(e), obj(e), prep(e), $f_1(e)$, $f_2(e)$>, which contains four basic components and two fine-grained properties.

The basic components refer to the former 4-tuple <tok(e), subj(e), obj(e), prep(e)>, where tok(e) stands for the <predicate, dependency> pair, which is the same as the protagonist representation. The other three elements (i.e., subj(e), obj(e), and prep(e)) are the same as the $v(e_{\mathrm{s}}, e_{\mathrm{o}}, e_{\mathrm{p}})$ of a multi-argument representation.

The two additional properties $f_1(e)$ and $f_2(e)$ refer to the sentence-level sentiment and animacy information, respectively. The sentence-level sentiment captures the overall tone of the event, consisting of three polarity labels (i.e., negative, neutral, and positive). The animacy information also consists of three types (i.e., animate, inanimate, and unknown), depending on whether the protagonist is a living entity.

For instance, given the narrative text "Jenny went to a restaurant and ordered a salad," the fine-grained property representation for the protagonist Jenny will be <(go, subj), jenny, -, restaurant, neural, animate>, <(order, subj), jenny, salad, -, neural, animate>.

3.5 More detailed information

Although similar event representation structures are used, different works may process details in different ways.

Some works extract only the headword of entity mentions, while other works extract the entire mention span. For example, consider the following event: "Jenny went into her favorite restaurant." The former works may represent it as go (Jenny, restaurant), while the latter works may see it as go (Jenny, her_favorite_restaurant). The latter works can capture more nuanced information, but have a greater risk of sparsity.

When it comes to predicates, some works consider only verbs, while other works also consider predicate adjectives, because some adjectives may play an important role in predicting the next event. Consider two events: "Jerry was angry" and "Jerry punched her son." The predicate adjective "angry" is crucial information to predict the follow-up event correctly. These works will represent the predicate adjective as an argument to the verb be or become, e.g., "Jerry was angry" → be(Jenny, angry).

Some works also add more elements to predicates, such as negation labels, particles, and clausal complements (xcomp) (Lee IT and Goldwasser, 2019). With negation labels, the event "She does not like the dinner" will result in the predicate "not_like" rather than "like" alone. Also, because some verbs, such as "go" and "have," are meaningless, it is necessary to include xcomp. In that case, the event "She went shopping yesterday" yields the predicate "go_shop" rather than "go" alone.

4 Script learning models

The script learning model is the core component of the script learning system, and undertakes the main tasks of encoding and matching events. Generally, models take the representations of the structured events as the input and take event-matching results as the output. A trained script learning model can encode script knowledge involving the actions and the entities that participate in the actions, and calculate the matching results based on the encoded script knowledge and event-matching algorithm. In the next step, the matching results can be applied to specific tasks to draw inferences that are commonsensible but not explicitly stated. In most cases, the matching results refer to similarity or coherence scores, which can somewhat measure the relationship between events.

There are various ways to classify the existing script learning models; recalling from Section 2, based on modeling methods, they can be roughly divided into three categories: rule-based models, count-based models, and deep learning based models. Rule-based models use complicated handcrafted rules to model script knowledge of specific domains; count-based models use statistical counting approaches to automatically learn broad-domain script knowledge from a large text corpus; deep learning based models introduce the neural network and embledding to capture richer script knowledge and overcome the sparsity issues.

From the perspective of different structures used to handle scripts, they can also be divided into three other categories: event pair based models, event chain based models, and event graph based models. Pair-based models focus on calculating associations between pairs of events by count- or vector-based methods; chain-based models capture the order information and long-term context information of the full event chain by RNN-based approaches; graph-based models extract script knowledge by constructing graph structures that can express denser and broader connections among events.

In this section, we discuss some representative models in detail, according to all of these six categories. Finally, we also introduce some other notable works outside the above categories that attempt to enhance traditional script learning models by injecting external knowledge.

Notably, because these categories are divided from various perspectives, there is some overlap between them inevitably. In addition, each script learning model may involve multiple categories, so when discussing specific categories below, we may emphasize some particular aspects.

4.1 Rule-based models

The handcrafted rule-based script learning models are the earliest use of script-related concepts in AI research. The seminal works include schemas by Rumelhart (1980), frames by Minsky (1975), semantic frames by Fillmore (1976), and scripts by Schank and Abelson (1977). Although these works established the core concept of script learning and made many beneficial attempts, they still have many obvious demerits and limitations. Below, we will briefly introduce and discuss these incipient works.

Schema (Rumelhart, 1980) is a broad concept of cognitive psychology and an important theoretical source of the script. A schema is a mental framework employed to organize and understand the world, and consists of a general template and some slots. The general template represents an abstract class containing the knowledge shared by all instances, while the slots can be filled in with different values to form a specific instance of this class. For example, a restaurant scenario has its own schema, which includes a sequence of events that take place in a restaurant, such as entering the restaurant, ordering food, eating food, paying the bill, and leaving the restaurant. When we go to a restaurant, we can activate the restaurant schema that is stored in our memory, and anticipate the upcoming events.

A frame (Minsky, 1975) is a variation of the schema, and is a data structure used in AI to represent knowledge about stereotyped situations. The frame has a hierarchical structure containing four different levels: syntactic surface frames, semantic surface frames, thematic frames, and narrative frames. Similar to the schema, the top levels (thematic frames and narrative frames) are fixed general templates with slots for different situations. In contrast, the bottom levels (surface syntactic frames and surface semantic frames) are specific instances that fill in slots with different values.

A semantic frame (Fillmore, 1976) can be roughly seen as a lexical-level Minsky frame, which was proposed to capture the abstract description of an individual activity and all of its possible roles. A semantic frame includes a set of predicates that can describe the main concept of the activity and the corresponding semantic roles, which can describe different entities involved in the activity. The same activity can be expressed via different surface realizations, with different predicates or sentence constructions. For example, consider the following three sentences:

(1) Microsoft purchased the Canadian company Maluuba.

(2) AI company Maluuba was bought by Microsoft on January 9, 2017.

(3) Microsoft purchased Maluuba for 30 million dollars.

Although different in surface forms, the core semantic meaning of them is similar, namely a commerce purchase activity. The purpose of the semantic frame is to capture the core semantic meaning across all the surface forms, by abstracting out the information about the main activity. The semantic frame also assigns proper semantic roles to the entities involved in the activity (e.g., Microsoft is the buyer role, and Maluuba is the role of the good). The semantic roles can help understand the meaning of the text by answering questions like "who did what to whom and by what means?" They are also useful for a lot of NLP tasks, such as question answering (Shen and Lapata, 2007), document summarization (Khan et al., 2015), and plagiarism detection (Osman et al., 2012).

The concept of script was first firmly proposed by Schank and Abelson (1977) to explain how commonsense knowledge about daily human activities can be used in language processing. They defined scripts as structured knowledge representations capturing the relationships between prototypical event sequences and their participants in a given scenario. In short, it is a sequence of prototypical events organized in temporal order. These events contain stereotypical human activities, such as eating in restaurants, cooking dinner, and making coffee.

To compare the relationship between the script and the concepts mentioned before, we can see the script as a specific schema or frame that focuses on stereotypical events in a particular scenario. We can also treat the script as multiple semantic frames, because it models a sequence of events rather than an individual activity.

The incipient script theory was primarily introduced to construct story understanding and generation systems (Schank and Abelson, 1977; Cullingford, 1978; DeJong, 1979; Schank, 1990). These script systems employ mainly complex handcrafted rules for modeling relations between events and use a set of separate models to treat different scenarios. Because rules are designed by humans, rule-based systems are limited to the bounded designer experience, and are extremely time-consuming. Although they could be applied in certain fields, their rule-based methods were characteristically limited to a few specific domains and did not scale to complex ones.

Some potential end-to-end connectionist methods were subsequently proposed to model script knowledge, such as DYNASTY (DYNAmic STory understanding sYstem) (Lee G et al., 1992) and DISCERN (DIstributed SCript processing and Episodic memoRy Network) (Miikkulainen, 1992, 1993). These works have some abilities in script understanding and inferring, but more rules require more computing power and more application scenarios require more training data. Due to the weak computational power and insufficient data at that time, the abilities were limited and these works did not achieve good generalization (Mueller, 1998; Gordon, 2001).

Nevertheless, these works are of great value to the subsequent works. From the structure and operation process of DISCERN, we can even see the rudiments of modern script learning systems. The input of DISCERN is a short narrative text about a stereotypical event, and a "lexico" module is used to map the input words into distributed representations. Then a "sentence parser" module processes each input sentence word by word to form the representation of the sentence. After that, a "story parser" module composes all the sentence representations to produce the representation of the story. Finally, the story representation is inputted into a "story generator" module and a "sentence generator" module to generate corresponding sentences about the story. The corresponding sentences can be used to solve story-related problems such as retrieving information, answering questions, and generating paraphrases. All the modules are trained separately, and subsequently trained jointly to fine-tune the modules.

The idea of employing distributed representations and combining the component representations to produce representations of the whole is similar to many modern deep learning based models. These deep learning based models widely employ distributed representations, and use embeddings to represent words, sentences, and events. They also create representations of a sentence by composing the words in the sentence, and create representations of an event by combining its components. In addition, the joint training and fine-tuning methods used by DISCERN are generally used by recent deep learning models. With improvements in computing power, model structure sophistication, and the surge of data volume, the ability to process and generalize these deep learning based models is far greater than that of DISCERN. We will discuss these models in detail in Sections 4.3–4.5.

4.2 Count-based models

Count-based script learning models greatly reduce the defects of rule-based models by automatically learning broad-domain script knowledge from a large text corpus. The key idea of count-based models is to measure the relationship of event pairs using statistical counting approaches.

Chambers and Jurafsky (2008) were the first to introduce a count-based approach for learning statistical script knowledge. They adopted PMI (Church and Hanks, 1990) to calculate the relationship scores over all the event pairs that occurred in the narrative chains.

As reviewed in Section 3.1, Chambers and Jurafsky (2008) used protagonists to represent events. They extracted the narrative chain from the corpus, which is a script-like event sequence that shares a common protagonist, and represented it as a series of <verb, dependency> pairs, all related to a common protagonist. Specifically, for each document in their training corpus, a coreference resolution was used to identify all the entities, and a dependency parser was used to identify all verbs that had a dependency relation with an entity. They connected all the <verb, dependency> pairs that shared the same entity as a subject, object, or preposition, to form a narrative event chain.

Chambers and Jurafsky (2008) followed the assumption that verbs sharing coreferring arguments are semantically connected. Thus, argument-sharing verbs are more likely to participate in the same narrative chain than verbs that are not sharing. Therefore, they measured the relationships between event pairs based on how often they shared grammatical arguments, and how often two predicate verbs occur in the same narrative chain.

PMI was adopted to calculate the specific scores of the relations. Given an occurring narrative chain $C = (c_1, c_2, ..., c_n)$, which contains n events, and a candidate event e as the possible new event, every event is represented as a <verb, dependency> pair. The PMI of C and e is the sum of the PMIs between the event e and each of the occurring events c_i in C, which can be formulated as

$$\begin{aligned}\mathrm{PMI}(C,e) &= \sum_{i=1}^{n} \mathrm{PMI}\left(c_i, e\right) \\ &= \sum_{i=1}^{n} \log \frac{P\left(c_i, e\right)}{P\left(c_i\right) P(e)}.\end{aligned} \tag{1}$$

The numerator is defined by (taking c_1 as an example)

$$P\left(c_1, e\right) = \frac{C\left(c_1, e\right)}{\sum_{c_i} \sum_{e_j} C\left(c_i, e_j\right)}, \tag{2}$$

where $C(c_1, e)$ can be obtained by counting the co-occurrence times of event pair (c_1, e) in the training data, regardless of the order.

The authors hypothesized that the most likely new event had the highest PMI score, so they did the above calculations for every candidate event in the training corpus and then chose the one with the highest PMI score as the prediction. The result can be obtained by maximizing the following expression:

$$\max_{0<j<m} \sum_{i=0}^{n} \mathrm{PMI}\left(c_i, e_j\right), \tag{3}$$

where n is the number of events in the occurring narrative chain, e_i is the ith event of the narrative chain, m is the number of candidate events in the training corpus, and c_j is the jth candidate event.

The work of Chambers and Jurafsky (2008) revived the trend of script learning, but their unsupervised framework still has some shortcomings. The framework ignores the temporal order, which may affect the performance, and has impoverished event representation, which cannot deal with multiple-argument events. Several researchers followed their framework and enhanced it in different aspects.

The first essential extension work was done by Jans et al. (2012). They introduced novel skip n-gram counting methods, an ordered PMI model and a bigram probability model, to replace the original PMI.

Following the insight that semantically related events do not have to be strictly adjacent, Jans et al. (2012) used skip-grams rather than regular n-grams to collect statistical data for the training model. The skip-grams method can reduce the sparsity and increase the size of the training data.

Particularly, the N-skip bigram strategy will find all event pairs that occur with 0 to N events intervening between them, from an event chain. Fig. 4 illustrates an example that contains event pairs collected by 0-, 1-, and 2-skip bigrams.

After obtaining the statistical data, the authors introduced two novel score functions to explicitly capture the temporal order information of event pairs. The first one is the ordered PMI (OP) score, a variation of the PMI, which explicitly takes into account the temporal order of event pairs in the chain. OP assumes that, in addition to the events occurring before the candidate events, the events occurring after the candidate events are still considered. So, given an insertion point, m, where the new event should be added, we can calculate the OP score as

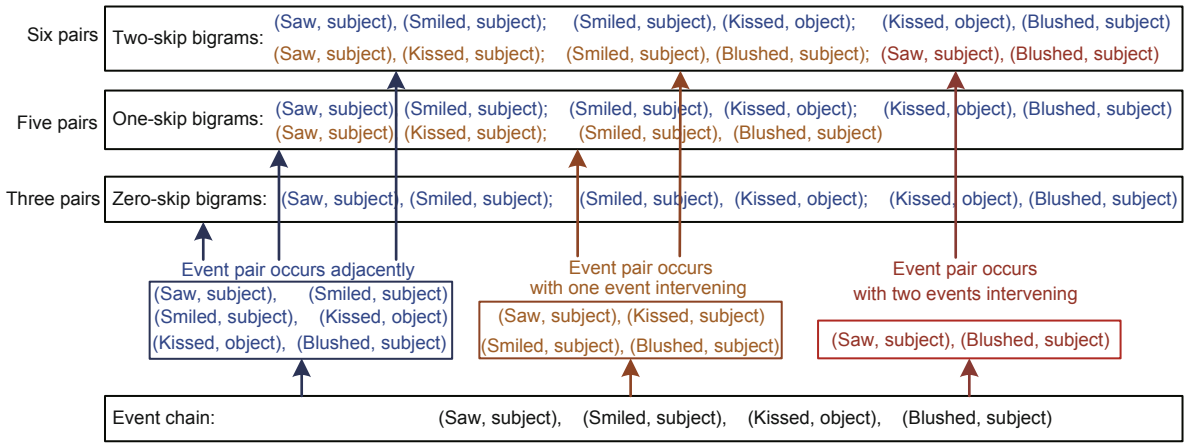

Fig. 4 Event pairs collected by N-skip bigrams. 0-skip bigrams collect three event pairs that occur adjacently. 1-skip bigrams collect five event pairs, including three 0-skip bigrams plus two 1-skip bigrams whose event pair occurs with one event intervening. 2-skip bigrams collect six event pairs, including three 0-skip bigrams and two 1-skip bigrams mentioned above, plus a 2-skip bigram whose event pair occurs with two events intervening

follows:

$$\begin{aligned}&\mathrm{OP}(C,e)\\&=\sum_{i=1}^{m}\mathrm{OP}\left(c_i,e\right)+\sum_{i=m+1}^{n}\mathrm{OP}\left(e,c_i\right)\\&=\sum_{i=1}^{m}\log\frac{P\left(c_i,e\right)}{P\left(c_i\right)P(e)}+\sum_{i=m+1}^{n}\log\frac{P\left(e,c_i\right)}{P(e)P\left(c_i\right)},\end{aligned}\tag{4}$$

$$P\left(c_1,e\right)=\frac{C\left(c_1,e\right)}{\sum_{c_1}\sum_{e_j}C\left(c_i,e_j\right)}.\tag{5}$$

OP treats the co-occurrence of two events, $C(c_i,e)$, as an asymmetric count (i.e., considering its order), whereas the PMI of Chambers and Jurafsky (2008) treats it as symmetric (i.e., regardless of the order).

The second score is the bigram probability (BP) score, which employs conditional probabilities $P(e|c_i)$ instead of $P(c_i,e)$ to compute the relational scores of the candidate event. The BP score is formulated as follows:

$$\begin{aligned}&\mathrm{BP}(C,e)\\&=\sum_{i=1}^{m}\mathrm{BP}\left(c_i,e\right)+\sum_{i=m+1}^{n}\mathrm{BP}\left(e,c_i\right)\\&=\sum_{i=1}^{m}\log P\left(e|c_i\right)+\sum_{i=m+1}^{n}\log P\left(c_i|e\right),\end{aligned}\tag{6}$$

$$P\left(e|c_i\right)=\frac{C\left(c_i,e\right)}{C\left(c_i\right)},\quad P\left(c_i|e\right)=\frac{C\left(e,c_i\right)}{C(e)}.\tag{7}$$

The authors compared these two novel scores with the PMI and observed that the 2-skip strategy plus the bigram probability function achieved the best empirical performance.

Another enhancement was made by Pichotta and Mooney (2014), who proposed multi-argument event representation $v(e_\mathrm{s},e_\mathrm{o},e_\mathrm{p})$ to replace the impoverished <verb, dependency> representation. As explained in detail in Section 3.3, the multi-argument representation directly models the interactions between entities and consequently captures richer semantic meanings.

The authors used the same 2-skip-bigram probability function as that in Jans et al. (2012), and made some further changes to make it applicable to their multi-argument representation. The previous works measured the relationships of event pairs by simply counting the co-occurrence times of each pair, but that is not sufficient for multi-argument representation.

For example, if there are two co-occurring event pairs, <ask(Mary, Bob, question), answer(Bob, -, -)> and <ask(Jerry, Tom, question), answer(Tom, -, -)>, we want to count two co-occurrence times of event pair <ask(X, Y, Z), answer(Y, -, -)>, for all distinct entities X, Y, and Z. However, if we keep the entities as they are and calculate the raw co-occurrence counts, we will count one time for pair <ask(Mary, Bob, question), answer(Bob, -, -)>, and another one time for pair <ask(Jerry, Tom, question), answer(Tom, -, -)>.

Pichotta and Mooney (2014) reformed the counting algorithm. They were motivated by the

observation that the essential factor in capturing the relationship between two events was their overlapping entities. In short, the central idea of their algorithm is that when counting the co-occurrence times of an event pair, the algorithm adds the number of co-occurrence times plus their overlapping entities. This algorithm has simple calculations and can capture pair-wise entity relationships between events.

They employed two evaluation tasks, predicting the verb with all arguments and predicting simpler <verb, dependency> pairs. The empirical results showed that modeling multi-argument events, instead of modeling <verb, dependency> pairs, could provide more power for both tasks.

Although both the representation and the performance were improved, the essential computing methods of the enhanced works were still based on the count, which had common sparsity issues.

Count-based models use discrete event representations, treat the whole structured representation of the event as an unbreakable atomic unit, and regard the components of event as fixed parts of the unit. The matching probability of an event pair is estimated by calculating the co-occurrence counts of two fixed event tuples. However, the underlying symbolic nature of tuple matching greatly limits the flexibility and generalization ability. Those works fail to express the semantic similarity between individual components of an event. For example, given two events cook(John, spaghetti, dinner) and prepare(Mary, pasta, dinner), since "cook" and "prepare" and "spaghetti" and "pasta" are semantically very similar, these two events should be semantically similar. However, count-based models would not take the similarity between these components into account, unless both events often occur in similar context.

In addition, count-based models measure the relationship among events merely by calculating co-occurrence counts of the entire event tuples. Consequently, the event tuple pairs that are never simultaneously observed in the training corpus are given zero probability (or a very small probability, if using a smoothing method), even if some of them are semantically probable. Therefore, count-based methods can make a respective good judgment about frequent events, but are less successful with rare ones. Moreover, count-based methods suffer from the curse of dimensionality (Bengio et al., 2003), because they rely on co-occurrence counts of events, but the number of instances required increases exponentially.

4.3 Deep learning based models

Recent deep learning based script learning models widely introduce neural networks and embedding to counter the shortcomings of count-based models. Embedding is a dense continuous vector of real values; it mitigates the sparsity issue and provides a useful approach for composing a predicate with its arguments.

In summary, as illustrated in Fig. 5, deep learning based models use embeddings to represent an event and its components (i.e., predicate and its arguments). The embedding of the entire event is computed by composing the word embeddings of its components, via a composition neural network. The event-matching works are based mainly on calculating conference scores or similarity scores of event embeddings, which can somewhat measure the semantic correlation between events. Both the parameters of the compositional process for computing the event embeddings and the parameters of the matching process for computing the similarities are automatically learned from the texts.

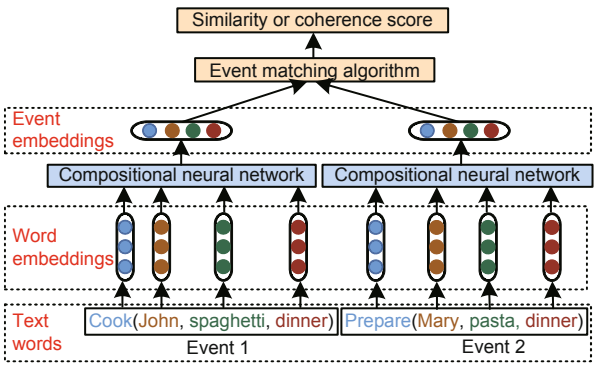

Fig. 5 Process of deep learning based models. An event and its components are represented by embeddings. The embedding of an event can be obtained by composing the word embeddings of its components, via a composition neural network. The event matching work is based on calculating conference scores or similarity scores of event embeddings

Learning and exploiting embeddings to represent words is beneficial for a range of NLP tasks, such as information extraction (Laender et al., 2002), semantic role labeling (Erk and Padó, 2008), word similarity analysis (Radinsky et al., 2011), word sense disambiguation (Navigli, 2009), and word and

spelling correction (Jones and Martin, 1997). The distributional hypothesis (Harris, 1954), namely, words that occur in the same context tend to have similar meanings, is the foundation of word embeddings. The word embeddings are learned so they can predict context words. Consequently, they encode some degree of semantic and syntactic properties of words: words that share similar contexts should be close to each other in the embedding vector space.

There are some works that focus on learning embeddings of phrases using a compositional model to compose the embeddings of individual words in the phrase (Baroni and Zamparelli, 2010; Socher et al., 2012). Motivated by them, Modi and Titov (2014a, 2014b) suggested that the embedding of an event can also be learned using a compositional neural network to compose the embeddings of individual components of the event.

In their works, the event was no longer treated as an unbreakable unit. Instead, it consisted of some separable components that contain a predicate and its arguments. All of the components were represented as embeddings, which were learned from predicting the prototypical event orderings. Because the embeddings are in the same and genetic dimensional vector space, we can easily obtain the embedding of the entire event by composing the embeddings of its components via a compositional neural network.

Similar to word embedding, event embedding can encode the semantic and syntactic properties of an event: the similar events should be close to each other in the embedding vector space, while dissimilar events should be far away from each other. With these properties, event embedding can be used to measure the relationship between almost any pair of events by calculating the vector distance between them, no matter whether they occurred together.

In addition, due to the continuous nature of real values, the embedding representation pushes the script learning system to learn the development patterns instead of the exact sequences of events. This in turn results in smooth modeling allowing the script learning system to predict unobserved events.

Although the above improvements effectively mitigated the sparsity issue of count-based models, Modi and Titov (2014a, 2014b) chose the event ordering task to learn embeddings and evaluate models, rather than the NC task (Chambers and Jurafsky, 2008), which was regarded as the predominant evaluation method at that time.

In fact, the NC task is essentially a language modeling task. For language models, the goal is to predict the center word by referring to the context words, while for script learning models, the goal is to predict the missing event by referring to the context events, and all of the events are made up of words. This observation strongly suggests the essential correlation between neural language models and script learning models. Following this, Rudinger et al. (2015) did exploratory work training a log-bilinear language model (LBL) (Mnih and Hinton, 2007) to resolve script learning tasks. Their model achieved a significantly better result than count-based models in the NC task, which led them to conclude that for script learning, either neural language models were a more effective approach or the NC task was not a suitable evaluation task. The event embeddings were also learned during the training process but merely regarded as a byproduct. The authors still considered the event as an unbreakable unit and directly learned representation vectors of them, so their event representations are distributed in external form but discrete in essence.

The alternative conclusions of Rudinger et al. (2015) were determined by subsequent works of Modi (2016) and Granroth-Wilding and Clark (2016). The performance of their neural network models overshadowed the previous count-based models, which indicated that neural network based methods are certainly more effective for script learning tasks. In addition, two novel and more appropriate evaluation tasks, the adversarial narrative cloze (ANC) task and the multiple choice narrative cloze (MCNC) task, were proposed to refine the NC test. We will discuss the details of NC, ANC, and MCNC in Section 5.

Modi (2016) proposed a probabilistic compositional model that had the same compositional neural network as that in Modi and Titov (2014b) to produce event embeddings, and a neural network based probabilistic model to encode the event sequence and predict the missing event. Word2vec, the vector-learning system of Mikolov et al. (2013), was also introduced to learn word embeddings.

Specifically, given the event sequence with a missing event ($e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_{k-1} \rightarrow ? \rightarrow e_{k+1} \rightarrow ... \rightarrow e_n$), the model is trained by predicting the missing event e_k. During training, instead of predicting the whole event at one time, the model

predicts the next events by incrementally using the context events and the previous prediction. As shown in Fig. 6, the model first predicts the verbal predicate of e_k using the embedding of context events. Then it predicts the position of the protagonist using the embedding of context events and the embeddings of the verbal predicate, predicted in the last step. In this way, the other arguments are predicted one by one. The authors considered this to be a more natural way of predicting events, because the verbal predicate information predicted before will influence the next possible arguments.

Notably, the model uses the embedding of the previous prediction to make the next prediction, instead of using gold embedding. In other words, if the previous prediction is wrong, the model will use the wrong embedding rather than the correct one stored in the training dataset to make the next prediction. The purpose of this design is to make the model more robust to noise and partially recover from mispredictions during the testing period. The authors also proposed a new testing task, the ANC task, to overcome the demerits of the standard NC task.

Similar to Modi (2016), Granroth-Wilding and Clark (2016) employed a compositional neural network and word2vec. In addition, they introduced a Siamese neural network to estimate the coherence scores of two event pairs. They evaluated their model using the MCNC task, a novel development of the NC test. Their experimental results showed that both the neural network and word2vec could yield further empirical improvements.

The above works not only prove the significant advantages of neural networks and embeddings in learning script knowledge, but also construct the basic framework of deep learning based script learning models. However, they focus only on learning associations between event pairs and ignore the temporal order of the event chain, so they are incapable of capturing order information and long-term context information. The following works introduce more advanced mechanisms, such as a recurrent neural network and attentional neural network, to model the full event chain. We will describe the details of these models in Section 4.5.

4.4 Pair-based models

Event pair-based models refer to models that focus on calculating associations between pairs of events. Because these models concern only the relations between event pairs, they are also known as weak-order models. Pair-based models are the dominant method in the literature before 2017 (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Granroth-Wilding and Clark, 2016).

The core idea of pair-based models is calculating the pair-wise event scores by various approaches. These scores can somewhat measure the semantic

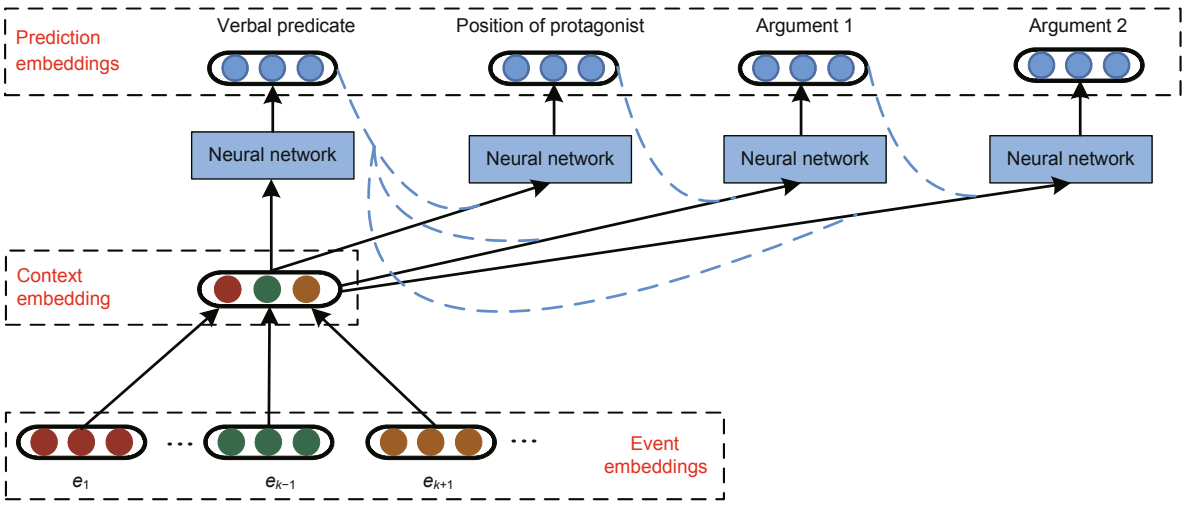

Fig. 6 Process of incrementally predicting the next event. First, the verbal predicate is predicted using context embedding. Then, the protagonist position is predicted using context embedding and the predicted verbal predicate embedding. In this way, the other arguments are predicted one by one using the context events and the previous prediction

correlation and compatibility between two events, namely, how strongly they are expected to appear in the same chain. These approaches include traditional count-based methods, such as PMI (Chambers and Jurafsky, 2008) or skip bigram probabilities (Jans et al., 2012), and vector-based methods, such as the Siamese network (Granroth-Wilding and Clark, 2016) or vector distance.

In a harder situation, when measuring the relations of event pairs that have subtle differences in surface realizations, the event embeddings, produced from widely used compositional neural networks, do not work well. They usually combine event components by concatenating or adding their word embeddings, and then applying a parameterized function to map the summed vector into the event embedding space. However, due to the overlap of words and the additive nature of parameters, it is hard to encode subtle differences in an event's surface realizations.

For example, the event pair "John threw bomb" and "John threw football" is semantically far apart, because they indicate entirely different scenarios (i.e., sports and terrorism). However, because their subject and predicate are the same, the embeddings of them, produced by compositional neural networks, are likely to be close in the vector space, which incorrectly indicates that they are closely related. On the contrary, the event pair "John threw bomb" and "John attacked embassy" is semantically close. Even though these two events have very different surface forms, they do not share similar word vectors, so their embeddings may be distinct, which incorrectly indicates that they have a distant relation.

To distinguish the relations of such event pairs, we need to understand deeper scenario-level semantics, of which the interactions between the predicate and its arguments are the key points. As the above example shows, the interactions of "football" or "bomb" with the predicate "threw" are what determine the precise semantics of the scenario. Weber et al. (2018) proposed a tensor-based composition model to combine the subject, predicate, and object, and then to produce the final event embedding. Tensor-based composition models can capture multiplicative interactions between event components and therefore represent deeper scenario-level semantics. With this ability, the event embeddings are sensitive to even a small change of a single component, which will be effective for multiple event-related tasks.

When resolving standard event prediction tasks (i.e., predicting which candidate event is most likely to occur given a context event sequence), pair-based models need to compute the correlation scores between a candidate event and a sequence of context events. Generally, they will compute the scores between the candidate event and each individual event in the chain, and then treat the average value of these scores as the final correlation score between the candidate event and the whole event chain. Finally, the candidate with the highest score is chosen as the final prediction. We can see this process in Fig. 7.

Although the relationships between the candidate event and each event in the chain have been considered, pair-based models are still limited to event pairs. They ignore the temporal order of events, and thus lack the ability to capture order information and long-term context information. As a result, improbable predictions may be produced by them. For example, given event <die, subj>, pair-based models may predict <live, subj> as the next event, simply because these two events often co-occur or have a high coherence score (Granroth-Wilding and Clark, 2016). This flaw will be countered by the event chain based models proposed by subsequent works.

Regardless of the above flaw, pair-based models still have their respective strengths compared to chain-based models. On one hand, they obviously inject less noise when modeling event chains with flexible order. On the other hand, because they are simpler, they are less likely to suffer from over-fitting issues and are less costly to employ. For these reasons, many recent script learning models have chosen to combine the advantages of pair- and chain-based approaches (Wang ZQ et al., 2017; Lv et al., 2019).

4.5 Chain-based models

Event chain based models are introduced to overcome the above-mentioned flaws. They use RNN-based approaches to model the full event chain and consider the temporal order information. The basis of chain-based models is the RNN, which is widely used for language models to express interactions between long-term sequences via intermediate hidden states. The basic process of chain-based models is shown in Fig. 8, in which an RNN provides embeddings with order information, and the event matching works are based on these embeddings.

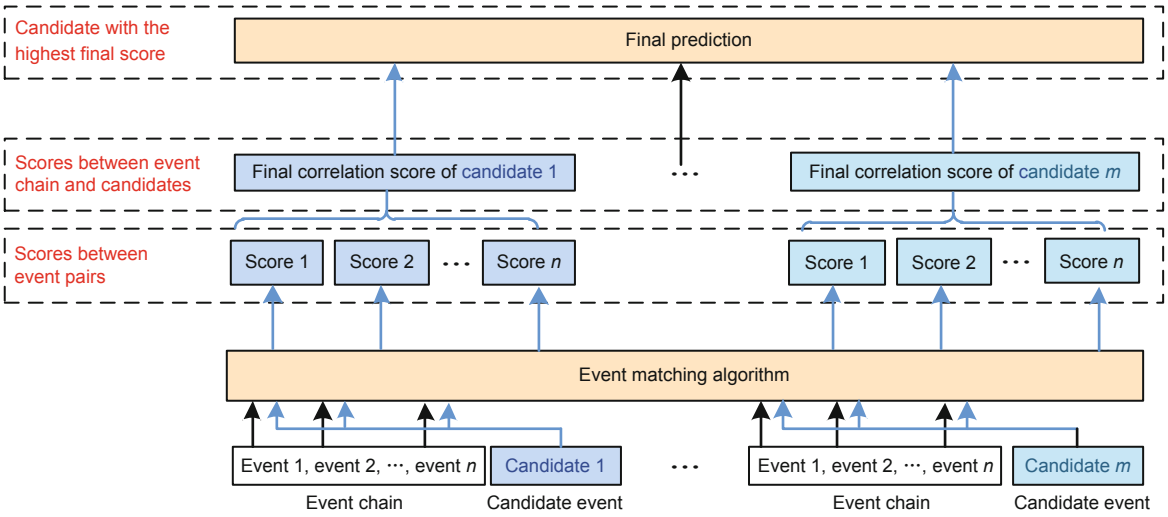

Fig. 7 Process of resolving the event prediction task by pair-based methods. First, calculate the scores between the candidate event and each individual event in the chain. Then, calculate the average value of these scores as the final score between the candidate event and the whole event chain. Finally, choose the candidate with the highest score as the final prediction

Nevertheless, basic RNNs suffer from a vanishing and exploding gradient problem. During the learning period, the gradient signal tends to either approach zero or diverge as it propagates through long time steps. So, LSTM (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) were devised to overcome this problem. They employ more complicated hidden units, which can provide encoded long-range propagation of events without losing long-term historical information.

A characteristic of script learning is that some events will be highly predictive of events far ahead in the chain, while some events are only locally predictive. Following this idea, Pichotta and Mooney (2016a) first adopted the LSTM model for the task of script learning. Their model represents an event with five components (i.e., 5-tuple $(v, e_{\mathrm{s}}, e_{\mathrm{o}}, e_{\mathrm{p}}, p)$, involving a verb predicate, subject, direct object, prepositional relation, and preposition). At each time step, one event component is inputted into the LSTM model. After inputting the entire event chain, the model will output the prediction of an additional event, which is also a component at each time step. Their work produced more accurate results, but it required handcrafted features to represent events and linguistic preprocessing to extract these features. In addition, it cannot predict additional events other than a given candidate set.

Two notable improved studies were conducted to overcome these shortcomings. The first one is by Pichotta and Mooney (2016b), in which the raw texts describing the previous event chain were directly used to predict the texts describing the missing event. The sentence-level RNN encoder-decoder model of Kiros et al. (2015) was employed to produce text predictions. Their word-level model trained by raw text was compared with an identical event-level model trained by structured event representations. The experimental results of the evaluation showed that, on the task of event prediction, there was only a marginal difference between the word-level model and the event-level model.

The second improved work was done by Hu et al. (2017). They developed an end-to-end LSTM model called CH-LSTM. Similar to Pichotta and Mooney (2016b), they took the raw texts describing the previous event chain as the input, and directly generated texts describing the possible next event.

CH-LSTM captures two-level structures of the event chain. At the first level, a single event is encoded using word embeddings as the input, and they are combined to produce event embedding as the output. At the second level, the event chain is encoded using event embeddings as the input, and the last hidden vector is outputted as the embedding of the entire previous event chain. CH-LSTM uses the event chain embedding to make a non-targeted

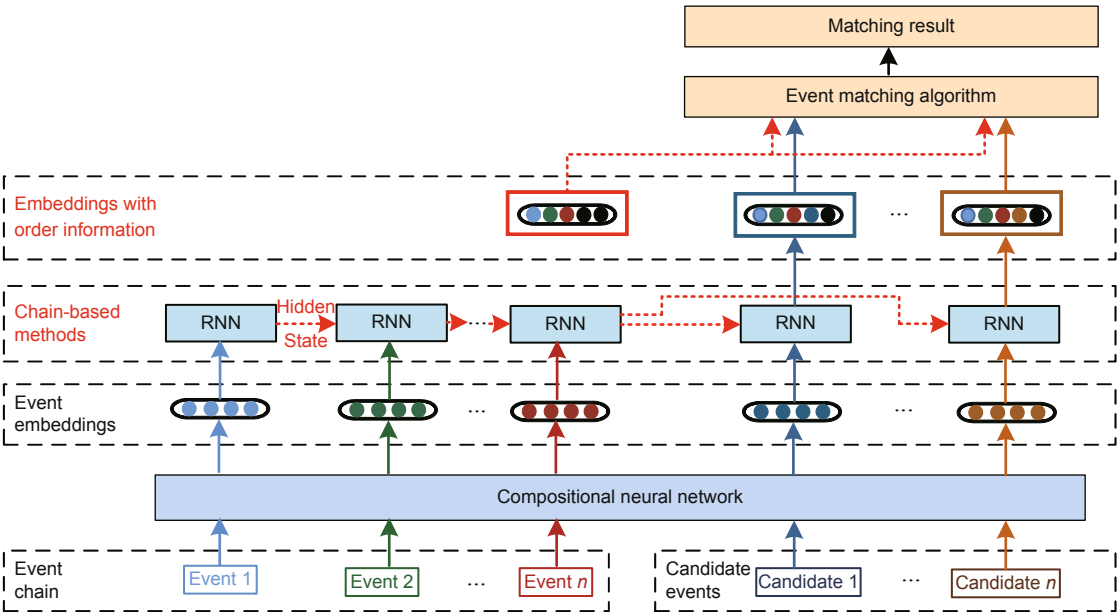

Fig. 8 Process of chain-based models. Inputting each event embedding inside an event chain into a recurrent neural network (RNN) can provide embeddings with order information via an intermediate hidden state. The event-matching process based on these embeddings can capture long-term context information

(unknown) next event prediction, which can generate new events that do not even exist in the training set.

Although the aforementioned works showed that LSTM captured significantly more event chain order information and made substantial experimental improvements compared to the previous methods, it may still suffer from the over-fitting problem, because an event chain in a script has a flexible order. To solve this issue, Wang ZQ et al. (2017) proposed a novel dynamic memory network model to integrate the advantages of chain-based temporal order learning and pair-based coherence learning. Fig. 9 shows an overview of their model.

As we mentioned at the end of Section 4.4, pair-based models are more practical when modeling event chains with flexible order, so Wang ZQ et al. (2017) used pair-based methods to measure the relationship between pairs of an occurring context event and a subsequent candidate event. To inject the order information, they represented events as the hidden states of LSTM, and used them to calculate the relation scores of event pairs.

In addition, considering the fact that the significance of different events varies for inferring a subsequent event, it is not appropriate to give an equal weight to each event as in the previous methods. Therefore, a dynamic memory network (Weston et al., 2015) was used to automatically induce different weights for each event in the inferring task.

The work of Wang ZQ et al. (2017) inspires us to resolve the script-related task by combining the advantages of pair- and chain-based models. Following this inspiration, we can exploit the event segment, which is a concept between individual events and event chains, to make further improvements.

An event segment refers to a part of script that involves several individual events in the same event chain. As Fig. 10 shows, given a script, for example, the script of restaurant visiting, there are various event segments within the chain, such as continuous event segment <"X read menu" "X order food" "X eat food"> or discontinuous event segment <"X read menu" "X order food" "X make payment">.

We can observe that both the individual events and the event segments are conducive to making more accurate subsequent event predictions. In particular, the individual event "X make payment" has a significant effect on predicting the subsequent event "X leave restaurant." Meanwhile, the event segment <"X read menu" "X order food" "X eat food"> has a strong semantic relation with the subsequent event "X leave restaurant." The event segment has a unique advantage; namely, it has richer semantic

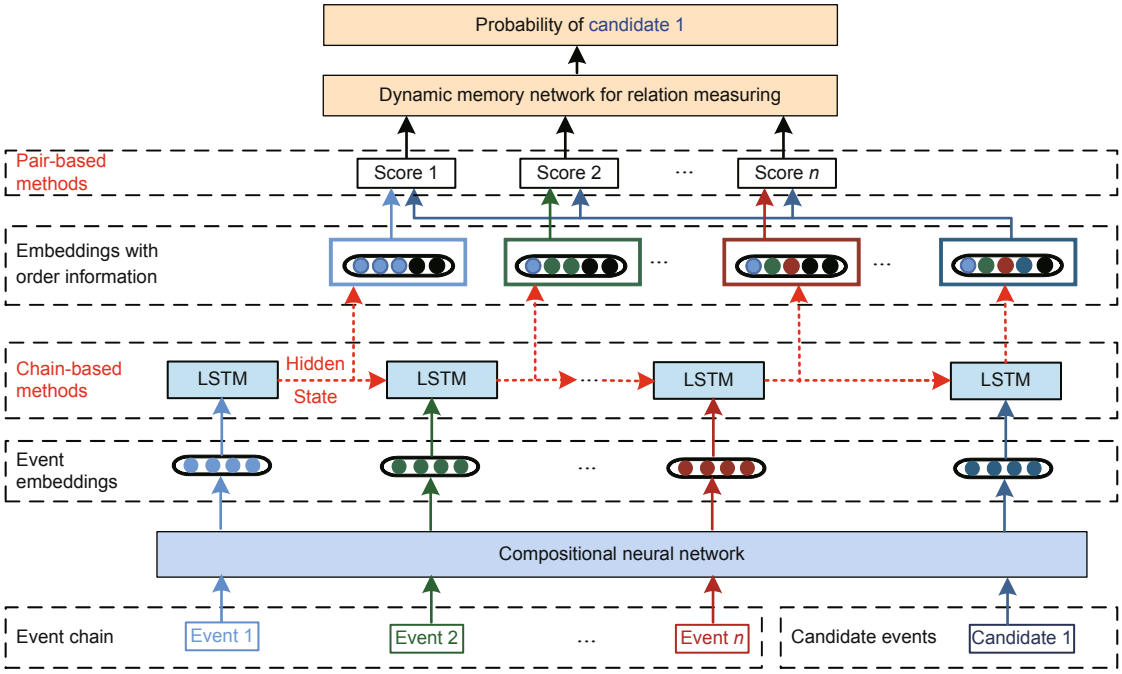

Fig. 9 Integrating the advantages of chain-based temporal order learning and pair-based coherence learning. For chain-based methods, the event embeddings of the event chain are first inputted into an LSTM to encode order information. For pair-based methods, the hidden states of LSTM are used to measure the relation between a pair of context event and subsequent candidate event

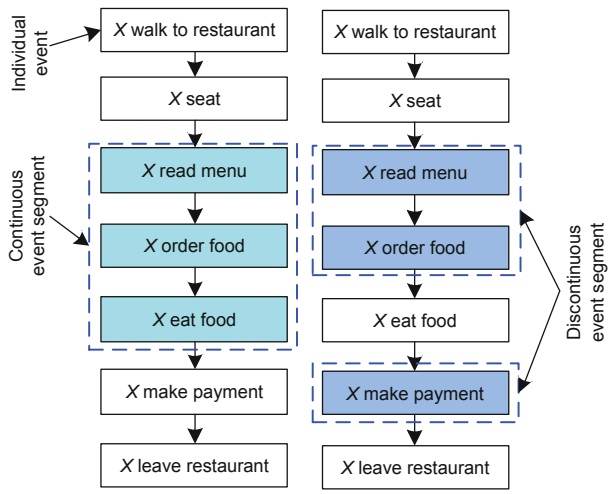

Fig. 10 Event segments of restaurant visiting. An event segment refers to several individual events in the same event chain. There are various event segments within an event chain, including a continuous event segment and a discontinuous event segment

information than individual events and introduces less noise than event chains, whereas previous approaches did not fully use them.

Lv et al. (2019) designed the SAM-Net model to fill in this gap. SAM-Net uses an LSTM to capture the order information of the event chain, and puts its hidden states into a self-attention mechanism (Lin ZH et al., 2017) to extract diverse event segments. SAM-Net can combine the information of individual events and the information of event segments. The former refers to the relations between a subsequent event and an individual event, while the latter refers to the relations between a subsequent event and an event segment.

Similar to Wang ZQ et al. (2017), SAM-Net considers that different events or event segments may have different semantic relations with the subsequent event. So, two attention mechanisms (Luong et al., 2015) were used to assign different corresponding weights for each individual event and event segment. The prediction of the subsequent event is based on the combination of these two attention mechanisms.

The standard MCNC task showed that SAM-Net apparently achieves better results compared to the previous models, and the idea of exploiting an event segment to draw more accurate inferences is a potential direction. Nevertheless, there is still no effective method for extracting event segments. The self-attention mechanism used by SAM-Net can

somewhat achieve this aim, but it is not accurate enough. Therefore, follow-up improvement research is anticipated.

4.6 Graph-based models

There is also a line of research that tries to express the relation between events and extract script knowledge by constructing a graph-based organization of events. As a general rule, the nodes of the graph refer to the events, and the edges refer to the relations between events; some edges may also have weights to measure the degree of the relationship or the transition probability between two events.

The graph structure is clearer and more intuitive from the human interpretability perspective to some extent. It is also closer to the organizational form of event development in reality. As in the real world, given any events, there are always many possible subsequent events, so there are also many different event chains occurring in the same scenario. When linking all of these events and event chains together according to their associations, a graph structure will be formed. Compared with pair- and chain-based models, graph-based models can express denser and broader connections among events, which contain richer script knowledge. In addition, once scripts are represented in a graph structure, a variety of existing graph-based algorithms can be used to solve script-related tasks.

The early work of Regneri et al. (2010) constructed the temporal script graph for specific scenarios from crowdsourced data. The data contains many different descriptions focusing on a single scenario. The scenario-specific paraphrase and temporal ordering information were extracted using a multiple sequence alignment (MSA) algorithm and connected to form a script graph.

Orr et al. (2014) employed the hidden Markov model (HMM) to construct a transition event graph. Their model clustered event descriptions into different types and learned an HMM over the sequence of event types. The structure and parameters were learned based on the expectation-maximization (EM) algorithm, and the script inferences were drawn according to the state transition probability.

Glavaš and Šnajder (2015) designed an end-to-end system with a three-stage pipeline, which can extract anchor, argument, and relation from natural language text, and link them to construct the event graph.

Here we discuss two more recent papers in detail. Zhao et al. (2017) modeled cause-effect relations between events into a graph structure. They proposed an abstract causality network on top of the specific events to reveal high-level general causality rules.

General causality patterns can be helpful in predicting future events correctly and reducing the sparsity issue caused by the causalities between specific events. The abstract causality network can discover general causalities on top of the specific causalities. For instance, the specific causality "a massive 8.9 magnitude earthquake hit northeast Japan on Friday → A large number of houses collapsed" can be abstracted to a more concise and general one: Earthquake → Houses collapsed.

The authors extracted event pairs with a causality relation from text snippets (in this paper, news headlines), and represented them by a set of verbs and nouns (e.g., "Hudson killed people"). The specific causality network is constructed by regarding each specific event as a node, and the causality between specific event pairs as an edge. The abstract causality network is constructed on top of the specific causality network. Particularly, the nouns of specific events are generalized to their hypernyms in WordNet (Miller, 1995) (e.g., the hypernym of the noun "chips" is "dishes"), and the verbs of specific events are generalized to their high-level classes in VerbNet (Schuler, 2005) (e.g., the verb "kill" belongs to the class "murder-42.1"). The nodes of the abstract causality network are created by replacing the original one with its generalization result, and the edges are also generated according to the specific one. For example, if there is an edge between the specific event "Hudson killed people" and the specific event "Hudson was sent to prison," then there will also be an edge between their corresponding abstract events, namely, the abstract event "murder-42.1, people" and the abstract event "send, prison". After constructing the abstract causality network, the event prediction task can be formulated as a link prediction task on the net. Given the cause (effect) event, the model will produce a list of corresponding effect (cause) events ranked by possibility.

Another notable graph-based model was introduced by Li ZY et al. (2018). They constructed a narrative event evolutionary graph (NEEG) to

express connections among events, and a scaled graph neural network (SGNN) to model event interactions and learn event representations.

As shown on the left of Fig. 11, NEEG uses the <verb, dependency> pair to stand for an event (i.e., a node of event graph), and regards each <verb, dependency> bigram as the relation between events (i.e., an edge of event graph). Their edges also have a weight that measures the transition probability between events and can be calculated from training data.

The task of inferring subsequent events can be solved by SGNN. Traditionally, a gated graph neural network (GGNN) was used to model the interactions among events. GGNN needs to input the whole graph, so it cannot effectively operate on a large-scale graph. To remedy the issue, SGNN borrows the idea of divide and conquer, and inputs a subgraph instead of the whole graph. As shown on the right of Fig. 11, the subgraph comprises only nodes of context events (red nodes) and a subsequent candidate event (green node), rather than all of the possible corresponding events. A graph network with a series of GGNNs is used to learn the representation of events. A Siamese network (Granroth-Wilding and Clark, 2016) is employed to compute the relatedness scores of the context and candidate events. The experimental results showed that their graph-based script learning model achieves high performance in standard MCNC tasks.

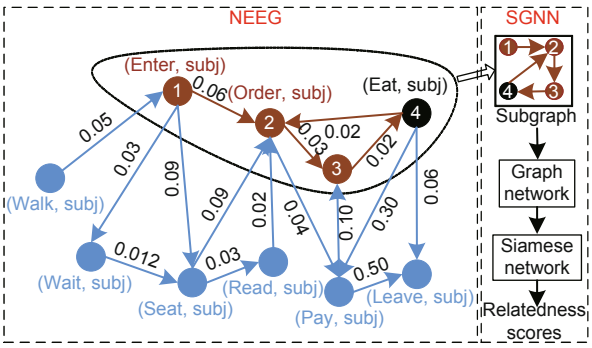

Fig. 11 Brief framework of NEEG and SGNN. NEEG regards <verb, dependency> pairs as nodes, and regards <verb, dependency> bigrams as edges. SGNN uses a subgraph instead of the whole graph as the input, and uses a series of GGNNs and a Siamese network to encode and match events. References to color refer to the online version of this figure

4.7 External knowledge models

At the end of this section, we will discuss some recent works that attempt to enhance traditional script learning models by injecting external knowledge, such as additional commonsense knowledge about intent and emotion of event participants, fine-grained properties of entities, or nuanced relations between events. Although these models do not strictly belong to one of the above-mentioned specific categories, they are enlightening and potentially useful, and some of them also achieve quite good performance.

Some fine-grained commonsense knowledge, such as the purpose or mental state of event participants, can be practical for understanding deep semantic meanings and generating better event representations. For instance, event "John threw bomb" and event "Jerry attacked embassy" have the same intent (i.e., bloodshed). This intent information can help map the embeddings of these two events into the neighbor vector space, even though they have very different surface forms. As for emotion information, consider the following events: "John threw bomb" and "John threw football." The former event may elicit feelings of anger or dread, while the latter may elicit feeling of joy. Based on this information, their embeddings should be far apart in the vector space, even though they are quite similar in the surface form.

However, Ding et al. (2019b) found that the event representation extracted from raw text lacks this commonsense knowledge, which is an important reason why it is difficult for script learning models to discriminate between the event pairs with subtle differences in their surface form. Hence, they used two commonsense knowledge corpora, namely Event2Mind (Rashkin et al., 2018) and ATOMIC (Sap et al., 2019), as the training dataset. In addition to an event description, there are multiple additional inference dimensions, produced by crowdsourcing, which are appended to each event in these two corpora. These inference dimensions contain richer commonsense knowledge, such as the cause and effect of an event, intents, and mental states of participants.

The authors leveraged intent and sentiment knowledge to help models produce more accurate embeddings and inferences of events. Specifically, they

used a neural tensor network to learn baseline event embeddings and defined a corresponding loss function to incorporate intent and sentiment information. Subsequently, the model jointly trained the combination of the loss functions on the event, intent, and sentiment, using the training data with intent and emotion labels. Experimental results showed that external commonsense knowledge could enhance the quality of event representations and improve the performance of downstream applications.

Lee IT and Goldwasser (2018) proposed a feature-enriched script learning model, featured event embedding learning (FEEL), and injected fine-grained event properties to enhance event embeddings. Following the idea that fine-grained event properties and relevant contextual information, such as argument, sentiment, animacy, time, or location, can potentially enhance the event representations, the authors injected two additional event properties, sentence-level sentiment and animacy information of the protagonist, into the script learning model.

The sentence-level sentiment captures the overall tone of the event, which can impact the probability of specific events. For example, the event "Jack likes the food" indicates a positive sentiment, increasing the possibility of "He tips the waiter" to be the next event. Animacy information of the protagonist is also an important factor that affects future events, because some events can be done only by living entities, and some events' meaning changes greatly with different types of animacy (e.g., "Jack is blue" and "the sky is blue").

FEEL represents an event by a 6-tuple $<\text{tok}(e)$, $\text{subj}(e)$, $\text{obj}(e)$, $\text{prep}(e)$, $f_1(e)$, $f_2(e)>$, where $f_1(e)$ and $f_2(e)$ refer to the sentence-level sentiment and animacy information, respectively. A hierarchical multitask model was used to jointly learn intra- and inter-event objectives. The intra-event objective (or local objective) expresses the connections between different components inside an individual event, namely, allowing the $\text{tok}(e)$, $\text{sub}(e)$, $\text{obj}(e)$, $\text{prep}(e)$, $f_1(e)$, and $f_2(e)$ to share information with each other. The inter-event objective (or contextual objective) expresses the relationship between two different events. The model was evaluated over three narrative tasks, including the traditional MCNC task and its two variants (MCNS and MCNE). The experimental results showed that these two properties could contribute improvements.

In their later work, Lee IT and Goldwasser (2019) made further improvement by introducing fine-grained multi-relations between events, such as Reason, Cause, and Contrast.

Previous models usually drew inferences based on vector similarity scores between event embeddings. However, when making commonsensible inferences, there are always many reasonable choices, so using an event similarity alone is too coarse to support some relevant inferences. For instance, given the event "Jenny went to her favorite restaurant," the high related possible events include "She was very hungry," "She ordered a meal," and many other choices. However, if we have more clues, for example, given that the type of relation between these two events is Reason, analogous to asking "Why did Jenny go there," then the event "She was very hungry" will obviously be a more reasonable choice. In the same way, if the type of relation is Temporal, analogous to asking "What did she do after that," clearly, the event "She ordered a meal" will be a better choice.

Considering that the nuanced types of relationships between events can make commonsense inferences more accurate, this paper presents a model to encode multiple nuanced relations, rather than only typical temporal relation or co-occurrence relations. As Fig. 12 shows, the authors extracted relation triplets $(e_{\text{h}}, r, e_{\text{t}})$ from the corpus, which consists of two related events $(e_{\text{h}}, e_{\text{t}})$ and the relation type between them (r), such as (e_{h}="Jenny went into her favorite restaurant," r=Reason, e_{t}="She was very hungry"). All of the $e_{\text{h}}, e_{\text{t}}$, and r are represented by embeddings, which can be jointly learned by the model.

The relation embeddings contain 11 types of relations, including a commonly used temporal relation and a co-occurrence relation, plus nine additional discourse relations. These discourse relations are introduced by an annotated corpus, Penn Discourse Tree Bank-2.0 (PDTB) (Prasad et al., 2008). The relation embeddings are learned by translation-based embedding models, which regard relations as translations in the embedding space. This paper adopted two of translation-based embeddings, TransE (Bordes et al., 2013) and TransR (Lin YK et al., 2015), to jointly learn embeddings for events and relations.

As for TransE, it embeds events and their relations in the same vector space, so that the distance

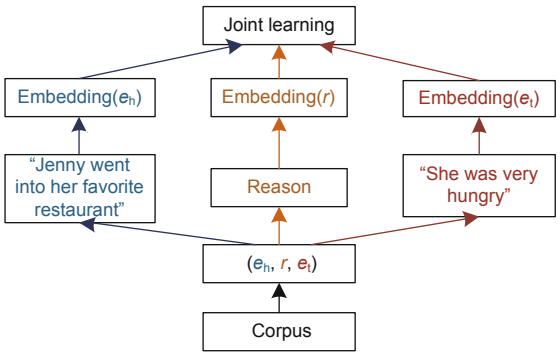

Fig. 12 Multi-relational script learning. Relation triplet (e_h, r, e_t) are extracted from the corpus. It consists of two related events (e_h, e_t) and the relation type between them (r). All of the e_h, e_t, and r are represented by embeddings, which can be jointly learned by the model

between event embeddings $(|e_h - e_t|)$ can directly reflect their relations (r). The relatedness score between events is formulated as

$$f_{\text{trass}}(t) = f_{\text{transe}}(e_h, e_t, r) = \|e_h - e_t + r\|_p^p, \quad (8)$$

where $e_h, e_t, r \in \mathbb{R}^{dr}$ are the embeddings from the event composition network. This formulation is a dissimilarity measure, with a lower score meaning a stronger relatedness.

Considering that relation and event are two completely different categories, it may not be appropriate to embed them into the same space. TansE has limitations in dealing with reflexive, 1-to-N, N-to-1, or N-to-N relations (Wang Z et al., 2014), so the authors also adapted TransR to encoder relations.

TransR separates the embedding space of relation $(r \in \mathbb{R}^{dr})$ and the embedding space of events $(e_h, e_t \in \mathbb{R}^{de})$. When performing operations, event embeddings can be mapped into the relation embedding space by multiplying a relation-specific parameter matrix $(M \in \mathbb{R}^{de*dr})$. The relatedness score between events is formulated as

$$\begin{aligned} f_{\text{trass}}(t) &= f_{\text{transe}}(e_h, e_t, r) \\ &= \|e_h M_r - e_t M_r + r\|_p^p. \end{aligned} \quad (9)$$

The model of Lee IT and Goldwasser (2019) was evaluated using several tasks, including three cloze tasks (MCNC, MCNS, MCNE), three relation-specific tasks, and a related downstream task. The results showed that the learned embedding could capture relation-specific information and improve performance for the downstream task.

5 Evaluation approaches

Essentially, the aim of the script learning system is to encode script knowledge and use it to draw reasonable inferences, so in the final process of script learning, some evaluation approaches are needed to test whether the model can effectively encode the script knowledge and exploit it.

In this section, we will introduce some representative evaluation approaches with specific tasks and corresponding metrics. The evaluation approaches can be employed to measure the capabilities of the script learning model by experimentally testing its performance on the task.

5.1 Narrative cloze test

The cloze test (Taylor, 1953) is a classic way to evaluate the language understanding ability of humans, by deleting a random word from a sentence and having a human attempt to fill in the blank.

Chambers and Jurafsky (2008) extended this approach to script learning and proposed the narrative cloze (NC) test. The NC test evaluates the inference ability of script learning models by holding out a single event from the event chain and letting the model predict the missing event given the remaining events. For example, given an event chain with a missing event: $(\text{walk}, \text{subject}) \rightarrow (\text{sit}, \text{subject}) \rightarrow (?) \rightarrow (\text{serve}, \text{object}) \rightarrow (\text{eat}, \text{subject}) \rightarrow (\text{leave}, \text{subject})$, the model is asked to predict the missing event (?) using the information of the remaining events. Particularly, it needs to compute the probability of each event in its entire event vocabulary and return a prediction list of likely events ranked by probability. The smaller the rank number of the correct event is, the more accurate the prediction will be. Chambers and Jurafsky (2008) used the average rank of all correct events as an evaluation metric. Obviously, a better performing model will have a lower average rank.

However, this metric is partially irrational; that is, it tends to punish the model with a long event vocabulary. If a large model with a long vocabulary and a small model with a short vocabulary both predict the correct event at the bottom of the list, then the large model will be more penalized than the small one, because the guess list of the large model is much longer.

Therefore, Jans et al. (2012) suggested Recall@N as a more reliable metric. It evaluates the percentage of missing events that fall under top-N predictions of the model. Specifically, if the rank number of the correct event is smaller than N, meaning that the first N events in the prediction list contain the correct one, then one score will be given. Recall@N is the average score of all predictions, namely, the percentage of prediction lists with correct events in their first N events. Contrary to the average rank metric, a better performing model will have a higher Recall@N.

Despite some relief, Recall@N still penalizes semantically reasonable predictions and solely awards events with exactly the same surface form. Therefore, Pichotta and Mooney (2014) introduced accuracy as a more robust metric, because it does not regard the event as an unbreakable atomic unit, and takes into account all of its components. Specifically, one point will be given if the model's top guess includes each component of the correct event. Then the score will be divided by the total number of possible points to yield a value between 0 and 1. More simply put, accuracy refers to the percentage of event components that are correctly predicted.

The NC test constructed the basic form of mainstream script learning evaluation methods, and it has been adopted by various subsequent works (Jans et al., 2012; Pichotta and Mooney, 2014, 2016a, 2016b; Rudinger et al., 2015). Despite the popularity, it still has two issues:

First, for any given event chains, there may be multiple subsequent choices that are possible and equally correct, but only one specific right answer. The NC test will penalize wrong answers equally, even if some of them are semantically plausible, which is somewhat arbitrary.

Second, it requires searching the entire event vocabulary; however, when considering complex event structures, such as multi-argument events, the vocabulary becomes extremely large, which leads to a burden of large computation.

5.2 Adversarial narrative cloze task

Modi (2016) attempted to address these issues by proposing the adversarial narrative cloze (ANC) task. This research pointed out that it is more realistic to evaluate script learning models that give credit for predicting semantically plausible alternatives as well.

In the ANC task, there are two event chains: one is the correct chain, and the other is the same but with one event replaced by random events. The models need to discriminate between real and corrupted event chains. Its form is as follows:

Correct chain:

(walk, subject) → (sit, subject)→ (eat, subject) → (leave, subject).

Incorrect chain:

(walk, subject) → (sit, subject)→ (eat, subject) → (swim, subject).

5.3 Multiple choice narrative cloze task

Granroth-Wilding and Clark (2016) also refined the NC test and proposed the multiple choice narrative cloze (MCNC) task, in which a multi-choice prediction rather than a ranking list is used. Particularly, a series of contextual events are given, and the model should choose the most likely next event from a set of optional candidates. There is only one correct subsequent event in the candidate set, while the other events are randomly sampled from the corpus with their protagonist replaced by the protagonist of the current chain. Its form is as follows:

Event chain: walk(X, restaurant)→ sit(X)→ order(X, food)→ __.

Optional choices:

C1: play(X, tennis).

C2: climb(X, mountain).

C3: ride(X, bicycle).

C4: eat(X, food).

C5: have(X, shower).

MCNC is a better fit for evaluating multi-argument events, because it can evaluate models' inference abilities without searching the entire event vocabulary. The way to calculate the accuracy of the MCNC task is also simpler and more intuitive, i.e., directly calculating the ratio of the correct prediction to the total test. In addition, in principle, the MCNC task can be completed by humans. Hence, task results from humans would be a valuable baseline for comparison with the model results.

5.4 Multiple choice narrative sequences task

The MCNC task is substituted for the NC task by most of subsequent related works (Wang ZQ et al., 2017; Li ZY et al., 2018; Weber et al., 2018; Lee IT

and Goldwasser, 2019; Lv et al., 2019). However, it does not capture the flow of the entire narrative chain. Therefore, it is hard to draw an inference over long event sequences. Lee IT and Goldwasser (2018) proposed two novel multiple-choice variants generalized from the MCNC task, to evaluate a model's ability to draw long sequence inferences instead of only predicting one event.

The first one is multiple choice narrative sequences (MCNS), which samples options for all the events on the chain, except for the first event used as the starting point. Its form is as follows: walk(X, restaurant) → __ → __ → __ → __.

The model needs to make a multiple choice narrative for every blank.

5.5 Multiple choice narrative explanation task

Another variant is multiple choice narrative explanation (MCNE), which gives both the first event and the final event and predicts all the intermediate events. Its form is as follows: walk(X, restaurant) → __ → __ → __ → leave(X, restaurant).

Note that regardless of the specific evaluation approach, the basic goal is to evaluate the ability of the script learning model to encode script knowledge and draw reasonable inferences. Therefore, the models with excellent evaluation performance should have learned rich commonsense knowledge and understood deep semantic meaning. Nevertheless, some researchers (Pichotta and Mooney, 2014; Mostafazadeh et al., 2016; Chambers, 2017) found an issue: some models are achieving high scores by optimizing performance on the task itself, instead of the learning process. For example, in the NC task, directly giving the prediction of the ranked list in accordance with the event's corpus frequency (e.g., always predicting common events "X said" or "X went") was shown to be an extremely strong baseline.

5.6 Story cloze test

To alleviate the above issues, Mostafazadeh et al. (2016) introduced the story cloze test (SCT), an open task, to help models make a more comprehensive and deeper evaluation. They collected a corpus full of commonsense short stories written in five sentences. When performing an evaluation task, four sentences are given as context, and models need to choose one correct ending of the story from two alternative sentences. For example:

"Karen was assigned a roommate in her first year of college. Her roommate asked her to go to a nearby city for a concert. Karen agreed happily. The show was absolutely exhilarating."

Given the above context, the model needs to choose between two alternative sentences, "Karen became good friends with her roommate" (right ending) and "Karen hated her roommate" (wrong ending).

Although the purpose of SCT and NC is relevant, there are some slight differences between the story on which SCT focuses and the script on which NC focuses. To some extent, the story is a kind of complex script that consists of more objections and multiple relations. SCT also calls for stronger inferential capability and understanding of the deeper-level story semantics, rather than shallow literal or statistical information, because it is asked to predict the entire sentence instead of an individual event. The experiment results also proved this. Mostafazadeh et al. (2016) chose some of the state-of-the-art script learning methods at that time for evaluation, but the experimental results showed that they all struggled to achieve a high score on SCT, and most of them were only slightly better than random or constant selection. This indicated that they have only learned shallow language knowledge instead of deeper understanding.

Recently, there have been a variety of works (Li Q et al., 2018; Zhou et al., 2019; Li ZY et al., 2019) attempting to resolve SCT, and some of them achieved high scores. For example, the three-stage transferable BERT training framework proposed by Li ZY et al. (2019) achieved rather good results with accuracy greater than 90%.

6 Conclusions and future directions

In this section, we conclude the main contents of this survey and briefly analyze the current script learning situation. In addition, we discuss some possible directions for future research, because we believe script learning research still has great potential, and there are many novel directions that need to be explored in the future.

6.1 Conclusions

This survey tried to provide a comprehensive review and introduce some representative script learning works. We first briefly introduced some basic script learning knowledge, including its definition, aim, significance, process, focus, and development timeline. Then we discussed in detail three main research topics: event representations, script learning models, and evaluation approaches. For each one, we introduced some typical and representative works, tried to compare their advantages and disadvantages, and summarized their development process. In the end, we discussed some possible directions for future research.

In summary, script learning is a relatively small but growing research direction with potential. It aims at encoding the commonsense knowledge of real life to help machines understand natural language and draw commonsensible inferences. Script learning systems have a long history and profound psychological foundation, and are also crucial components in constructing AI systems that are designed to achieve human-level performance.

Despite its significance, it still has some drawbacks and limitations, such as lacking a standard and high-quality corpus, lacking systematic evaluation frameworks, and lacking effective approaches for extracting and encoding the most important information of events. Or, more broadly, it lacks a deeper understanding of the mechanisms of human cognition and comprehension of commonsense knowledge. This knowledge is considered self-evident. On one hand, it seems that everyone naturally knows it, and therefore there is no need to explain it. On the other hand, it seems that we are unable to present appropriate and effective proof at present. Hence, there is still much work that needs to be done in the future.

For now, event chain based plus deep learning based models are the most popular script learning approaches. The trend in this direction is to further improve them by introducing more advanced mechanisms and more complex structures. Most of these works extracted events from the New York Times (NYK) portion of the Gigaword corpus as a dataset and evaluated their models by the MCNC test.

Furthermore, very recently, some other notable works have attempted to enhance traditional script learning models by constructing event graphs or injecting external knowledge. Some new evaluation frameworks, such as SCT, MCNS, and MCNG, and several novel corpora, such as InScript, Event2Mind, and ATOMIC, were also proposed. These innovative works are very enlightening and have potential, and some of them have achieved quite good performance. We are looking forward to more revolutionary improvements shortly.

Finally, we list some representative script learning works, as well as their dataset, event representations, models, and evaluation approaches, in Table 1. Because these works have quite different experimental setups, it is not likely that a fair comparison can be conducted among them. Consequently, we do not list the specific experimental results and detailed processes, but only the main methods they used.

6.2 Future directions

6.2.1 Establishing a standard corpus and evaluation system

There is no standard corpus or evaluation system for script learning. The majority of related works use NLP tools to automatically extract events from the news articles in the New York Times (NYK) portion of the Gigaword corpus, and the NC or MCNC test is used to evaluate the extraction results. However, this method has the following defects: the types of chosen raw texts can vary widely, the quality of extraction results relies heavily on NLP tools, and the content of texts may lack detailed information about quite common and mundane normal life events. In addition, Mostafazadeh et al. (2016) and Chambers (2017) found that automatically generating event chains for evaluation may not test relevant script knowledge. Some script learning models that achieve high test scores tend to capture frequent event patterns instead of script knowledge. If there is no effective evaluation system, we cannot accurately judge whether a model can really work. Therefore, the establishment of a standardized corpus and evaluation system is particularly important for the development of script learning.

There are a few other works (Regneri et al., 2010; Modi et al., 2017; Ding et al., 2019b; Lee IT and Goldwasser, 2019) that attempt to solve this issue using a crowdsourced corpus, such as OMICS (Gupta and Kochenderfer, 2004), PDTB (Prasad et al., 2008), InScript (Modi et al., 2016),

Table 1 A comparison of representative script learning works

Reference	Dataset	Representation	Model	Evaluation
Chambers and Jurafsky (2008)	NYKG	PR + DCR	CB + PB	NC
Jans et al. (2012)	Reuters + Andrew Lang Fairy Tale	PR + DCR	CB+ PB	NC
Balasubramanian et al. (2013)	NYKG	PR + DCR	CB + PB	Human evaluation
Pichotta and Mooney (2014)	NYKG	PR + DCR	CB + PB	Human evaluation
Modi and Titov (2014a)	OMICS + NYKG	MAR + DTR	DLB + PB	Event ordering + Event paraphrasing
Modi and Titov (2014b)	NYKG	MAR + DTR	DLB + PB	Event ordering
Rudinger et al. (2015)	NYKG	PR + DCR	DLB + PB	NC
Modi (2016)	Movies summary	MAR + DTR	DLB + PB	NC + ANC
Granroth-Wilding and Clark (2016)	NYKG	MAR + DTR	DLB + PB	MCNC
Pichotta and Mooney (2016a)	English language Wikipedia	MAR + DTR	DLB + CB	NC
Pichotta and Mooney (2016b)	English language Wikipedia	RTR + DTR	DLB + CB	NC
Modi et al. (2017)	InScript + Movies summary	MAR + DTR	DLB + PB	NC + ANC
Hu et al. (2017)	Chinese news event dataset of Sina News	RTR + DTR	DLB + CB	Other test
Wang ZQ et al. (2017)	NYKG	MAR + DTR	DLB + CB	MCNC
Zhao et al. (2017)	NYK	RTR + DTR	DLB + GB	Other test
Li Q et al. (2018)	ROCStories	RTR + DTR	DLB + EK	SCT
Li ZY et al. (2018)	NYKG	PR+ MAR + DTR	DLB + CB + GB	MCNC
Lee IT and Goldwasser (2018)	NYKG	FGPR + DTR	DLB + PB + EK	MCNC + MCNS + MCNE
Lee IT and Goldwasser (2019)	PDTB + NYKG	MAR + DTR	DLB + PB + EK	MCNC + MCNS + MCNE
Ding et al. (2019b)	Event2Mind + ATOMIC	MAR + DTR	DLB + PB + EK	MCNC
Lv et al. (2019)	NYKG	MAR + DTR	DLB + CB	MCNC
Li ZY et al. (2019)	SNLI + MNLI + MC_NLI + IMDB + Twitter + SWAG + ROCStories	RTR + DTR	DLB + EK	SCT

NYK: New York Times corpus; NYKG: NYK portion of the Gigaword corpus. PR: protagonist representation; DCR: discrete representation; DTR: distributed representation; MAR: multi-argument representation; RTR: raw text representation; FGPR: fine-grained property representation. CB: count-based; PB: pair-based; DLB: deep learning based; GB: graph-based; EK: external knowledge; NC: narrative cloze; ANC: adversarial narrative cloze; MCNC: multiple choice narrative cloze; SCT: story cloze test; MCNS: multiple choice narrative sequences; MCNE: multiple choice narrative explanation

Event2Mind (Rashkin et al., 2018), or ATOMIC (Sap et al., 2019). These crowdsourced corpora have advantages such as focusing on specific prototypical scenarios, containing rich script knowledge, having structure representations, and having clear annotated information. Specifically, the InScript Corpus (Modi et al., 2016) was designed for script learning, all of the relevant verbs and noun phrases were annotated with event types and participant types, and it contained many subtle actions describing a particular scenario. These advantages can help models effectively process data and learn script knowledge, as well as avoid many of the problems caused by unstructured texts (Chambers, 2017).

However, this is far from a trivial issue. On one hand, the crowdsourced corpora require domain expertise, professional knowledge, and plenty of manual annotation and quality control. Thus, they are small in size but prohibitive in construction cost. On the other hand, automatically understanding unstructured natural texts is a vital goal and a necessary capability of advanced AI systems. Therefore, collecting a standard corpus that has both quantity and quality and establishing an effective evaluation system are important directions for future research.

6.2.2 Constructing the event graph

As we discussed in Section 4.6, the graph structure is clearer and more intuitive from the human interpretability perspective, and to some extent, it is also closer to the organizational form of event development in reality. It can express denser and broader connections among events, compared with the event pair or event chain. In addition, once scripts are represented in a graph structure, a variety of off-the-shelf graph-based algorithms can be used to solve event-related tasks. However, there are only a few models that represent a script as graph structure and use graph-based approaches to solve event-related problems, so it is a research direction that deserves more attention in the future.

For example, the event logic graph (ELG) proposed by Ding et al. (2019a) is a potential event-related knowledge base that aims to discover the evolutionary patterns and logic of events. ELG integrates event-related knowledge into conventional knowledge graph techniques, which have been progressing fast recently, and organizes event evolutionary patterns into a commonsense knowledge base. The experimental results also show that ELG is effective for script-related tasks.

6.2.3 Using pre-trained language models

Recent works have shown that the two-stage framework that includes a pre-trained language model and fine-tuning operation can bring great improvements to many NLP tasks. Specifically, a language model can be pre-trained on large-scale unsupervised corpora, such as CoVe (McCann et al., 2017), ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), or GPT (Radford et al., 2019), and fine-tuned on target tasks, such as natural language inference (NLI) or reading comprehension. The pre-trained language models can learn universal language representations, which can lead to better generalization performance and speed up convergence for downstream NLP tasks. The pre-trained language models can also avoid heavy work and the complex process of training a model from scratch (Qiu et al., 2020).

It is natural to think that script learning tasks may also benefit from applying this framework. In addition, these novel pre-trained models can replace the embedding learning systems that have been widely used by script learning models for a long time, such as word2vec (Mikolov et al., 2013), glove (Pennington et al., 2014), or deepwalk (Perozzi et al., 2014). This direction is of great research value but there are only a few existing works. Here we briefly introduce two representative works.

Li ZY et al. (2019) did relevant work that designed a transferable BERT (TransBERT) training framework to solve an SCT task. In addition to a pre-trained BERT model, they introduced three semantically related supervised tasks (including NLI, sentiment classification, and next action prediction) to further pre-train the BERT model. Zheng et al. (2020) proposed a unified fine-tuning architecture (UniFA), which is a scalable and stackable framework consisting of four key components, i.e., BERT-base-uncased, raw-text, intra-event, and inter-event components, connected by a multi-step fine-tuning method. With the merits of cascaded training, multi-step fine-tuning can contribute to minimizing loss and also avoid being trapped in local optima. They also employed a scenario-level variational auto encoder (S-VAE) to implicitly represent scenario-level knowledge. In particular, S-VAE regards the event

chain as a dialog-generation process and further introduces a stochastic latent variable to guide the event representation.

6.2.4 Learning script by reinforcement learning

The script learning system is somewhat naturally similar to the reinforcement learning framework. The process of generating correct events one by one through an event chain is similar to the situation wherein an agent traverses through the space of events (Kaelbling et al., 1996; Li JW et al., 2016; Sutton and Barto, 2018).

In the reinforcement learning setup, an agent will search for all the possible states of the environment to achieve the highest rewards. The action of the agent refers to changing the state, and a corresponding reward will be given to the agent according to whether the action is good or bad. The policy of the agent refers to a sequence of actions taken by it, and the optimal policy is the policy that achieves the highest rewards.

In the script learning setup, each discrete state can represent an event. The action of the agent may refer to transferring to another event, based on current and previous states. If the agent transfers to a correct state (i.e., predicting the correct next event), then it will obtain a reward. The optimal policy of the agent may represent the prediction of the most likely event chain.

Reinforcement learning can help with many script learning tasks that are hard to solve now, for example, the event segment extraction task. As we mentioned in Section 4.5, Lv et al. (2019) used a self-attention mechanism to solve this task, but this approach is not accurate enough. We may design a reinforcement learning framework in which an agent traverses the event chain to extract the most valuable event segments. The state refers to whether the corresponding event is part of the extracted event segment, and the alternative action refers to changing the state and moving to the next event or keeping the state and moving to the next event. After traversing the whole event chain, the final extracted event segment will be used to predict the next event, and a corresponding reward will be given based on the accuracy of the prediction.

6.2.5 Injecting external fine-grained knowledge

As we discussed in Section 4.7, some recent works attempt to enhance traditional script learning models by injecting external knowledge. They chose fine-grained event properties, such as intent, sentiment, and animacy of event participants, or nuanced relations between events, such as Reason, Temporal, and Contrast.

These works indicate that fine-grained event properties and relevant contextual information such as argument, sentiment, animacy, time, or location, can potentially enhance script learning models, so other script learning models may also be improved by injecting external fine-grained knowledge.

This knowledge, however, is not necessarily the more the better. The experimental results of Lee IT and Goldwasser (2018) showed that although each property individually improves the performance, including too many properties might hurt it. Therefore, the questions of which and how many properties to choose are appropriate and need further study.

6.2.6 Building an interpretable system

Although deep learning technology has developed rapidly and reached impressive performance in many fields recently, its deep and complex nonlinear architecture makes the decision-making process highly non-transparent.

Deep learning models are known to be black boxes, so it is hard to explain the reason behind their decisions. As for deep learning based script learning systems, the lack of interpretability exists mainly in terms of two aspects: first, the output of the script learning system (i.e., the event prediction results) is hardly explainable to system users; second, the operation mechanism of how machines make a decision (i.e., the event encoding and matching algorithm) can hardly be investigated by system designers. Without letting users know why specific results are provided, the system may be less effective in persuading users to accept the results, which may further decrease the system's trustworthiness. Without letting designers know the mechanism behind the algorithm, it will be difficult for them to promote the system effectively.

In a broader sense, explainable artificial intelligence (XAI) (Arrieta et al., 2020) has recently sparked wider attention from the general AI

community. Many of the recent advances in machine learning have been trying to open the black box. For instance, Koh and Liang (2017) introduced a framework to analyze deep learning models based on influence analyses; Pei et al. (2017) proposed a white-box testing mechanism to explore the nature of deep learning models. The interpretability-related works will help us understand the exact meaning of each feature in a neural network and how they interact to produce the final outputs. However, the research in interpretable script learning systems is still in the initial stage, and there is much to be further studied.

Contributors

Yi HAN drafted the manuscript. Linbo QIAO, Jianming ZHENG, and Hefeng WU helped organize the manuscript. Dongsheng LI and Xiangke LIAO led the preparation of the manuscript. Yi HAN and Linbo QIAO revised and finalized the paper.

Compliance with ethics guidelines

Yi HAN, Linbo QIAO, Jianming ZHENG, Hefeng WU, Dongsheng LI, and Xiangke LIAO declare that they have no conflict of interest.

References

Arrieta AB, Díaz-Rodríguez N, Del Ser J, et al., 2020. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inform Fus*, 58:82-115. https://doi.org/10.1016/j.inffus.2019.12.012

Balasubramanian N, Soderland S, Mausam, et al., 2013. Generating coherent event schemas at scale. Proc Conf on Empirical Methods in Natural Language Processing, p.1721-1731.

Baroni M, Zamparelli R, 2010. Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space. Proc Conf on Empirical Methods in Natural Language Processing, p.1183-1193. https://doi.org/10.5555/1870658.1870773

Bengio Y, Ducharme R, Vincent P, et al., 2003. A neural probabilistic language model. *J Mach Learn Res*, 3:1137-1155. https://doi.org/10.5555/944919.944966

Bordes A, Usunier N, Garcia-Durán A, et al., 2013. Translating embeddings for modeling multi-relational data. Proc 26th Int Conf on Neural Information Processing Systems, p.2787-2795. https://doi.org/10.5555/2999792.2999923

Bower GH, Black JB, Turner TJ, 1979. Scripts in memory for text. *Cogn Psychol*, 11(2):177-220. https://doi.org/10.1016/0010-0285(79)90009-4

Chambers N, 2017. Behind the scenes of an evolving event cloze test. Proc 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-Level Semantics, p.41-45. https://doi.org/10.18653/v1/w17-0905

Chambers N, Jurafsky D, 2008. Unsupervised learning of narrative event chains. Proc 46th Annual Meeting of the Association for Computational Linguistics, p.789-797.

Chambers N, Jurafsky D, 2009. Unsupervised learning of narrative schemas and their participants. Proc Joint Conf of the 47th Annual Meeting of the ACL and the 4th Int Joint Conf on Natural Language Processing of the AFNLP, p.602-610. https://doi.org/10.5555/1690219.1690231

Chung J, Gulcehre C, Cho K, et al., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. https://arxiv.org/abs/1412.3555

Church KW, Hanks P, 1990. Word association norms, mutual information, and lexicography. *Comput Ling*, 16(1):22-29. https://doi.org/10.5555/89086.89095

Cullingford RE, 1978. Script Application: Computer Understanding of Newspaper Stories. PhD Thesis, Yale University, New Haven, CT, USA.

DeJong GF, 1979. Skimming Stories in Real Time: an Experiment in Integrated Understanding. PhD Thesis, Yale University, New Haven, CT, USA.

Devlin J, Chang MW, Lee K, et al., 2019. BERT: pre-training of deep bidirectional transformers for language understanding. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.4171-4186. https://doi.org/10.18653/v1/n19-1423

Ding X, Li ZY, Liu T, et al., 2019a. ELG: an event logic graph. https://arxiv.org/abs/1907.08015

Ding X, Liao K, Liu T, et al., 2019b. Event representation learning enhanced with external commonsense knowledge. Proc Conf on Empirical Methods in Natural Language Processing and the 9th Int Joint Conf on Natural Language Processing, p.4896-4905. https://doi.org/10.18653/v1/D19-1495

Erk K, Padó S, 2008. A structured vector space model for word meaning in context. Proc Conf on Empirical Methods in Natural Language Processing, p.897-906. https://doi.org/10.5555/1613715.1613831

Fillmore CJ, 1976. Frame semantics and the nature of language. *Ann N Y Acad Sci*, 280(1):20-32. https://doi.org/10.1111/j.1749-6632.1976.tb25467.x

Glavaš G, Šnajder J, 2015. Construction and evaluation of event graphs. *Nat Lang Eng*, 21(4):607-652. https://doi.org/10.1017/S1351324914000060

Gordon AS, 2001. Browsing image collections with representations of common-sense activities. *J Am Soc Inform Sci Technol*, 52(11):925-929. https://doi.org/10.1002/asi.1143

Granroth-Wilding M, Clark S, 2016. What happens next? Event prediction using a compositional neural network model. Proc 30th AAAI Conf on Artificial Intelligence, p.2727-2733. https://doi.org/10.5555/3016100.3016283

Gupta R, Kochenderfer MJ, 2004. Common sense data acquisition for indoor mobile robots. Proc 19th National Conf on Artifical Intelligence, p.605-610. https://doi.org/10.5555/1597148.1597246

Harris ZS, 1954. Distributional structure. *Word*, 10(2-3):146-162.

Hochreiter S, Schmidhuber J, 1997. Long short-term memory. *Neur Comput*, 9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu LM, Li JZ, Nie LQ, et al., 2017. What happens next? Future subevent prediction using contextual hierarchical LSTM. Proc 31st AAAI Conf on Artificial Intelligence, p.3450-3456. https://doi.org/10.5555/3298023.3298070

Jans B, Bethard S, Vulic, et al., 2012. Skip N-grams and ranking functions for predicting script events. Proc 13th Conf of the European Chapter of the Association for Computational Linguistics, p.336-344.

Jones MP, Martin JH, 1997. Contextual spelling correction using latent semantic analysis. Proc 5th Conf on Applied Natural Language Processing, p.166-173. https://doi.org/10.3115/974557.974582

Kaelbling LP, Littman ML, Moore AW, 1996. Reinforcement learning: a survey. *J Artif Intell Res*, 4:237-285. https://doi.org/10.1613/jair.301

Khan A, Salim N, Kumar YJ, 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Appl Soft Comput*, 30:737-747. https://doi.org/10.1016/j.asoc.2015.01.070

Kiros R, Zhu YK, Salakhutdinov R, et al., 2015. Skip-thought vectors. Proc 28th Int Conf on Neural Information Processing Systems, p.3294-3302. https://doi.org/10.5555/2969442.2969607

Koh PW, Liang P, 2017. Understanding black-box predictions via influence functions. Proc 34th Int Conf on Machine Learning, p.1885-1894.

Laender AHF, Ribeiro-Neto BA, Da Silva AS, et al., 2002. A brief survey of web data extraction tools. *ACM SIGMOD Rec*, 31(2):84-93. https://doi.org/10.1145/565117.565137

Lee G, Flowers M, Dyer MG, 1992. Learning distributed representations of conceptual knowledge and their application to script-based story processing. In: Sharkey N (Ed.), Connectionist Natural Language Processing. Springer, Dordrecht, p.215-247. https://doi.org/10.1007/978-94-011-2624-3_11

Lee IT, Goldwasser D, 2018. FEEL: featured event embedding learning. Proc 32nd AAAI Conf on Artificial Intelligence.

Lee IT, Goldwasser D, 2019. Multi-relational script learning for discourse relations. Proc 57th Annual Meeting of the Association for Computational Linguistics, p.4214-4226. https://doi.org/10.18653/v1/p19-1413

Li JW, Monroe W, Ritter A, et al., 2016. Deep reinforcement learning for dialogue generation. Proc Conf on Empirical Methods in Natural Language Processing, p.1192-1202. https://doi.org/10.18653/v1/D16-1127

Li Q, Li ZW, Wei JM, et al., 2018. A multi-attention based neural network with external knowledge for story ending predicting task. Proc 27th Int Conf on Computational Linguistics, p.1754-1762.

Li ZY, Ding X, Liu T, 2018. Constructing narrative event evolutionary graph for script event prediction. Proc 27th Int Joint Conf on Artificial Intelligence, p.4201-4207. https://doi.org/10.5555/3304222.3304354

Li ZY, Ding X, Liu T, 2019. Story ending prediction by transferable BERT. Proc 28th Int Joint Conf on Artificial Intelligence, p.1800-1806. https://doi.org/10.24963/ijcai.2019/249

Lin YK, Liu ZY, Sun MS, et al., 2015. Learning entity and relation embeddings for knowledge graph completion. Pro 29th AAAI Conf on Artificial Intelligence.

Lin ZH, Feng MW, Dos Santos CN, et al., 2017. A structured self-attentive sentence embedding. Proc 5th Int Conf on Learning Representations.

Luong T, Pham H, Manning CD, 2015. Effective approaches to attention-based neural machine translation. Proc Conf on Empirical Methods in Natural Language Processing, p.1412-1421. https://doi.org/10.18653/v1/d15-1166

Lv SW, Qian WH, Huang LT, et al., 2019. SAM-Net: integrating event-level and chain-level attentions to predict what happens next. Proc AAAI Conf on Artificial Intelligence, p.6802-6809. https://doi.org/10.1609/aaai.v33i01.33016802

Mausam, Schmitz M, Bart R, et al., 2012. Open language learning for information extraction. Proc Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, p.523-534. https://doi.org/10.5555/2390948.2391009

McCann B, Bradbury J, Xiong CM, et al., 2017. Learned in translation: contextualized word vectors. Proc 31st Int Conf on Neural Information Processing Systems, p.6297-6308. https://doi.org/10.5555/3295222.3295377

Miikkulainen R, 1992. Discern: a distributed neural network model of script processing and memory. University Twente, Connectionism and Natural Language Processing, p.115-124.

Miikkulainen R, 1993. Subsymbolic Natural Language Processing: an Integrated Model of Scripts, Lexicon, and Memory. MIT Press, Cambridge, USA.

Mikolov T, Chen K, Corrado G, et al., 2013. Efficient estimation of word representations in vector space. https://arxiv.org/abs/1301.3781

Miller GA, 1995. WordNet: a lexical database for English. *Commun ACM*, 38(11):39-41. https://doi.org/10.1145/219717.219748

Minsky M, 1975. A framework for representing knowledge. In: Winston PH (Ed.), The Psychology of Computer Vision. McGraw-Hill Book, New York, USA.

Mnih A, Hinton G, 2007. Three new graphical models for statistical language modelling. Proc 24th Int Conf on Machine Learning, p.641-648. https://doi.org/10.1145/1273496.1273577

Modi A, 2016. Event embeddings for semantic script modeling. Proc 20th SIGNLL Conf on Computational Natural Language Learning, p.75-83. https://doi.org/10.18653/v1/k16-1008

Modi A, Titov I, 2014a. Inducing neural models of script knowledge. Proc 18th Conf on Computational Natural Language Learning, p.49-57. https://doi.org/10.3115/v1/w14-1606

Modi A, Titov I, 2014b. Learning semantic script knowledge with event embeddings. Proc 2nd Int Conf on Learning Representations.

Modi A, Anikina T, Ostermann S, et al., 2016. InScript: narrative texts annotated with script information. Proc 10th Int Conf on Language Resources and Evaluation.

Modi A, Titov I, Demberg V, et al., 2017. Modeling semantic expectation: using script knowledge for referent prediction. *Trans Assoc Comput Ling*, 5(2):31-44. https://doi.org/10.1162/tacl_a_00044

Mostafazadeh N, Chambers N, He XD, et al., 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.839-849. https://doi.org/10.18653/v1/n16-1098

Mueller ET, 1998. Natural Language Processing with ThoughtTreasure. Signiform, New York, USA.

Navigli R, 2009. Word sense disambiguation: a survey. *ACM Comput Surv*, 41(2):10. https://doi.org/10.1145/1459352.1459355

Orr JW, Tadepalli P, Doppa JR, et al., 2014. Learning scripts as hidden Markov models. Proc 28th AAAI Conf on Artificial Intelligence, p.1565-1571. https://doi.org/10.5555/2892753.2892770

Osman AH, Salim N, Binwahlan MS, et al., 2012. Plagiarism detection scheme based on semantic role labeling. Proc Int Conf on Information Retrieval & Knowledge Management, p.30-33. https://doi.org/10.1109/InfRKM.2012.6204978

Pei KX, Cao YZ, Yang JF, et al., 2017. DeepXplore: automated whitebox testing of deep learning systems. Proc 26th Symp on Operating Systems Principles, p.1-18. https://doi.org/10.1145/3132747.3132785

Pennington J, Socher R, Manning C, 2014. GloVe: global vectors for word representation. Proc Conf on Empirical Methods in Natural Language Processing, p.1532-1543. https://doi.org/10.3115/v1/d14-1162

Perozzi B, Al-Rfou R, Skiena S, 2014. DeepWalk: online learning of social representations. Proc 20th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.701-710. https://doi.org/10.1145/2623330.2623732

Peters M, Neumann M, Iyyer M, et al., 2018. Deep contextualized word representations. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.2227-2237. https://doi.org/10.18653/v1/n18-1202

Pichotta K, Mooney R, 2014. Statistical script learning with multi-argument events. Proc 14th Conf of the European Chapter of the Association for Computational Linguistics, p.220-229. https://doi.org/10.3115/v1/e14-1024

Pichotta K, Mooney RJ, 2016a. Learning statistical scripts with LSTM recurrent neural networks. Proc 30th AAAI Conf on Artificial Intelligence, p.2800-2806. https://doi.org/10.5555/3016100.3016293

Pichotta K, Mooney RJ, 2016b. Using sentence-level LSTM language models for script inference. Proc 54th Annual Meeting of the Association for Computational Linguistics, p.279-289. https://doi.org/10.18653/v1/p16-1027

Prasad R, Dinesh N, Lee A, et al., 2008. The Penn discourse Treebank 2.0. Proc Int 6th Conf on Language Resources and Evaluation, p.2961-2968.

Qiu XP, Sun TX, Xu YG, et al., 2020. Pre-trained models for natural language processing: a survey. https://arxiv.org/abs/2003.08271

Radford A, Narasimhan K, Salimans T, et al., 2019. Improving language understanding by generative pre-training. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.4171-4186.

Radinsky K, Agichtein E, Gabrilovich E, et al., 2011. A word at a time: computing word relatedness using temporal semantic analysis. Proc 20th Int Conf on World Wide Web, p.337-346. https://doi.org/10.1145/1963405.1963455

Rashkin H, Sap M, Allaway E, et al., 2018. Event2Mind: commonsense inference on events, intents, and reactions. Proc 56th Annual Meeting of the Association for Computational Linguistics, p.463-473. https://doi.org/10.18653/v1/P18-1043

Regneri M, Koller A, Pinkal M, 2010. Learning script knowledge with web experiments. Proc 48th Annual Meeting of the Association for Computational Linguistics, p.979-988. https://doi.org/10.5555/1858681.1858781

Rudinger R, Rastogi P, Ferraro F, et al., 2015. Script induction as language modeling. Proc Conf on Empirical Methods in Natural Language Processing, p.1681-1686. https://doi.org/10.18653/v1/d15-1195

Rumelhart DE, 1980. Schemata: the building blocks of cognition. In: Spiro RJ (Ed.), Theoretical Issues in Reading Comprehension. Erlbaum, Hillsdale, p.33-58.

Sap M, Le Bras R, Allaway E, et al., 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. Proc AAAI Conf on Artificial Intelligence, p.3027-3035. https://doi.org/10.1609/aaai.v33i01.33013027

Schank RC, 1983. Dynamic Memory: a Theory of Reminding and Learning in Computers and People. Cambridge University Press, New York, USA.

Schank RC, 1990. Tell Me a Story: a New Look at Real and Artificial Memory. Charles Scribner, New York, USA.

Schank RC, Abelson RP, 1977. Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures. L. Erlbaum, Hillsdale, USA.

Schuler KK, 2005. VerbNet: a Broad-Coverage, Comprehensive Verb Lexicon. PhD Thesis, University of Pennsylvania, Pennsylvania, USA.

Shen D, Lapata M, 2007. Using semantic roles to improve question answering. Proc Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, p.12-21.

Socher R, Huval B, Manning CD, et al., 2012. Semantic compositionality through recursive matrix-vector spaces. Proc Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, p.1201-1211. https://doi.org/10.5555/2390948.2391084

Sutton RS, Barto AG, 2018. Reinforcement Learning: an Introduction (2nd Ed.). MIT Press, Cambridge, USA.

Taylor WL, 1953. "Cloze procedure": a new tool for measuring readability. *J Mass Commun Q*, 30(4):415-433. https://doi.org/10.1177/107769905303000401

Terry WS, 2006. Learning and Memory: Basic Principles, Processes, and Procedures. Allyn and Bacon, Boston, USA.

Tulving E, 1983. Elements of Episodic Memory. Oxford University Press, New York, USA.

Wang Z, Zhang JW, Feng JL, et al., 2014. Knowledge graph embedding by translating on hyperplanes. Proc 28th AAAI Conf on Artificial Intelligence, p.1112-1119. https://doi.org/10.5555/2893873.2894046

Wang ZQ, Zhang Y, Chang CY, 2017. Integrating order information and event relation for script event prediction. Proc Conf on Empirical Methods in Natural Language Processing, p.57-67.
https://doi.org/10.18653/v1/d17-1006

Weber N, Balasubramanian N, Chambers N, 2018. Event representations with tensor-based compositions. Proc 32nd AAAI Conf on Artificial Intelligence, p.4946-4953.

Weston J, Chopra S, Bordes A, 2015. Memory networks.
https://arxiv.org/abs/1410.3916

Zhao SD, Wang Q, Massung S, et al., 2017. Constructing and embedding abstract event causality networks from text snippets. Proc 10th ACM Int Conf on Web Search and Data Mining, p.335-344.
https://doi.org/10.1145/3018661.3018707

Zheng JM, Cai F, Chen HH, 2020. Incorporating scenario knowledge into a unified fine-tuning architecture for event representation. Proc 43rd Int ACM SIGIR Conf on Research and Development in Information Retrieval, p.249-258.
https://doi.org/10.1145/3397271.3401173

Zhou MT, Huang ML, Zhu XY, 2019. Story ending selection by finding hints from pairwise candidate endings. *IEEE/ACM Trans Audio Speech Lang Process*, 27(4):719-729.
https://doi.org/10.1109/TASLP.2019.2893499

Yi HAN, first author of this invited paper, received his BS and MS degrees from the National University of Defense Technology (NUDT), Changsha, China in 2016 and 2018, respectively, and is currently a PhD candidate at the College of Computer Science, NUDT. His research interests include natural language processing, event extraction, and few-shot learning.

Linbo QIAO, corresponding author of this invited paper, received his BS, MS, and PhD degrees in computer science and technology from NUDT, Changsha, China in 2010, 2012, and 2017, respectively. Now, he is an assistant research fellow at the National Lab for Parallel and Distributed Processing, NUDT. He worked as a research assistant at the Chinese University of Hong Kong from May to Oct. 2014. His research interests include structured sparse learning, online and distributed optimization, and deep learning for graph and graphical models.

Jianming ZHENG received the BS and MS degrees from NUDT, China in 2016 and 2018, respectively, and is currently a PhD candidate at the School of System Engineering, NUDT. His research interests include semantics representation, few-shot learning and its applications in information retrieval. He has several papers published in SIGIR, WWW, COLING, IPM, Cognitive Computation, etc.

Hefeng WU received the BS degree in Computer Science and Technology and the PhD degree in Computer Application Technology from Sun Yat-sen University, China in 2008 and 2013, respectively. He is currently a full research scientist with the School of Data and Computer Science, Sun Yat-sen University. His research interests include computer vision, multimedia, and machine learning. He has served as a reviewer for many academic journals and conferences, including TIP, TCYB, TSMC, TCSVT, PR, CVPR, ICCV, NeurIPS, and ICML.

Dongsheng LI received his BS degree (with honors) and PhD degree (with honors) in computer science from the College of Computer Science, NUDT, Changsha, China in 1999 and 2005, respectively. He was awarded the prize of National Excellent Doctoral Dissertation by the Ministry of Education of China in 2008, and the National Science Fund for Distinguished Young Scholars in 2020. He is now a full professor at the National Lab for Parallel and Distributed Processing, NUDT. He is a corresponding expert of *Frontiers of Information Technology & Electronic Engineering*. His research interests include parallel and distributed computing, cloud computing, and large-scale data management.

Xiangke LIAO received his BS degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China in 1985, and his MS degree from NUDT, Changsha, China in 1988. He is currently a full professor of NUDT, and an academician of the Chinese Academy of Engineering. His research interests include parallel and distributed computing, high-performance computer systems, operating systems, cloud computing, and networked embedded systems.