Engineering 6 (2020) 275–290

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Artificial Intelligence—Review

Progress in Neural NLP: Modeling, Learning, and Reasoning

Ming Zhou, Nan Duan, Shujie Liu, Heung-Yeung Shum *

Microsoft Research Asia, Beijing 100080, China

ARTICLE INFO

Article history:
Received 30 April 2019
Revised 30 August 2019
Accepted 13 October 2019
Available online 7 January 2020

Keywords:
Natural language processing
Deep learning
Modeling, learning, and reasoning

ABSTRACT

Natural language processing (NLP) is a subfield of artificial intelligence that focuses on enabling computers to understand and process human languages. In the last five years, we have witnessed the rapid development of NLP in tasks such as machine translation, question-answering, and machine reading comprehension based on deep learning and an enormous volume of annotated and unannotated data. In this paper, we will review the latest progress in the neural network-based NLP framework (neural NLP) from three perspectives: modeling, learning, and reasoning. In the modeling section, we will describe several fundamental neural network-based modeling paradigms, such as word embedding, sentence embedding, and sequence-to-sequence modeling, which are widely used in modern NLP engines. In the learning section, we will introduce widely used learning methods for NLP models, including supervised, semi-supervised, and unsupervised learning; multitask learning; transfer learning; and active learning. We view reasoning as a new and exciting direction for neural NLP, but it has yet to be well addressed. In the reasoning section, we will review reasoning mechanisms, including the knowledge, existing non-neural inference methods, and new neural inference methods. We emphasize the importance of reasoning in this paper because it is important for building interpretable and knowledge-driven neural NLP models to handle complex tasks. At the end of this paper, we will briefly outline our thoughts on the future directions of neural NLP.

1. Introduction

As an important branch of artificial intelligence (AI), natural language processing (NLP) studies the interactions between humans and computers via natural language. It studies fundamental technologies for the meaning expressions of words, phrases, sentences, and documents, and for syntactic and semantic processing such as word breaking, syntactic parsers, and semantic parsing and develops applications such as machine translation (MT), question-answering (QA), information retrieval, dialog, text generation, and recommendation systems. NLP is vital to search engines, customer support systems, business intelligence, and spoken assistants.

The history of NLP dates back to the 1950s. In the beginning of NLP research, rule-based methods were used to build NLP systems, including word/sentence analysis, QA, and MT. Such rules edited by experts were utilized in algorithms for various NLP tasks starting from MT. Normally, designing rules required significant human efforts. Furthermore, it is difficult to organize and manage rules when the number of rules is large. In the 1990s, along with the rapid development of the internet, large amounts of data became available, which enabled statistical learning methods to work on NLP tasks. With human-designed features, statistical learning models were learned by using labeled/mined data. The statistical learning method brought significant improvements to many NLP tasks, typically in MT and search engine technology. In 2012, deep learning approaches were introduced to NLP following deep learning's success in object recognition with ImageNet [1] and in speech recognition with Switchboard [2]. Deep learning approaches quickly outperformed statistical learning methods with surprisingly better results. As of the present, the neural network-based NLP (referred to as "neural NLP" hereafter) framework has achieved new levels of quality and has become the dominating approach for NLP tasks, such as MT, machine reading comprehension (MRC), chatbot, and so forth. For example, the Bible system from Microsoft achieved a human parity result on the Chinese-to-English news translation task of workshop on MT in 2017. R-NET and NLNet from Microsoft Research Asia (MSRA) achieved human-quality results on the Stanford Question Answering Dataset (SQuAD) evaluation task on both the exact match (EM) score and

* Corresponding author.
E-mail address: hshum@microsoft.com (H.-Y. Shum).

https://doi.org/10.1016/j.eng.2019.12.014

the fuzzy-match score (F_1 score). Recently, pre-trained models such as generative pre-training (GPT) [3], bidirectional encoder representations from transformers (BERT) [4], and XLNet [5] have demonstrated strong capabilities in multiple NLP tasks. The neural NLP framework works well for supervised tasks in which there is abundant labeled data for learning neural models, but still performs poorly for low-resource tasks where there is limited or no labeled data.

This paper reviews the notable progress of the neural NLP framework in three categories of efforts: ① neural NLP modeling aimed at designing appropriate network structures for different tasks; ② neural NLP learning aimed at optimizing the model parameters; and ③ reasoning aimed at generating answers to unseen questions by manipulating existing knowledge with inference techniques. Based on deep analysis of current technologies and the challenges of each of these aspects, we seek to identify and sort out future directions that are critical to advancing NLP technology.

2. Modeling

An NLP system consumes natural language sentences and generates a class type (for classification tasks), a sequence of labels (for sequence-labeling tasks), or another sentence (for QA, dialog, natural language generation, and MT). To apply neural NLP approaches, it is necessary to solve the following two key issues:

(1) Encode the natural language sentence (a sequence of words) in the neural network.

(2) Generate a sequence of labels or another natural language sentence.

From these two aspects, in this section, we will introduce several popularly used neural NLP models, including word embedding, sentence embedding, and sequence-to-sequence modeling. Word embedding maps words in the input sentences into continuous space vectors. Based on the word embedding, complex networks such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and self-attention networks can be used for feature extraction, considering the context information of the whole sentence to build context-aware word embedding, or integrating all the information of the sentence to construct the sentence embedding. Context-aware word embedding can be used for sequential labeling tasks such as part-of-speech (POS) tagging and named-entity recognition (NER), and sentence embedding can be used for sentence-level tasks, such as sentiment analysis and paraphrase classification. Sentence embedding can also be used as input to another RNN or self-attention network to generate another sequence, which forms the encoder-decoder framework for the sequence-to-sequence modeling. Given an input sentence, the sequence-to-sequence modeling can be used to generate an answer for a question (i.e., a QA task), or to perform a translation in another language (i.e., an MT task).

2.1. Word embedding and sentence embedding

Word/sentence embedding attempts to map words and sentences from a discrete space into a semantic space, in which the semantically similar words/sentences have similar embedding vectors.

2.1.1. Context-independent word embedding

To map a word into a continuous semantic vector, Ref. [6] proposed the continuous bag-of-words (CBOW) and skip-gram models, based on which the implementation tool word2vec is used to learn word-embedding vectors with a large monolingual corpus. As shown in Fig. 1 [6], the CBOW model predicts the central word using its surrounding words in a window, while the skip-gram model predicts the surrounding words of the given word. These two models are designed based on the principle of "knowing a word by the company it keeps" [7]. In addition, to utilize the benefits of global co-occurrence statistics and meaningful linear substructures, Ref. [8] proposed a global log-bilinear regression model (GloVe) to learn word embedding.

Word2vec and GloVe learn a constant embedding vector for a word; the embedding is the same for a word in different sentences. For example, we can learn an embedding vector for the word "bank," and the embedding remains the same, regardless of whether the word "bank" is used in the sentence "an ant went to the river bank" or in the sentence "that is a good way to build up a bank account." Ostensibly, the embedding of the word "bank" in the first sentence should be different from that of "bank" in the second sentence. To deal with this issue, context information of the sentence is used to predict a dynamic word embedding.

2.1.2. RNN-based context-aware word embedding

ELMo [9] leverages the bidirectional recurrent neural network (the long short-term memory (LSTM) network is particularly used) to model the context information, in which the word embedding is the concatenation of the hidden states of a forward RNN and a backward one, modeling the context at the left side and the right side, respectively. For example, given the input sentence "an ant went to the river bank." as shown in Fig. 2, the forward RNN first takes the first word "an" as the input and generates the first hidden state, which contains the information of the first word. When the second word "ant" is inputted, the RNN combines the information of the first hidden state and the second word to generate the second hidden state, which should contain the information of the first two words. When the word "bank" is inputted, the previous hidden state should contain all the previous information of "an ant went to the river." Taking it as context information, the new hidden state of "bank" contains dynamic information from the given sentence.

2.1.3. Self-attention-based context-aware word embedding

Ref. [3] proposed GPT, which leverages the self-attention network to train a multi-layer left-to-right language model. Compared with the RNN used in ELMo, which is also a left-to-right language model, the self-attention network used in GPT allows direct interaction between the current word and each previous word, which leads to better context representations. Ref. [4] proposed BERT, which leverages the self-attention network to jointly consider both the left and right context information in the sentence. Whereas the RNN processes the input sentence in a sequential order, from left to right or from right to left, as shown in Fig. 3, the self-attention network takes all the remaining words of the word "bank" as the context to build the context-aware word embedding, which is a weighted sum of all the representations of the words in the sentence, whose weight is calculated by normalizing and computing the similarity between the current word "bank" and all the words in the sentence. To consider the ordering information, a position index is also used to enrich the input by summing the word embedding and position embedding. To alleviate BERT's pre-train–fine-tune discrepancy issue, which means that artificial symbols such as [MASK] used by BERT during pre-training are absent from real data at fine-tuning time, Ref. [5] proposed XLNet, a pre-training method that enables the learning of bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order.

2.1.4. CNN-based context-aware word embedding

Both ELMo and BERT can consider all the context information in the input sentence to generate the dynamic embedding for a given

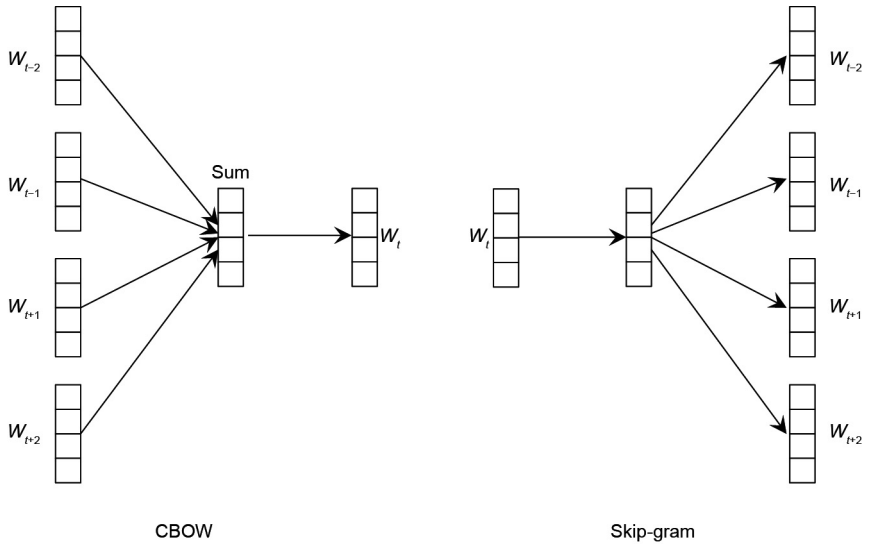

Fig. 1. Context-independent word-embedding methods [6]. CBOW: using the context words in a window to predict the central word. Skip-gram: using the central word to predict the context words in a window. W_t is the tth word in the sentence.

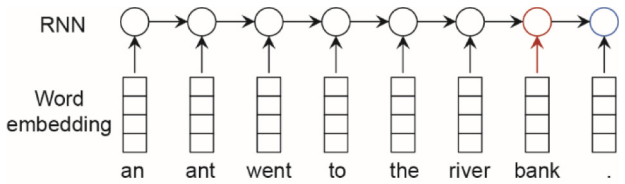

Fig. 2. RNN-based context-aware word embedding.

word. In contrast to using all the words as context, a CNN can be used to generate the dynamic embedding with only the surrounding words as context [10]. As shown in Fig. 4, the CNN uses a window to slide along the sequence of the input sentence. Using the embeddings of the words in the window, linear mappings (i.e., filters) are used to generate a representation vector to integrate the information of the input words. For example, to generate the dynamic embedding for the word "bank," a window with size 3 can be used to cover the span of "river bank," and the word "river" can be used to generate a disambiguating dynamic embedding for the word "bank."

2.1.5. Sentence embedding

Based on the representation of each word in the input sentence, the sentence representation can be obtained, via RNN, self-attention network, and CNN. For the RNN, the last hidden state (the blue one) should contain all the information in the sentence by consuming the input words one by one. For the self-attention network, a sentence-ending symbol, </S>, can be added, and its hidden state (the blue one) can be used as the representation of the whole sentence. For CNN, a max pooling layer can be used to select the maximum value for each dimension and generate one semantic vector (with the same size as the convolution layer output) to summarize the whole sentence, which is processed by a feed-forward network (FFN) to generate the final sentence representation. The generated sentence embedding can be used in other tasks, such as predicting the sentiment class (i.e., positive or negative) or predicting another sequence (MT).

2.2. Sequence-to-sequence modeling

2.2.1. Task of sequence-to-sequence modeling

Sequence-to-sequence modeling attempts to generate one sequence with another sequence as input. Many NLP tasks can be formulated as a sequence-to-sequence task, such as MT (i.e., given the source language word sequence, generate the target language word sequence), QA (i.e., given the word sequence of a question, generate the word sequence of an answer), and dialog (i.e., given the word sequence of user input, generate the word sequence of response).

2.2.2. Encoder–decoder framework

Ref. [11] proposed an encoder–decoder framework for sequence-to-sequence modeling. As shown in Fig. 5, the encoder–decoder framework contains two parts: an encoder and a decoder. The encoder is an RNN to encode the input sentence into a semantic representation, by consuming the words from left to right, one by one. The final hidden states should contain the

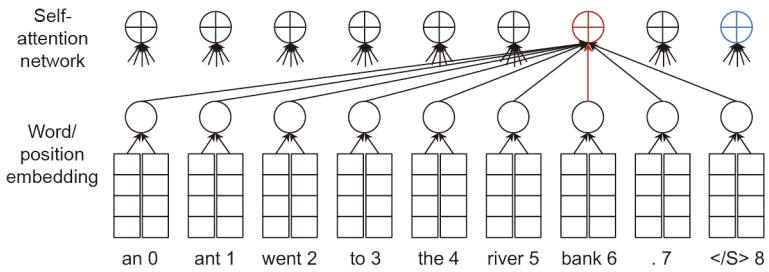

Fig. 3. Self-attention-based context-aware word embedding. </S>: sentence-ending symbol.

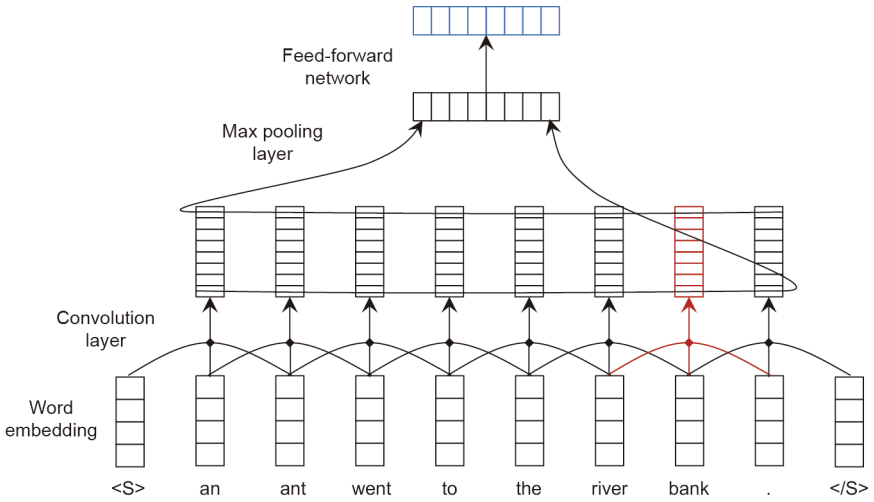

Fig. 4. CNN-based context-aware word embedding.

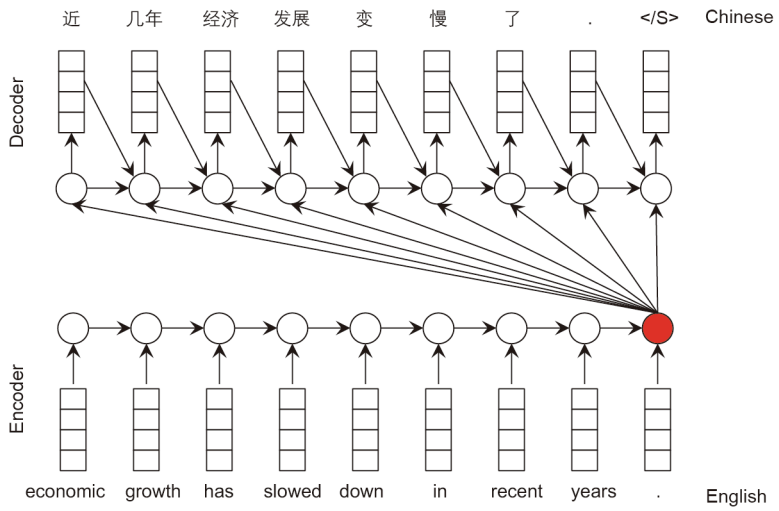

Fig. 5. Encoder–decoder framework for MT from English to Chinese.

information of all the words in the sentence, and are used as the context vector (i.e., representation) of the input sentence. Based on the context vector, another decoder RNN is used to generate the target sequence one word after another until the sentence-ending symbol (</S>) is generated. The decoder RNN takes the previous word, previous hidden state, and source sentence context vector as input to generate the current hidden state in order to predict the next target word.

The original encoder–decoder framework has several drawbacks: ① Only the last hidden state is used to model the source sentence. With such a size-fixed vector, it is difficult to model any sentence in the source language. ② The information of previous words consumed by the encoder is difficult for the RNN cell to maintain to influence the target word. ③ It is difficult to use only one context vector to predict all the words in the target sentence.

2.2.3. Attention-based encoder–decoder framework

In order to deal with these problems, the attention-based encoder–decoder framework [12] makes it possible for the neural network to pay attention to different parts of the input and to directly align the input sequence and the output result. As shown in Fig. 6, the attention mechanism leverages all the hidden states of the encoder and the previous decoder hidden state to compute a context vector, which is a weighted sum of the hidden state of the encoder, and the weights are computed and normalized by the similarity between the hidden states of the decoder and encoder. Together with the previous decoder hidden state and the previous target word, this context vector is used as the input to the decoder RNN to generate the next decoder hidden state. In this way, not only are all the encoder hidden states leveraged, but the decoder hidden state also directly relates to the corresponding encoder hidden states.

2.2.4. All-attention-based encoder–decoder framework

To use the strong modeling capacity of self-attention, Transformer [13] (as shown in Fig. 7) uses multi-head self-attention to replace the original attention mechanism and RNN cells in the encoder and decoder. Multi-head attention is a combination of attention networks. By projecting each query, key, and value into N vectors with linear layers, N attention networks are used to generate N context vectors, which are concatenated into one

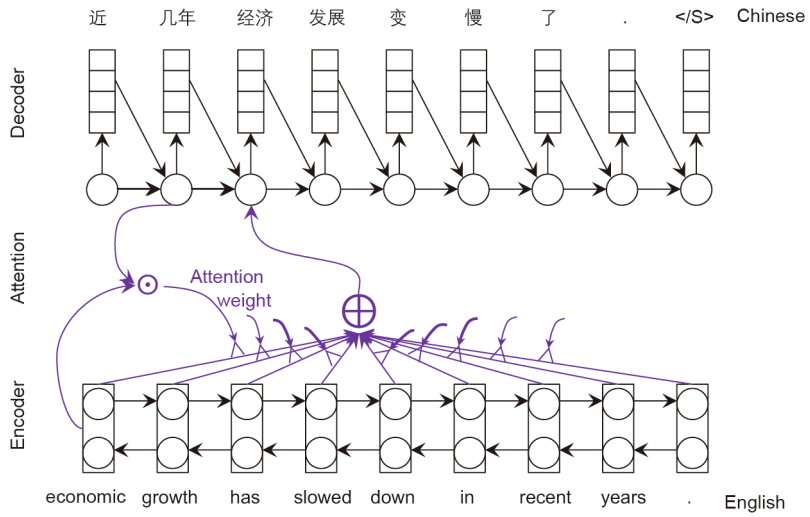

Fig. 6. Attention-based encoder–decoder framework for MT from English to Chinese.

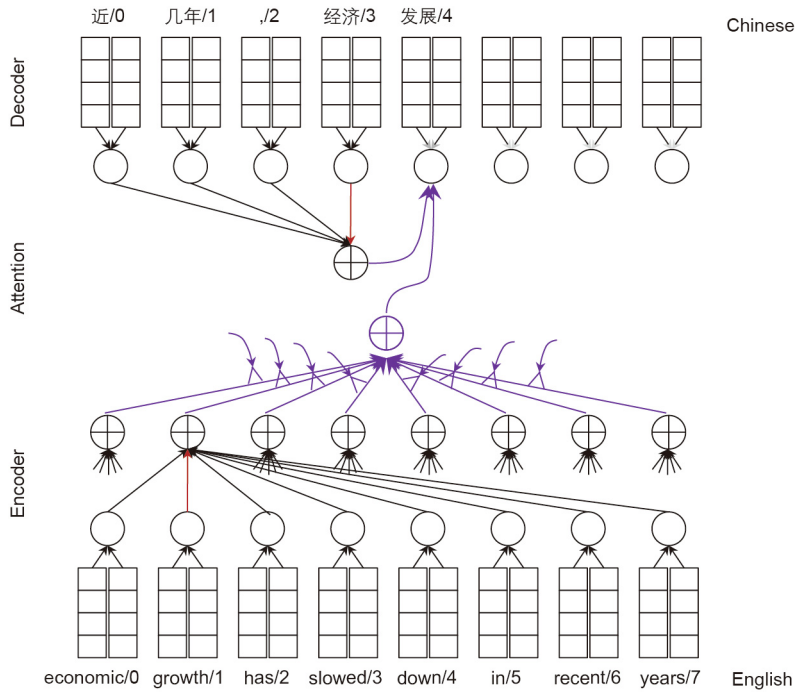

Fig. 7. All-attention-based encoder–decoder framework for MT from English to Chinese.

context vector. For the decoder self-attention, only the previous hidden states are used to compute the decoder context vector, because future words cannot be used during the inference.

2.3. Summary

In this section, we introduced the network structures to learn word embedding, sentence embedding, and sequence generation. In order to improve modeling for various NLP tasks, several directions still require further exploration:

- **Prior knowledge modeling.** Even though word-embedding methods that are trained with a huge amount of data can model certain kinds of commonsense knowledge [6], the problem of how to integrate the linguistic prior knowledge information, such as WordNet and HowNet, with specific NLP tasks should receive more attention [14].
- **Document/multi-turn modeling.** Leveraging sentence context by means of word embedding effectively improves the performance of various tasks, but the problem of how to model long-distance context, such as other sentences in the same document (or even in another document), is still an ongoing research work [15]. For example, given the English sentence "the mouse is on the table," it is impossible to tell whether the word "mouse" should be translated to the Chinese "鼠标" (a computer device) or the Chinese "老鼠" (an animal) based on the information in the given sentence. To do semantic disambiguation, the context of the document should be leveraged. Similarly, the problem of how to better model the context information in multi-turn tasks such as chatbots and dialog systems [16] is still challenging.
- **Non-autoregressive generation.** The current sequence-to-sequence model generates the output sentence one word at

a time in an autoregressive way, meaning that the word generated in the previous time-step is used as the input to generate the next word. Such an autoregressive generation process leads to the exposure bias problem, in which the mistake made in previous steps will be amplified in subsequent steps. To deal with this problem, non-autoregressive structures have been proposed, which show a performance drop compared with the autoregressive structure [17]. More attention should be paid to developing better non-autoregressive structures in the future.

3. Learning

New and efficient training algorithms have been proposed to optimize the large number of parameters in deep learning models. To train the neural network, stochastic gradient descent (SGD) [18] is often used, which is usually based on back-propagation methods [19]. Momentum-based SGD has been proposed in order to introduce momentum to speed up the training process. The AdaGrad [20], AdaDelta [21], Adam [22], and RMSProp methods attempt to use different learning ratios for different parameters, which further improves the efficiency and stabilizes the training process. When the model is very complex, parallel training methods are used to leverage many computing devices—even hundreds or thousands (central processing units, graphics processing units, or field programmable gate arrays). Depending on whether the parameters are updated synchronously or not, distributed training methods can be grouped into synchronous SGD and asynchronous SGD.

In addition to the progress that has been achieved in general optimization methods, better training methods have been proposed for specific NLP tasks. When large amounts of training data are available for rich resource tasks, using supervised learning methods, deep learning models achieve very good performance. For some specific tasks, such as MT for language pairs with a large volume of parallel data such as English–Chinese, neural models do a good job, sometimes achieving human parity results in the shared tasks. For many NLP tasks, however, it is difficult to acquire large amounts of labeled data. Such tasks are often referred to as low-resource tasks, including MT of sentiment analysis for rare languages. By using unlabeled data to enhance the models trained with a small amount of labeled data, semi-supervised learning methods can be used. Without any labeled data, unsupervised learning methods can be leveraged to learn NLP models. Another way to leverage unlabeled data is to pre-train models, which will be transferred to specific tasks with transfer learning. Instead of leveraging in-task labeled data, labeled data from other tasks can also be used with the help of multitask learning. If there is no data that can be used, human resources could be introduced to create the training data using active learning in order to maximize the model performance with a given budget.

3.1. Supervised learning

Based on labeled data, supervised learning attempts to learn the mapping model from input to output. For classification tasks, the supervised learning method can train the model by maximizing the log-likelihood of the correct label or minimizing the cross-entropy loss. For sequence-to-sequence tasks, it is also possible to maximize the likelihood of the correct target sentence conditioned on the input sentence. For a given sentence pair (x,y) in the training data, where $X=(x_1,x_2,\ldots,x_{|x|})$ is the input sentence and $Y=(y_1,y_2,\ldots,y_{|y|})$ is the output sentence, the likelihood of the conditional probability is defined as follows:

$$p_{\hat{e}}(Y|X)=\prod_{i=1}^{|y|}p_{\hat{e}}(y_i|y_{i-1},\ldots,y_1,X) \quad (1)$$

where $p_{\hat{e}}(y_i|y_{i-1},\ldots,y_1,X)$ is the softmax layer output of the decoder in the sequence-to-sequence model. Based on the likelihood, the log-likelihood loss function is defined as follows:

$$\text{LOSS}=\sum_{(X,Y)\in\text{TrainingData}}-\log p_{\hat{e}}(Y|X) \quad (2)$$

Based on this loss function, training algorithms (such as Adam or AdaDelta) can be used to optimize the parameters. Instead of only maximizing the generation probability of the golden target, in order to consider the task-specific error in the training process, Ref. [23] proposed minimum-risk (maximum-bilingual evaluation understudy (BLEU) [24]) training for the sequence-to-sequence generation model. The model is first optimized using the cross-entropy loss with the bilingual training corpus, and then fine-tuned by maximizing the expected BLEU of the generated translation candidates (where BLEU is the evaluation metrics for MT, which measures the n-gram accuracy of the generated candidates against a human-made reference). To deal with the exposure bias problem caused by the autoregressive decoding of the sequence-to-sequence model from left to right, Ref. [25] introduced two Kullback–Leibler (KL) divergences into the training objective to maximize the agreement between the candidates generated by left-to-right and right-to-left models. A deliberation network [26] is a method to refine the translation candidate based on two-pass decoding that simulates the human translation process; the first pass generates the initial translation and the second pass refines it. In order to deal with the label bias problem, as mentioned in Section 2.3, Ref. [27] proposed sampling the context words not only from the ground truth sequence, but also from the predicted sequence in order to bridge the gap between the training and inference of MT training.

3.2. Semi-supervised and unsupervised learning

Semi-supervised and unsupervised learning use unlabeled data to improve the model performance. For semi-supervised learning, first, the model is usually trained with labeled data; next, it is fine-tuned with the help of unlabeled data. There are many semi-supervised learning methods, such as self-learning, generative methods, and graph-based methods [28]. In these methods, pseudo data generated by models are usually used to fine-tune the model itself. In neural machine translation (NMT), to control the errors or noise generated in semi-supervised learning, weights/rewards are usually leveraged to filter the bad translation candidates; examples of weights/rewards include the expected BLEU method [29] and the dual-learning approach [30]. To utilize unlabeled data to improve the performance of the sequence-to-sequence model, back-translation [31] uses a reverse translation model to translate the target monolingual data in order to build a pseudo bilingual corpus, which is used to fine-tune the source-to-target (S2T) model. The joint training method (as shown in Fig. 8) extends the back-translation method to iteratively boost the S2T and target-to-source (T2S) translation models in a unified generalized (T2S) expectation–maximization framework by leveraging both the source and target monolingual corpus [32]. The bilingual corpus is used to train the NMT models first, including S2T and T2S models. With the source monolingual data, the S2T model is used to generate pseudo data to fine-tune the T2S model, and the target monolingual data is used to fine-tune the S2T model. This training process is iterated until the performance on hold-out data is no longer improved.

Leveraging the deep learning technique, deep generative models have been proposed for unsupervised learning, such as the variational auto-encoder (VAE) [33] and generative adversarial networks (GANs) [34]. The VAE net follows the auto-encoder framework, in which there is an encoder to map the input to a

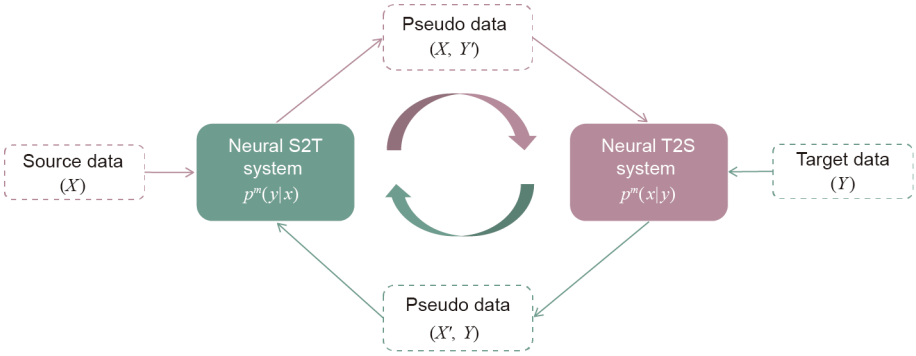

Fig. 8. Joint training of S2T and T2S NMT models. *m* means the *m*th epoch; Y' is the translation of the source monolingual data X; X' is the translation of the target monolingual data Y.

semantic vector, and a decoder to reconstruct the input. Unlike the original auto-encoder methods, a VAE assumes that the distribution of the generated semantic vectors should be as close as possible to a standard normal distribution. Similar to the VAE nets, GANs also have two parts: a generator, which uses a given semantic vector to generate the output, and a discriminator, which tries to distinguish between the generated samples and the real samples. With an adversarial training loss function, the generator tries to output samples that are similar to real ones, in order to fool the discriminator, while the discriminator tries to distinguish between the real and fake samples. A great deal of research work has attempted to apply VAEs and GANs to natural language-generation tasks [35,36].

Without using a bilingual corpus, but only using a small dictionary and a large monolingual corpus, unsupervised learning methods can be used for MT methods [37]. This method uses the joint training method to boost the S2T and T2S translation models jointly with the source and target monolingual corpus by generating a pseudo bilingual corpus. As there is no real bilingual data, the generated pseudo data may contain errors and noise, which will be reinforced in the subsequent iterative training process, when such examples are used for the model training. To deal with this problem, Ref. [38] introduced statistical machine translation (SMT) models as posterior regularization (PR) to filter this noise. The SMT and NMT models are optimized jointly and boost each other incrementally in a unified expectation–maximization framework. The whole procedure of this method consists of two parts (as shown in Fig. 9 [38]): model initialization and unsupervised NMT with SMT as PR. Given a language pair *X*–*Y*, two initial SMT models are first built with language models pre-trained using monolingual data, and word translation tables are inferred from cross-lingual embeddings. Then the initial SMT models will generate pseudo data to warm up two NMT models. The NMT models are trained using not only the pseudo data generated by SMT models, but also those generated by reverse NMT models with the joint training method. After that, the NMT-generated pseudo data is fed to SMT models. As PR, SMT models filter out noise and infrequent errors by constructing strong phrase tables with good and frequent translation patterns, and then generate denoised pseudo data to guide the subsequent NMT training. Benefiting from this process, NMT then produces better pseudo data for SMT to extract phrases of higher quality, while compensating for the deficiency in smoothness inherent in SMT via back-translation. The NMT and SMT models boost each other until final convergence. Compared with the work presented in Ref. [37], this method can significantly improve the translation results on four translation tasks with gaps of 1.4 BLEU points from French to English, 3.5 BLEU points from English to French, 3.1 BLEU points from German to English, and 2.2 BLEU points from English to German. Ref. [39] further introduced GPT methods to the unsupervised NMT tasks and proposed a cross-lingual language model pre-training to achieve the new state-of-the-art performance.

3.3. Multitask learning

Multitask learning attempts to leverage the information from other related tasks to improve the performance on the desired task. When a large amount of training data for the desired task is not available, a training corpus of related tasks can be introduced using a multitask learning approach. Ref. [10] proposed a unified neural network architecture with various NLP tasks including POS, chunking, named entity recognition, and semantic role labeling (SRL). This method learns internal representations based on vast amounts of mostly unlabeled data. This work was a milestone in learning features automatically with a neural network for NLP, which inspired the subsequent deep learning trend that has been applied in the field of NLP.

Ref. [40] proposed treating ten NLP tasks, including QA, MT, summarization, natural language inference, and so forth, as QA tasks, and built a multitask question-answering network (MQAN) to model them in a unified framework, as shown in Fig. 10 [40]. The inputs of the MQAN are a question and a context document. This input format is natural for the original QA task. For the MT task, the question is roughly "What is the translation from language X to language Y?" and the context is the source language sentence. For the summarization, the question is "What is the summary?" and the context is the document to be summarized. The question and the context are encoded with BiLSTM, followed by dual co-attention to build conditional representations for both sequences. These conditional representations are processed with another two BiLSTMs, followed by two self-attention networks and two BiLSTMs to obtain the final encoding representations of the question and context. To generate the output, an attention mechanism is leveraged to focus on the necessary encoding hidden states, and a multi-pointer generator is used to decide whether to copy from the question and context or generate a new word. This model achieved a state-of-the-art result on WikiSQL with a 72.4% EM and an 80.4% execution accuracy. With multitask learning, the MQAN can lead to better generalization for zero-shot learning on the zero-shot relation extraction task on dataset QA-ZRE, with a gain of 11 F_1 points over the highest single-task models. Ref. [41] proposed MT-deep neural network (DNN), a multitask learning-based DNN based on BERT. By adding more specific tasks into the pre-training, MT-DNN obtains very good results on ten natural language understanding (NLU) tasks, including SNLI, SciTail, and eight out of nine GLUE [42] tasks. The effectiveness of MT-DNN also shows that different but related tasks can boost each other via multitask learning.

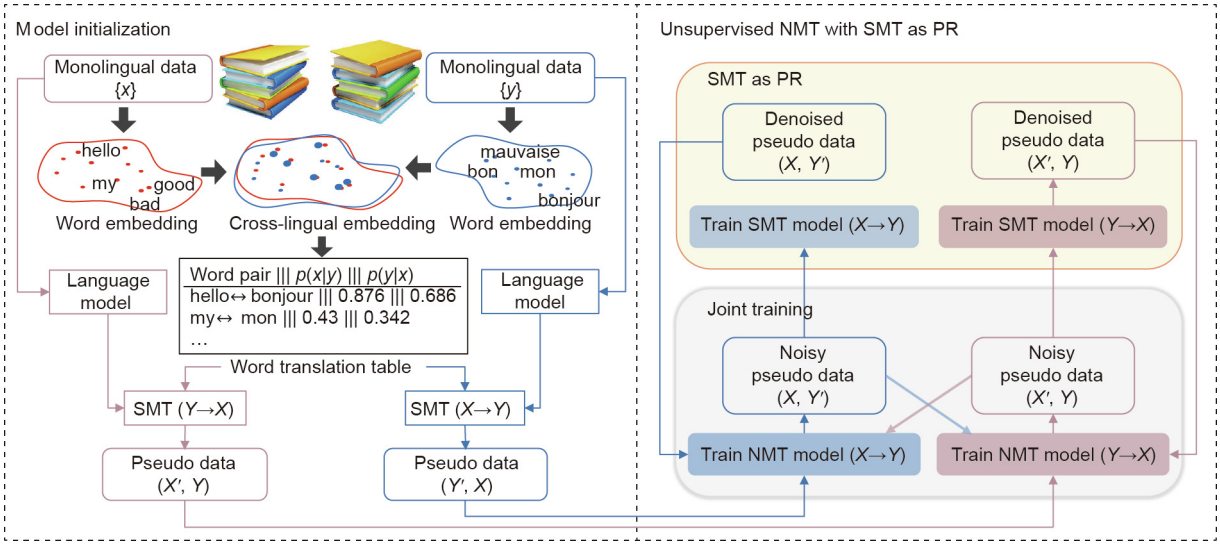

Fig. 9. Illustration of the unsupervised NMT training [38].

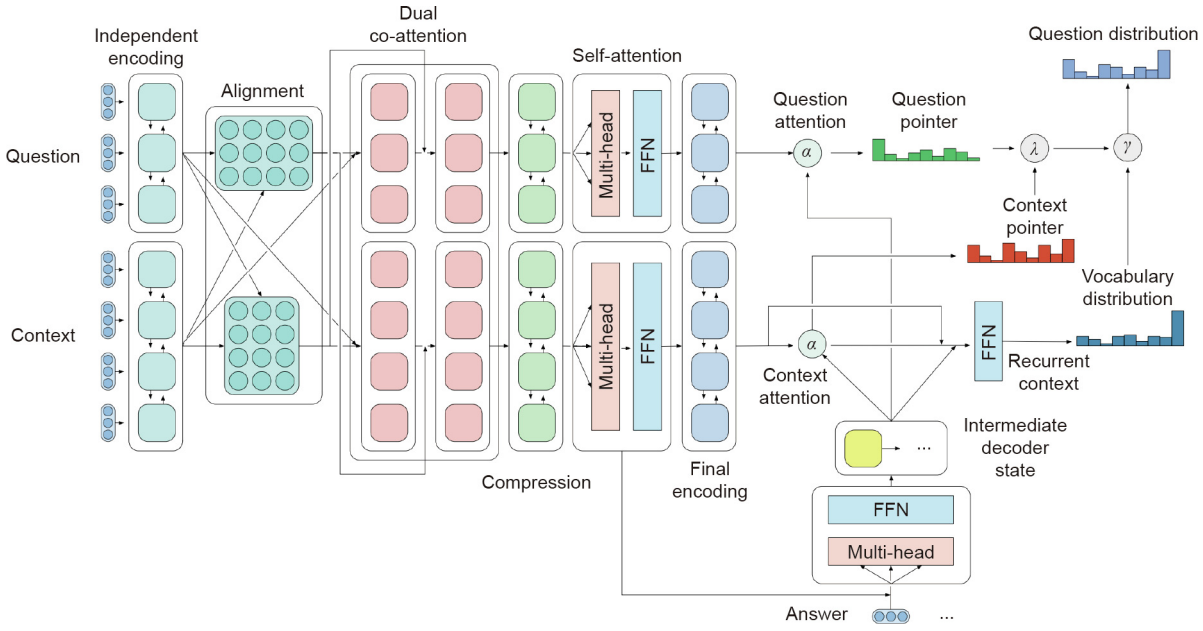

Fig. 10. Network structure of the MQAN [40]. α is the attention weights; γ and λ are the scalars to switch the output distributions.

3.4. Pre-trained models and transfer learning

Task-unspecific models are pre-trained first, and can then be transferred to specific tasks with a fine-tuning process. With pre-trained word embedding or sentence embedding, transfer learning can be used on top of them to fine-tune the task-specific models [43]. In recent years, many pre-trained models have been proposed, such as word2vec [6,8], ELMo [9], GPT [3], BERT [4], and XLNet [5], as introduced in Section 2.1, which are commonly used in NLP tasks such as MRC and QA.

Zero-shot and one-shot transfer learning have been explored with no or only a few labeled samples. Ref. [44] proposed a zero-shot transfer learning method for text classification, in which the model is trained to learn relationships, sentences, and categories with a large dataset, and is then transferred to new categories with no training data. Ref. [45] used semantic parsing to map natural language explanations on classifying concepts to formal constraints relating features grounded in the observed attributes of unlabeled data. Such constraints are combined using PR to yield a classifier for the zero-shot classification task. For some tasks in which there is a large amount of training data in a rich language (e.g., English), but little or no data in other languages (e.g., Romanian), cross-lingual transfer learning can be used to transfer the model trained in the rich language to a model for the rare language. Multitask learning over different MT language pairs can be transferred to enable zero-shot translation for language pairs with no bilingual corpus [46].

Another direction for transferring a pre-trained model to a wide variety of new and unseen situations is "learning to learn"—also known as meta learning. First proposed by Ref. [47], meta learning has recently become a hot topic, and is used for pre-trained model transfer, hyper-parameter tuning, and neural network optimization. By directly optimizing an initial model that can be effectively transferred with the help of a few examples, model agnostic meta

learning (MAML) [48] has been proposed to learn a task agnostic model. The learned model can be quickly adapted (with a small number of update steps) to several related tasks. Ref. [49] leveraged MAML to optimize NMT models using 18 European languages. By treating each example as a unique pseudo-task, the original structured query-generation task is reduced to a few-shot problem for which meta learning is used, bringing significant performance improvement.

Ref. [50] introduced an effective multitask learning framework to train a transferable sentence representation by combining a set of objectives including multi-lingual NMT, natural language inference, constituency parsing, and skip-thought vectors (i.e., to predict the previous or following sentence), and the learned representation can be transferred to a low-complexity classifier on a new task. By switching between different tasks, the gated recurrent unit (GRU) based RNN network can learn different aspects of the sentence. With the NMT and parsing tasks, syntactic properties can be better encoded. Sentence characteristics such as length and word order have been found to be better encoded by parsing. The learned representations are visualized using dimensionality reduction and nearest-neighbor exploration on three different datasets (movie reviews, question type classification, and Wikipedia classification), and sentences are found to be clustered reasonably well according to the labels. The learned representations are used as features (without parameters updating), and transferred to sentiment-classification tasks (movie reviews, product reviews, subjectivity/objectivity classification) with gains of 1.1%–2.0%, question type classification with gains of 6%, and paraphrase identification (Microsoft research paraphrase corpus) with gains of 2.3%.

3.5. Active learning

Active learning can interactively query the user in order to selectively label the data, with the aim of maximizing the performance gain and minimizing the labeling costs. To deal with the low-resource problem, one straightforward method is to label more data, which presents the challenge of identifying which data should be labeled in order to improve the model performance best. To deal with this problem, active learning methods [51] can be used to automatically and iteratively select useful instances to be labeled to maximize the model performance. With the labeled data, the learning model can be trained and applied to the unlabeled data. Based on the labeled result, the active selection model can predict a best selection for the editors to label based on signals such as uncertainty. The new labeled data can be used to obtain a better learning model, and this process can be iterated until an acceptable performance is achieved.

3.6. Summary

In this section, we reviewed several typical training methods, including supervised, semi-supervised, and unsupervised learning; multitask learning; transfer learning; and active learning. When enough labeled data is available, the supervised method can achieve very good performances on many NLP tasks, such as MT and MRC. For other tasks, where there is insufficient labeled data, we have introduced several learning methods to enhance the model performance, including semi-supervised and unsupervised learning with unlabeled data, transfer learning with pre-trained models, multitask learning with labeled data of other tasks, and active learning to annotate the most valuable samples. To improve the performance of NLP models, we think that more research should be conducted in the following topics:

- **The topological training process.** Although multitask learning and transfer learning can leverage the data from related tasks to enhance models for a desired task, the relationship between various NLP tasks should be further studied. For instance, a topological training process should be developed in the future, based on which the pre-trained models on fundamental tasks (e.g., language models, POS, and parsing) can be better transferred to higher-level tasks (e.g., MT and QA).
- **Reinforcement learning (RL).** RL has also been explored in many NLP model training cases, but it seems that it has not achieved satisfactory results. For example, in the dialog system for custom service, the error or loss is not available in each turn of the dialog session. The only information is whether the tickets are booked successfully or not, or how many turns are used. For such scenarios, where only a long-term reward is available, RL can be used to learn a policy network to maximize the expected reward. The RL process still suffers from the exponential search space with respect to the length of the natural language sentence [52].
- **GANs.** Even though many research efforts have tried to apply GANs to NLP tasks, such as MT [53] and natural language generation [54], there are several challenges. It is difficult for the discriminator to pass the gradient signal to the generator directly, as it does in image and speech processing, due to the discrete output of the generator. And the GAN is very sensitive to the random initialization and small deviations from the best hyper-parameter choice [36]. Such challenges amplify the difficulty of GAN training.

4. Reasoning

Neural approaches have achieved good progress on many NLP tasks, such as MT and MRC. However, they still have some unsolved problems. For example, most neural network models behave like a black box, which never tells how and why a system has solved a problem in the way it did. Besides, for tasks such as QA and dialog systems, only knowing the literal meanings of input utterances is often not enough. To generate the right responses, external and/or context knowledge may also be needed. To build such interpretable and knowledge-driven systems, reasoning is necessary.

In this paper, we define reasoning as a mechanism that can generate answers to unseen questions by manipulating existing knowledge with inference techniques. Based on this definition, a reasoning system (Fig. 11) should have two components:

- Knowledge, such as a knowledge graph, common sense, rules, assertions extracted from raw texts, etc.;
- An inference engine, to generate answers to questions by manipulating existing knowledge.

Next, we use two examples to illustrate why reasoning is important to NLP tasks.

The first example is a knowledge-based QA task. Given the question "When was Bill Gates' wife born?," the QA model must parse it into the logical form for answer generation: $\lambda x \lambda y$. DateOfBirth$(y, x) \wedge$ Spouse(Bill Gates, y), where a knowledge graph-based reasoning is needed. Starting with this question, new questions can be further appended to this context, such as: "What's his/her job?" To answer such context-aware questions, co-reference resolution determines which person is meant by "his/her." This is also a reasoning procedure, which needs the commonsense knowledge that "his" can only refer to men and "her" can only refer to women.

The second example is a dialog task. For example, if a user says "I am very hungry now," it would be more suitable to reply: "Let me recommend some good restaurants to you" instead of "Let me recommend some good movies to you." This also requires reasoning, as the dialog system should know that hungry will lead to actions such as looking for restaurants instead of watching films.

In the remainder of this section, we will first introduce two types of knowledge: the knowledge graph and common sense.

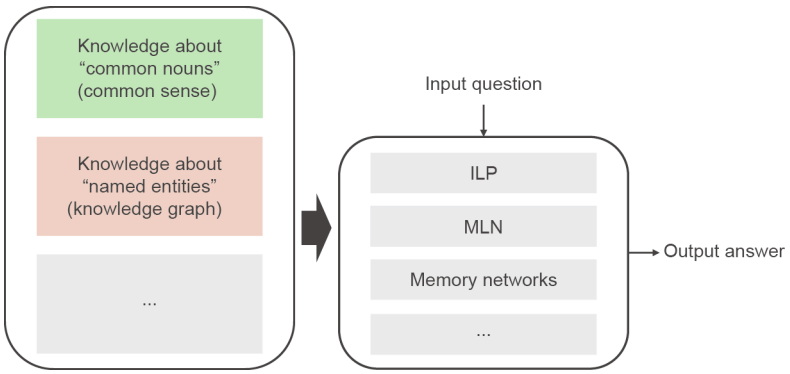

Fig. 11. Overview of a reasoning system. ILP: integer linear programming; MLN: Markov logic network.

Next, we will describe typical inference approaches, which have been or are being studied in the NLP area.

4.1. Knowledge

Knowledge plays an important role in reasoning-driven NLP tasks. It may refer to any information (e.g., dictionaries, rules, knowledge graphs, annotations of specific tasks) that can guide NLP systems to complete specific tasks. Here, we focus on two types of knowledge for reasoning: the knowledge graph and common sense.

4.1.1. The knowledge graph

A knowledge graph is a directed graph $\{V, E\}$, which consists of nodes V and edges E. Each node $v \in V$ denotes an entity. Each edge $e \in E$ denotes a predicate between the two entities it connects with. Each triple $< v_1, e, v_2 >$ denotes a fact, where $v_1, v_2 \in V$ and $e \in E$.

For example, $<$ Microsoft, Founder, Bill Gates $>$ is a knowledge triple from a knowledge graph, where Microsoft is the subject entity, Bill Gates is the object entity, and Founder is the predicate indicating that Bill Gates is the founder of Microsoft. A knowledge graph is essential to those NLP tasks where parsing a natural language into machine-executable structured queries is indispensable, such as semantic searches, knowledge-based QA, and task-oriented dialog.

The approaches of knowledge graph construction can be grouped into three categories:

- **Handcrafted methods.** In these methods, knowledge graphs are manually constructed by human experts. Knowledge graphs built in this way, such as WordNet [55], are usually of high quality, but cover limited facts. In addition, the cost of maintenance is usually very high.
- **Crowdsourcing methods.** In these methods, community members construct knowledge graphs in a collaborative way. Compared with handcrafted methods, knowledge graphs built in this way, such as DBpedia [56], Freebase [57], and WikiData [58], are large scale and high quality.
- **Information-extraction methods.** These methods extract structured knowledge from web documents. KnowItAll [59], YAGO [60], and NELL [61] are representatives of this method. Compared with the first two approaches, these methods can extract more knowledge from free texts; however, they also bring a lot of noise into the resulting knowledge graphs.

4.1.2. Common sense

Common sense refers to common knowledge of things in the world, such as their properties, relationships, and interactions, which all humans are expected to know. Such knowledge is mostly location-, language-, and culture-independent, and is rarely explicitly expressed in texts.

For example, “a father is a male” is property common sense, “the sun is in the sky” is spatial common sense, and “plants grow from seeds” is procedural common sense.

Building a commonsense knowledge base (CKB) is difficult. At present, there are three major methods (which are similar to the knowledge graph), but all suffer from significant problems.

- **Handcrafted methods.** CYC [62] is a CKB built by human experts in this way, which focuses on things that are rarely written down or said, such as “every human has exactly one father and exactly one mother.” The goal of CYC is to enable AI systems to perform human-like reasoning and to be less brittle when confronting unseen situations. The sizes of CKBs such as CYC are limited, as human labeling is expensive.
- **Crowdsourcing methods.** ConceptNet [63] is a CKB built by absorbing knowledge from knowledge graphs such as WordNet, Wiktionary, Wikipedia, DBpedia, and Freebase. Compared with CYC, ConceptNet covers many more assertions. However, a large portion of the ConceptNet assertions is actually not common sense, as it comes from existing knowledge bases and are about “named entities,” such as Bill Gates and White House, instead of “common nouns,” such as human and building.
- **Information extraction methods.** WebChild [64] is a CKB containing commonsense assertions that connect nouns with adjectives via fine-grained relations such as hasShape and hasTaste. Such assertions are extracted from web documents based on seed assertions from WordNet. Due to extraction errors, WebChild contains a great deal of noise. It now covers more than four million commonsense assertions.

4.2. Inference engine

Although this paper focuses on neural NLP methods, we will still start from two typical non-neural inference methods: integer linear programming (ILP) and Markov logic networks (MLNs), as they have been studied a great deal and have already been used in some NLP tasks. After that, we will introduce memory networks as a typical neural inference method. We will also describe their applications in two reasoning tasks: semantic parsing and response generation, which are based on the knowledge graph and common sense, respectively.

4.2.1. Non-neural inference methods: ILP and MLN

ILP is an optimization framework that maximizes a linear objective function over a finite set of variables x, subject to a set of linear inequality constraints:

maxmize $w^{\mathrm{T}}x$

subject to $Ax \leq b$

and $x \in \mathbb{Z}^n$

where w is parameters to be optimized; A and b are pre-defined parameters in constraints; n is the number of variables.

The constraints used in ILP can be considered as prior knowledge, and the optimization procedure is to reason out a global prediction by incorporating learned models based on such prior knowledge. This method can be used in the information extraction task [65], whose objective is to recognize named entities and their relationships from texts. Fig. 12 gives an example.

Usually, an NER and a relation extractor (RE) will be trained separately and used in this task in a cascaded manner. First, the NER detects entity mentions from input and assigns possible types to each mention. Then, the RE assigns possible relations to each entity pair. Five prediction tables are generated for these three named entities and two relations, which are shown in Fig. 13.

If local predictions with the highest probabilities are always chosen, errors will occur. For example, the type of Brooklyn will be recognized as Person and the relation between Adam and Anne will be classified as PlaceOfBirth. However, if it is known that the object type of PlaceOfBirth should be Location instead of Person, such errors can be avoided. Inference with such prior knowledge is a reasoning procedure, which can be done by ILP.

In order to obtain correct predictions based on the above five local prediction tables, we formulate it as an ILP problem and manually define the following four constraints: ① Each entity mention can be assigned an entity type only once. ② Each entity pair can be assigned a relation only once. ③ The type assignment to each entity mention should be consistent with the assignments to its neighboring relations. For example, when Anne is tagged as Person, if type of Brooklyn is also recognized as Person, then the relation assignment between Anne and Brooklyn cannot be PlaceOfBirth, as its object entity type should be Location, which is inconsistent with Person. ④ The relation assignment to each entity pair should be consistent with the assignments to these two entities. Based on local NER and RE outputs and these four constraints, ILP can obtain the global optimized output (Fig. 14).

This time, the type of Brooklyn is recognized as Location, as it is consistent to the relation (i.e., PlaceOfBirth) assigned to Anne and Brooklyn. Similarly, the relation between Adam and Anne can be correctly recognized as SpouseOf.

ILP can be used in many NLP tasks, such as QA [66–68] and semantic role labeling [69,70]. It is especially useful when the size of the training data is small, as prior knowledge (used as constraints in ILP) can play more important roles in such scenarios. However, this framework is still independent from existing neural network methods.

An MLN [71] $L = (F_l, w_l)$ is defined as a set of pairs, where each F_l is a first-order logic formula and w_l is the weight of F_l. It combines probability and first-order logic in a unified model and can generate a Markov network $M_{L,C}$ by grounding all variables in formulas to constants $C = \{c_1, \ldots, c_{|c|}\}$. The basic idea of an MLN is to soften hard constraints in first-order logic: When a world violates one formula in the knowledge graph, it is less probable, but not impossible.

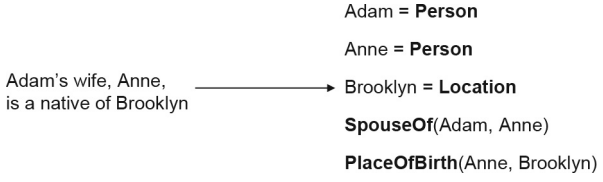

Fig. 12. An example of an information-extraction task.

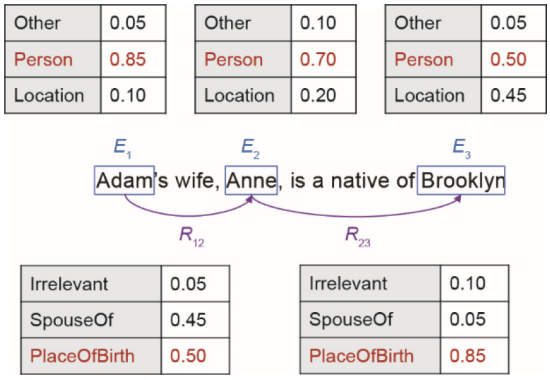

Fig. 13. Sub-optimal results without using ILP. *E*: entities occurred in the sentence; *R*: relations between entities.

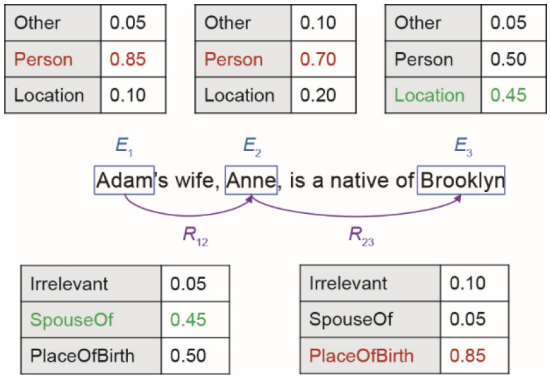

Fig. 14. Optimal results when using ILP.

Ideally, an MLN can be applied to reasoning-required NLP tasks by the following steps. Here, we use the famous "friend smoking" case as an example:

(1) Given a world described in natural language, parse it into a set of first-order logic formulas $F = \{F_1, \ldots, F_{|F|}\}$. For example, given two sentences:

Smoking causes cancer

Friends have similar smoking habits

The sentences are parsed into two first-order logic formulas:

$$\forall x \text{Smokes}(x) \Rightarrow \text{Cancer}(x)$$

$$\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Longleftrightarrow \text{Smokes}(y))$$

In practice, each formula has a weight, which should be learned from existing relational databases by algorithms such as the pseudo-likelihood algorithm [72].

(2) Given L and a set of constants C, a Markov network $M_{L,C}$ is defined as follows: ① $M_{L,C}$ contains one binary node for each possible grounding of each predicate in L. The value of the node is 1 if the ground atom is true, and 0 otherwise. ② $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L. For

example, given two constants Anna(A) and Bob(B), the ground $M_{L,C}$ is described in Fig. 15.

(3) Given a set of evidences, such as Friends(A, B) = 1 and Cancer(A) = 1, reason the most likely state of the world, such as Smokes(A) = ?, Smokes(B) = ?, and Cancer(B) = ?. This inference procedure can answer questions such as "What is the probability that Anna smokes, given that Anna has cancer?" In practice, algorithms such as MC-SAT [66] can be used to solve such inference tasks.

As a statistical relational learning approach, the MLN has been applied to some NLP tasks already, such as semantic parsing [73], information extraction [74], and entity disambiguation [75]. However, it is still difficult to apply it to real tasks, as the size of the ground Markov network is usually very large, which will raise the computation issue. For example, based on the analysis of Ref. [76], the size of the corresponding ground Markov networks is exponential in $|C|$, which is the number of constants. If the constants come from a practical knowledge base (such as Freebase, which contains 39 million entities), the inference complexity in the resulting ground Markov networks will be enormous. In addition, similar to ILP, the MLN is independent of neural models. The necessities and solutions of designing neural versions of the MLN are still worthy of discussion and study.

4.2.2. Neural inference methods: MemNN and its variants

A memory network (MemNN) [77–79] is a neural network architecture in which reasoning can be supported by processing knowledge stored in a long-term memory component multiple times before outputting a result.

We take the key-value memory networks (KV-MemNN) proposed by Ref. [79] as an example to show how this framework can support reasoning. Fig. 16 [79] gives an overview of the KV-MemNN.

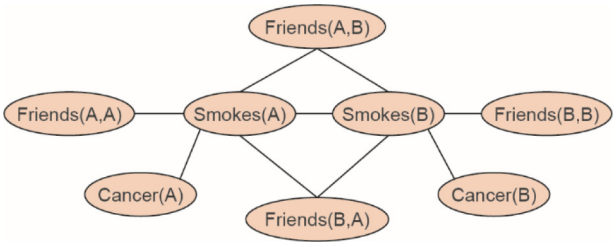

Fig. 15. An example of a Markov network.

In KV-MemNN, the long-term memory is represented as pairs of vectors $(k_1, v_1), ..., (k_M, v_M)$, where k_M denotes a key and v_M denotes the value of k_M. Knowledge from both structured knowledge bases and unstructured texts can be represented as key-value pairs. For example, given a triple $< e_1, r, e_2 >$ from a knowledge graph, the key is composed of the subject entity e_1 and predicate r, and the value is the object entity e_2. Given an input question x, a small subset of key-value pairs $(k_{h_1}, v_{h_1}), ..., (k_{h_N}, v_{h_N})$ will first be retrieved from memory based on their similarities with x. Then, a relevance probability is computed to each key as follows:

$$p_{h_i} = \mathrm{softmax}\left[A\Phi_X(x) \cdot A\Phi_K(k_{h_i})\right] \tag{3}$$

where $\Phi_X(x)$ and $\Phi_K(k_{h_i})$ map x and k_{h_i} into feature maps of dimension D, and A is a $d \times D$ matrix. d means a hyperparameter of the model, and there is no physical meaning for it. Next, an output vector $\boldsymbol{o}$ is generated by taking the weighted sum of the value vectors based on their corresponding relevance probabilities:

$$\boldsymbol{o} = \sum_i p_{h_i} A\Phi_V(v_{h_i}) \tag{4}$$

If a task needs multi-turn reasoning based on the stored knowledge, then the input question x will be updated to

$$\boldsymbol{q}_2 = R_1(\boldsymbol{q} + o) \tag{5}$$

where $\boldsymbol{q} = A\Phi_X(x)$ is the question vector, and R_1 is a $d \times d$ matrix. This procedure will repeat H times using different R_j and the final vector $\boldsymbol{q}_{H+1}$ will be used to predict outputs.

MemNN is widely used in many reasoning-required NLP tasks. For example, Ref. [77] first proposed the MemNN framework and applied it to bAbI, which is a reasoning-required QA task. Ref. [79] proposed KV-MemNN and evaluated it on two QA datasets: WikiMovies and WikiQA [80]. Ref. [81] described an end-to-end goal-oriented dialog method based on MemNN.

In a broader sense, MemNN is a special case of memory-augmented neural networks (MANNs), which extend the capabilities of neural networks by coupling them to external memory resources. In such methods, neural models interact with memory by attentional processes. As MANNs have many variants, we use two examples to illustrate the applications of a MANN in two reasoning-required tasks (i.e., semantic parsing and response generation), which are based on the knowledge graph and on commonsense knowledge, respectively.

Ref. [82] proposed a unified semantic parsing approach to handle a knowledge-graph-based conversational QA task, in which a dialog memory motivated by MemNN is introduced to cope with

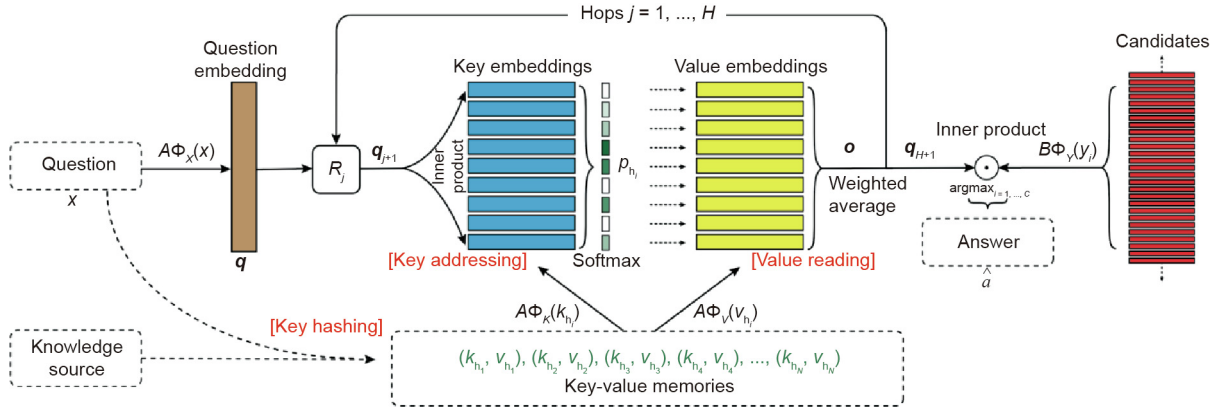

Fig. 16. Overview of KV-MemNN [79]. a means the answer of the question; B means a $d \times D$ matrix, which can be constrained to be identical to A. R_j means a $d \times d$ matrix, which is used to update the representation of the input question in jth hop.

co-reference and ellipsis phenomena in multi-turn interactions. Two examples are given below.

In order to answer the first question:

(1) Where was Donald Trump born?

The semantic parser first generates its logical form:

$$\ddot{e}x.\text{PlaceOfBirth}\,(\text{Donald Trump}, x)$$

and then executes it against a knowledge base to get the answer New York. This is called context-independent semantic parsing, as only the current question is needed.

In order to answer the second question:

(2) Where did he graduate from?

Both questions (1) and (2) are needed, as "he" in question (2) refers to Donald Trump in question (1). This is called context-dependent semantic parsing.

Given a context-independent question, (e.g., question (1)), semantic parsing is done as an action sequence prediction task. Starting from a root symbol "Start," a grammar-guided decoder recursively rewrites the leftmost nonterminal (i.e., the semantic category) in the logical form by applying a legitimate action. It terminates when no nonterminal is left. Fig. 17 illustrates this procedure.

A list of actions is defined in Table 1, where each action consists of three parts: a semantic category; a function symbol, which might be omitted; and a list of arguments, each of which can be a semantic category, a constant, or an action subsequence. The first 15 actions (A1–A15) are designed to cover typical operations in semantic parsing. We take A5 as an example: "num" is the semantic category, which denotes the returned type of count(set); "count" is the function symbol, which returns the number of elements in set; and "set" is the only argument of count. A16–A18 are designed to instantiate entity e, predicate r, and number num, respectively. A19–A21 are designed to reuse the previously predicated action subsequences.

Given a context-dependent question (e.g., question (2)), a dialog memory is used to maintain all entities, predicates, and action subsequences that come from either the generated logical forms of previous questions or the predicted answers. Such contents are considered to be the conversation history and will be used to restore the missing information (co-reference and entity/relation ellipsis) of the current question. For the example in Fig. 18, as "he" in current question actually refers to Donald Trump in the previous question, the semantic parser can copy the action subsequence (i.e., A15 e_{DT}) from the dialog memory to complete the logical form generation.

Experiments are conducted on the CSQA dataset [83], which consists of 2×10^5 dialogs with 1.6×10^6 turns over 1.28×10^7 entities from WikiData, and this method achieves state-of-the-art results on both context-independent and context-dependent questions.

Ref. [84] proposed a commonsense-aware encoder–decoder framework for the response-generation task in open-domain dialog systems. ConceptNet is used in this work to understand the background information of a given user utterance and then facilitate response generation.

In the encoder part, each word x_i in the input utterance $X = x_1, \ldots, x_n$ is used to retrieve a graph from the entire commonsense knowledge base. The retrieved graph is then encoded as a graph vector g_i, which is the weighted sum of the head and tail vectors of the triples contained in the graph. Here, each attention weight measures the association of a relation to its head entity and tail entity. Lastly, the embedding of each input word $\text{vec}(x_i)$ is obtained by concatenating the word vector and the graph vector $\text{vec}(x_i) = [\boldsymbol{w}(x_i); \boldsymbol{g}_i]$, and is fed to the GRU cell of the encoder. In this procedure, the graph vector is obtained based on an MemNN-like method.

In the decoder part, a knowledge–aware generator is designed to generate a response by making full use of the retrieved graphs. This part plays two roles: ① It attentively reads the retrieved graphs to obtain a graph–aware context vector, and uses the vector to update the decoder's state. This procedure is similar to MemNN. ② It adaptively chooses a generic word or an entity from the retrieved graphs for response generation.

Experiments are conducted on the Commonsense Conversation dataset, and this method achieves state-of-the-art results compared with the traditional sequence-to-sequence approach with copy.

With deep learning, different types of knowledge can be represented and used in neural inference methods for various reasoning tasks. We also see that such approaches still lack sufficient reasoning flexibilities. For example, the reasoning hop in MemNN is fixed without considering different input contents, which can be further improved in the future.

Table 1
List of actions, each consisting of a semantic category, function symbol, and list of arguments.

Action	Operation	Note
A1–A3	start → set|num|bool	
A4	set → find(set, r)	Set of entities with a r edge to e
A5	num → count(set)	Total number of set
A6	bool → in(e, set)	Whether e is in set
A7	set → union(set_1, set_2)	Union of set_1 and set_2
A8	set → inter(set_1, set_2)	Intersection of set_1 and set_2
A9	set → diff(set_1, set_2)	Instances included in set_1 but not included in set_2
A10	set → larger(set, r, num)	Subset of set linking to more than num entities with relation r
A11	set → less(set, r, num)	Subset of set linking to less than num entities with relation r
A12	set → equal(set, r, num)	Subset of set linking to num entities with relation r
A13	set → argmax(set, r)	Subset of set linking to most entities with relation r
A14	set → argmin(set, r)	Subset of set linking to least entities with relation r
A15	set → $\{e\}$	
A16–A18	e|r|num → constant	Instantiation for entity e, predicate r or number num
A19–A21	set|num|bool → action_{i-1}	Replicate previous action sequence (w/o or w/ instantiation)

4.3. Reasoning–aware shared tasks

Based on different types of knowledge used, we classify recently proposed reasoning tasks into four categories:

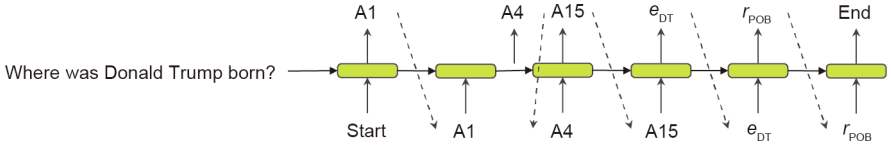

Fig. 17. An example of context-independent semantic parsing. DT: Donald Trump.

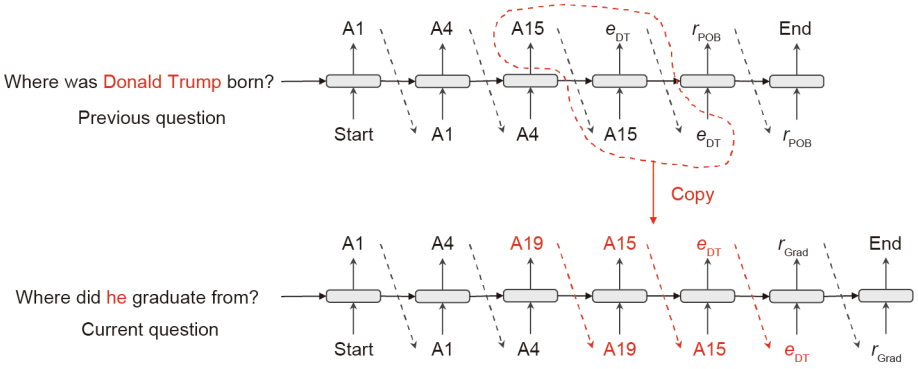

Fig. 18. An example of content-dependent semantic parsing.

- **Tasks based on knowledge graphs.** These tasks include WikiSQL [85], LC-QuAD [86], CSQA [83], and ComplexWebQuestions [87], in which various types of questions, such as multi-hop, multi-constraint, superlative, and multi-turn questions, must be answered based on knowledge graphs. In these tasks, reasoning is required and is done by performing semantic parsing explicitly.
- **Tasks based on commonsense knowledge.** These tasks include the Winograd Schema Challenge [88], ARC [89], CommonsenseQA [90], and ATOMIC [91], in which questions can be answered based on different types of commonsense knowledge, such as temporal, spatial, and causal common sense. In these tasks, reasoning can either be performed explicitly by designing specific inference models, or performed implicitly by end-to-end training.
- **Tasks based on texts.** These tasks include HotpotQA [92], NarrativeQA [93], MultiRC [94], and CoQA [95], in which answers can be obtained by reasoning across paragraphs, documents, or conversation turns. Currently, state-of-the-art results on these tasks are achieved by end-to-end neural models. One reason for the current poor performance is that existing knowledge bases still suffer from low coverage of open-domain natural language texts.
- **Tasks based on both texts and visual content.** These tasks include GQA [96] and VCR [97], in which the goal is to answer a natural language question based on a given image. As the questions in these two datasets are either multi-hop questions or require commonsense knowledge, a model needs a strong reasoning ability to achieve a good performance.

As data annotation is expensive, these datasets are often small in size, or big but generated automatically using templates. The models learned from such datasets lack strong generalization abilities. Recently, pre-trained models such as ELMo, GPT, BERT, and XLNet have shown good generalization performances across different NLP tasks. For the next step, it is practical to fine-tune existing pre-trained models with reasoning–aware datasets with the aim of building more generalized reasoning systems.

4.4. Summary

This section briefly reviewed the progress of reasoning in NLP. Typical (non-neural and neural) inference methods were introduced, including ILP, MLN, and MemNN, all of which have been successfully used in various NLP tasks, such as QA, dialog systems, and information extraction.

In order to build more powerful reasoning systems, some challenges remain:

- **Knowledge extraction.** Due to the limited coverage of existing knowledge sources, only a small portion of queries issued to current human–computer interaction systems (such as search, QA, and dialog engines) can be fully understood and addressed. Therefore, knowledge extraction is still a long-term research task in order to acquire high-quality knowledge to support reasoning.
- **Reasoning with explicit knowledge and pre-trained models.** Typical reasoning approaches are built based on explicit knowledge bases. Recently, pre-trained models such as GPT, BERT, and XLNet have shown strong abilities on reasoning-required NLP tasks, such as the Winograd Schema Challenge [88] and SWAG [98]. Therefore, the question of how to combine explicit knowledge and pre-trained models into a unified reasoning framework is worth exploring.
- **Datasets and metrics.** Although the latest approaches can perform well on many NLP tasks, it is still difficult to tell how much reasoning capability an NLP model has. In order to measure such abilities, large-scale datasets and specific metrics should be built for the community.

In general, reasoning is critical to advance NLP, as reasoning can bring existing knowledge into all NLP tasks for better natural language understanding and generation.

5. Conclusion

This paper reviewed the latest progress of neural NLP from three perspectives: modeling, learning, and reasoning. In general, NLP is entering a new era, in which neural network-based models dominate in research and application systems. For rich-source tasks with enough training data (e.g., NMT and SQuAD), supervised learning can do pretty well. For low-resource tasks with little training data, semi-supervised and unsupervised learning, multitask learning, transfer learning, and active learning can be used; these methods can either generate more pseudo training data for model training, or leverage the knowledge learned from existing models and tasks. Different reasoning mechanisms have also been explored and used for tasks in which reasoning is required, such as commonsense QA, chitchat, and dialog systems.

Looking ahead, we can see many exciting directions. For example, pre-trained models such as GPT, BERT, and XLNet have already shown strong performances on various NLP tasks, and there is no doubt that they will continue to evolve for both natural language understanding and generation tasks. It is also worth exploring how to integrate such pre-trained models into various NLP tasks with transfer learning, multitask learning, meta learning, and so on. Research on memory-argument neural networks and their variants will further advance reasoning approaches, with the help of new knowledge-extraction techniques. Furthermore, by combing NLP with multi-modal tasks such as speech recognition, image/video captioning, and QA, new research and application scenarios will emerge. It is an incredibly exciting time to work on NLP from research to applications.

Compliance with ethics guidelines

Ming Zhou, Nan Duan, Shujie Liu, and Heung-Yeung Shum declare that they have no conflict of interest or financial conflicts to disclose.

References

[1] Deng J, Dong W, Socher R, Li LJ, Li K, Li FF. ImageNet: a large-scale hierarchical image database. In: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition; 2009 Jun 20–25; Miami, FL, USA; 2009. p. 248–55.

[2] Xiong W, Wu L, Alleva F, Droppo J, Huang X, Stolcke A. The Microsoft 2017 conversational speech recognition system. In: Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing; 2018 Apr 15–20; Calgary, AB, Canada; 2018. p. 5934–8.

[3] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training [Internet]. [cited 2019 Apr 29]. Available from: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[4] Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics; 2019 Jun 2–7; Minneapolis, MN, USA; 2019. p. 4171–86.

[5] Yang ZL, Dai Z, Yang YM, Carbonell J, Salakhutdinov R, Le QV. XLNet: generalized autoregressive pretraining for language understanding. 2019. arXiv:1906.08237.

[6] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. 2013. arXiv:1301.3781.

[7] Firth JR. A synopsis of linguistic theory 1930–1955. In: Firth JR. Studies in linguistic analysis. Oxford: Blackwell; 1957. p. 1–31.

[8] Pennington J, Socher R, Manning C. GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing; 2014 Oct 25–29; Doha, Qatar; 2014. p. 1532–43.

[9] Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. Deep contextualized word representations. In: Proceedings of the 2018 Annual Conference of the North American Chapter of the Association for Computational Linguistics; 2018 Jun 1–6; New Orleans, LA, USA; 2018.

[10] Collobert R, Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning; 2008 Jul 5–9; Helsinki, Finland; 2008. p. 160–7.

[11] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. 2014. arXiv:1406.1078.

[12] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. 2014. arXiv:1409.0473.

[13] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31rd Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA; 2017.

[14] Yu M, Dredze M. Improving lexical embeddings with semantic knowledge. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics; 2014 Jun 23–25; Baltimore, MD, USA; 2014. p. 545–50.

[15] Zhang J, Luan H, Sun M, Zhai F, Xu J, Zhang M, et al. Improving the transformer translation model with document-level context. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; 2018 Oct 31– Nov 4; Brussels, Belgium; 2018. p. 533–42.

[16] Wu Y, Wu W, Xing C, Xu C, Li Z, Zhou M. A sequential matching framework for multi-turn response selection in retrieval-based chatbots. Comput Linguist 2019;45(1):163–97.

[17] Gu J, Bradbury J, Xiong C. Li VOK, Socher R. Non-autoregressive neural machine translation. 2017. arXiv:1711.02281.

[18] Friedman JH. Stochastic gradient boosting. Comput Stat Data Anal 2002;38 (4):367–78.

[19] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(9):533–6.

[20] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 2011;12(Jul):2121–59.

[21] Zeiler MD. ADADELTA: an adaptive learning rate method. 2012. arXiv:1212.5701.

[22] Kingma DP, Ba J. Adam: a method for stochastic optimization. In: Proceedings of the 2015 International Conference on Learning Representations; 2015 May 7–9; San Diego, CA, USA; 2015.

[23] Shen S, Cheng Y, He Z, He W, Wu H, Sun M, et al. 2016. Minimum risk training for neural machine translation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics; 2016 Aug 7–12; Berlin, Germany; 2016.

[24] Papineni K, Roukos S, Ward T, Zhu WJ. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics; 2002 Jul 7–12; Philadelphia, PA, USA; 2002. p. 311–8.

[25] Zhang Z, Wu S, Liu S, Li M, Zhou M, Xu T. Regularizing neural machine translation by target-bidirectional agreement. In: Proceedings of the 31rd AAAI Conference on Artificial Intelligence; 2019 Jan 27–Feb 1; Honolulu, HI, USA; 2019.

[26] Xia Y, Tian F, Wu L, Lin J, Qin T, Yu N, et al. Deliberation networks: sequence generation beyond one-pass decoding. In: Proceedings ot the 31rd Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA; 2017.

[27] Zhang W, Feng Y, Meng F, You D, Liu Q. Bridging the gap between training and inference for neural machine translation. 2019. arXiv:1906.02448.

[28] Zhu XJ. Semi-supervised learning literature survey. Madison: University of Wisconsin-Madison; 2005.

[29] Cheng Y, Xu W, He Z, He W, Wu H, Sun M, et al. Semi-supervised learning for neural machine translation. 2016. arXiv:1606.04596.

[30] He D, Xia Y, Qin T, Wang L, Yu N, Liu T, et al. Dual learning for machine translation. In: Proceedings of the 30th International Conference on Neural Information Processing Systems; 2016 Dec 5–10; Barcelona, Spain; 2016. p. 820–8.

[31] Sennrich R, Haddow B, Birch A. Improving neural machine translation models with monolingual data. 2015. arXiv:1511.06709.

[32] Zhang Z, Liu S, Li M, Zhou M, Chen E. Joint training for neural machine translation models with monolingual data. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence; 2018 Feb 2–7; New Orleans, LA, USA; 2018.

[33] Kingma DP, Welling M. Auto-encoding variational bayes. 2013. arXiv:1312.6114.

[34] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Proceedings on Neural Information Processing Systems (NIPS 2014); 2014 Dec 8–13; Montreal, QC, Canada; 2014. pp. 2672–80.

[35] Hu W, Tan Y. Generating adversarial malware examples for black-box attacks based on GAN. 2017. arXiv:1702.05983.

[36] Semeniuta S, Severyn A, Gelly S. On accurate evaluation of GANs for language generation. 2018. arXiv:1806.04936.

[37] Lample G, Conneau A, Denoyer L. Ranzato M. Unsupervised machine translation using monolingual corpora only. 2017. arXiv:1711.00043.

[38] Ren S, Zhang Z, Liu S, Zhou M, Ma S. Unsupervised neural machine translation with SMT as posterior regularization. 2019. arXiv:1901.04112.

[39] Conneau A, Lample G. Cross-lingual language model pretraining. 2019. arXiv:1901.07291.

[40] McCann B, Keskar NS, Xiong C, Socher R. The natural language decathlon: multitask learning as question answering. 2018. arXiv:1806.08730.

[41] Liu X, He P, Chen W, Gao J. Multi-task deep neural networks for natural language understanding. 2019. arXiv:1901.11504.

[42] Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR. GLUE: a multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP; 2018 Oct 31– Nov 4; Brussels, Belgium; 2018. p. 353–5.

[43] Zoph B, Le QV. Neural architecture search with reinforcement learning. 2016. arXiv:1611.01578.

[44] Pushp PK, Srivastava MM. Train once, test anywhere: zero-shot learning for text classification. 2017. arXiv:1712.05972.

[45] Srivastava S, Labutov I, Mitchell T. Zero-shot learning of classifiers from natural language quantification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics; 2018 Jul 15–20; Melbourne, VIC, Australia; 2018. p. 306–16.

[46] Johnson M, Schuster M, Le QV, Krikun M, Wu Y, Chen Z, et al. Google's multilingual neural machine translation system: enabling zero-shot translation. Trans Assoc Comput Linguist 2017;5:339–51.

[47] Schmidhuber J. Evolutionary principles in self-referential learning: on learning how to learn [dissertation]. München: Technische Universität München; 1987.

[48] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. 2017. arXiv:1703.03400.

[49] Gu JT, Wang Y, Chen Y, Cho K, Li VOK. Meta-learning for low-resource neural machine translation. 2018. arXiv:1808.08437.

[50] Subramanian S, Trischler A, Bengio Y, Pal CJ. Learning general purpose distributed sentence representations via large scale multi-task learning. 2018. arXiv:1804.00079.

[51] Settles B. Active learning literature survey. Madison: University of Wisconsin-Madison; 2009.

[52] He J, Chen J, He X, Gao J, Li L, Deng L, et al. Deep reinforcement learning with a natural language action space. 2015. arXiv:1511.04636.

[53] Wu L, Xia Y, Zhao L, Tian F, Qin T, Lai J, et al. Adversarial neural machine translation. 2017. arXiv:1704.06933.

[54] Alzantot M, Sharma Y, Elgohary A, Ho BJ, Srivastava M, Chang KW. Generating natural language adversarial examples. 2018. arXiv:1804.07998.

[55] Miller GA. WordNet: a lexical database for English. Commun ACM 1995;38 (11):39–41.

[56] Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z. DBpedia: a nucleus for a web of open data. In: Proceedings of the 2007 International Semantic Web Conference; 2007 Nov 11–15; Busan, Korea; 2007. p. 722–35.

[57] Bollacker KD, Evans C, Paritosh P, Sturge T, Taylor J. Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data; 2008 Jun 9–12; Vancouver, BC, Canada; 2008. p. 1247–50.

[58] Vrandečić D, Krötzsch M. Wikidata: a free collaborative knowledgebase. Commun ACM 2014;57(10):78–85.
[59] Etzioni O, Cafarella M, Downey D, Kok S, Popescu AM, Shaked T, et al. Web-scale information extraction in knowitall: (preliminary results). In: Proceedings of the 13th International Conference on World Wide Web; 2004 May 17–20; New York, NY, USA; 2004. p. 100–10.
[60] Fabian MS, Gjergji K, Gerhard WE. YAGO: a core of semantic knowledge unifying WordNet and Wikipedia. In: Proceedings of the 16th International Conference on World Wide Web; 2007 May 8–12; Banff, AL, Canada; 2007. p. 697–706.
[61] Carlson A, Betteridge J, Kisiel B, Settles B, Hruschka ER Jr, Mitchell TM. Toward an architecture for never-ending language learning. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence; 2010 Jul 11–15; Atlanta, GA, USA; 2010.
[62] Lenat DB. CYC: a large-scale investment in knowledge infrastructure. Commun ACM 1995;38(11):33–8.
[63] Liu H, Singh P. ConceptNet—a practical commonsense reasoning tool-kit. BT Technol J 2004;22(4):211–26.
[64] Tandon N, De Melo G, Weikum G. Acquiring comparative commonsense knowledge from the web. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence; 2014 Jul 27–31; Quebec City, QC, Canada; 2014.
[65] Roth D, Yih W. A linear programming formulation for global inference in natural language tasks. In: Proceedings of the 8th Conference on Computational Natural Language Learning; 2004 May 6–7; Boston, MA, USA; 2004.
[66] Khashabi D, Khot T, Sabharwal A, Clark P, Etzioni O, Roth D. Question answering via integer programming over semi-structured knowledge. 2016. arXiv: 1604.06076.
[67] Khot T, Sabharwal A, Clark P. Answering complex questions using open information extraction. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics; 2017 Jul 30–Aug 4; Vancouver, BC, Canada; 2017. p. 311–6.
[68] Khashabi D, Khot T, Sabharwal A, Roth D. Question answering as global reasoning over semantic abstractions. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence; 2018 Feb 2–7; New Orleans, LA, USA; 2018.
[69] Punyakanok V, Roth D, Yih WT, Zimak D. Semantic role labeling via integer linear programming inference. In: Proceedings of the 20th International Conference on Computational Linguistics; 2004 Aug 23–27; Geneva, Switzerland; 2004.
[70] Srikumar V, Roth D. A joint model for extended semantic role labeling. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing; 2011 Jul 27–31; Edinburgh, UK; 2011. p. 129–39.
[71] Richardson M, Domingos P. Markov logic networks. Mach Learn 2006;62(1–2):107–36.
[72] Besag J. Statistical analysis of non-lattice data. Statistician 1975;24(3):179–95.
[73] Poon H, Domingos P. Unsupervised semantic parsing. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing; 2009 Aug 6–7; Singapore, Singapore; 2009.
[74] Pawar S, Bhattacharya P, Palshikar GK. End-to-end relation extraction using Markov logic networks. 2017. arXiv:1712.00988.
[75] Dai HJ, Tsai RTH, Hsu WL. Entity disambiguation using a Markov-logic network. In: Proceedings of the 5th Internatioanl Joint Conference on Natural Language Processing; 2011 Nov 8–13; Chiang Mai, Thailand; 2011. p. 846–55.
[76] Culotta A, McCallum A. Practical Markov logic containing first-order quantifiers with application to identity uncertainty. In: Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing; 2006 Jun 9; New York, NY, USA; 2006. p. 41–8.
[77] Weston J, Bordes A, Chopra S, Rush AM, van Merriënboer B, Joulin A, et al. Towards AI-complete question answering: a set of prerequisite toy tasks. 2015. arXiv:1502.05698.
[78] Sukhbaatar S, Szlam A, Weston J, Fergus R. End-to-end memory networks. In: Proceedings of the 2015 Neural Information Processing Systems Conference; 2015 Dec 7–12; Montreal, QC, Canada; 2015.
[79] Miller AH, Fisch A, Dodge J, Karimi AH, Bordes A, Weston J. Key-value memory networks for directly reading documents. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing; 2016 Nov 1–5; Austin, TX, USA; 2016. p. 1400–9.
[80] Yang Y, Yih WT, Meek C. WikiQA: a challenge dataset for open-domain question answering. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; 2015 Sep 17–21; Lisbon, Portugal; 2015. p. 2013–8.
[81] Bordes A, Boureau YL, Weston J. Learning end-to-end goal-oriented dialog. In: Proceedings of the 2017 International Conference on Learning Representations; 2017 Apr 24–26; Toulon, France; 2017.
[82] Guo D, Tang D, Duan N, Zhou M, Yin J. Dialog-to-action: conversational question answering over a large-scale knowledge base. In: Proceedings of the 2018 Neural Information Processing Systems Conference; 2018 Dec 3–8; Montreal, QC, Canada; 2018. p. 2942–51.
[83] Saha A, Pahuja V, Khapra MM, Sankaranarayanan K, Chandar S. Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In: Proceedings of the 32nd AAAI Conference on Artificial Intellgence; 2018 Feb 2–7; New Orleans, LA, USA; 2018.
[84] Zhou H, Young T, Huang M, Zhao H, Xu J, Zhu X. Commonsense knowledge aware conversation generation with graph attention. In: Proceeding of the 27th International Joint Conference on Artificial Intelligence; 2018 Jul 13–19; Stockholm, Sweden; 2018. p. 4623–9.
[85] Zhong V, Xiong C, Socher R. Seq2SQL: generating structured queries from natural language using reinforcement learning. 2017. arXiv:1709.00103.
[86] Trivedi P, Maheshwari G, Dubey M, Lehmann J. LC-QuAD: a corpus for complex question answering over knowledge graphs. In: Proceedings of the 2017 International Semantic Web Conference; 2017 Oct 21–25; Vienna, Austria; 2017. p. 210–8.
[87] Talmor A, Berant J. The web as a knowledge-base for answering complex questions. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology; 2018 Jun 3–5; New Orleans, LA, USA; 2018. p. 641–51.
[88] Levesque HJ, Davis E, Morgenstern L. The winograd schema challenge. In: Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning; 2012 Jun 10–14; Rome, Italy; 2012.
[89] Clark P, Cowhey I, Etzioni O, Khot T, Sabharwal A, Schoenick C, et al. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. 2018. arXiv:1803.05457.
[90] Talmor A, Herzig J, Lourie N, Berant J. CommonsenseQA: a question answering challenge targeting commonsense knowledge. 2018. arXiv:1811.00937.
[91] Sap M, Le Bras R, Allaway E, Bhagavatula C, Lourie N, Rashkin H, et al. ATOMIC: An atlas of machine commonsense for if-then reasoning. In: Proceedings of the 31rd AAAI Conference on Artificial Intelligence; 2019 Jan 27–Feb 1; Honolulu, HI, USA; 2019.
[92] Yang Z, Qi P, Zhang S, Bengio Y, Cohen WW, Salakhutdinov R, et al. HotpotQA: a dataset for diverse, explainable multi-hop question answering. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; 2018 Oct 31–Nov 4; Brussels, Belgium; 2018. p. 2369–80.
[93] Kočiský T, Schwarz J, Blunsom P, Dyer C, Hermann KM, Melis G, et al. The narrativeQA reading comprehension challenge. Trans Assoc Comput Linguist 2018;6:317–28.
[94] Khashabi D, Chaturvedi S, Roth M, Upadhyay S, Roth D. Looking beyond the surface: a challenge set for reading comprehension over multiple sentences. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology; 2018 Jun 3–5; New Orleans, LA, USA; 2018. p. 252–62.
[95] Reddy S, Chen D, Manning CD. CoQA: a conversational question answering challenge. 2018. arXiv:1808.07042.
[96] Hudson DA, Manning CD. GQA: a new dataset for real-world visual reasoning and compositional question answering. 2019. arXiv:1902.09506.
[97] R. Zellers Y. Bisk A. Farhadi Y. Choi From recognition to cognition: visual commonsense reasoning. In: Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition; 2019 Jun 16–20; Long Beach, CA, USA; 2019. 6720–31.
[98] Zellers R, Bisk Y, Schwartz R, Choi Y. SWAG: a large-scale adversarial dataset for grounded commonsense inference. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; 2018 Oct 31–Nov 4; Brussels, Belgium; 2018. p. 93–104.