A DQN-based Approach to Finding Precise Evidences for Fact Verification

Hai Wan[1], **Haicheng Chen**[1], **Jianfeng Du**[2,3]*, **Weilin Luo**[1], **Rongzhen Ye**[1]
[1] School of Computer Science and Engineering,
Sun Yat-sen University, Guangzhou 510006, P.R.China
[2] Guangzhou Key Laboratory of Multilingual Intelligent Processing,
Guangdong University of Foreign Studies, Guangzhou 510006, P.R.China
[3] Pazhou Lab, Guangzhou 510330, P.R.China
wanhai@mail.sysu.edu.cn, jfdu@gdufs.edu.cn,
{chenhch8, luowlin3, yerzh}@mail2.sysu.edu.cn

Abstract

Computing *precise evidences*, namely minimal sets of sentences that support or refute a given claim, rather than larger evidences is crucial in *fact verification* (FV), since larger evidences may contain conflicting pieces some of which support the claim while the other refute, thereby misleading FV. Despite being important, precise evidences are rarely studied by existing methods for FV. It is challenging to find precise evidences due to a large search space with lots of local optimums. Inspired by the strong exploration ability of the deep Q-learning network (DQN), we propose a DQN-based approach to retrieval of precise evidences. In addition, to tackle the label bias on Q-values computed by DQN, we design a post-processing strategy which seeks best thresholds for determining the true labels of computed evidences. Experimental results confirm the effectiveness of DQN in computing precise evidences and demonstrate improvements in achieving accurate claim verification.[1]

1 Introduction

With the growing false information, such as fake news, political deception and online rumors, automatic fact-checking systems have emerged to automatically identify and filter this information. *Fact verification (FV)* is a special fact-checking task that aims to retrieve related evidences from a text corpus to verify a textual claim.

Taking Figure 1 as example, an existing method for FV first retrieves related documents from the given corpus at stage 1 (namely the *document retrieval* stage), then finds key sentences from the documents at stage 2 (namely the *sentence selection* stage), and finally treats the set of key sentences as an evidence to verify the claim at stage

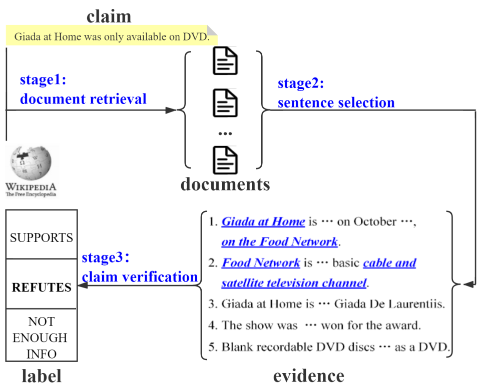
claim
Giada at Home was only available on DVD.
stage1:
document retrieval
documents
WIKIPEDIA
The Free Encyclopedia
stage2:
sentence selection
SUPPORTS
REFUTES
NOT ENOUGH INFO
label
stage3:
claim verification
1. Giada at Home is ⋯ on October ⋯, on the Food Network.
2. Food Network is ⋯ basic cable and satellite television channel.
3. Giada at Home is ⋯ Giada De Laurentiis.
4. The show was ⋯ won for the award.
5. Blank recordable DVD discs ⋯ as a DVD.
evidence

Figure 1: The pipeline for FV on FEVER. Underlined words in blue italics given in evidence provide key information to determine the truthfulness of the claim. “SUPPORTS” / “REFUTES” / “NOT ENOUGH INFO” indicates that the evidence can support / refute / is insufficient for supporting or refuting the claim. Both the evidence and label are output by FV.

3 (namely the *claim verification* stage). As can be seen in this example, it is desirable to retrieve an evidence consisting of the first two sentences only, since it does not contain unnecessary sentences to determine the truthfulness of the claim and can alleviate human efforts to further validate the evidence. More importantly, an evidence containing unnecessary sentences may involve conflicting pieces some of which support the claim while the other refute the claim. Thus, it is crucial to compute minimal sets of sentences that can determine the truthfulness of the claim. In this paper, we refer to a minimal set of sentences that supports or refutes a given claim as a *precise evidence*.

Existing methods for FV do not target the retrieval of precise evidences. Most existing studies (Thorne et al., 2018b; Nie et al., 2019; Zhou et al., 2019; Liu et al., 2020; Zhong et al., 2020; Ye et al., 2020; Subramanian and Lee, 2020; Wang et al., 2020) formulate FV as a three-stage pipeline

*Corresponding author
[1]Source code and data are available at https://github.com/sysulic/DQN-FV.

task as illustrated in Figure 1. This way makes the retrieval of precise evidences extremely difficult since the sentence selection stage is required to select a precise set of relevant sentences rather than a fixed number of sentences as in existing methods. To the best of our knowledge, TwoWingOS (Yin and Roth, 2018) is the only method by now which does not follow the three-stage pipeline. Instead, it exploits a supervised training scheme to train the last two stages jointly and is able to compute precise evidences. However, it exhibits a significantly worse performance than other state-of-the-art methods for FV, especially in terms of the recall of evidences. Therefore, there is still a need for designing new methods to compute precise evidences. These methods are expected to achieve better performance than TwoWingOS.

It is challenging to compute precise evidences. On one hand, the search space for precise evidences is very large. For example, in the benchmark Fact Extraction and VERification (FEVER) dataset (Thorne et al., 2018b) the average number of sentences for each claim input to the sentence selection stage is 40, and an output evidence has up to 5 sentences. Hence there are up to $\sum_{i=1}^{5} \mathrm{C}_{40}^{i} = 760,098$ candidates in the search space. On the other hand, greedy search of precise evidences easily falls into a local optimum. As shown in our experiments (see Table 6), a greedy search method does not perform well.

Inspired by the strong exploration ability of the Deep Q-learning Network (DQN) (Mnih et al., 2015), we develop a DQN-based approach to retrieval of precise evidences. In this approach, we first employ DQN to compute candidate pairs of precise evidences and their labels, and then use a post-processing strategy to refine candidate pairs. We notice that Q-values computed by DQN has label bias due to two reasons. On one hand, the label "NOT ENOUGH INFO" does not locate at the same concept level as "SUPPORTS" or "REFUTES". On the other hand, there is not a fixed range for Q-values, making Q-values hard to accurately estimate. Thus, a post-processing strategy is needed to tackle the label bias on Q-values. We develop such a strategy to seek best thresholds in determining the true labels of computed evidences.

Our experimental results on FEVER (Thorne et al., 2018b) confirm that our DQN-based approach is effective in finding precise evidences. More importantly, the approach is shown to outperform state-of-the-art methods for FV.

2 Related Work

2.1 Fact Extraction and Claim Verification

The FEVER 1.0 shared task (Thorne et al., 2018b) aims to develop an automatic fact verification system to determine the truthfulness of a textual claim by extracting related evidences from Wikipedia. Thorne et al. (2018a) has formalized this task, released a large-scale benchmark dataset FEVER (Thorne et al., 2018b), and designed the three-stage pipeline framework for FV, which consists of the document retrieval stage, the sentence selection stage and the claim verification stage. Most existing methods follow this framework and mainly focus on the last stage (Liu et al., 2020). For the document retrieval stage, most methods reuse the document retrieval component of top-performing systems (Hanselowski et al., 2018; Yoneda et al., 2018; Nie et al., 2019). For the sentence selection stage, there are three approaches commonly used, including keyword matching, supervised classification, and sentence similarity scoring (Thorne et al., 2018b). For the claim verification stage, most recent studies formulate this task as a graph reasoning task (Zhou et al., 2019; Liu et al., 2020; Ye et al., 2020; Zhong et al., 2020; Subramanian and Lee, 2020; Wang et al., 2020). Different from most existing methods that focus on claim verification, Yin and Roth (2018) proposed a supervised training method named TwoWingOS to jointly conduct sentence selection and claim verification.

Nowadays pre-trained language models like BERT (Devlin et al., 2019) have been widely used in claim verification (Li et al., 2019; Zhou et al., 2019; Soleimani et al., 2020). Following this way we employed RoBERTa (Liu et al., 2019), an enhanced version of BERT, as the sentence encoder in our DQN-based approach in experiments.

2.2 Deep Q-learning Network

Reinforcement learning (RL) is about an agent interacting with the environment, objective to maximize the cumulative rewards of a sequence of states and actions by adjusting its policies. Q-Learning (Mnih et al., 2015) is a popular reinforcement learning technique. It aims to approximate the optimal value function $Q^*(o, a)$ to measure the expected long-term rewards for a given pair of state o and action a. Deep Q-learning Network (DQN)

(Mnih et al., 2015) is a combination of deep learning and Q-Learning. It typically uses the following Equation (1) derived from the Bellman equation (Cao and ZhiMin, 2019) to approximate the optimal Q-value function:

$$Q(o^{(t)}, a^{(t)}) = \mathbb{E}_{o^{(t+1)}}[r^{(t)} + \lambda \max_{a'} Q(o^{(t+1)}, a')], \quad (1)$$

where $o^{(t)}, a^{(t)}, r^{(t)}$ respectively denote the state, action and reward at step t, and $\lambda \in [0, 1]$ is a discounted factor for future rewards.

3 Approach

3.1 Problem Setting

Given a set of candidate sentences $S = \{s_1, s_2, \dots\}$, a claim c, a set of precise evidences $\mathcal{E} \subset 2^S$, and a true label $y \in Y = \{\mathsf{T,F,N}\}$ that determines whether every precise evidence supports or refutes the claim, where $\mathsf{T/F/N}$ denotes "SUPPORTS"/"REFUTES"/"NOT ENOUGH INFO", we aim to train a model to predict a precise evidence; more precisely, to train a model for retrieving an evidence $\hat{E} \subset S$ and predicting a label $\hat{y} \in Y$ such that $\hat{y} = y$ and $\hat{E} = E$ for some $E \in \mathcal{E}$. This goal is different from the goal targeted by existing methods, which aim to retrieve an evidence $\hat{E} \subset S$ and predict a label $\hat{y} \in Y$ such that $\hat{y} = y$ and $E \subseteq \hat{E}$ for some $E \in \mathcal{E}$.

We define the four ingredients of DQN namely states, actions, transitions and rewards as follows:

- **State**. A state o is a tuple $(c, \hat{E}, \hat{y})$ for c a claim, $\hat{E}$ a set of sentences and $\hat{y}$ a label.
- **Action**. An action a is a sentence in S.
- **Transition**. A transition at step t is a tuple $(o^{(t)}, a^{(t)}, o^{(t+1)})$, where $o^{(t)} = (c, \hat{E}^{(t)}, \hat{y})$, $o^{(t+1)} = (c, \hat{E}^{(t+1)}, \hat{y})$ and $\hat{E}^{(t+1)} = \hat{E}^{(t)} \cup \{a^{(t)}\}$.
- **Reward**. The reward r for a transition $(o^{(t)}, a^{(t)}, o^{(t+1)})$ is defined as

$$r^{(t)} = \begin{cases} 1, & \hat{y} = y \wedge (y = \mathsf{N} \vee \exists E \in \mathcal{E} : a^{(t)} \in E) \\ -1, & \hat{y} \neq y \wedge |\hat{E}^{(t+1)}| = K \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where the number K is a hyper-parameter, and $|S|$ denotes the cardinality of a set S.

3.2 The DQN-based Model

The core of our proposed approach is the DQN-based model, illustrated in Figure 2.

3.2.1 Sentence Encoding Module

We employ RoBERTa in this module to extract the final hidden state of $\langle\text{s}\rangle$ as the sentence representation, where $\langle\text{s}\rangle$ and $\langle\text{/s}\rangle$ mentioned in the following are the special classification tokens in RoBERTa. Specifically, following KGAT (Liu et al., 2020), we first concatenate the claim c, the document title l, and a sentence s (*resp.* an action a) as "$\langle\text{s}\rangle c \langle\text{/s}\rangle l \langle\text{/s}\rangle s \langle\text{/s}\rangle$" (*resp.* "$\langle\text{s}\rangle c \langle\text{/s}\rangle l \langle\text{/s}\rangle a \langle\text{/s}\rangle$") and then feed it into RoBERTa to obtain the sentence representation $\mathbf{h}_s \in \mathbb{R}^{d_0}$ (*resp.* the action representation $\mathbf{h}_a \in \mathbb{R}^{d_0}$), where d_0 is the dimension of the representation. We also feed the claim "$\langle\text{s}\rangle c \langle\text{/s}\rangle$" alone to obtain the claim representation $\mathbf{h}_c \in \mathbb{R}^{d_0}$.

3.2.2 Evidence Encoding Module

This module is used to get an aggregated evidence representation. It consists of two sub-modules.

Context sub-module. It is obvious that the sentences in an evidence are always contextual dependent, so we apply two different networks BiLSTM (Nguyen et al., 2016) and Transformer (Vaswani et al., 2017) for comparison. These two different networks are widely used to encode contextual-aware information of sequential text in the NLP community. Formally, we either define

$$[\mathbf{h}'_{\hat{E}_0}, \dots, \mathbf{h}'_{\hat{E}_{|\hat{E}|-1}}] = \text{BiLSTM}(\mathbf{h}_a, H_{\hat{E}}) \quad (3)$$

if the BiLSTM network is used, or define

$$[\mathbf{h}'_{\hat{E}_0}, \dots, \mathbf{h}'_{\hat{E}_{|\hat{E}|-1}}] = \text{Transformer}(\mathbf{h}_a, H_{\hat{E}}) \quad (4)$$

if the Transformer is used, where $H_{\hat{E}} = [\mathbf{h}_{\hat{E}_0}, \dots, \mathbf{h}_{\hat{E}_{|\hat{E}|-1}}]$, $\mathbf{h}_{\hat{E}_i} \in \mathbb{R}^{d_0}$ is the i-th sentence representation in $\hat{E}$, $\mathbf{h}'_{\hat{E}_i} \in \mathbb{R}^{d_1}$ is the corresponding context-aware sentence representation in $\hat{E}$, and d_1 is the dimension of the representation.

Aggregation sub-module. This sub-module is used to fuse the sentence representations in evidences to obtain an aggregated evidence representation. We also apply two different networks in this sub-module: Transformer and attention. Unlike the Transformer with self-attention in the first sub-module, the query in this sub-module is the claim and the key/value is the context-aware sentence representation from the first sub-module. For the

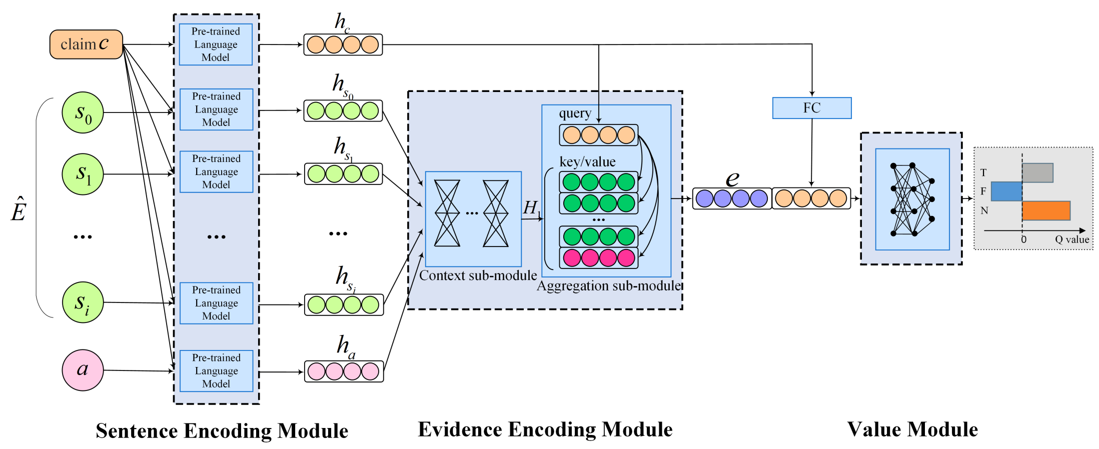

Figure 2: The architecture of the DQN-based model. The input is a state and an action, and the output is the Q-value of each label. The sentence encoding module is used to compute the sentence representation. The evidence encoding module is used to compute the evidence representation. The value module is used to predict the Q-value for each label.

attention network, we define

$$\mathbf{e} = \sum_{i=0}^{|\hat{E}|-1} \alpha_i \cdot \mathbf{h}'_i \quad (5)$$

$$\alpha_i = \frac{\exp(\mathrm{MLP}([\mathbf{h}_c; \mathbf{h}'_i]))}{\sum_{j=0}^{|\hat{E}|-1} \exp(\mathrm{MLP}([\mathbf{h}_c; \mathbf{h}'_j]))} \quad (6)$$

where $\mathbf{e} \in \mathbb{R}^{d_1}$ is the aggregated evidence representation, $\mathrm{MLP}(\cdot) = \mathrm{Linear}(\mathrm{ReLU}(\mathrm{Linear}(\cdot)))$ is a two-layer fully connected network using rectified linear unit as the activation function, and $[;]$ denotes the concatenation of two vectors.

3.2.3 Value Module

This module is used to obtain the Q-value vector for all labels, simply written as $Q(o, a; \theta)$ for θ denoting the set of learnable parameters, which is formally defined as

$$Q(o, a; \theta) = \mathrm{MLP}([\mathbf{h}_c \mathbf{W}; \mathbf{e}]) \quad (7)$$

where $\mathrm{MLP}(\cdot) = \mathrm{Linear}(\mathrm{ReLU}(\mathrm{Linear}(\cdot)))$ is similar to $\mathrm{MLP}(\cdot)$ used in Equation (6) except that different parameters in linear layers are used, $\mathbf{W} \in \mathbb{R}^{d_0 \times d_0}$ is a learnable matrix, and $Q(o, a; \theta) \in \mathbb{R}^{d_2}$ for d_2 the number of different labels.

3.3 Objective Function

Given a transition $(o^{(t)}, a^{(t)}, o^{(t+1)})$ and its reward $r^{(t)}$, we use the Double Deep Q-learning Network (DDQN) (Mnih et al., 2015) technique to train our model through the temporal difference error (Mnih et al., 2015). This error δ is formally defined as

$$\delta = Q_{\hat{y}}(o^{(t)}, a^{(t)}; \theta) - v(o^{(t+1)}, r^{(t)}) \quad (8)$$

where $v(\cdot)$ denotes the target value defined as

$$v(o, r) = \begin{cases} r, & \text{if } |\hat{E}| = K \\ r + \lambda \hat{Q}_{\hat{y}}(o, a^*; \hat{\theta}) & \text{otherwise} \end{cases} \quad (9)$$

for $a^* = \arg\max_{a \in S \setminus \hat{E}} Q_{\hat{y}}(o, a; \theta)$.

In the above equation, $\hat{Q}(\cdot; \hat{\theta})$ is the target network in DDQN, $Q_{\hat{y}}$ denotes the Q-value of $\hat{y}$ for $\hat{y}$ the predicted label in o, $\hat{E}$ is the predicted evidence in o, and $\lambda \in [0, 1]$ is a hyper-parameter representing the discount factor.

We use the Huber loss to minimise δ:

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{((o^{(t)}, a^{(t)}, o^{(t+1)}), r^{(t)}) \in \mathcal{B}} \mathcal{L}(\delta) \quad (10)$$

$$\mathcal{L}(\delta) = \begin{cases} \frac{1}{2}\delta^2 & \text{if } |\delta| \leq 1 \\ |\delta| - \frac{1}{2} & \text{otherwise} \end{cases} \quad (11)$$

where $\mathcal{B}$ is a batch of transition-reward pairs.

3.4 Algorithms

3.4.1 Model Training

Algorithm 1 shows how to train the DQN-based model. First, we initialize three replay memories, the DQN-based model, and the target network in Line 1-3. Then, in Line 9-17, we obtain the training

Algorithm 1: Model training for DQN, where the memory capacity M, the maximum evidence size K, the maximum number of epochs T and the reset interval C are hyper-parameters.

```
1  initialize a replay memory with a capacity $M$ for each
     label: $R_{\hat{y}} = \emptyset, \forall \hat{y} \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}$.
2  initialize DQN $Q(o,a;\theta)$ with random weights $\theta$.
3  initialize the target network $\hat{Q}(o,a;\hat{\theta})$ with $\hat{\theta} = \theta$.
4  for $e = 1 \to T$ do
5      shuffle the training set $\mathcal{D}$.
6      foreach $(c, y, \mathcal{E}, S) \in \mathcal{D}$ do
7          initialize one state for each label:
             $o_{\hat{y}}^{(0)} = (c, \hat{E}^{(0)}, \hat{y}), \forall \hat{y} \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}$,
             where $\hat{E}^{(0)} = \emptyset$.
8          for $t = 0 \to K-1$ do
9              foreach $\hat{y} \in \{T, F, N\}$ do
10                 if $random() < \epsilon\text{-}greedy$ then
11                     $a^{(t)} =$
                         random_select$(S \setminus \hat{E}^{(t)})$,
                         where $\hat{E}^{(t)}$ comes from $o_{\hat{y}}^{(t)}$.
12                 else
13                     $a^{(t)} =$
                         $\arg\max_{a \in S \setminus \hat{E}^{(t)}} Q_{\hat{y}}(o_{\hat{y}}^{(t)}, a; \theta)$,
                         where $Q(\cdot)$ is defined in
                         Eq. (7) and $Q_{\hat{y}}$ denotes the
                         Q-value of $\hat{y}$.
14                 end
15                 $o_{\hat{y}}^{(t+1)} = (c, \hat{E}^{(t+1)}, \hat{y})$, where
                     $\hat{E}^{(t+1)} = \hat{E}^{(t)} \cup \{a^{(t)}\}$ and
                     $\hat{E}^{(t)}$ comes from $o_{\hat{y}}^{(t)}$.
16                 calculate $r^{(t)}$ based on Eq. (2).
17                 store $((o_{\hat{y}}^{(t)}, a^{(t)}, o_{\hat{y}}^{(t+1)}), r^{(t)})$
                     into $R_y$.
18             end
19             sample a mini-batch of
                 transition-reward pairs from $R_{\mathsf{T}}$, $R_{\mathsf{F}}$,
                 $R_{\mathsf{N}}$ and update $Q(o,a;\theta)$ based on
                 Eq. (8)–(11).
20             for every $C$ steps reset the target
                 network $\hat{Q}(o,a;\hat{\theta})$ by $\hat{\theta} = \theta$.
21         endfor
22     end
23 endfor
24 return $Q(o,a;\theta)$
```

Algorithm 2: Candidate retrieval for a claim c from a set S of sentences, where K is the maximum evidence size.

```
1  initialize $\hat{E}_{\hat{y}} = [], q_{\hat{y}} = [], \forall \hat{y} \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}$.
2  initialize one state for each label:
     $o_{\hat{y}}^{(0)} = (c, \hat{E}^{(0)}, \hat{y}), \forall \hat{y} \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}$, where
     $\hat{E}^{(0)} = \emptyset$.
3  for $t = 0 \to K-1$ do
4      foreach $\hat{y} \in \{T, F, N\}$ do
5          $a^{(t)} = \arg\max_{a \in S \setminus \hat{E}^{(t)}} Q_{\hat{y}}(o_{\hat{y}}^{(t)}, a; \theta)$
6          $q^{(t)} = Q_{\hat{y}}(o_{\hat{y}}^{(t)}, a^{(t)})$
7          $o_{\hat{y}}^{(t+1)} = (c, \hat{E}^{(t+1)}, \hat{y})$, where
             $\hat{E}^{(t+1)} = \hat{E}^{(t)} \cup \{a^{(t)}\}$ and $\hat{E}^{(t)}$ comes
             from $o_{\hat{y}}^{(t)}$.
8          store $\hat{E}^{(t+1)}$ into $\hat{E}_{\hat{y}}$ and $q^{(t)}$ into $q_{\hat{y}}$.
9      end
10 endfor
11 return $\left\{(\hat{E}_{\hat{y}}, q_{\hat{y}})\right\}_{\hat{y} \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}}$
```

Algorithm 3: Making final prediction from $\left\{\langle\langle \hat{E}_{\hat{y}}^{(1)}, \ldots, \hat{E}_{\hat{y}}^{(K)} \rangle, \langle q_{\hat{y}}^{(0)}, \ldots, q_{\hat{y}}^{(K-1)} \rangle\rangle\right\}_{\hat{y} \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}}$, using thresholds $\alpha_{\mathsf{T}}, \alpha_{\mathsf{F}}, \alpha_{\mathsf{N}}$ for different labels.

```
1  let $t_y = \arg\max_{0 \le t \le K-1} q_y^{(t)}, \forall y \in \{\mathsf{T},\mathsf{F},\mathsf{N}\}$.
2  let $\hat{E} = \hat{E}_{\hat{y}}^{(t_{\hat{y}}+1)}$, where $\hat{y} = \arg\max_{y \in \{\mathsf{T},\mathsf{F}\}} q_y^{(t_y)}$.
3  if $q_N^{(t_N)} > \max\{q_T^{(t_T)}, q_F^{(t_F)}\}$ and
     $\min_{0 \le t \le K-1} q_N^{(t)} - \max_{\hat{y} \in \{T,F\}} q_{\hat{y}}^{(t_{\hat{y}})} > \alpha_N$ then
4      $\hat{y}' = \mathsf{N}$
5  else if $q_T^{(t_T)} > q_F^{(t_F)}$ then
6      if $q_T^{(t_T)} - \max_{\hat{y} \in \{F,N\}} q_{\hat{y}}^{(t_T)} > \alpha_T$ then $\hat{y}' = \mathsf{T}$ ;
7      else $\hat{y}' = \mathsf{N}$ ;
8  else
9      if $q_F^{(t_F)} - \max_{\hat{y} \in \{T,N\}} q_{\hat{y}}^{(t_F)} > \alpha_F$ then $\hat{y}' = \mathsf{F}$ ;
10     else $\hat{y}' = \mathsf{N}$ ;
11 end
12 return $(\hat{E}, \hat{y}')$
```

transition-reward pairs by letting the DQN-based model interact with the environment in an ϵ-greedy exploration-exploitation way (Mnih et al., 2015). Finally, in Line 19, we sample a mini-batch of transition-reward pairs to update the DQN-based model, while in Line 20, for every C steps we reset the target network to the DQN-based model.

3.4.2 Candidate Retrieval

Algorithm 2 shows how to retrieve a pair (candidate list, score list) for each label, where the candidate list stores progressively enlarged sentence sets, where each sentence set is a candidate of the predicted evidence, and the score list stores the strengths that the corresponding candidates support the label. We enlarge the two-list pair for each label through a greedy-search way (Line 3-10). Specifically, for each label, we first select the action with the largest Q-value (Line 5), then update the state by adding the chosen action into its predicted evidence (Line 7), and finally add the evidence and score into the corresponding list (Line 8).

Algorithm 4: Searching for best thresholds, where $\min q_{\hat{y}}$ is short for $\min_t q_{\hat{y}}^{(t)}$ and $\max q_{\hat{y}}$ for $\max_t q_{\hat{y}}^{(t)}$, for all $\hat{y} \in \{\mathsf{T}, \mathsf{F}, \mathsf{N}\}$.

```
1  construct $V = \{(q_\mathsf{T}, q_\mathsf{F}, q_\mathsf{N}, y)\}$ from the
     development set by Algorithm 2.
2  initialize $C_{\hat{y}} = L_{\hat{y}} = L'_{\hat{y}} = [], \forall \hat{y} \in \{\mathsf{T}, \mathsf{F}, \mathsf{N}\}$.
3  foreach $(q_T, q_F, q_N, y) \in V$ do
4      if $\max q_N > \max\{\max q_T, \max q_F\}$ then
5          $v = \min q_\mathsf{N} - \max\{\max q_T, \max q_F\}$
6          store $v$ into $L_\mathsf{N}$ and $(v, y)$ into $C_\mathsf{N}$.
7      end
8  end
9  sort $L_\mathsf{N}$ in ascending order.
10 calculate the medians of adjacent values in $L_\mathsf{N}$ and
     store them into $L'_\mathsf{N}$.
11 $\alpha_\mathsf{N} = \arg\max_{\alpha \in L'_\mathsf{N}} \sum_{(v,y) \in C_\mathsf{N}} \mathbb{1}((v > \alpha \wedge y = \mathsf{N}) \vee (v \leq$
     $\alpha \wedge y \neq \mathsf{N}))$
12 foreach $(q_T, q_F, q_N, y) \in V$ do
13     if $\max q_N \leq \max\{\max q_T, \max q_F\}$ or
         $\min q_N - \max\{\max q_T, \max q_F\} \leq \alpha_N$ then
14         $t_{\hat{y}} = \arg\max_t q_{\hat{y}}^{(t)}, \forall \hat{y} \in \{\mathsf{T}, \mathsf{F}\}$
15         if $q_T^{(t_T)} > q_\mathsf{F}^{(t_F)}$ then
16             $v = q_\mathsf{T}^{(t_\mathsf{T})} - \max\{q_\mathsf{F}^{(t_\mathsf{T})}, q_\mathsf{N}^{(t_\mathsf{T})}\}$
17             store $v$ into $L_\mathsf{T}$ and $(v, y)$ into $C_\mathsf{T}$.
18         else
19             $v = q_\mathsf{F}^{(t_\mathsf{F})} - \max\{q_\mathsf{T}^{(t_\mathsf{F})}, q_\mathsf{N}^{(t_\mathsf{F})}\}$
20             store $v$ into $L_\mathsf{F}$ and $(v, y)$ into $C_\mathsf{F}$.
21         end
22     end
23 end
24 foreach $\hat{y} \in \{T, F\}$ do
25     sort $L_{\hat{y}}$ in ascending order.
26     calculate the medians of adjacent values in $L_{\hat{y}}$
         and store them into $L'_{\hat{y}}$.
27     $\alpha_{\hat{y}} = \arg\max_{\alpha \in L'_{\hat{y}}} \sum_{(v,y) \in C_{\hat{y}}} \mathbb{1}(v > \alpha \wedge y =$
         $\hat{y}) - \mathbb{1}(v > \alpha \wedge y = \mathsf{N})$
28 end
29 return $(\alpha_\mathsf{T}, \alpha_\mathsf{F}, \alpha_\mathsf{N})$
```

3.4.3 Final Prediction

Algorithm 3 shows how to compute the target evidence-label pair from the (candidate list, score list) pairs obtained by Algorithm 2, where the thresholds are determined by Algorithm 4. In this algorithm, we first use the condition given by Algorithm 4 to predict N (Line 3), and then refine the prediction of T (Line 6) and F (Line 9) in turn. In Line 2, we focus on the evidences with the highest score for T and F, while we ignore the evidence for N, due to the following reasons: (1) there are no supporting sentences in the evidence for N; (2) we follow a strategy commonly used in existing methods for FV, *i.e.*, focusing only on the evidence for T and F.

Split	SUPPORTS	REFUTES	NEI
Train	80,035	29,775	35,639
Dev	6,666	6,666	6,666
Test	6,666	6,666	6,666

Table 1: Dataset statistics for FEVER

3.4.4 Threshold Searching

Algorithm 4 shows how to search for the best thresholds $(\alpha_\mathsf{T}, \alpha_\mathsf{F}, \alpha_\mathsf{N})$ to maximize the Label Accuracy (LA) over the development set. We first call Algorithm 2 to construct a set of tuples $(q_\mathsf{T}, q_\mathsf{F}, q_\mathsf{N}, y)$ from the development set, each of which corresponds to a development instance, where q_T, q_F and q_N are respectively the output score lists for the three labels T, F and N, and y is the corresponding true label (Line 1). We then go through the following two stages. The first stage (Line 3-11) finds a threshold α_N that can maximize LA for label N, where maximizing LA is amount to maximizing the difference between the number of correctly and incorrectly predicted instances. The second stage (Line 12-28) finds the thresholds α_T and α_F that can maximize LA for label T and F, respectively, where those instances that satisfy the conditions for N are neglected (Line 13).

4 Experiments

4.1 Experimental setting

4.1.1 Dataset

Our experiments are conducted on the large-scale benchmark dataset FEVER (Thorne et al., 2018a), which consists of 185,455 annotated claims with a set of 5,416,537 Wikipedia documents from the June 2017 Wikipedia dump. All claims are labeled as "SUPPORTS", "REFUTES", or "NOT ENOUGH INFO". What's more, each claim for "SUPPORTS" and "REFUTES" is accompanied by some evidences extracted from Wikipedia documents. The dataset partition is kept the same with Thorne et al. (2018b) as shown in Table 1.

4.1.2 Evaluation Metrics

The task has five evaluation metrics: 1) FEVER, the primary scoring metric that measures the accuracy of claim verification with a requirement that the predicted evidences fully covers the ground-true evidences for SUPPORTS and REFUTES claims; 2) Label Accuracy (LA), the accuracy of claim verification without considering the validity of the

Method	α_{T}	α_{F}	α_{N}
T-T	-1.23361155390739	-1.26671668887138	0.0153777748346328
T-A	-0.0631487071514129	0.0747150778770446	-1.48811344802379
BiLSTM-T	0.184351719915866	-0.64785711467266	-0.465365642681717
BiLSTM-A	-0.0904324240982532	-0.795884847640991	-0.403448916971683

Table 2: The thresholds determined by Algorithm 4. T-T, T-A, BiLSTM-T, and BiLSTM-A denote the architectures of the evidence encoding module, which are respectively Transformer-Transformer, Transformer-Attention, BiLSTM-Transformer, and BiLSTM-Attention.

predicted evidences; 3) Precision (Pre), the macro-precision of the evidences for SUPPORTS and REFUTES claims; 4) Recall, the macro-recall of the evidences for SUPPORTS and REFUTES claims; 5) F1, the F_1-score of the evidences for SUPPORTS and REFUTES claims. We choose F1 as our main metric because it can directly show the performance of methods on retrieval of precise evidences.

4.1.3 Implementation Details

Document retrieval. The document retrieval stage is kept the same as previous work (Hanselowski et al., 2018; Zhou et al., 2019; Liu et al., 2020; Ye et al., 2020). Given a claim, the method first utilizes the constituency parser from AllenNLP (Gardner et al., 2018) to extract potential entities from the claim. Then it uses the entities as search queries to find the relevant documents via the online MediaWiki API[2]. The convinced articles are reserved (Hanselowski et al., 2018).

Sentence selection and claim verification. We implement our DQN-based model with PyTorch and train it with the AdamW (Loshchilov and Hutter, 2019) optimizer while keeping the sentence encoding module frozen and inheriting the RoBERTa implementation from Wolf et al. (2020)[3]. Specifically, the learning rate is 5e-6, the batch size is 128, the training epochs is 30, the iteration steps (or largest evidence size, *i.e.*, K) is 5, the discount factor λ is 0.95, and the layer number of the context sub-module is 3. Prioritized experience replay memory (Schaul et al., 2016) with a capacity of 10,000 is used to store transitions. The target network is reset when DQN is updated every 10 times. The probability of ϵ-greedy policy starts at 0.9 and decays exponentially towards 0.05, and the rate of the decay is $\frac{1}{2000}$. Table 2 shows the thresholds α_{T}, α_{F} and α_{N} computed by Algorithm 4. All experiments were conducted on an NVIDIA GTX 2080ti 10GB GPU.

[2]`https://www.mediawiki.org/wiki/API:Main_page`

[3]`https://github.com/huggingface/pytorch-transformers`

4.1.4 Baselines

We compare our method with the following baselines, including six methods that focus on claim verification and one joint method **TwoWingOS** (Yin and Roth, 2018). The six methods include: (1) **GEAR** (Zhou et al., 2019) uses two kinds of attentions to conduct reasoning and aggregation in a graph model; (2) **KGAT** (Liu et al., 2020) employes the Kernel Graph Attention Network to capture fine-grained information over evidences for more accurate claim verification; (3) **DREAM** (Zhong et al., 2020) introduces semantic structures for evidences obtained by semantic role labeling in claim verification; (4) **CorefBERT** (Ye et al., 2020) extends KGAT and can explicitly model co-reference relationship in context; (5) **HESM** (Subramanian and Lee, 2020) is a framework that can encode and attend the claim and evidence sets at different levels of hierarchy; (6) **DGAT** (Wang et al., 2020) is a double graph attention network that performs well in multi-domain datasets. The join method **TwoWingOS** (Yin and Roth, 2018) exploits a two-wing optimization strategy that optimizes sentence selection and claim verification in a jointly supervised training scheme.

4.2 Results and Analysis

As shown in Table 3, we implement four versions of the evidence encoding module and evaluate them on the DEV set and the blind TEST set. The FEVER metric of the top six methods is calculated with the imprecise evidences, so we introduce the FEVER@5 metric for a fair comparison. We analyze our method from the following four aspects.

Comparison with the state-of-the-art methods. Results in Table 3 show that all versions (except BiLSTM-A) with post-processing significantly outperform the state-of-the-art methods on FEVER, Pre, and F1, especially for T-A on F1, which shows the superiority of our method in retrival of precise evidences. However, none of the four versions of our method can achieve the best result on FEVER@5, LA, and Recall. The reason for low recall is that the number of sentences in precise evidences is less than that in imprecise evidences, which means other methods have a higher probability to recall the ground-true evidences than ours. Besides, the relatively low LA is caused by the

Method		DEV FEVER@5	DEV FEVER	DEV LA	DEV Pre	DEV Recall	DEV F1	TEST FEVER@5	TEST FEVER	TEST LA	TEST Pre	TEST Recall	TEST F1
GEAR		70.69	-	74.84	24.08	86.72	37.69	67.10	-	71.60	-	-	36.87
KGAT		**76.11**	-	78.29	27.79	**94.37**	42.34	70.38	-	74.07	25.21	**87.47**	39.14
DREAM		-	-	-	26.60	87.33	40.79	70.60	-	**76.85**	25.63	85.57	39.45
CorefBERT		-	-	-	-	-	-	**71.80**	-	-	-	-	39.14
HESM		73.44	-	75.77	-	-	-	71.48	-	74.64	-	-	52.78
DGAT		-	-	-	-	-	-	66.91	-	71.79	-	-	-
TwoWingOS		-	56.16	**78.90**	47.73	53.81	50.59	-	54.33	75.99	44.68	49.91	47.15
Ours	T-T	72.83	71.55	78.18	50.42	81.82	62.39	70.16	68.91	75.74	48.76	79.91	60.56
	(w./o.)	72.90	70.00	74.87				70.43	68.23	73.13			
	T-A	73.32	**72.79**	78.35	**54.75**	79.92	**64.98**	70.81	**70.28**	76.14	**52.24**	77.93	**62.55**
	(w./o.)	73.29	72.60	78.12				70.82	70.18	76.00			
	BiLSTM-T	73.15	63.77	73.91	48.06	71.06	57.34	70.54	61.51	70.20	45.97	69.43	55.32
	(w./o.)	73.19	55.39	63.55				70.81	53.21	61.68			
	BiLSTM-A	72.99	70.88	77.79	35.50	76.54	48.50	70.11	68.21	75.53	33.76	74.50	46.46
	(w./o.)	73.20	65.65	71.21				70.55	63.38	69.32			

Table 3: Performance on DEV set and blind TEST set of FEVER (%). FEVER@5 and FEVER are computed based on imprecise and precise evidences, repectively. The result obtained with/without *post-processing* (namely threshold searching and final prediction) is displayed in each architecture's first/second row ("w."/"o."). We directly output the evidence with the highest score in the candidate list and its corresponding label if post-processing is not performed. Pre, Recall, and F1 keep unchanged because they are not affected by the post-processing. '-' denotes a missing value.

#	T-T	T-A	BiLSTM-T	BiLSTM-A	KGAT
LA	78.18	78.35	73.91	77.79	78.29
LA*	82.82	82.48	**84.93**	83.95	79.08

Table 4: Comparison between our method and KGAT on LA (%). LA and LA* are respectively evaluated on the DEV set and its subset constructed by selecting the samples where the ground-true evidences are successfully recalled.

Method	Avg.	Std.
Three-stage pipeline	4.00	0.07
Our method (T-A)	1.07	0.89

Table 5: Comparison of the number of unnecessary sentences in predicted evidences.

width	FEVER@5	FEVER	LA	Pre	Recall	F1
1	60.73	54.91	72.69	52.76	58.57	55.51
(w./o.)	50.09	46.55	53.00			
2	60.74	54.94	72.69	52.84	58.66	55.59
(w./o.)	50.09	46.53	53.00			
3	60.70	54.96	72.69	52.84	58.67	55.60
(w./o.)	50.10	46.54	53.00			
4	60.67	54.95	72.69	52.81	58.66	55.58
(w./o.)	50.09	46.54	53.00			
5	60.68	54.95	72.69	52.84	58.68	55.61
(w./o.)	50.09	46.54	53.00			

Table 6: The beam-search result of KGAT on the DEV set (%). The width (k) means to select the top-k results at each search step. The result obtained with/without *post-processing* (namely threshold searching and final prediction) is displayed in each width's first/second row ("w."/"o."). We employed the KGAT source code released by Liu et al. (2020) to implement beam-search for finding precise evidences and the evaluation data for KGAT was kept the same as ours.

low Recall of precise evidences. To further clarify this point, we evaluate our method on a subset of the DEV set where the ground-true evidences are recalled successfully. Our method improves significantly the performance on this subset, as shown in Table 4, which justifies our point of view. FEVER is affected by the LA and Recall, thereby the low FEVER@5 is also due to the low recall of precise evidences. In addition, the results reported in Table 5 show that our method can significantly reduce the number of unnecessary sentences in a predicted evidence.

Comparison between different versions. As shown in Table 3, T-T and T-A perform respectively better than BiLSTM-T and BiLSTM-A on almost all metrics except that T-T is slightly worse than BiLSTM-A on FEVER@5, which suggests Transformer can encode better context-aware representations than BiLSTM in our context sub-module. Moreover, we find that T-A performs better than T-T on almost all metrics except Recall and that BiLSTM-A is worse than BiLSTM-T on Pre and F1. This contrary result shows that the performance of the aggregation sub-module is impacted by the context sub-module. Thus, the choice between Transformer and Attention should depend on the context sub-module. Overall, T-A achieves the best performance among all the four versions of our

#	label	claim	ground-true evidences		predicted evidences	
				GEAR	KGAT	Our method (T-A)
1	F	Savages was exclusively a German film.	(Savages(2012 film), 3)	***(Savages(2012 film), 3)*** (Savages(band), 0) (Savages(2012 film), 6) (Savages(band), 2) (Savages(band), 4)	***(Savages(2012 film), 3)*** (Savages(2012 film), 6) (Savages(2012 film), 0) (Savages(band), 5) (Savages(band), 0)	***(Savages(2012 film), 3)***
2	T	Ed Gein murdered people around Plainfield, Wisconsin.	(Ed Gein, 2) (Ed Gein, 1)	***(Ed Gein, 1)*** (Ed Gein, 0) (Ed Gein, 6) ***(Ed Gein, 2)*** (Ed Gein, 5)	***(Ed Gein, 1)*** ***(Ed Gein, 2)*** (Ed Gein, 0) (Ed Gein(band), 2) (Ed Gein, 4)	***(Ed Gein, 2)*** ***(Ed Gein, 1)***
3	T	Marnie is a film that was created in the United States.	(Marnie(film), 0)	***(Marnie(film), 0)*** (Marnie, 0) (Marnie(film), 2) (Marnie(film), 6) (Marnie(dis...tion), 12)	***(Marnie(film), 0)*** (Marnie, 0) (Marnie(film), 2) (Marnie(film), 6) (Marnie(film), 5)	***(Marnie(film), 0)*** (Marnie(film), 5)
4	F	First Motion Picture Unit produced zero films.	(First ... Unit, 1) (First ... Unit, 4) (First ... Unit, 0)	***(First ... Unit, 0)*** ***(First ... Unit, 4)*** ***(First ... Unit, 1)*** (Zero(2016 film), 0) (First ... Unit, 8)	***(First ... Unit, 1)*** ***(First ... Unit, 4)*** ***(First ... Unit, 0)*** (First ... Unit, 2) (Zero(2016 film), 0)	***(First ... Unit, 1)*** ***(First ... Unit, 4)*** ***(First ... Unit, 0)*** (First ... Unit, 2)

Table 7: Cases in FEVER. We list the predicted evidences of GEAR, KGAT and our method. (title, i) denotes the i-th sentence in the corresponding wiki document. In predicted evidences, the sentences highlighted in ***blue bold italics and underline*** are sentences in the target evidence while others in black are unnecessary ones.

proposed method.

Comparison on retrieval of precise evidences. TwoWingOS is a supervised-learning method that can also find precise evidences. Although it achieves slightly better performance on LA than ours, its F1 and other metrics are much worse, indicating that it performs worse than our method except for BiLSTM-A in retrieval of precise-evidences. We also enhance KGAT to conduct beam-search for finding precise evidences and report the results in Table 6. The F1 score of KGAT is always higher than TwoWingOS but is still lower than our method except for BiLSTM-A.

Comparison between the methods with and without post-processing. It can be seen from Table 3 and Table 6 that, post-processing (namely threshold searching and final prediction from candidates) consistently improves FEVER and LA. Although with post-processing, our method (except T-A) achieves slightly lower scores on FEVER@5, KGAT still achieves significantly higher scores on FEVER@5 as on other metrics. These results show that post processing is very important in retrieval of precise evidences.

4.3 Case Study

In Table 7 we provide some cases to demonstrate the effectiveness of our method (T-A) in retrieving precise evidences. In case#1 and case#2, our method exactly finds ground-true evidences without introducing any unnecessary sentence, while GEAT and KGAT cannot. In case#3 and case#4, our method generates less unnessary sentences in prdicted evidents than GEAT and KGAT do.

5 Conclusion and Future Work

In this paper, we have proposed a novel DQN-based approach to finding precise evidences for fact verification. It provides a method to solve the precise-evidence problem by first employing a DQN to compute some candidates and then introducing a post-processing strategy to extract the target evidence and its label from the candidates. Experimental results show that the approach achieves state-of-the-art performance in terms of retrieval of precise evidences. Besides, to the best of our knowledge, it is the first attempt to employ DQN in the fact verification task.

Future work will incorporate external knowledge into our approach to improve the retrieval recall.

Acknowledgments

This work is supported by the Guangdong Province Science and Technology Plan projects (2017B010110011), the National Natural Science Foundation of China (No. 61876204, 61976232, and 51978675), the National Key R&D Program of China (No.2018YFC0830600), Guangdong Province Natural Science Foundation (No. 2018A030313086), All-China Federation of Returned Over-seas Chinese Research Project (No. 17BZQK216).

References

LiChun Cao and ZhiMin. 2019. An overview of deep reinforcement learning. In *CACRE*, pages 17:1–17:9.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *NLP-OSS Workshop*, pages 1–6.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-athene: Multi-sentence textual entailment for claim verification. In *FEVER*, pages 103–108.

Tianda Li, Xiaodan Zhu, Quan Liu, Qian Chen, Zhigang Chen, and Si Wei. 2019. Several experiments on investigating pretraining and knowledge-enhanced models for natural language inference. *CoRR*, abs/1904.12104.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained fact verification with kernel graph attention network. In *ACL*, pages 7342–7351.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Ngoc-Khuong Nguyen, Anh-Cuong Le, and Hong Thai Pham. 2016. Deep bi-directional long short-term memory neural networks for sentiment analysis of social data. In *IUKM*, volume 9978, pages 255–268.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *AAAI*, pages 6859–6866.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *ICLR*.

Amir Soleimani, Christof Monz, and Marcel Worring. 2020. BERT for evidence retrieval and claim verification. In *ECIR*, volume 12036, pages 359–366.

Shyam Subramanian and Kyumin Lee. 2020. Hierarchical evidence set modeling for automated fact extraction and verification. In *EMNLP*, pages 7798–7809.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and VERification (FEVER) shared task. In *FEVER*, pages 1–9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Yongyue Wang, Chunhe Xia, Chengxiang Si, Beitong Yao, and Tianbo Wang. 2020. Robust reasoning over heterogeneous textual information for fact verification. *IEEE Access*, 8:157140–157150.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*, pages 38–45.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential reasoning learning for language representation. In *EMNLP*, pages 7170–7186.

Wenpeng Yin and Dan Roth. 2018. Twowingos: A two-wing optimization strategy for evidential claim verification. In *EMNLP*, pages 105–114.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. UCL machine reading group: Four factor framework for fact finding (HexaF). In *FEVER*, pages 97–102.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *ACL*, pages 6170–6180.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: graph-based evidence aggregating and reasoning for fact verification. In *ACL*, pages 892–901.