

Understand the Dynamic World: An End-to-End Knowledge Informed Framework for Open Domain Entity State Tracking

Mingchen Li
Georgia Sate University
Atlanta, USA
lmingchen96@gmail.com

Lifu Huang
Virginia Tech
Blacksburg, USA
lifuh@vt.edu

ABSTRACT

Open domain entity state tracking aims to predict reasonable state changes of entities (i.e., [attribute] of [entity] was [before_state] and [after_state] afterwards) given the action descriptions. It's important to many reasoning tasks to support human everyday activities. However, it's challenging as the model needs to predict an arbitrary number of entity state changes caused by the action while most of the entities are implicitly relevant to the actions and their attributes as well as states are from open vocabularies. To tackle these challenges, we propose a novel end-to-end Knowledge Informed framework for open domain Entity State Tracking, namely Kiest, which explicitly retrieves the relevant entities and attributes from external knowledge graph (i.e., ConceptNet) and incorporates them to autoregressively generate all the entity state changes with a novel dynamic knowledge grained encoder-decoder framework. To enforce the logical coherence among the predicted entities, attributes, and states, we design a new constraint decoding strategy and employ a coherence reward to improve the decoding process. Experimental results show that our proposed Kiest framework significantly outperforms the strong baselines on the public benchmark dataset – OpenPI.¹

CCS CONCEPTS

• Computing methodologies → Natural language processing.

KEYWORDS

Open Domain Entity State Tracking, Knowledge Informed Generation, Constraint Decoding, Coherent Reward

ACM Reference Format:

Mingchen Li and Lifu Huang. 2023. Understand the Dynamic World: An End-to-End Knowledge Informed Framework for Open Domain Entity State Tracking. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*, July 23–27, 2023, Taipei, Taiwan. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3539618.3591781>

¹The source code, data, and checkpoints will be publicly available at https://github.com/VT-NLP/open_domain_entity_state_tracking.



This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGIR '23, July 23–27, 2023, Taipei, Taiwan
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9408-6/23/07.
<https://doi.org/10.1145/3539618.3591781>

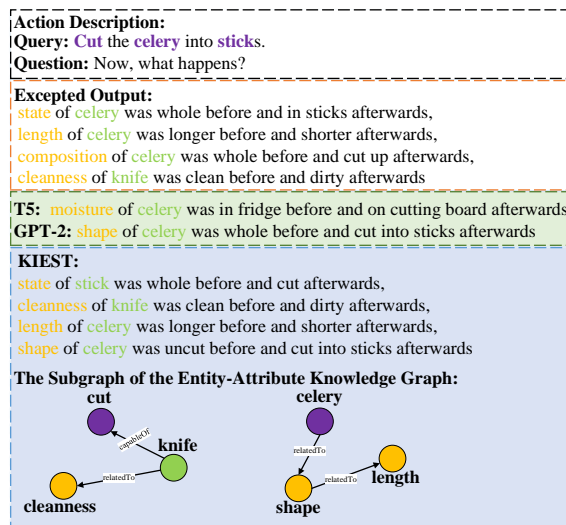


Figure 1: An example demonstrating the input, expected outputs and system outputs (from T5 [21], GPT-2 [20] and our Kiest approach) for open-domain entity state tracking. The purple words in the action description highlight the keywords that are used to retrieve the additional entity and attribute knowledge, e.g., the concepts shown in the bottom entity-attribute knowledge graph. The green and orange words in the expected outputs and system outputs highlight the entities and attributes for describing the entity state changes.

1 INTRODUCTION

Open domain entity state tracking [30] aims to predict all the state changes of entities that are related to a given action description, where each state change can be described with a template, e.g., “[attribute] of [entity] was [before_state] and [after_state] afterwards”. It's an important task to many reasoning and information retrieval systems to better understand human daily activities and make recommendations, e.g., the subsequent actions that humans need to perform. Figure 1 shows an example, where given the action, *Cut the celery into sticks.*, we need to infer all the entity state changes that are related to the action, such as *length of celery was longer before and shorter afterwards*, *cleanness of knife was clean before and dirty afterwards*, and so on.

There are two particular challenges for the open domain entity state tracking task: first, given that the actions are from open domains, the entities, attributes, and states that are related to state

changes are usually from open set vocabularies, e.g., in the example of Figure 1, the entity such as *knife* is not explicitly mentioned in the action description and there is no close-set attribute vocabulary to indicate the state changes of entities, making it hard to be formulated as a reading comprehension task as previous studies [6, 16, 27] which only focus on a few predefined entities and states (e.g., *location*, *existence*). In addition, there could be an arbitrary number of entity state changes that are caused by the action while the system is expected to predict most, if not all, state changes of entities. Previous study [30] tackles these challenges by exploring an autoregressive language model, such as GPT-2 [20], to directly generate all the entity state changes (green box in Figure 1). However, as shown in Figure 1, these approaches suffer from very low coverage of the entities and attributes for the predicted state changes, and without any constraint, the models easily generate state changes that are not related to the context of the action or consistent with human commonsense, e.g., the generated “*fridge*” (*before_state*) from T5 is not coherent with the attribute “*moisture*”.

In this work, we argue that knowledge graphs (KGs), such as ConceptNet [12], can provide meaningful knowledge to inform the model of relevant entities and attributes to the given action description, and thus facilitate the model to better predict entity state changes. For example, as shown in Figure 1, given the source concepts, such as *cut*, from the action description, ConceptNet provides more implicitly relevant concepts, such as *knife*, *cleanliness* and so on. Motivated by this, we propose a novel end-to-end Knowledge Informed framework for open domain Entity State Tracking, namely KIEST, which consists of two major steps: (1) retrieving and selecting all the relevant entities and attributes to the given action description from ConceptNet; (2) incorporating the external entity and attribute knowledge into a novel *Dynamic Knowledge Grained Encoder-Decoder* framework, which is based on an autoregressive pre-trained language model, such as T5 [21], to generate all the entity state changes. To encourage the model to generate coherent and reasonable state changes, we further propose a constrained decoding strategy and an *Entity State Change Coherence Reward (ESCCR)* to improve the generation process. The experimental results demonstrate the effectiveness of our proposed KIEST framework with a significant improvement over the strong baselines. The contributions of this work can be summarized as:

- To the best of our knowledge, we are the first to incorporate the external entity and attribute knowledge to inform the model to better generate entity state changes with higher coverage;
- We design a novel Dynamic Knowledge Grained Encoder-Decoder approach to dynamically incorporate the external knowledge and autoregressively generate all the entity state changes;
- We also design a new constrained decoding strategy and an automatic reward function to estimate the coherence of entity state changes so as to encourage the model to generate state changes that are more coherent to the context actions and human commonsense;
- We conduct a thorough analysis of our method, including an ablation study, demonstrating the robustness of our framework.

2 RELATED WORK

2.1 Entity State Tracking

Tracking the state changes of entities is important to understand the natural language text describing actions and procedures. Most previous studies [3, 6, 15, 16, 18, 23, 25] only focus on a particular domain with a set of predefined entities and attributes. For example, Mishra et al. [16] tackle this problem as a question-answering task and only focus on *location* and *existence* attributes to track the entity states in process paragraphs. Faghihi and Kordjamshidi [6] propose a Time-Stamped Language Model to understand the *location* changes of entities. PiGLET [36] predicts the post-state of an entity by giving its pre-state, a specific attribute and context, while EVENTS REALM [27] determines whether an entity has a state change with respect to the set of given attributes. Recently, [30] further extended entity state tracking to open domain actions and explore pre-trained language models, such as GPT-2 [20], to autoregressively generate all the entity state changes. Compared with all these studies, our work focuses on open-domain entity state tracking and aims to encourage the model to generate entity state changes with high coverage and are more coherent to the context and human commonsense.

2.2 Knowledge Informed Language Understanding and Generation

Many studies [1, 2, 8, 10, 13, 28, 29, 32, 34, 38] have been proposed to incorporate external knowledge to better understand the text or generate the expected output. For example, on the language understanding task, [13] injects expanded knowledge into the language model by adding the entity and relation from the knowledge graph as additional words. Different from the masking strategy of BERT [5], [29] proposes an entity-level masking strategy to incorporate the informative entities into the language model. [2] verbalize extracted facts aligning with input questions as natural language and incorporate them as prompts to the language model to improve story comprehension. For open-domain question answering, [8] incorporate the informative entities extracted from the input question and passage with the output of language model T5 to jointly optimize the knowledge representations based on their proposed relation-aware GNN. Inspired by these studies, we retrieve the relevant entities and attributes to the action description from external knowledge graphs and further dynamically incorporate them to better predict the entity state changes.

3 METHODOLOGY

Given a procedural paragraph with a sequence of action descriptions, we aim to predict all the state changes of entities related to each action. We follow [30] and formulate the task of open domain entity state tracking as follows. The **Input** $x = (x_c, x_q)$ consists of a *context* x_c that includes one or a few history action descriptions and a *query* x_q that is the concatenation of a description for the current action and a short question “*what happens?*”, and the **Output** is a set of entity state changes $s = \{s_j\}$ while each s_j follows a template: *[attribute] of [entity] was [before_state] before and [after_state] afterwards.*

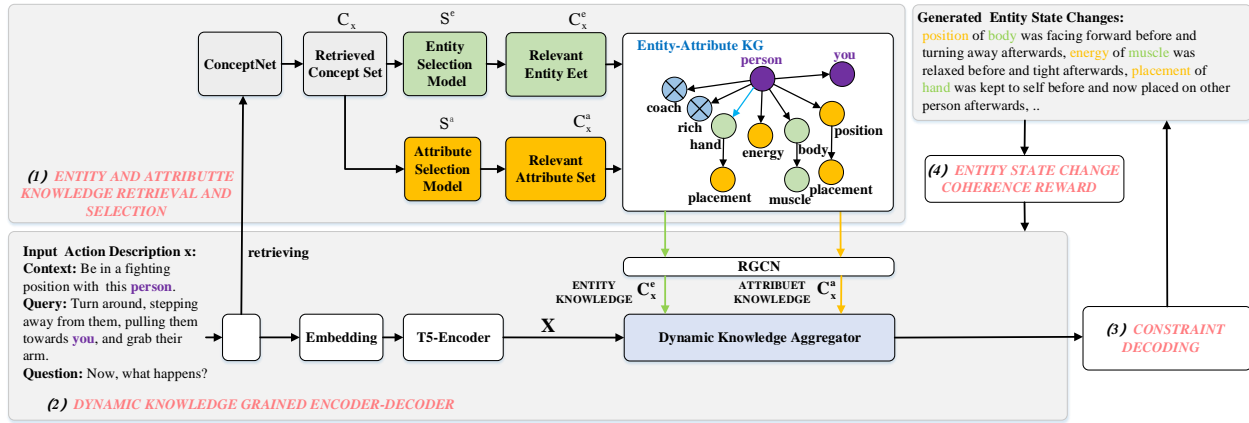


Figure 2: Overview of the KIEST. In the Entity-attribute KG, the green circle, orange circle, and blue circle refer to the entity, attribute and noise entity separately. The black line and blue line refer to the relation *relatedTo* and *capableof* separately.

Intuitively, when asked to write state changes given an action description, humans will always call to mind the scenarios based on the action description, and then conceive the relevant entities and their state changes about certain attributes. We thus follow this human cognitive process and propose KIEST, a knowledge-informed framework to track the entity states given open domain actions. Specifically, as illustrated in Figure 2, KIEST consists of two main steps: (1) it first retrieves and selects all the candidate entities and attributes that are related to the action description from an external knowledge graph. Here, we use ConceptNet [12] given its high coverage of open-domain concepts; (2) it then dynamically incorporates the relevant entity and attribute knowledge into a **Dynamic Knowledge Grained Encoder-Decoder** to generate all the entity state changes while a constrained decoding strategy and a **Entity State Change Coherence Reward** are employed to encourage the decoder to generate state changes that are more coherent to the action descriptions and human commonsense. Next, we provide details for each of the components.

3.1 Entity and Attribute Knowledge Retrieval and Selection

We observe that state changes usually happen to the entities that are either directly included in the action description or are conceptually relevant to the key entities and actions (called as *anchors*) included in the action description. For example, in Figure 2, the state change happens to the entity *body*, which is closely related to the anchor *person* in the action description. Similarly, the entity related attributes are also conceptually related to the anchors contained in the action description.

Motivated by this, we propose to acquire a rich set of entities and attributes that are relevant to state changes from ConceptNet [26], a general KG covering about 4,716,604 open-domain concepts and their relations. Specifically, given the input $x = (x_c, x_q)$, we first find all the spans in x that are included as concepts in ConceptNet and take each span as an anchor to retrieve the connected concepts within H hops in ConceptNet, denoted as C_x . Taking the action description in Figure 2 as an example, given the context and query

as input, we first extract the anchors from them, such as *person* and *you*, and then take each anchor as a query to ConceptNet and obtain a set of relevant concepts, such as *coach*, *rich*, *hand*, *placement*, *energy*, *body*, *position*, *muscle* and so on.

For each input x , there are hundreds or thousands of neighboring concepts retrieved in C_x while most of them are not related to entity state changes. For instance, in Figure 2, the retrieved concepts, such as *rich* and *coach*, are not relevant to any state changes. Based on this observation, we further design two selection models S^e and S^a to select the most relevant entity and attribute knowledge to the action description from C_x . Both selection models share the same architecture and training objective. Taking S^e as an example, it takes the following information as input: (1) action description x ; (2) positive entity set $C^p = \{e^p\}$, which is constructed based on the entities included in the human-annotated state changes; (3) negative entity set $C^n = \{e^n\}$, which is created by randomly sampling the entities from state changes annotated for other actions while ensuring each e^n is not included in C^p . To differentiate the positive and negative entities, S^e utilizes a pre-trained BERT model $f(\cdot)$ [5, 11] to extract a semantic representation $f(x)$, $f(e^p)$, $f(e^n)$ for x , e^p , e^n , respectively, and select the positive entities by measuring their distance to the action description. S^e is optimized with the following triple loss:

$$\max(\|f(x) - f(e^p)\| - \|f(x) - f(e^n)\| + \alpha, 0)$$

where $\|\cdot\|$ denotes the Euclidean Distance and α is a margin parameter, which we set to 1 as default. During training, the triplet loss reduces the distance between $f(x)$ and $f(e^p)$ while enlarging the distance between $f(x)$ and $f(e^n)$. At inference time, we calculate the similarity scores between x and each candidate concept from C_x and select the candidates as relevant entity knowledge C_x^e if their scores are higher than a threshold ϵ . ϵ is regarded as a hyper-parameter and is discussed in section 6.1.

We utilize the same architecture for the attribute selection model S^a and the same way to obtain the positive and negative set of attributes for each input x to optimize S^a with the triplet loss. Note that, during inference, for each input x , we apply S^a to select the relevant attribute knowledge C_x^a from the same set of candidate

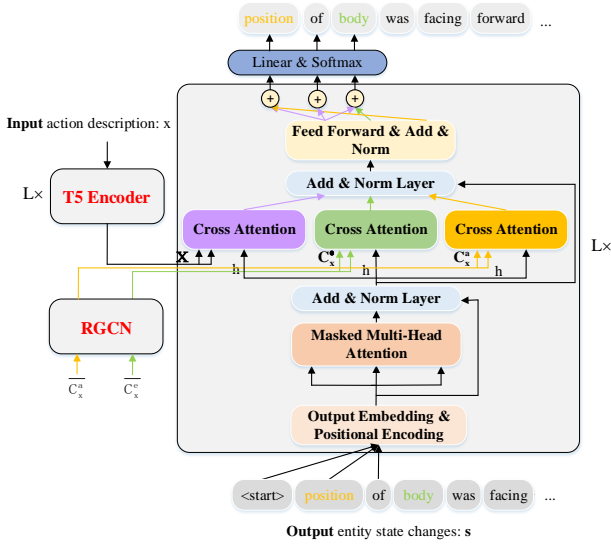


Figure 3: Overview of the DKGED decoder which dynamically incorporates additional relevant entities or attributes to generate entity state changes.

concepts C_x . After selecting the relevant entities and attributes for input x , we construct a heterogeneous knowledge graph, named entity-attribute knowledge graph (KG): $\mathcal{G} = (\mathcal{E}^x, \overline{C}_x^e, \overline{C}_x^a, \mathcal{R})$, where \mathcal{E}^x denotes the set of anchor concepts extracted from the input x and \mathcal{R} denotes the links among the concepts of \mathcal{E}^x , C_x^e and C_x^a . We remove a concept from \overline{C}_x^e and \overline{C}_x^a if it's not connected with any other concepts, thus \overline{C}_x^e and \overline{C}_x^a are subsets of C_x^e and C_x^a , respectively.

3.2 Dynamic Knowledge Grained Encoder-Decoder Framework

After selecting entities and attributes from ConceptNet that are relevant to the entity state changes, we aim to further incorporate them to improve the generation of the entity state changes. To do this, we propose a Dynamic Knowledge Grained Encoder-Decoder (DKGED) framework to dynamically select the relevant entity or attribute knowledge when we predict the tokens of each slot in the state change template, i.e., “[attribute] of [entity] was [before_state] and [after_state] afterwards”. Specifically, DKGED takes the action description x , relevant entity set \overline{C}_x^e and attribute set \overline{C}_x^a as input and generates a set of entity state changes $s = \{s_j\}$. We illustrate the DKGED at the bottom of Figure 2 and describe its major components as follows.

DKGED first computes the hidden representation $\mathbf{X} = [w_1, \dots, w_n]$ of each token w_n in the raw action description x :

$$\mathbf{X} = \text{Encoder}(w_1, \dots, w_n)$$

where $\text{Encoder}(\cdot)$ is a L -layer Transformer encoder of T5. To capture the interaction among the relevant entities and attributes in the entity-attribute KG, we utilize the Relational Graph Convolutional Network (RGCN) [24] to learn the embedding matrix C_x^e and C_x^a

for the entity set \overline{C}_x^e and attribute set \overline{C}_x^a , respectively:

$$C_x^e, C_x^a = \text{RGCN}(\overline{C}_x^e, \overline{C}_x^a)$$

After encoding the input action description and the relevant entity and attribute knowledge, DKGED further autoregressively generates a set of entity state changes, denoted as a concatenation of a sequence of words $[s_1, \dots, s_j]$, while dynamically incorporating the relevant entities or attributes. Specifically, as shown in Figure 3, the DKGED adopts a L -layers of Transformer-based decoder, which converts the previous tokens into vector representations together with positional embedding and encodes them with a masked multi-head attention (MHA) in each layer:

$$\tilde{h}_i^l = \text{LN}(\text{MHA}(h_{1:i-1}^l + h_i^{l-1}))$$

where $h_{1:i-1}^l, \tilde{h}_i^l$ denote the vector representation of previous tokens and current token at layer l , respectively. h_i^{l-1} is the representation of the current token at layer $l-1$. LN denotes the Layer Normalization layer.

At each generation step i , DKGED dynamically incorporates the relevant entity knowledge, attribute knowledge, and the input action description to predict the output token, based on the position of i in the state change template: “[attribute] of [entity] was [before_state] and [after_state] afterwards”. Intuitively, when the current decoding step aims to predict an attribute, the external attribute knowledge is more meaningful than entity knowledge. Similarly, when the current decoding step is to predict an entity, the external entity knowledge is more helpful. Other positions, such as the template tokens (e.g., *of*, *was*, *afterwards*) and states can be decoded based on the input action descriptions. To achieve this goal, we design three Cross Attention mechanisms while each Cross Attention is the same as Multi-Head Attention defined in Transformer [31]:

$$\begin{aligned} A_i^l &= \text{LN}(\text{CA}(\tilde{h}_i^l, C_x^a) + \tilde{h}_i^l) & i \in \{[e^b, e^e]\} \\ E_i^l &= \text{LN}(\text{CA}(\tilde{h}_i^l, C_x^e) + \tilde{h}_i^l) & i \in \{[a^b, a^e]\} \\ H_i^l &= \text{LN}(\text{CA}(\tilde{h}_i^l, \mathbf{X}) + \tilde{h}_i^l) & \text{other} \end{aligned}$$

where A_i^l, E_i^l , and H_i^l is a contextual representation by attending over all the relevant attribute knowledge, entity knowledge, and all the input representations, respectively. e^b and e^e are the start and end word positions of the entity in the template. a^b and a^e are the start and end word positions of the attribute in the template. During the decoding process, these positions are known based on the previously generated template tokens, such as *of*, *was*.

Based on these contextual representations and the decoding position i in the state change template, we further apply a Feed Forward layer to obtain an overall feature representation h_i^l . The feature representation from the last layer h_i^{l-1} is then used to predict a probability distribution over the whole vocabulary V based on Softmax

$$P(s_i | ((s_{1:i-1}, x, \overline{C}_x^e, \overline{C}_x^a); \theta)) = \text{softmax}(\mathbf{W} \cdot h_i^l)$$

where \mathbf{W} is a learnable parameter matrix.

3.3 Constrained Decoding

At each decoding step, existing studies [7, 9, 21] usually predict the output token from the whole vocabulary V , which results in outputs that are not related to the input. To improve the decoding process, we further design a *constrained decoding* strategy to select a subset of candidate tokens that are more related to the input action description from the vocabulary V . Specifically, given an input x and a candidate token c from V , we calculate a *cosine similarity score* between c and each token in x based on their contextual representations from a pre-trained T5 encoder, and use the highest score as the relevance score ξ_x between c and x :

$$r_n = \text{cosine_similarity}(c, w_n), w_n \in x$$

$$\xi_x = \max(r_0, r_1, \dots, r_n)$$

where w_n is the n -th word in x , c and w_n is the contextual representation of c and w_n from a pre-trained T5 encoder. For each input x , after computing the relevance scores for all the candidate tokens in V , we only select the tokens with relevance scores higher than γ , which is regarded as a hyper-parameter and discussed in section 6.1.

3.4 Entity State Change Coherence Reward

An important issue with the generated state changes from baseline models such as GPT-2 [20] is that the *entity*, *attribute*, *before_state* and *after_state* are not coherent enough or aligned with human commonsense. For example, in the generated output “*composition of flowers were in dead flowers before and fresh flowers afterwards*”, the generated states, e.g., *dead/fresh flowers*, are not relevant to the attribute *composition*. To address this problem, we further design a classifier-based automatic metric to evaluate the coherence of each entity state change and use it as a reward to improve the generation process. To train the classifier, we use all the human-annotated entity state changes in the training set of OpenPI [30] as the positive instances S^p , and create the same number of negative instances S^n by randomly replacing the *entity*, *attribute*, *before_state*, *after_state* of each $s \in S^p$ with a concept sampled from other positive instances but the same slot. Then, each entity state change s is fed as input to a T5 model which outputs a score “1” if the entity state change is coherent, otherwise, “0”. We fine-tune the T5-based classifier with the same number of positive and negative training instances. The classifier is optimized with the cross-entropy objective:

$$p(l|s) = \text{softmax}(\text{T5_Encoder}(s, \theta))$$

where l is 0 or 1 and $p(l|s)$ is the probability indicating the coherence or incoherence of the entity state change s . θ denotes the set of parameters of the classifier.

After training the classifier, we use it to estimate a coherence score for each generated entity state change and compute a reward based on the coherence score to further optimize the knowledge-grained encoder-decoder framework. The reward is computed by:

$$R_{cls} = \log(p_s)(p(l_1|y) - p(l_0|y))$$

where y is the generated target sequence sampled from the model’s distribution at each time step in decoding, p_s is the output probability of the sampled tokens. l_0 and l_1 denote the label 0 and 1,

respectively. We then further optimize the DKGED framework with the reward using policy learning. The policy gradient is calculated by:

$$\nabla_{\phi} J(\phi) = E[R_{cls} \cdot \nabla_{\phi} \log(P(y|x; \phi))]$$

where ϕ represents the model parameters. x is the action description.

The overall learning objective for our proposed KIEST is the combination of the RL and cross-entropy (CE) loss:

$$L_{CE} = - \sum_{i=1}^k \log(s_i | s_{1:i-1}, x, \overline{C_x^e}, \overline{C_x^a})$$

$$L_{RL} = - \frac{1}{t} \sum_{j=1}^t R_{cls}^j$$

$$L_{overall} = (1 - \lambda) \cdot L_{CE} + \lambda \cdot L_{RL}$$

where $\lambda \in [0, 1]$ is a tunable parameter.

4 EXPERIMENTAL SETUP

4.1 Dataset

We evaluate our approach on OpenPI [30], which, to the best of our knowledge, is the only public benchmark dataset for open domain entity state tracking. It comprises 23,880, 1,814, and 4,249 pairs of action description and entity state change in the training, development and test sets, respectively. We also design the following steps to correct the annotation errors of OpenPI:

- We first correct all the spelling errors contained in OpenPI, such as *liquor* (“liquor”), *skier* (“skier”), *necklace* (“necklace”), *apperance* (“appearance”), *compostion* (“composition”) and so on.
- some annotated entity state changes are not following the template “[*attribute*] of [*entity*] was [*before_state*] and [*after_state*] afterwards”, such as “*flexibility of was hard before and soft afterwards*”, “*location of vegetable*”. We thus remove all these entity state changes from OpenPI.

4.2 Baselines and Evaluation Metrics

We compare the performance of KIEST with several strong baselines based on the state-of-the-art pre-trained generative models, including **T5 base/large** [21] and **GPT-2 base/large** [20]. Note that the only previous work [30] on open domain entity state tracking utilized GPT-2-base [20] to generate the state changes. We also design two variants of KIEST by removing the classifier-based coherence reward (denoted as KIEST w/o ESC), and removing both the classifier-based coherence reward and the constrained decoding strategy (denoted as KIEST w/o ESC+CD). Same as [30], we evaluate all the models based on generative evaluation metrics, including Exact Match [22], BLEU-2 [19], and ROUGE-L [4].

4.3 Implementation Details

In DKGED, the embedding of each node in $\overline{C_x^a}$ or $\overline{C_x^e}$ is initialized by the pre-trained model BERT² with 1024-dim. During training, we use AdamW [14] as the optimizer and Cross Entropy as the

²<https://huggingface.co/bert-base-uncased>

Table 1: Results of various approaches for open domain entity state tracking on OpenPI.

Approach	Exact Match			BLEU-2			ROUGE-L		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
GPT2-base [30]	8.82	7.00	4.13	22.85	19.08	16.38	39.54	35.18	32.66
GPT2-large [20]	10.82	7.69	5.24	25.35	20.59	18.16	41.88	35.92	34.08
T5-base [21]	20.10	5.44	3.37	34.38	15.89	14.72	49.86	29.95	29.45
T5-large [21]	15.22	7.46	5.64	31.30	20.33	19.29	47.83	35.97	35.52
KIEST w/o attribute	19.83	7.44	6.66	37.25	21.79	21.68	52.73	36.15	36.78
KIEST w/o entity	19.41	7.30	6.56	36.95	21.69	21.77	52.74	36.70	37.36
KIEST w/o selection	13.61	9.41	6.26	30.49	27.16	22.87	47.19	42.88	39.29
KIEST w/o ESC+CD	19.77	7.67	6.92	37.66	22.05	22.17	53.12	36.63	37.37
KIEST w/o ESC	21.52	7.72	7.17	39.73	22.53	22.92	54.77	37.10	38.02
KIEST (Our Approach)	20.91	9.54	7.84	38.14	26.29	24.24	53.43	40.77	39.40

loss function with a learning rate of 0.00005. We use label smoothing [17] to prevent the model from being over-confident. The batch size is 6. For the overall loss function in section.3.4, we tune the λ value in 0, 0.1, 0.3, 0.5, 0.7, 0.9. We find that KIEST works well when the weight of L_{RL} is 0.1. For the constrained decoding strategy, we tune the threshold γ in $\{0, 0.2, 0.4, 0.6, 0.8\}$, and when $\gamma = 0.4$, our approach achieves the best performance.

In our entity selection model, we use AdamW Optimizer [14] with the learning rate of 0.00002. We also use gradient clipping [37] to constrain the max L_2 -norm of the gradients to be 1. We tune the threshold ϵ in $\{0.3, 0.5\}$ ³, and hops H in $\{1, 2\}$. We find that our model works well when $\epsilon = 0.5, H = 2$. By evaluation, we use the same configuration as above in our attribute selection model. By using selection models, the maximum number of filtered entities and attributes in all action descriptions are 1000 and 140, so when training our model, the maximum number of filtered entities and attributes are set to 1000 and 140 in each action description.

To train the classifier for estimating the coherence of each entity state change, we set the dropout rate to 0.1 in the last linear layer, batch size to 32, and use AdamW optimizer [14] with the learning rate of 0.00002. We also apply gradient clipping [37] to constrain the maximum value of L_2 -norm of the gradients to be 1. The accuracy of predicting *coherence* or *incoherence* for the development and test sets of OpenPI is both 98%, demonstrating that the performance of the classifier has been high enough to be used as a reward function.

5 RESULTS AND DISCUSSION

Table 1 presents the experimental results of various approaches based on the metrics, including Exact Match, BLEU-2 and ROUGE-L. We have the following observations: (1) our KIEST significantly outperforms all the strong baselines and its variants across all evaluation metrics; (2) T5-large model shows obvious superiority compared with GPT-2-base/large model. (3) by adding the classifier-based coherence reward, our KIEST approach significantly improves over the baseline KIEST w/o ESC, especially on recall of all evaluation metrics, demonstrating the effectiveness of the classifier-based

³We found there are 60%, 52%, 40% entities in C_x^e belongs to the training entity set when $\epsilon = 0.3, 0.5, 0.7$ respectively. To obtain abundant knowledge, we discuss the model performance when $\epsilon = 0.3, 0.5$.

coherence reward especially in encouraging the model to generate more valid entity state changes. The overall precision is dropped after adding the classifier-based coherence reward. We guess the reason is that KIEST tends to generate more entity state changes than KIEST w/o ESC. (4) the constrained decoding can help remove most of the noisy tokens from the target vocabulary, thus the precision of the approach is significantly improved, which can be seen by comparing KIEST w/o ESC+CD with KIEST w/o ESC.

Table 2: Reduction of storage and run time on OpenPI dataset. S-Space refers to Storage Space, R-Time refers to Running Time (hours: minutes: seconds), w/o selection refers to without selection model, w/ selection refers to with selection model.

	Training		Inference	
	R-Time	S-Space	R-Time	S-Space
w/ selection	04:23:11	3,793 MB	00:29:46	38,425 MB
w/o selection	71:30:21	4,845 MB	00:38:09	40,112 MB

Ablation studies based on different types of external knowledge and selection strategy are shown in the middle of Table 1. We have the following observations: (1) By comparing the performance of only injecting the entity knowledge C_x^e (w/o attribute) or attribute knowledge C_x^a (w/o entity)⁴ to our model, our model still get a significant improvement over the best generation model T5-large, demonstrating that by incorporating the relevant entity or attribute knowledge, the generation of entity state changes can be significantly improved, which validates our motivation that the relevant entity and attribute knowledge from the knowledge graph can help improve the coverage of the generation of entity state changes. (2) Notably, without the entity and attribute selection (KIEST w/o selection), the precision of KIEST significantly drops on all evaluation metrics, which validates our assumption that without selection, the massive concepts retrieved from ConceptNet are likely to introduce noise and hurt the model’s performance.

⁴When injecting the entity/attribute knowledge, the generation of other positions only uses the knowledge X.

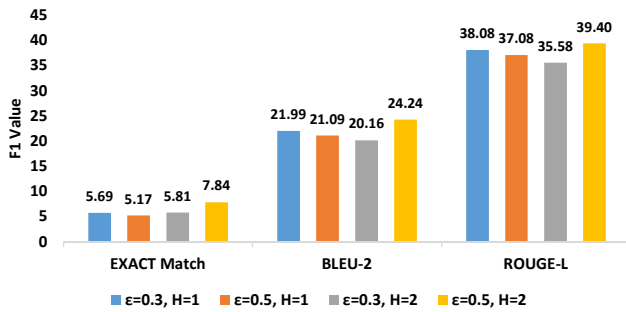


Figure 4: F1 results with different ϵ and H settings.

To evaluate the effectiveness of our proposed selection method in reducing storage and running time, we compared the storage space and run time between KIEST with and without selection on the OpenPI dataset. As presented in Table 2, our model without selection required 4,845 MB of storage whereas KIEST with selection only used 3,793 MB, resulting in a 23.6% reduction in storage costs. This reduction is primarily due to the selection approach which reduced noise from the original entity and attribute sets. Furthermore, the selection method further significantly reduced the running time cost for both training and inference.

6 ANALYSIS

6.1 The Impact of Hyper-Parameters

In our proposed model, there are two parameters controlling the size of the relevant entities and attributes retrieved from the ConceptNet: (1) $\epsilon \in \{0.3, 0.5\}$ which is the threshold to select relevant entities and attributes based on their distance to the input action description, and (2) $H \in \{1, 2\}$ which is the number of hops to retrieve the concepts that are related to the anchors of the action description from ConceptNet. We analyze the impact of these two hyper-parameters in Figure 4. As we can see, when $\epsilon = 0.5$, KIEST selects entities and attributes that are more relevant to the input action description and consistently provides better performance than the setting of $\epsilon = 0.3$. In addition, $H = 1$ is also better than $H = 2$, indicating that when we set $H = 2$, too many concepts are retrieved from ConceptNet which leads to too much noise and hurts the performance of both the selection models and KIEST.

We also analyze the impact of $\gamma \in \{0, 0.2, 0.4, 0.6, 0.8\}$, which is the threshold to select tokens for the target output vocabulary. As Figure 5 shows, KIEST achieves the best performance when γ is 0.2 or 0.4. When $\gamma < 0.2$, too many candidate tokens are considered at each decoding step and thus leading to irrelevant output. When $\gamma > 0.4$, too few candidate tokens are included in the target vocabulary and thus also leading to negative impacts on the generation of entity state changes.

6.2 Content Richness and Coherence of Outputs

We measure the content richness of KIEST based on the generated entity state changes and compare it with the two strong baseline models, GPT-2-Large and T5-Large. As shown in Figure 6, KIEST tends to generate longer outputs, e.g., more entity state changes with sequence length in the range of 16-31 than the baseline models.

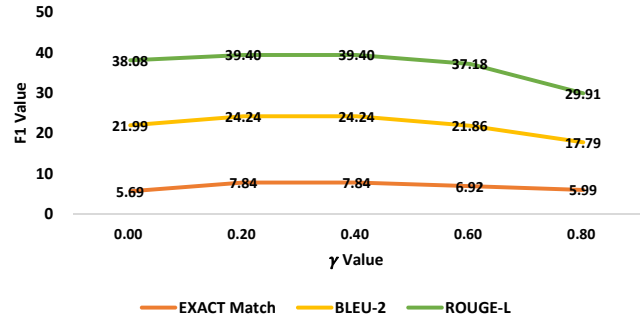


Figure 5: F1 results with different γ settings.

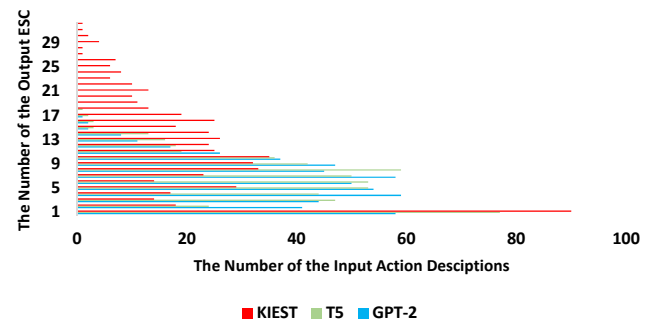


Figure 6: Number of entity state changes for each action description generated by different approaches.

The results imply that our knowledge-informed framework encourages the model to generate more entity state changes with higher coverage.

To evaluate the coherence of the entity state changes generated by KIEST, its variants and the baseline models, we use the trained classifier in Section 3.4 as an automatic metric. The output of the classifier is normalized into $[0, 1]$ with the sigmoid function. As shown in Figure 7, KIEST achieves the highest coherence compared with the GPT-2 and T5, suggesting that our knowledge-informed model with the constraint decoding and coherent reward method is effective to improve the coherence of entity state change. For further analysis, KIEST w/o ESC+CD has a lower score compared to GPT-2, suggesting that simply incorporating the additional entity and attribute knowledge into the language model but without any control on the decoding process may have a negative impact on the coherence of the output. Without using the classifier-based coherence reward (i.e., KIEST w/o ESC), the average coherence score of the outputs is lower than the coherence score of KIEST, demonstrating the effectiveness of the coherence reward.

6.3 Is ChatGPT a Good Open Domain Entity State Tracker?

Recently, ChatGPT from OpenAI has shown significant advances in various downstream NLP tasks. Here, we systematically compare the state-of-the-art GPT models, including GPT-3.5-turbo⁵, GPT-4⁶,

⁵<https://platform.openai.com/docs/models/gpt-3-5>

⁶<https://platform.openai.com/docs/models/gpt-4>

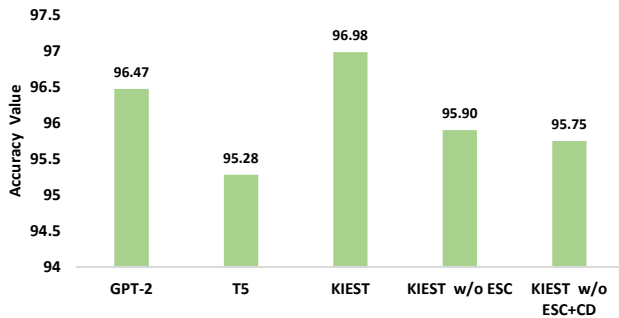


Figure 7: Automatic coherence evaluation of different models.

with KIEST on open domain entity state tracking task, to answer the potential research question: "Is ChatGPT a good open domain entity state tracker?"

We first design prompts to instruct the GPT models to predict the entity state changes for each input action description. As shown in Figure 8, we design two prompts: (1) Prompt 1 is based on a task description defined manually; (2) prompt 2 contains a task definition from [30] and 8 demonstration examples that are randomly selected from the OpenPI training dataset. Table 3 presents the zero-shot performance of GPT-3.5-turbo and GPT-4 on the whole testing set of OpenPI. Our analysis reveals that different prompts yield significantly different results for both GPT-3.5-turbo and GPT-4. Although GPT-4 outperforms GPT-3.5-turbo in terms of Exact Match and BLEU-2 scores, both models fail to achieve satisfactory results on this task compared to all the supervised models including KIEST. These findings suggest that, without any tuning, the large language models still have limited capability to perform complex reasoning tasks.

Table 3: F1 performance of GPT-3.5-turbo and GPT-4 on OpenPI. P1 refers to Prompt1, and P2 refers to Prompt2.

Model	Exact Match	BLEU-2	ROUGE-L
GPT-3.5-turbo (P1)	0.34	6.74	21.25
GPT-3.5-turbo (P2)	0.38	8.28	24.06
GPT-4 (P1)	1.25	11.02	27.28
GPT-4 (P2)	0.56	10.34	26.78
KIEST	7.84	24.24	39.40

We further manually analyze the prediction errors of GPT-3.5-turbo and GPT-4 on the test set of OpenPI and summarize them into the following categories (examples for each category are shown in Table 4):

- **Wrong Template Prediction:** Despite being provided with output templates and examples, GPT-3.5-turbo and GPT-4 still generated a considerable number of output errors that are not based on the provided template. Furthermore, our analysis indicates that GPT-3.5-turbo is more prone to generating erroneous templates than GPT-4.

Table 4: The examples of prediction error about GPT-3.5-turbo (P1/P2) and GPT-4 (P1/P2).

Model	Wrong Template Prediction:	
GPT-3.5-turbo (P1)	Action Input:	Identify the state where the car is licensed. Now, what happens?
	Generated ESC:	ent is car, attr is licensed state, valpre is unknown and valpost is the state where the car was licensed
GPT-3.5-turbo (P2)	Action Input:	Identify the state where the car is licensed. Subscribe to an online information search website. Now, what happens?
	Generated ESC:	access to information was limited before and expanded afterwards.
GPT-4 (P1)	Action Input:	Identify the state where the car is licensed. Subscribe to an online information search website. Now, what happens?
	Generated ESC:	information search website subscription status was inactive before and active afterwards
Unable to Extract Entities or Attributes:		
GPT-3.5-turbo (P1)	Action Input:	Grind roots, seeds and other hard herbs in a coffee grinder or mortar and pestle. Mix the ground or dried herb with a bit of oil or water until it forms a moist paste. Now, what happens?
	Generated ESC:	attr of None was None before and None afterwards
GPT-3.5-turbo (P2)	Action Input:	It needs to be SMART: specific, measurable, attainable, realistic and timed. Identify the necessary steps. Achieving your goals can often take a long time and can be very frustrating. Now, what happens?
	Generated ESC:	None, as no entities with attributes are presented in the given sentence.
GPT-4 (P2)	Action Input:	Gather the supplies you will need: Now, what happens?
	Generated ESC:	There will be no change.
Generating Irrelevant Entity State Change:		
GPT-3.5-turbo (P1)	Action Input:	Do something that smells great, like bake bread. Now, what happens?
	Generated ESC:	state of hunger of people around was not hungry before and potentially hungry afterwards
GPT-4 (P1)	Action Input:	Do something that smells great, like bake bread. Now, what happens?
	Generated ESC:	smell of flowers were not present before and present afterwards
GPT-4 (P2)	Action Input:	Have a spa experience. Read a good book. Now, what happens?
	Generated ESC:	knowledge of plot book was unknown before and in parent's mind afterwards

- **Unable to Extract Entities or Attributes:** Our analysis indicates that GPT-3.5-turbo and GPT-4 struggle to comprehend the semantic information of input action descriptions and accurately identify the corresponding entities, attributes, and states.
- **Generating Irrelevant Entity State Changes:** During our analysis, we observed that GPT-3.5-turbo and GPT-4 occasionally generate irrelevant Entity State Changes. As an

System Prompt 1: You are a smart and intelligent entity state tracking (EST) system. I will provide you the definition of the element you need to extract (the entity, the attribute, the before state, and after state about the entity of this attribute), the sentence from where your extract, the elements, and the output format with examples

System Prompt 2: You are a smart and intelligent entity state tracking (EST) system. Input: As input we are given a procedural text comprising current step (step_i) as query and all past step as context We denote the input as $x=(xq,xc)$, where xq is the step for which we need the state changes (i.e. the query) and xc is the context. Output: The output is a set of zero or more state changes $y= \{y_i\}$. A state change y_i is of the form: $attr$ of ent was $valpre$ before and $valpost$ afterwards. Here, $attr$ is the attribute or state change type, and ent is the changed entity. $valpre$ is the precondition (i.e., the state value before), and $valpost$ is the postcondition (i.e., the state value afterwards). Pre/postcondition adj or $relp(yprei)$ can be an adjectival phrase or a relational phrase. In this task, $attr$, ent , $valpre$ and $valpost$ are open form text i.e. they are not tied to any fixed, constrained vocabulary." ok now your task is to give the input and output the output definition, Consider the running example: $x=(context: The window of your car is foggy, query: Rub half potato on the window)$. Then, $\{y\} = \{ transparency of window was fogged before and partially clear afterwards, stickiness of window was smooth before and sticky afterwards \}$. In y_1 , $attr = transparency$, $ent = window$, $valpre = fogged$ and $valpost = partially clear$. in the next part I will give you the example about the input and output"

User Prompt: Are you clear about your role?

Assistant Prompt: Sure, I'm ready to help you with your EST task. Please provide me with the necessary information to get started.

Guidelines Prompt:
Entity Definition:
 "1. ent : it is the entity, Initiator of the action"
 "2. $attr$: it is the attribute, the attribute about this entity"
 "3. $valpre$: it is the before state, the before state about this attribute"
 "4. $valpost$: it is the after state, the after state about this attribute"
Output Format:
 " $attr$ of ent was $valpre$ before and $valpost$ afterwards"
 "If no entities are presented in any categories keep it None, The output is a set of zero or more state changes"
Examples:
 1: "**Sentence:** Apply insecticide to peonies. Now what happens"
 "**Output:** the location of insecticide was in bottle before and on peonies afterwards, the health of bugs were healthy before and dying afterwards"
 2: "**Sentence:** Dip the peony flowers in water. Now what happens"
 "**Output:** the moisture of flowers was dry before and wet afterwards..."

Figure 8: Example of Prompt 1 and Prompt 2 defined for GPT-based models.

illustration in Tabel 4, these examples demonstrate that GPT-3.5-turbo and GPT-4 are capable of generating semantically plausible entity state changes, but in some cases, they may not be aligned with the information presented in the input action description. Such as for the generations of model GPT-3.5-turbo (P1) and GPT-4 (P1), the entities “*hunger*” and “*flowers*” in the generated output are not relevant to the context described in the input.

7 LIMITATION

In this work, we explored various state-of-the-art large language models for the open-domain entity state-tracking task. Though such models are capable of generating relevant and fluent outputs given the input action description, they still cannot generate all the accurate entity state changes with high coverage. In our proposed K_{IE}ST framework, we attempted to locate the relevant entities and attributes from ConceptNet. Such knowledge is still noisy and the coverage is not satisfying. Going forward, we plan to develop more appropriate prompts to stimulate and instruct the language models such as GPT-3.5 or GPT-4 to generate more coherent entity state changes. In addition, this work only considers textual descriptions of actions while their corresponding image or video illustrations are also beneficial for predicting the relevant entities and state changes. In the future, we will extend entity state tracking to multimodality

by leveraging the state-of-the-art multimodal pre-trained language models [33] and instruction tuning [35].

8 CONCLUSION

This paper introduces K_{IE}ST, a knowledge-informed framework for open domain entity state tracking. It consists of two major steps: first, retrieving and selecting the relevant entity and attribute knowledge from an external knowledge base, i.e., ConceptNet; and then dynamically incorporating the entity and attribute knowledge into an encoder-decoder framework with an effective knowledge decoding strategy and a classifier-based entity state change coherence reward. Experimental results and extensive analysis on the public benchmark dataset – OpenPI demonstrate the effectiveness of our overall framework and each component with significant improvements over the strong baselines, including T5, GPT-2, GPT-3.5, and GPT-4.

ACKNOWLEDGEMENTS

This research is based upon work supported by U.S. DARPA KMASS Program # HR001121S0034. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

REFERENCES

- [1] Khalid Al Khatib, Lukas Trautner, Henning Wachsmuth, Yufang Hou, and Benno Stein. 2021. Employing argumentation knowledge graphs for neural argument generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 4744–4754.
- [2] Berkeley R Andrus, Yeganeh Nasiri, Shilong Cui, Benjamin Cullen, and Nancy Fulda. 2022. Enhanced story comprehension for large language models through dynamic document-based knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 10436–10444.
- [3] Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2017. Simulating action dynamics with neural process networks. *arXiv preprint arXiv:1711.05313* (2017).
- [4] Arman Cohan and Nazli Goharian. 2016. Revisiting summarization evaluation for scientific articles. *arXiv preprint arXiv:1604.00400* (2016).
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [6] Hossein Rajaby Faghihi and Parisa Kordjamshidi. 2021. Time-stamped language model: Teaching language models to understand the flow of events. *arXiv preprint arXiv:2104.07635* (2021).
- [7] Zhe Hu, Hou Pong Chan, Jiachen Liu, Xinyan Xiao, Hua Wu, and Lifu Huang. 2022. PLANET: Dynamic Content Planning in Autoregressive Transformers for Long-form Text Generation. *arXiv preprint arXiv:2203.09100* (2022).
- [8] Mingxuan Ju, Wenhao Yu, Tong Zhao, Chuxu Zhang, and Yanfang Ye. 2022. Grape: Knowledge Graph Enhanced Passage Reader for Open-domain Question Answering. *arXiv preprint arXiv:2210.02933* (2022).
- [9] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
- [10] Mingchen Li, Junfan Chen, Samuel Mensah, Nikolaos Aletras, Xiulong Yang, and Yang Ye. 2022. A Hierarchical N-Gram Framework for Zero-Shot Link Prediction. *arXiv preprint arXiv:2204.10293* (2022).
- [11] Mingchen Li and Shihao Ji. 2022. Semantic Structure based Query Graph Prediction for Question Answering over Knowledge Graph. *International Conference on Computational Linguistics (COLING)* (2022).
- [12] Hugo Liu and Push Singh. 2004. ConceptNet—a practical commonsense reasoning tool-kit. *BT technology journal* 22, 4 (2004), 211–226.
- [13] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2901–2908.
- [14] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
- [15] Kaixin Ma, Filip Ilievski, Jonathan Francis, Eric Nyberg, and Alessandro Oltramari. 2022. Coalescing Global and Local Information for Procedural Text Understanding. *arXiv preprint arXiv:2208.12848* (2022).
- [16] Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. *arXiv preprint arXiv:1805.06975* (2018).
- [17] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems* 32 (2019).
- [18] Sheshera Mysore, Zach Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. *arXiv preprint arXiv:1905.06939* (2019).
- [19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.
- [20] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.
- [22] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*.
- [23] Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018. Modeling naive psychology of characters in simple commonsense stories. *arXiv preprint arXiv:1805.06533* (2018).
- [24] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.
- [25] Janvijay Singh, Fan Bai, and Zhen Wang. 2022. Frustratingly Simple Entity Tracking with Effective Use of Multi-Task Learning Models. *arXiv preprint arXiv:2210.06444* (2022).
- [26] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.
- [27] Evangelia Spiliopoulou, Artidoro Pagnoni, Yonatan Bisk, and Eduard Hovy. 2022. EvEntS ReLM: Event Reasoning of Entity States via Language Models. *arXiv preprint arXiv:2211.05392* (2022).
- [28] Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. *arXiv preprint arXiv:2010.00309* (2020).
- [29] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).
- [30] Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi Mishra, Dheeraj Rajagopal, Peter Clark, Michal Guerin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. *arXiv preprint arXiv:2011.08092* (2020).
- [31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [33] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*. PMLR, 23318–23340.
- [34] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics* 9 (2021), 176–194.
- [35] Zhiyang Xu, Ying Shen, and Lifu Huang. 2022. MultiInstruct: Improving Multi-Modal Zero-Shot Learning via Instruction Tuning. *arXiv preprint arXiv:2212.10773* (2022).
- [36] Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. FIGLeT: Language grounding through neuro-symbolic interaction in a 3D world. *arXiv preprint arXiv:2106.00188* (2021).
- [37] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881* (2019).
- [38] Ranran Haoran Zhang, Qianying Liu, Aysa Xuemo Fan, Heng Ji, Daojian Zeng, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Minimize exposure bias of Seq2Seq models in joint entity and relation extraction. *arXiv preprint arXiv:2009.07503* (2020).