OpenFact: Factuality Enhanced Open Knowledge Extraction

Linfeng Song*, Ante Wang*,†, Xiaoman Pan, Hongming Zhang, Dian Yu, Lifeng Jin, Haitao Mi, Jinsong Su†, Yue Zhang‡ and Dong Yu
Tencent AI Lab, Bellevue, WA, USA
†School of Informatics, Xiamen University, China
‡School of Engineering, Westlake University, China
lfsong@global.tencent.com

Abstract

We focus on the *factuality* property during the extraction of an OpenIE corpus named *OpenFact*, which contains more than 12 million high-quality knowledge triplets. We break down the *factuality* property into two important aspects—*expressiveness* and *groundedness*—and we propose a comprehensive framework to handle both aspects. To enhance expressiveness, we formulate each knowledge piece in OpenFact based on a semantic frame. We also design templates, extra constraints, and adopt human efforts so that most OpenFact triplets contain enough details. For groundedness, we require the main arguments of each triplet to contain linked Wikidata[1] entities. A human evaluation suggests that the OpenFact triplets are much more accurate and contain denser information compared to OPIEC-Linked (Gashteovski et al., 2019), one recent high-quality OpenIE corpus grounded to Wikidata. Further experiments on knowledge base completion and knowledge base question answering show the effectiveness of OpenFact over OPIEC-Linked as supplementary knowledge to Wikidata as the major KG.

1 Introduction

Open information extraction (OIE, Etzioni et al., 2008) aims to extract factual and informative knowledge, which can further be used to enhance major knowledge bases (Martinez-Rodriguez et al., 2018) or on downstream tasks (Stanovsky et al., 2015). Most current OIE systems (Fader et al., 2011; Mausam et al., 2012; Del Corro and Gemulla, 2013; Angeli et al., 2015) organize knowledge into subject-relation-object (SRO) triples, and they use templates to extract such knowledge triples. Figure 1a and 1b show a Wikipedia[2] fragment and the corresponding SRO triple from OPIEC-Linked (Gashteovski et al., 2019), a major OIE corpus.

Previous OIE efforts (Fader et al., 2011; Mausam et al., 2012; Del Corro and Gemulla, 2013; Angeli et al., 2015; Stanovsky et al., 2018; Cui et al., 2018) mainly focus on extracting SRO triples that are closer to human annotated outputs on benchmarks. However, they overlook the *factuality* issue, where many extracted triples may convey incomplete or even incorrect information. This issue can be categorized into two main aspects. The first aspect is the lack of *expressiveness*, where complex knowledge is poorly organized or critical information is dropped due to the fixed SRO schema and the template-based extraction system. Taking Figure 1b as an example, the relation (‘‘*surpassed Washington Monument to become*’’) is a long span containing multiple events, and important temporal information (‘‘*during its construction*’’) is not captured. The second aspect regards *groundedness* that measures the degree of a triplet being linked to known entities. For instance, a standard OIE system can extract ‘‘<*the tower, has, three levels for visitors*>’’ from input sentence ‘‘*The tower has three levels for visitors.*’’ The extracted triple lacks groundedness particularly because it is unclear what ‘‘*the tower*’’ refers to. Lacking factuality diminishes the usability of the OIE triples as supplementary knowledge to major KGs.

Later work proposes to use either nested SRO triples (Bhutani et al., 2016) or n-ary triples (Christensen et al., 2011; Akbik and Löser, 2012; Gashteovski et al., 2017; Zhao et al., 2021) to enrich SRO schema with properties such as polarity. A recent effort (Gashteovski et al., 2019)

*Equal contribution, work done during an internship of Ante Wang at Tencent AI Lab.

[1]https://www.wikidata.org/.

[2]https://www.wikipedia.org/.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 686–702, 2023. https://doi.org/10.1162/tacl_a_00569
Action Editor: Doug Downey. Submission batch: 9/2022; Revision batch: 12/2022; Published 6/2023.
© 2023 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

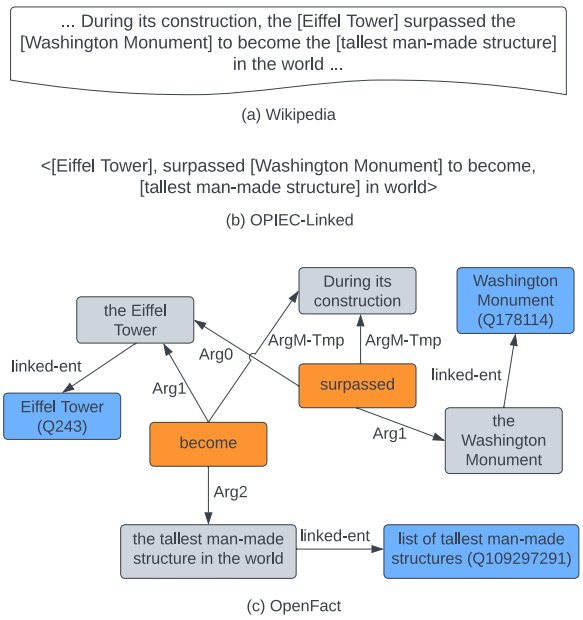

Figure 1: (a) A Wikipedia fragment and (b) the corresponding OPIEC-Linked (Gashteovski et al., 2019) triples. Throughout this paper, we use ‘‘[]’’ to indicate the mentions of Wikidata entities. (c) Our OpenFact fragment, where orange, gray, and blue boxes represent relations, arguments, and Wikidata entities, respectively.

further extracts more substantial properties (e.g., space and time), and a refined[3] OIE corpus (OPIEC-Linked) of 5.8 millions triples is extracts from Wikipedia where both the subject and the object contains linked Wikidata entities. Although this effort can partially improve expressiveness and groundedness, the factuality issue is not discussed and formally addressed. Besides, simply keeping extra properties or linked entities without a comprehensive solution may not ensure that most triples contain precise factual information. For instance, though both subject and object are grounded for the triple in Figure 1b, it erroneously mixes two events (i.e., ‘‘*surpass*’’ and ‘‘*become*’’) and lacks critical information (e.g., time) to describe precise factual information.

In this paper, we propose a comprehensive solution to address the factuality issue, which is then used to construct a corpus of OIE knowledge named *OpenFact*. The corpus contains more than 12 million accurate factual knowledge triplets extracted from Wikipedia, a reliable knowledge source with broad coverage. Comparing with previous OIE corpora, factuality is thoroughly enhanced in OpenFact. Specifically, as shown in Figure 1c, OpenFact is based on semantic role labeling (SRL, Palmer et al., 2005) for expressiveness.[4] Here each OpenFact triplet contains a predicate, which may mention a relation or an event, and several core arguments from the corresponding semantic frame. To further ensure that each OpenFact knowledge triplet describes detailed factual information, we require it to contain a certain combination of semantic arguments (e.g., (*Arg0*, *Arg1*, *ArgM-Tmp*)), and we manually design multiple such combinations to ensure coverage. We also resort to some affordable human efforts to manually verify some extracted knowledge triplets, which are then used to train a quality checker to automatically filter low-quality triplets from the remaining data.

To improve groundedness, we extend the core arguments (*Arg0*, *Arg1*, *Arg2*, *ArgM-Loc*) with associated Wikidata entities to ensure that the knowledge triplet is not about some ambiguous entity (e.g., ‘‘the tower’’). As the result, our OpenFact triplets (e.g., ‘‘*surpassed(Arg0:the [Eiffel Tower], Arg1:the [Washington Monument], ArgM-Tmp:During its construction)*’’) can provide more expressive and grounded information than previous OIE triples (e.g., the example in Figure 1b).

A human evaluation on 200 randomly sampled triplets show that 86.5% of OpenFact triplets convey precise factual information, while it is only 54.0% for OPIEC-Linked. Further experiments on knowledge base completion (KBC) and knowledge base question answering (KBQA) benchmarks (Safavi and Koutra, 2020; Yih et al., 2016) over Wikidata show that OpenFact provides more useful complementary information than OPIEC-Linked and significantly improves highly competitive baseline systems.[5]

In summary, we make the following contributions:

- We propose a comprehensive approach to address the factuality issue of open information extraction from two key aspects: *expressiveness* and *groundedness*.

[3]OPIEC-Linked is obtained by applying filtering rules on the initial OPIEC corpus to enhance quality.

[4]Though this shares a similar spirit to SRLIE (Christensen et al., 2011), SRLIE converts the extracted semantic frames back into traditional SRO or n-ary triples and loses some critical information.

[5]Code and data are available at `https://github.com/Soistesimmer/OpenFact`.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

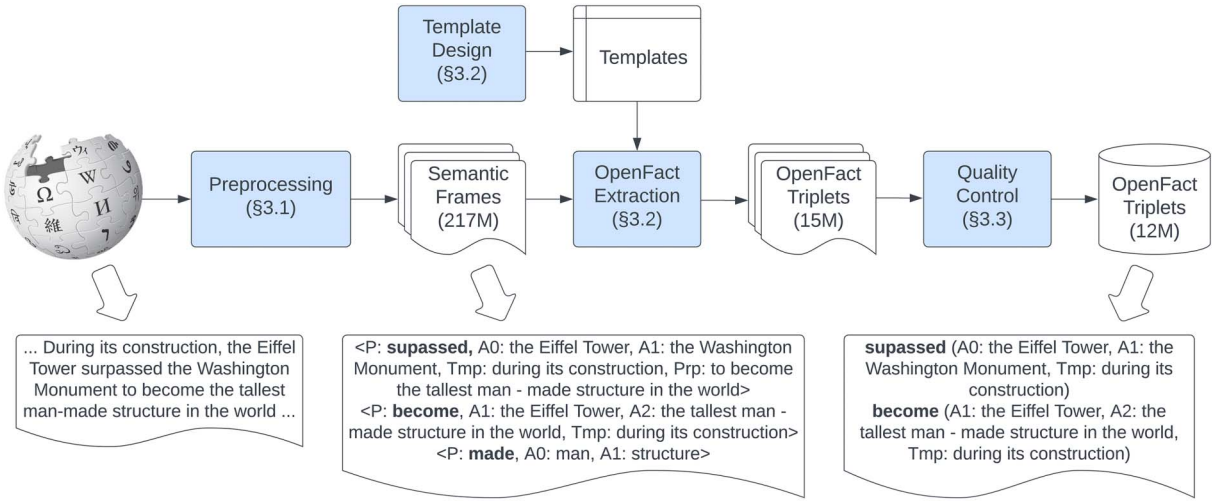

Figure 2: The pipeline for OpenFact construction with the corresponding outputs for the example in Figure 1a. Numbers (e.g., ‘‘12M’’) indicates the number of extracted items. ‘‘P’’, ‘‘A0’’, ‘‘A1’’, ‘‘A2’’, and ‘‘Tmp’’ are short for *predicate*, *Arg0*, *Arg1*, *Arg2*, and *ArgM-Tmp*, respectively.

- Using our approach, we construct *OpenFact*, an OIE corpus containing 12 million accurate factual knowledge triplets.
- Human evaluation and experiments on both KBQA and KBC show superiority of OpenFact over OPIEC, a state-of-the-art OIE corpus.

2 OpenFact Construction

We choose the Wikipedia dump of May 20, 2021, to construct OpenFact, as Wikipedia has been considered to be a reliable knowledge source with broad coverage. Figure 2 shows the overall pipeline, which includes Wiki dump preprocessing (§2.1), triplet extraction (§2.2), and quality control (§2.3). We also conduct an analysis on all the extracted OpenFact triplets (§2.4).

2.1 Preprocessing

We conduct three preprocessing steps (*Wikipedia dump cleaning*, *text processing*, and *SRL parsing*) to obtain the input data for triplet extraction.

Wikipedia Dump Cleaning In a Wikipedia dump, each page contains not only the main content but also the special tokens and keywords from a markup language. Therefore, the first step is to clean the special markup tokens and reformat each page into plain text. For example, the markup {{convert|5464|km|mi||sp=us}} should be reformatted into the plain text *‘‘5,464 kilometer (3,395 mile)’’*. To this end, we adopt a popular tool (Pan et al., 2017) to remove all undesired markups (e.g., text formatting notations, HTML tags, citations, external links, templates) and only keep the internal links that naturally indicate the mentions of Wikidata entities.

Text Processing After obtaining the Wikipedia dump in plain texts, we perform sentence segmentation, tokenization, and entity mention detection using SpaCy.[6] Since there are plenty of missing entity links in the original Wikipedia pages (Adafre and de Rijke, 2005), we adopt a dense retrieval based entity linker (Wu et al., 2020) to link some extracted entity mentions to Wikipedia. Note that we only keep the linking results when the model probability of the top candidate exceeds 0.5 to ensure quality. Finally, we map all Wikipedia titles to Wikidata IDs.

SRL Parsing The last step in preprocessing is to obtain the semantic frames from SRL parsing. Specifically, we adopt the SRL parser from AllenNLP,[7] an implementation of Shi and Lin (2019), to parse for all the sentences extracted from Wikipedia by preprocessing. Since standard SRL parsing omits the situations where the main predicate can be a verbal phrase (e.g., ‘‘come up with’’), we further extend those predicates into phrasal predicates. To this end, we collect 390 frequent verbal phrases for matching.

[6] https://spacy.io/.

[7] https://demo.allennlp.org/semantic-role-labeling.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

<linking_verb:present, Arg1, Arg2> *is(Arg1=The [Eiffel Tower], Arg2=a [wrought-iron] [lattice tower] on the [Champ de Mars] in [Paris], [France])*
<reg_verb:past_active, Arg0, Arg1, ArgM-Tmp> *supassed(Arg1=the [Eiffel Tower], Arg2=the [Washington Monument], ArgM-Tmp=during its construction)*
<reg_verb:past_passive, Arg1, ArgM-Tmp> *constructed(Arg1=The [Eiffel Tower], ArgM-Tmp=from 1887 to 1889)*

Table 1: Example templates and a representative OpenFact triplet. The requirements for the main predicate is in ''*verb:tense_voice*'' format with the *verb* being either a linking verb or a regular verb.

2.2 Triplet Extraction

In this stage, we design multiple templates to obtain candidate OpenFact triplets from the extracted semantic frames in the previous stage. Table 1 lists example templates and the corresponding OpenFact triplets. Each template contains the requirements on the main predicate and a subset of six important semantic roles: *Arg0* (agent), *Arg1* (patient), *Arg2* (beneficiary), *ArgM-Tmp* (time), *ArgM-Loc* (location), and *ArgM-Neg* (negation). Particularly, we make requirements on predicates regarding the verb type (linking or regular), tense (present or past), and voice (active or passive) in order to ensure the quality of the automatically extracted triplets. Besides, we further impose the following restrictions when designing and applying the templates:[8]

- Each template should take at least two arguments, and their combination needs to capture meaningful factual information. We manually design multiple combinations that satisfy this requirement. For instance, (*Arg1*, *ArgM-Tmp*, *ArgM-Loc*) can be used to extract ''*born(Arg1=Barack Obama, ArgM-Tmp=1961, ArgM-Loc=Honolulu, Hawaii)*''. On the other hand, (*ArgM-Tmp*, *ArgM-Loc*) is not a meaningful combination, as it can barely match a sound example.
- If a sentence matches multiple templates, only the one with the highest number of arguments is used for triplet extraction. For each pair of templates, we guarantee that the union of their arguments is also a valid template, so that there is no ambiguity on choosing templates.
- All extracted semantic arguments except for *ArgM-Tmp* and *ArgM-Neg* need to contain at least one linked Wikidata entity. This ensures that the corresponding triplet can be grounded to some well-known entity, increasing the chance of the triplet describing some factual information.
- Each triplet with a past-tense (or past-participle) predicate needs to contain *ArgM-Tmp* to alleviate time-related ambiguity. For instance, ''*surpassed(Arg0=the Eiffel Tower, Arg1=the Washington Monument)*'' becomes ambiguous without the time property.

With these restriction rules, we extract around 15 million candidate triplets from 217 million semantic frames. Though SpaCy-based text processing and SRL parsing can introduce errors, the imposed restrictions help limit further propagation of these errors. Besides, we introduce a neural quality controller (§2.3) to prune erroneous triplets.

2.3 Quality Control

Despite the imposed restrictions during extraction, some obtained triplets can still contain too much noise. Our observation reveals that lacking enough details is the major reason. For example, ''*invented(Arg0=A German company, Arg1=the first practical jet aircraft, ArgM-Tmp=in 1939)*'' is not precise, because its *Arg0* (''*A German company*'') misses essential details. To further improve our data, we resort to human annotators to examine the quality for some triplets, before training a model to automatically examine the quality of the remaining triplets. In particular, we ask annotators to label whether a triplet contain quality issues, and what semantic roles cause the issue.[9] We hire professional annotators and set additional requirements on English proficiency and a bachelor's degree.

[8] We release all templates for triplet extraction in the Appendix.

[9] Detailed guidelines will be provided in the Appendix given more space.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

Quality Checker With the annotated data, we train a quality checker to automatically examine all remaining triplets. Particularly, an input triplet t is first serialized into a token sequence with special BERT tokens [CLS] and [SEP]:

$$t_{serial} = [\text{[CLS]}, t_p, \text{[SEP]}, t_{a_1}, \ldots t_{a_N}], \quad (1)$$

where t_p and t_{a_i} represent the tokens of the main predicate and an argument inside t, respectively, and N indicates the number of arguments in t. The arguments are linearized in the order of their appearance in the original sentence. Next, the quality checker adopts a BERT-base (Devlin et al., 2019) encoder with multiple binary classification heads for predicting the quality for the overall triplet and each semantic role, respectively.

$$\boldsymbol{H} = \text{BERT}(t_{serial}), \quad (2)$$

$$p(q_x|t) = \text{sigmoid}(\boldsymbol{W}_x \cdot \boldsymbol{h}_{\text{[CLS]}}), \quad (3)$$

where $x \in X = [\texttt{whole}, \texttt{a0}, \texttt{a1}, \texttt{a2}, \texttt{loc}, \texttt{tmp}]$, which represents all aspects of each triplet t for quality evaluation, $\boldsymbol{W}_x$ and $p(q_x|t)$ are the linear classifier and the quality score of the corresponding item, respectively, and $\boldsymbol{h}_{\text{[CLS]}}$ indicates the BERT hidden state for the [CLS] token.

Training We adopt binary cross-entropy loss to train the checker:

$$\mathcal{L} = -\sum_{x \in X} \Big(\hat{q}_x \log p(q_x|t) + (1 - \hat{q}_x)(1 - \log p(q_x|t)) \Big), \quad (4)$$

where $\hat{q}_x \in [0, 1]$ is the annotated quality for aspect x. AdamW (Loshchilov and Hutter, 2018) is used as the optimizer with learning rate and number of epochs set to 10^{-5} and 10, respectively.

Inference During inference, we only use the linear classifier for the `whole` aspect (Eq. 3 with $x = \texttt{whole}$) to determine triplet quality, while the quality classification results on the other aspects (`a0`, `a1`, `a2`, `loc`, and `tmp`) only serve as auxiliary losses during training. Following standard practice, we keep each triplet if the score of the `whole` classifier exceeds 0.5.

2.4 Data Statistics

We obtain more than 12 million OpenFact triplets after all the steps in Figure 2, where each triplet takes 18.1 words on average (μ) with standard deviation (σ) being 10.7 words. Here we conduct more detailed analyses in the following aspects:

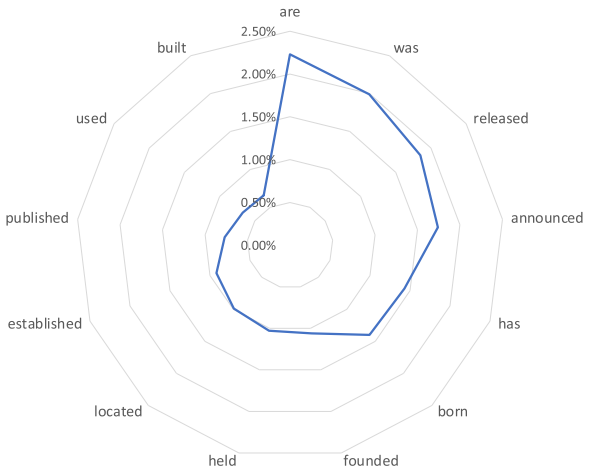

Figure 3: Percentages for the most frequent predicates except for ‘‘*is*’’.

Predicates We find 14,641 distinct predicates among all the extracted OpenFact triplets. Figure 3 shows the percentages of the most frequent predicates. We exclude the most frequent predicate ‘‘*is*’’, which counts for 36% of all triplets for better visualization. Predicate ‘‘*is*’’ usually involves in definitions in Wikipedia, such as ‘‘The Eiffel Tower is a wrought-iron lattice tower on the Champ de Mars in Paris, France’’. Our extraction procedure favors those definition-based triplets, as they usually give enough details. The remaining top frequent predicates form a flat distribution with the percentage ranging from 2.25% to 0.75%. Many of them (e.g., ‘‘*released*’’, ‘‘*announced*’’, ‘‘*held*’’, and ‘‘*established*’’) do not have obvious corresponding relations in Wikidata.

Entities Figure 4 shows the distribution of OpenFact triplets over the number of contained Wikidata entities, where more than 66% of the triplets contain more than three entities. This indicates that most of our triplets capture more complex relations than standard Wikidata triples, which only capture the relation between two entities. On the other hand, 0.39% of triplets contain only 1 entity, as they are obtained from the templates with only one main argument. Typical examples are ‘‘*constructed(Arg1=The [Eiffel Tower], ArgM-Tmp=from 1887 to 1889*’’.

Arguments Figure 5 shows the distribution of OpenFact triplets over the number of arguments.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

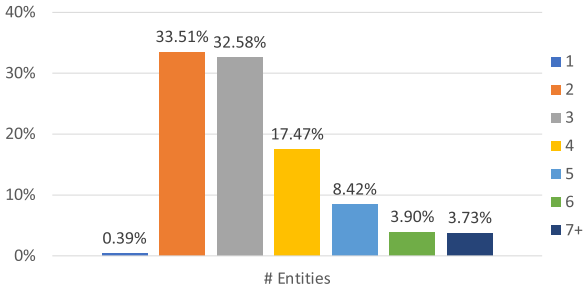

Figure 4: Distribution of OpenFact triplets over the number of entities.

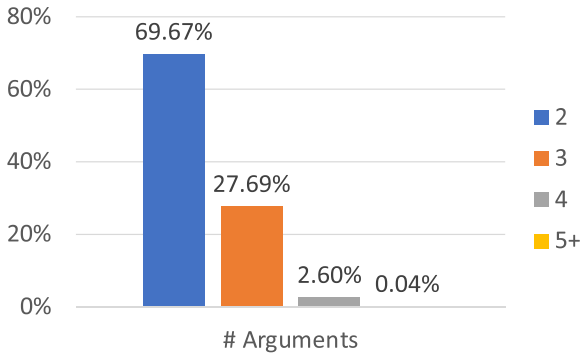

Figure 5: Distribution of OpenFact triplets over the number of arguments.

Most triplets ($> 97.3\%$) contain either 2 or 3 arguments, indicating that most of the triplets consist a moderate amount of information. The most frequent argument combination are (*Arg1, Arg2*), (*Arg0, Arg1, ArgM-Tmp*), (*Arg1, ArgM-Tmp*), and (*Arg0, Arg1*) that count for 42.2%, 27.7%, 17.1%, and 13.2%, respectively. The (*Arg1, Arg2*) combination accounts for the highest percent as it covers all the triplets with a linking verb (e.g., "*is*").

Figure 6 further visualizes the average lengths for all the argument types, where most types take 4.5 to 13 words except for *ArgM-Neg*. The reason is that it only needs one word to represent negation in most cases, while it can requires a lot of words for other components.

3 OpenFact for Downstream Tasks

We choose knowledge base completion (KBC) and knowledge base question answering (KBQA) to evaluate the usefulness of OpenFact.

3.1 Knowledge Base Completion

Automatic KBC aims to infer missing facts based on existing information in a knowledge graph. For example, given a knowledge triple $(h, r, ?)$ with a head h and relation r, the goal is to infer

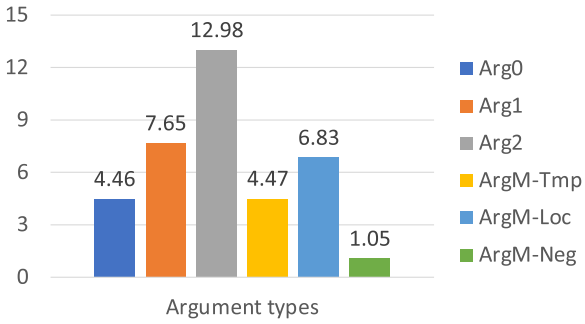

Figure 6: The argument average lengths.

the tail entity, which is missing from the input KG. This task is important due to the fact that knowledge graphs are usually highly incomplete (West et al., 2014).

Traditional efforts on KBC (Nickel et al., 2011; Bordes et al., 2013; Trouillon et al., 2016; Dettmers et al., 2018; Balažević et al., 2019) focus on learning the knowledge graph embedding (KGE). On the other hand, adopting pretrained language models (PLMs) to this task by linearizing each candidate KG triple into a sequence has recently gained popularity (Yao et al., 2019; Saxena et al., 2022).

Baseline: Combine KGE with PLM as a Pipeline We first build our baseline system by combining a major KGE system (ComplEx; Trouillon et al., 2016) with a PLM model (KG-BERT; Yao et al., 2019) into a pipeline to leverage the advantages of both approaches. Inside the pipeline, the KGE system first produces a list of k-best candidates, which are then ranked by the PLM model to pick the final output. We choose ComplEx and KG-BERT, which both show highly competitive performances in several major benchmarks.

Formally, given an input query $(h, r, ?)$, we first adopt ComplEx to infer the k-best tail entities (e.g., t_i) over all the entities in a knowledge graph with defined scoring function:

$$\phi(r, h, t_i) = \texttt{Re}(\langle \boldsymbol{w}_r, \boldsymbol{e}_h, \boldsymbol{e}_{t_i} \rangle), \quad (5)$$

where $\boldsymbol{e}_h$, $\boldsymbol{e}_{t_i}$ are the embeddings for h, t_i, respectively, and $\boldsymbol{w}_r$ is the learned representation vector for relation r. $\texttt{Re}(\boldsymbol{x})$ denotes the real vector component for vector $\boldsymbol{x}$, and $\langle \cdot, \cdot, \cdot \rangle$ represents the Hermitian product.[10]

[10]For details are shown in §3 of Trouillon et al. (2016).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

As the next step, given each hypothesis triple (h, r, t_i) with th i-th candidate tail entity t_i, we use the KG-BERT model to determine the likelihood of this triple being valid, before picking the hypothesis triple with the highest likelihood score as the final output. In particular, each triple (h, r, t_i) is first linearized into a sequence of tokens:

$$\mathrm{X} = \big[\texttt{[CLS]}\, h\, \texttt{[SEP]}\, r\, \texttt{[SEP]}\, t\, \texttt{[SEP]}\big], \quad (6)$$

where `[CLS]` and `[SEP]` are BERT (Devlin et al., 2019) special tokens. Then a BERT encoder with a linear layer (`MLP`) is adopted to predict the likelihood $p(t_i|h, r)$ of t_i being the correct tail:

$$\begin{aligned} p(t_i|h, r) &= \texttt{sigmoid}\big(\texttt{MLP}(\mathbf{H}_{\texttt{[CLS]}})\big), \\ \mathbf{H} &= \texttt{BERT}(\mathrm{X}), \end{aligned} \quad (7)$$

where $\mathbf{H}_{\texttt{[CLS]}}$ is the output hidden state of `[CLS]` from BERT.

ComplEx and KG-BERT are trained separately, where we completely follow the previous work (Safavi and Koutra, 2020) to prepare the ComplEx system. As high-quality negative samples are essential to train a robust KG-BERT (Lv et al., 2022) model, we collect the top-k link prediction results from the pretrained ComplEx system as the negative triples instead of randomly selection from the overall KG as early practices. KG-BERT is trained with binary cross-entropy loss to determine the correctness of each input triple:

$$\begin{aligned} \mathcal{L}_{\text{KG-BERT}} = &-y_i \log(p(t_i|h, r)) - \\ &(1 - y_i) \log(1 - p(t_i|h, r)), \end{aligned} \quad (8)$$

where y_i ($y_i \in \{0, 1\}$) is the label on whether t_i is the valid tail for query $(h, r, ?)$.

Finally, we calculate the ranking score for a candidate triple considering both the scores from both ComplEx and KG-BERT:

$$score_{\texttt{Rank}} = score_{\texttt{ComplEx}} + \omega \cdot score_{\texttt{KG-BERT}}, \quad (9)$$

where ω is a scaling hyperparameter.

OpenFact for KBC We do not change the model structure but propose to incorporate one relevant OpenFact triplet to enhance the corresponding candidate KG triple, so that the KG-BERT model in the pipeline system can leverage richer information to make more accurate ranking decisions.

Particularly, given a KG triple (h, r, t), we first gather the OpenFact triplets (e.g., $P(A_h, A_t)$) where h and t are contained by its arguments (i.e., $h \in A_h$ and $t \in A_r$) correspondingly. For simplicity, we only keep the most similar OpenFact triplet if multiple ones are obtained, where a similarity score is calculated by comparing a linearized[11] OpenFact triplet with the linearized target KG triple using SentenceBERT (Reimers and Gurevych, 2019). We then concatenate them as a sequence of tokens:

$$\begin{aligned} \mathrm{X'} = \texttt{[CLS]}\, h\, \texttt{[SEP]}\, r\, \texttt{[SEP]}\, t \\ \texttt{[SEP]}\, h\, A_h\, P\, A_t\, t\, \texttt{[SEP]}. \end{aligned} \quad (10)$$

This is similar with Eq. 6, and the resulting token sequence is then taken to the KG-BERT model to pick the final tail entity as in Eq. 7.

3.2 Knowledge Base Question Answering

Knowledge Base Question Answering (KBQA) aims to answer factoid questions based on a knowledge base. It is an important task and such systems have been integrated in popular web search engines and conversational assistants. The main challenge of this task is to properly map each query in natural language to KG entities and relations. Since knowledge graphs are often sparse with many missing links, this poses additional challenges, especially increasing the need for multi-hop reasoning (Saxena et al., 2020). We further evaluate OpenFact on this task, as OpenFact triplets are in a closer format to natural language than KG triples. Besides, the dense knowledge provide by OpenFact may help bridge many multi-hop situations.

Baseline: UniK-QA (Oguz et al., 2022) It has been recently proposed to solve multiple QA tasks as one unified process. The system consists of a pretrained Dense Passage Retriever (DPR; Karpukhin et al., 2020) and a Fusion-in-Decoder (FiD; Izacard and Grave, 2021) model. We adopt UniK-QA as our baseline due to its strong performances over multiple KBQA benchmarks.

Given an input question q and the associated KG triples obtained by entity linking and 2-hop expansion on the target KG, UniK-QA first uses

[11] Each OpenFact triple is empirically linearized in the order of "*Arg0 Neg Prd Arg1 Arg2 Loc Tmp*."

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

System	CoDEx-S			CoDEx-M		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
ComplEx†	0.465	0.372	0.646	0.337	0.262	0.476
KG-BERT	0.263	0.183	0.429	0.209	0.155	0.307
Pipeline	0.522	0.420	0.724	0.392	0.300	0.567
+OPIEC	0.526	0.426	0.726	0.394	0.305	0.568
+OpenFact	**0.535**	**0.430**	**0.736**	**0.403**	**0.313**	**0.578**

Table 2: Main results on knowledge base completion. † denotes results from previous work.

the DPR module to rank the KG triples. Next, the sorted KG triples are linearized and combined into passages of at most 100 tokens, after which top-n passages $[p_1, \ldots, p_n]$ are fed into the FiD model for generating the final answer autoregressively:

$$\begin{aligned} a_j &= \texttt{FiD-Dec}([\mathbf{H_1}, \ldots, \mathbf{H_n}], a_{<j}) \\ \mathbf{H_i} &= \texttt{FiD-Enc}([q, p_i]), \end{aligned} \tag{11}$$

where a_j is the j-th token of the final answer.

We fix the pretrained DPR and train the FiD model with standard cross-entropy loss:

$$\mathcal{L}_{\text{FiD}} = -\sum_{j=1}^{|a|} \log p(a_j | q, [p_1, \ldots, p_n]), \tag{12}$$

where the conditional probability is calculated by the FiD model. $|a|$ indicates the size of answer a.

OpenFact for KBQA Similar to KBC tasks, we do not change the baseline model structure and only incorporate OpenFact as extra knowledge. Since the UniK-QA baseline (Oguz et al., 2022) consumes all related KG triples as passages in a FiD manner (Izacard and Grave, 2021), we propose to incorporated the relevant OpenFact triples as additional passages for UniK-QA. Particularly, given an entity e linked to the input question q and an expanded entity $\hat{e}$ within two hops of KG expansion, we obtain OpenFact triplets (e.g., $P(A_e, A_{\hat{e}})$) where e and $\hat{e}$ are contained by their arguments (i.e., $e \in A_e$ and $\hat{e} \in A_{\hat{e}}$). We then follow UniK-QA to linearize each OpenFact triple as a sequence of tokens before ranking and combining them into multiple passages $[p'_1, \ldots, p'_m]$. Finally, both $[p_1, \ldots, p_n]$ and $[p'_1, \ldots, p'_m]$ are consumed by the FiD module to predict the final answers as Eq. 11.

4 Experiments

We evaluate OpenFact regarding both the quality and its usefulness on two major downstream tasks.

4.1 Human Evaluation on Knowledge Quality

Binary Evaluation As mentioned in §2.3, we ask humans to annotate 70,967 OpenFact triplets regarding whether (*yes* or *no*) each triplet and its arguments are accurate. Taking 60K annotated triplets to train the quality checker, we leave the remaining 10,967 for held-out testing. The percent of qualified triplets in the whole test set is already 70.4% as determined by human annotators. This indicates that the imposed restrictions and templates during automatic triplet extraction (§2.2) are effective for achieving a descent quality level.

On the same hold-out test set, our quality checker (§2.3) reports an F1 score of 85.1 for detecting erroneous triplets, where the precision and recall are 83.6 and 86.7, respectively. The checker filters 30.7% test instances using the inference method in §2.3. As the result, it yields an accuracy of 94.3% for the remaining triplets. This indicates that the combination of our restrictions, templates and automatic quality checking is effective for generating high-quality knowledge.

Fine-grained Evaluation We further conduct a more fine-grained human evaluation to directly compare OpenFact with OPIEC-Linked by randomly sampling 200 knowledge triplets from each knowledge source. Particularly, for each triplet, we want to evaluate to what level it conveys precise factual information. The levels are ''*Good*'', meaning that the triplet is accurate; ''*Mixed*'', meaning that some triplet arguments are mixed (e.g., the time information is contained in *Arg1*)

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

OIE KN	Good	Mixed	Missing	Wrong
OPIEC	0.540	0.000	0.210	0.250
OpenFact	0.865	0.015	0.120	0.000

Table 3: Distribution over various quality levels.

but enough detailed information is present; ''*Missing*'', meaning that some essential information (e.g., time or location) is missing; and ''*Wrong*'', meaning that it conveys wrong information or lacks critical information to be understandable.

Here for each knowledge triplet, we ask 3 annotators to judge its quality, and we use majority voting to make the final judgement. For the rare cases where there are 3 distinct decisions (actually we find none), we ask the annotators to discuss and converge their ratings. To avoid potential bias, we hire new annotators instead of these who annotated the OpenFact triplets for automatic quality checking (§2.3). The annotators are encouraged to search the internet to verify triplet correctness whenever necessary. The inner-annotator agreement is *0.787*, indicating *substantial agreement*.

Table 3 shows the results, where a significantly higher percent of OpenFact triplets are rated as ''*Good*'' than OPIEC-Linked. Besides, no OpenFact triplets are rated as ''*Wrong*'', while the number is 25.0% for OPIEC-Linked. We find that most OPIEC-Linked triplets (including those rated as ''*Good*'' such as ''<*Caramboxin, is, toxin*>'') are quite short, not consisting of much useful knowledge. Many ''*Wrong*'' triplets contain obvious errors, such as ''<*QUANT_S_1 Turks, is, QUANT_O_1 Italians*>'' and ''<*Lexus, is, Acura*>''. All results suggest that OpenFact has significantly higher quality and is much denser than OPIEC-Linked.

We further analyze the erroneous triplets of OpenFact and show representative cases in Table 4, where the top and bottom groups correspond to the ''Mixed'' and ''Missing'' categories, respectively. For the ''Mixed'' category, we list all the 3 instances rated as this type out of the 200 instances for human evaluation. From these instances and other OpenFact data, we find that most defects in this category are caused by SRL parsing errors, like erroneous argument-type detection (e.g., the first example where the Arg1 and Arg2 arguments are mistakenly swapped) and erroneous argument-boundary detection (e.g., the second and third examples where the time information is mistakenly mixed into other arguments).

Mixed	linked(Arg1=with [Australia] and [New Zealand], Arg2=Every important island or group in [Oceania])
	was(Arg1=[Cuba], Arg2=able to exchange one ton of sugar for 4.5 tons of [Soviet] oil, ArgM-Tmp=in 1987)
	helped(Arg0=the [Wiesbaden] lab's rapid reports, Arg1=to predict the [Israeli] - [British] - [French] attack on [Egypt] three days before it began on 29 October, Arg2=the [U.S.] government, ArgM-tmp=as Detachment B took over from A and flew over targets that remain classified)
Missing	born(Arg1=a [Tajik], [Russian], and [Soviet] composer, ArgM-Loc=in the city of [Dushanbe], [Tajik SSR])
	bear(Arg0=Her [passports], Arg1=the stamps of two dozen countries, including [Algeria] . . . and the [Union of Soviet Socialist Republics])
	reaches(Arg0=the [African] dust, Arg1 = the [United States])
	turned(Arg1=[Soros], Arg2=into a pariah in [Hungary] and a number of other countries)

Table 4: Error analysis on OpenFact.

For the ''Missing'' category, we observe two prevalent types of defects, and we list two typical examples for each type. As shown in the first two examples of this category, one major type of defect is caused by the absent of critical entity information in the arguments. For instance, the missing name of ''the Soviet composer'' in the first example makes the whole instance too vague to be factual. The second example suffers from a similar situation as it fails to mention who ''the owner of the passports'' is. Although we force each argument to contain at least one entity during the extraction process, it cannot guarantee that all essential entities are included. The other major type of defects are due to the missing of critical time or location information, as it is nontrivial to determine whether a OpenFact triplet requires time or location to be factual. For the last two cases of this category, both the time when ''the African dust reaches the United States'' and the time when ''Soros is turned into a pariah in Hungary ...'' are missing.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

OIE KN	CoDEx-S			CoDEx-M		
	2-hop	3-hop	4-hop	2-hop	3-hop	4-hop
OPIEC	0.629	0.077	0.012	0.344	0.055	0.011
OpenFact	0.843	0.351	0.248	0.560	0.069	0.062

Table 5: The percent of devset KG triples where head and tail can be connected by one OIE triplet.

4.2 Evaluation on KBC

Datasets We evaluate on CoDEx-S and CoDEx-M (Safavi and Koutra, 2020) that are extracted from Wikidata with CoDEx-M being *much more challenging* than CoDEx-S. The entity number and the distinct relation-type number for CoDEx-S/CoDEx-M are 2,034/17,050 and 42/51, respectively. The number of train/dev /test instances for CoDEx-S and CoDEx-M are 32,888/1,827/1,828 and 185,584/10,310/10,311, respectively.

Systems In addition to *ComplEx*, *KG-BERT*, and *Pipeline* (§3.1), we also compare our model (+*OpenFact*) with (+*OPIEC*): They both add extra knowledge (OpenFact vs OPIEC-Linked) to the *Pipeline* system. All three pipeline systems share the same model architecture except for the adopted knowledge (None vs OPIEC vs OpenFact), which helps validate the effectiveness of OpenFact.

Settings For fair comparison, the KG-BERT for both the baselines and our model are trained using Adam (Kingma and Ba, 2014) for 4 epochs with linear scheduler and initial learning rate set to 5×10^{-5}. Their batch sizes on CoDEx-S and CoDEx-M are 256 and 1024, respectively. We set k as 25 to get a considerable amount of negative samples. Besides, to balance the positive and negative sample numbers, we penalize negative samples with their instance weights as 0.2.

We evaluate in the form of link prediction, where a system is required to predict the missing entities from inputs that are in the format of $(?, r, t)$ or $(h, r, ?)$. The performance is evaluated with mean reciprocal rank (MRR) and Hits@k. We set ω as 2.5/5/5 on CoDEx-S and 2.5/5/7.5 on CoDEx-M for Pipeline/Pipeline+OPIEC/ Pipeline+OpenFact according to development experiments.

Main Results Table 2 shows the main test results on CoDEx-S and CoDEx-M for link prediction, where we also compare with the reported numbers of *ComplEx*. We can draw the following conclusions. **First**, the *Pipeline* model shows significantly better performances than using just a single module (*ComplEx* and *KG-BERT*), as it can enjoy the benefits from both knowledge graph embedding and a large-scale PLM. **Secondly**, enhancing the KG-BERT module of *Pipeline* with the knowledge from either OPIEC-Linked or OpenFact can further improve the performance. Comparatively, using OpenFact gives larger performance gains over OPIEC-Linked with the gaps on CoDEx-S and CoDEx-M being 0.9 MRR points (0.4 Hits@1 points) and 0.9 MRR points (0.8 Hits@1 points), respectively. This indicates that OpenFact is a better supplement than OPIEC-Linked for major KGs like Wikidata.

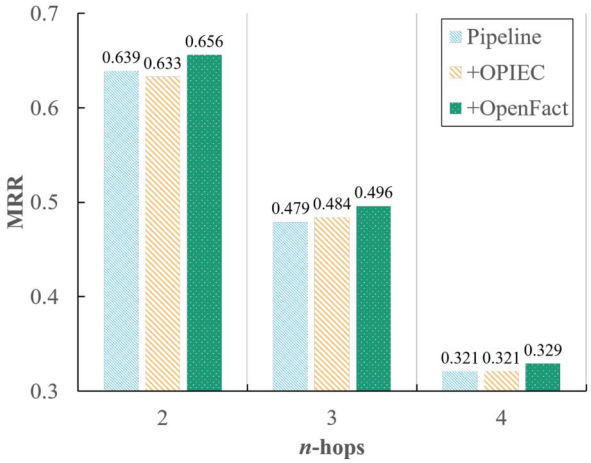

Figure 7: The MRR scores on the CoDEx-m test set across different hops between head and tail entities.

Analysis We further analyze a few aspects to pinpoint where the gains of OpenFact come from. As listed in Table 5, we first compare OpenFact with OPIEC-Linked regarding the coverage rates on the devset KG triples across different number of hops. OpenFact consistently covers more percents of KG triples than OPIEC-Linked across all numbers of hops. For both OPIEC-Linked and OpenFact, the coverage rate drops when increasing the number of hops, while the advantages of OpenFact become more significant. For instance, OpenFact covers 20.6 (24.8% vs 1.2%) and 5.6 (6.2% vs 1.1%) times more than OPIEC-Linked on the 4-hop instances of CoDEx-S and CoDEx-M, respectively.

As shown in Figure 7, we also compare the MRR score of OpenFact with OPIEC-Linked

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

regarding the number of hops from the head entity to the tail entity on the known part of the input knowledge graph. In particular, we choose CoDEx-M (the more challenging dataset) and categorize its test instances into 3 groups according to the number of hops. We observe consistent performance gains from +*OPIEC* to +*OpenFact* for all groups. The percent of *relative gains* along the number of hops are 3.6% (0.656/0.639), 2.4% (0.496/0.484), and 2.4% (0.329/0.321), indicating that OpenFact can be helpful for a large variety of situations instead of for certain cases.

4.3 Evaluation on KBQA

Dataset We evaluate on the WebQSP dataset (Yih et al., 2016), which consists of 3,098/1,639 training/testing question-answer pairs. Since WebQSP uses Freebase (Bollacker et al., 2008) as the knowledge source, we map each Freebase entity id to the corresponding Wikidata id if there is any,[12] and we find the corresponding Wikidata entities for nearly half of the Freebase entities.

Comparing Systems One system for comparison is the *UniK-QA* (Oguz et al., 2022) baseline, which only accesses the associated KG triples of each input question. In addition, we also compare our model (+*OpenFact*) using extra OpenFact knowledge with another baseline (+*OPIEC*) using OPIEC-Linked (Gashteovski et al., 2019) knowledge. For fair comparison, all three systems take the same number of parameters.

Settings Following previous work (Oguz et al., 2022), we split 10% training data as the development set and take *Hits@1* as the evaluation metric. Following the setup of vanilla UniK-QA, we use the pretrained DPR checkpoint from Karpukhin et al. (2020) without further finetuning, and we adopt the standard T5-base (Raffel et al., 2020) checkpoint to initialize the FiD model, which is then trained using the Adam optimizer (Kingma and Ba, 2014) with batch size, learning rate, and training steps set to 64, 10^{-4}, and 1,000, respectively.

Main Results Figure 8 visualizes the Hits@1 scores of *UniK-QA* with only associated KG triples, +*OPIEC* with extra OPIEC-Linked knowledge, and +*OpenFact* with extra OpenFact triplets. For both +*OPIEC* and +*OpenFact*, we report the scores with growing amounts of passages (knowledge), as we usually obtain multiple pieces of relevant triplets. Both +*OPIEC* and +*OpenFact* improve the final performance, while +*OpenFact* gives significantly more gains with 10 or more extra passages. Particularly with at most 20 extra passages, +*OpenFact* outperforms *UniK-QA* and +*OPIEC* by 1.4 and 1.2 Hits@1 points. The reason can be that OpenFact provides richer high-quality knowledge compared to OPIEC-Linked. According to our statistics, the average number of relevant passages by OPIEC-Linked and OpenFact are 4.7 and 12.8, respectively.

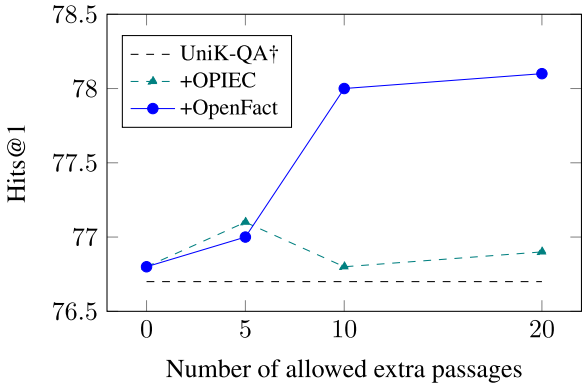

Figure 8: Main UniK-QA performance with the number of allowed extra passages from either OPIEC-Linked or OpenFact. † denotes the score reported in Oguz et al. (2022).

5 Related Work

Though open information extraction has received increasing attention, there are only a few publicly available large-scale Open-IE databases, such as TextRunner (Yates et al., 2007), ReVerb (Fader et al., 2011), PATTY (Nakashole et al., 2012), WiseNet 2.0 (Moro and Navigli, 2013), DefIE (Bovi et al., 2015), and OPIEC (Gashteovski et al., 2019). On the other hand, NELL (Carlson et al., 2010; Mitchell et al., 2018) uses a predefined seed ontology (including 123 categories and 55 relations) instead of extracting from scratch. All these Open-IE corpora organize data into traditional SRO or n-ary triples. Some incorporate either automatic entity linking results (Lin et al., 2012; Nakashole et al., 2012; Bovi et al., 2015) or the natural links retained from the source data (Moro and Navigli, 2013; Gashteovski et al., 2019) simply for alleviating the ambiguity of extracted triples.

[12] We build the mapping from the ''Freebase ID'' property (P646) of each Wikidata entity.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

On the other hand, we focus on extracting triplets that contain enough expressiveness and groundedness to retain factuality. To do that, we adopt a comprehensive framework that imposes carefully designed restrictions and templates to filter unqualified triplets. We also resort to human annotations in descent-scale to train a quality checker for further data filtering. Previous studies either rely on system confidence scores and lexical features for heuristics-based filtering (Kolluru et al., 2020) or clean data to enable self-training (Nayak et al., 2021). In addition, our triplets adopt the format of semantic frames that contains rich semantic roles. Thus, our triplets can be more expressive than standard SRO or n-ary triples.

6 Conclusion

We have introduced OpenFact, an OIE corpus of 12 million accurate factual knowledge triplets extracted from Wikipedia. Different from existing OIE work, we focused on the factuality of extracted triplets with a comprehensive approach to enhance such property. Specifically, OpenFact is based on frame semantics instead of the standard SRO or n-ary schema, and we design automatic restriction rules and multiple dedicated templates to enhance expressiveness. Besides, we extended the core arguments of each OpenFact triplet with linked Wikidata-entity mentions for groundedness. We also resorted to a quality checker trained with descent-scale human annotations to further improve our knowledge quality. A human study revealed that 86.5% of OpenFact triplets are precise, while it is only 54.0% for OPIEC-Linked (Gashteovski et al., 2019), a recent high-quality OIE corpus closest to ours in spirit. Further experiments on knowledge base completion and knowledge base question answering show that OpenFact provides more useful knowledge than OPIEC-Linked, significantly improving very strong baselines.

Future work includes designing dynamic templates where the combination of arguments is based on both prior knowledge and the predicate (Zeng et al., 2018).

References

Sisay Fissaha Adafre and Maarten de Rijke. 2005. Discovering missing links in Wikipedia. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 90–97. https://doi.org/10.1145/1134271.1134284

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 52–56.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354. https://doi.org/10.3115/v1/P15-1034

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194. https://doi.org/10.18653/v1/D19-1522

Nikita Bhutani, H. V. Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 55–64. https://doi.org/10.18653/v1/D16-1006

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. https://doi.org/10.1145/1376616.1376746

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, 26.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015. Large-scale information extraction from textual definitions through deep syntactic and semantic analysis. *Transactions of the Association for Computational Linguistics*, 3:529–543. https://doi.org/10.1162/tacl_a_00156

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v24i1.7519

Janara Christensen, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, pages 113–120. https://doi.org/10.1145/1999676.1999697

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 407–413. https://doi.org/10.18653/v1/P18-2065

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 355–366. https://doi.org/10.1145/2488388.2488420

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v32i1.11573

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74. https://doi.org/10.1145/1409360.1409378

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. Minie: Minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/D17-1278

Kiril Gashteovski, Sebastian Wanner, Sven Hertling, Samuel Broscheit, and Rainer Gemulla. 2019. {OPIEC}: An open information extraction corpus. In *Automated Knowledge Base Construction (AKBC)*.

Gautier Izacard and Édouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Mausam, and Soumen Chakrabarti. 2020. IMoJIE: Iterative memory-based joint open information extraction. In *Proceedings of the ACL*, pages 5871–5886. https://doi.org/10.18653/v1/2020.acl-main.521

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 84–88.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? A reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581. https://doi.org/10.18653/v1/2022.findings-acl.282

Jose L. Martinez-Rodriguez, Ivan López-Arévalo, and Ana B. Rios-Alvarado. 2018. OpenIE-based approach for knowledge graph construction from text. *Expert Systems with Applications*, 113:339–355. https://doi.org/10.1016/j.eswa.2018.07.017

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115. https://doi.org/10.1145/3191513

Andrea Moro and Roberto Navigli. 2013. Integrating syntactic and semantic analysis into the open information extraction paradigm. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145.

Tapas Nayak, Navonil Majumder, and Soujanya Poria. 2021. Improving distantly supervised relation extraction with self-ensemble noise filtering. In *Proceedings of the RANLP 2021*, pages 1031–1039. https://doi.org/10.26615/978-954-452-072-4_116

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*. https://doi.org/10.18653/v1/2022.findings-naacl.115

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106. https://doi.org/10.1162/0891201053630264

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. https://doi.org/10.18653/v1/D19-1410

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

Tara Safavi and Danai Koutra. 2020. Codex: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350. https://doi.org/10.18653/v1/2020.emnlp-main.669

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2022.acl-long.201

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507. https://doi.org/10.18653/v1/2020.acl-main.412

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 303–308. https://doi.org/10.3115/v1/P15-2050

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895. https://doi.org/10.18653/v1/N18-1081

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080. PMLR.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 515–526. https://doi.org/10.1145/2566486.2568032

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.519

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael J. Cafarella, Oren Etzioni, and Stephen Soderland. 2007. Textrunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26. https://doi.org/10.3115/1614164.1614177

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Ying Zeng, Yansong Feng, Rong Ma, Zheng Wang, Rui Yan, Chongde Shi, and Dongyan Zhao. 2018. Scale up event extraction learning via automatic training data generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v32i1.12030

Yang Zhao, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2021. Knowledge graphs enhanced neural machine translation. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4039–4045. https://doi.org/10.24963/ijcai.2020/559

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

A The Data Annotation Guideline for Quality Control (§2.3)

Task Definition Given a triplet that contains a predicate, its arguments, and their contained entity mentions, the goal is to detect whether the triplet contains enough details to convey precise factual information. If it does not, the annotator needs to point out *all* arguments that cause the issue. Table 6 lists a few examples to better visualize the goal of this annotation. The top group and bottom group contain positive and negative examples with erroneous parts labeled in red. There are generally two main types of triplets: One type of triplets describe some constant status, such as the *first* case in the upper group; the other type of triplets describe an event, such as the *remaining two* cases in the upper group. Due to the time-sensitive nature of most events, an event-type require usually requires some specific time property to be precise.

Possible argument types are Arg0, Arg1, Arg2, ArgM-Tmp (time), ArgM-Loc (location), and ArgM-Neg (negation). Here Arg0 and Arg1 can be considered as the subject and object for regular verbs, while Arg2 is called beneficiary. Taking ‘‘John gave an apple to Tom’’ for example, the predicate is ‘‘gave’’, ‘‘John’’ is Arg0 (the executor to perform ‘‘give’’), ‘‘an apple’’ is Arg1 (the thing being given), and ‘‘Tom’’ is Arg2 (the beneficiary for this action). Note that the meaning of beneficiary is not literal. For instance, the beneficiary of ‘‘John threw a rock to Tom’’ is also ‘‘Tom’’.

For the *first* example in the bottom group, we should select both ‘‘Arg0’’ and ‘‘ArgM-Tmp’’ as erroneous arguments. ‘‘Arg0’’ is selected because the subject of event ‘‘visited’’ is required to make the whole triplet precise. Since ‘‘Arg0’’ is missing, ‘‘ArgM-Tmp’’ becomes unclear because it is unclear what ‘‘his’’ refers. For the *second* example in the bottom group, we need to choose ‘‘Arg1’’, which should be the place of being ‘‘ruled’’, not the title of the ruler. For the *third* example in the bottom group, we need to choose ‘‘Arg0’’, because ‘‘he’’ is not precise without knowing what this pronoun refers to.

contains(Arg0=[Upper Redwater Lake], Arg1=fish populations of [walleye] , [smallmouth bass] and [northern pike])
risen(Arg1=hazelnut production in Turkey, ArgM-Tmp=since 1964, when a law on a Guarantee of Purchase was introduced, after which a large part of the peasants in the Black Sea region became hazelnut cultivators)
undertaken(Arg1=the Velhas Conquistas, Arg0=by the Portuguese, ArgM-Tmp=during the 16th and 17th centuries)
visited (ArgM-Tmp=During his long service, Arg1=Prince Charles ’ Gloucestershire home, Highgrove)
ruled(Arg0=Kettil Karlsson, Arg1=as Regent of Sweden, ArgM-Tmp=for half a year in 1465 before dying from bubonic plague)
graduated(Arg0=he, Arg1=Bachelor of Laws (LL.B.), ArgM-Tmp=in 1930)

Table 6: Examples for annotation introduction.

B All Templates for Triplet Extraction

As listed in Table 7. Note that we only select the argument combination with the highest number of arguments for each type of predicate (e.g., `linking_verb:present`).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023

Requirements on Predicates	Argument Combination
`linking_verb:present`	<Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg1, Arg2, ArgM-Loc, ArgM-Neg> <Arg1, Arg2, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg1, Arg2, ArgM-Loc> <Arg1, Arg2, ArgM-Tmp> <Arg1, Arg2>
`linking_verb:past`	<Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg1, Arg2, ArgM-Tmp>
`reg_verb:present_active`	<Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Neg> <Arg0, Arg1, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, ArgM-Tmp, ArgM-Neg> <Arg0, Arg1, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg0, Arg1, Arg2, ArgM-Tmp> <Arg0, Arg1, Arg2, ArgM-Loc> <Arg0, Arg1, Arg2> <Arg0, Arg1, ArgM-Tmp, ArgM-Loc> <Arg0, Arg1, ArgM-Tmp> <Arg0, Arg1, ArgM-Loc> <Arg0, Arg1>
`reg_verb:past_active`	<Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg0, Arg1, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, ArgM-Tmp, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg0, Arg1, Arg2, ArgM-Tmp> <Arg0, Arg1, ArgM-Tmp, ArgM-Loc> <Arg0, Arg1, ArgM-Tmp>
`reg_verb` : `present_passive`	<Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg1, Arg2, ArgM-Loc, ArgM-Neg> <Arg1, Arg2, ArgM-Neg> <Arg1, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg1, ArgM-Tmp, ArgM-Neg> <Arg1, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg0, Arg1, Arg2, ArgM-Tmp> <Arg0, Arg1, Arg2, ArgM-Loc> <Arg0, Arg1, Arg2> <Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg1, Arg2, ArgM-Tmp> <Arg1, Arg2, ArgM-Loc> <Arg1, Arg2> <Arg1, ArgM-Tmp, ArgM-Loc> <Arg1, ArgM-Tmp> <Arg1, ArgM-Loc>
`reg_verb:past_passive`	<Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg1, Arg2, ArgM-Tmp, ArgM-Neg> <Arg1, ArgM-Tmp, ArgM-Loc, ArgM-Neg> <Arg1, ArgM-Tmp, ArgM-Neg> <Arg0, Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg0, Arg1, Arg2, ArgM-Tmp> <Arg1, Arg2, ArgM-Tmp, ArgM-Loc> <Arg1, Arg2, ArgM-Tmp> <Arg1, ArgM-Tmp, ArgM-Loc> <Arg1, ArgM-Tmp>

Table 7: All templates used in Triplet Extraction (§2.2).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00569/2141030/tacl_a_00569.pdf by guest on 07 September 2023