Mapping and Cleaning Open Commonsense Knowledge Bases with Generative Translation

Julien Romero[1(✉)] and Simon Razniewski[2]

[1] Téléom SudParis, IPParis, Palaiseau, France
julien.romero@telecom-sudparis.eu
[2] Bosch Center for AI, Renningen, Germany
simon.razniewski@de.bosch.com

Abstract. Structured knowledge bases (KBs) are the backbone of many knowledge-intensive applications, and their automated construction has received considerable attention. In particular, open information extraction (OpenIE) is often used to induce structure from a text. However, although it allows high recall, the extracted knowledge tends to inherit noise from the sources and the OpenIE algorithm. Besides, OpenIE tuples contain an open-ended, non-canonicalized set of relations, making the extracted knowledge's downstream exploitation harder. In this paper, we study the problem of mapping an open KB into the fixed schema of an existing KB, specifically for the case of commonsense knowledge. We propose approaching the problem by *generative translation*, i.e., by training a language model to generate fixed-schema assertions from open ones. Experiments show that this approach occupies a sweet spot between traditional manual, rule-based, or classification-based canonicalization and purely generative KB construction like COMET. Moreover, it produces higher mapping accuracy than the former while avoiding the association-based noise of the latter. Code and data are available. (https://github.com/Aunsiels/GenT, julienromero.fr/data/GenT)

Keywords: Open Knowledge Bases · Generative Language Models · Schema Matching

1 Introduction

Motivation and Problem Open Information Extraction (OpenIE) automatically extracts knowledge from a text. The idea is to find explicit relationships, together with the subject and the object they link. For example, from the sentence "In nature, fish swim freely in the ocean.", OpenIE could extract the triple *(fish, swim in, the ocean)*. Here, the text explicitly mentions the subject, the predicate, and the object. Therefore, if one uses OpenIE to construct a knowledge base (we call it an Open Knowledge Base, open KB) from a longer text, one obtains many predicates, redundant statements, and ambiguity.

OpenIE is often used for commonsense knowledge base (CSKB) construction. Previous works such as TupleKB [18], Quasimodo [27,28] or Ascent [19–21] use OpenIE to extract knowledge from different textual sources (textbooks,

T. R. Payne et al. (Eds.): ISWC 2023, LNCS 14265, pp. 368–387, 2023.
https://doi.org/10.1007/978-3-031-47240-4_20

query logs, question-answering forums, search engines, or the Web), and then add additional steps to clean and normalize the obtained data. Another example is ReVerb [10], which was used to get OpenIE triples from a Web crawl. The output of OpenIE typically inherits noise from sources and extraction, and the resulting KBs contain an open-ended set of predicates. This generally is not the case for knowledge bases with a predefined schema. Famous instances of this type are manually constructed, like ConceptNet [31] and ATOMIC [15]. They tend to have higher precision. Besides, they are frequently used in downstream applications such as question-answering [11,42], knowledge-enhanced text generation [43], image classification [40], conversation recommender systems [47], or emotion detection [46]. These applications assume there are a few known predicates so that we can learn specialized parameters for each relation (a matrix or embeddings with a graph neural network). This is not the case for open KBs.

Still, many properties of open KBs, such as high recall and ease of construction, are desirable. In this paper, we study *how to transform an open KB into a KB with a predefined schema*. More specifically, we study the case of commonsense knowledge, where ConceptNet is by far the most popular resource. From an open KB, we want to generate a KB with the same relation names as ConceptNet. This way, we aim to increase precision and rank the statements better while keeping high recall. Notably, as we reduce the number of relations, we obtain the chance to make the statements corroborate. For example, *(fish, live in, water, freq:1)*, *(fish, swim in, water, freq:1)* and *(fish, breath in, water, freq:1)* can be transformed into *(fish, LocatedIn, water, freq:3)*, and therefore they all help to consolidate that statement.

Transforming open triples to a predefined schema raises several challenges. In the simplest case, the subject and object are conserved, and we only need to predict the correct predefined predicate. This would be a classification task. For example, *(fish, live in, water)* can be mapped to *(fish, LocatedAt, water)* in ConceptNet. We could proceed similarly in cases where subject and object are inverted, like mapping *(ocean, contain, fish)* to *(fish, LocatedAt, ocean)*, with just an order detection step. However, in many cases, the object is not expressed in the same way or only partially: *(fish, live in, the ocean)* can be mapped to *(fish, LocatedAt, ocean)*. In other cases, part or all of the predicate is in the object, like *(fish, swim in, the ocean)* that can be mapped to *(fish, CapableOf, swim in the ocean)*. Here, the initial triple could also be mapped to *(fish, LocatedAt, ocean)*, showing that the mapping is not always unique. Other problems also arise, like with (near) synonyms. For example, we might want to map *(fish, live in, sea)* to *(fish, LocatedAt, ocean)*.

Approach and Contribution. We propose to approach the mapping of an open KB to a predefined set of relations as a translation task. We start by automatically aligning triples from the source and target KB. Then, we use these alignments to finetune a generative language model (LM) on the translation task: Given a triple from an open KB, the model produces one or several triples in the target schema. The generative nature of the LM allows it to adapt to the abovementioned problems while keeping a high faithfulness w.r.t. the source KB. Besides, we

show that this improves the precision of the original KB and provides a better ranking for the statements while keeping a high recall. Our contributions are:

1. We define the problem of open KB mapping, delineating it from the more generic KB canonicalization and the more specific predicate classification.
2. We propose a generative translation model based on pre-trained language models trained on automatically constructed training data.
3. We experimentally verify the advantages of this method compared to traditional manual and rule-based mapping, classification, and purely generative methods like COMET.

2 Previous Work

2.1 Commonsense Knowledge Bases

ConceptNet. ConceptNet [31], built since the late 1990s via crowdsourcing, is arguably today's most used commonsense knowledge base. Due to user-based construction, it has high precision. ConceptNet comprises a limited set of predefined relations and contains non-disambiguated entities and concepts. For example, we find *(mouse, PartOf, computer)* and *(mouse, PartOf, rodent family)*. Thus, when mapping an open KB to ConceptNet, one needs to focus mainly on the predicates and, to some extent, the modification of the subject and object.

Open Knowledge Base. An open knowledge base (open KB) is a collection of SPO triples *(subject, predicate, object)* with no further constraints on the components. This means that they are not canonicalized. For example, the triples *(The Statue of Liberty, is in, New York)* and *(Statue of Liberty, located in, NYC)*, although equivalent, could be present in the same knowledge base. The subject and the object are noun phrases (NP), whereas the predicate is a relational phrase (RP). As a comparison, knowledge bases with a predefined schema like Wikidata [36], YAGO [33] or ConceptNet [31] come with a set of predefined predicates and/or entities for the subjects and the objects. This paper will call such a knowledge base a *Closed Knowledge Base* (closed KB).

This paper will use two open KBs: Quasimodo and Ascent++. Quasimodo [28] is an open commonsense knowledge base constructed automatically from query logs and question-answering forums. Ascent++ [19] is also an open commonsense knowledge base created from Web content. The extraction follows a classical pipeline and outputs an open KB in both cases.

2.2 From Open KBs to Closed KBs

Open Knowledge Base Canonicalization. The task of open KB canonicalization [12] consists of turning an open triple *(s, p, o)*, where s and o are an NP and p is an RP, into an equivalent (semantically) new triple *(s_e, p_e, o_e)*, where s_e and o_e represent entities (generally through a non-ambiguous NP), and p_e is a non-ambiguous and unique representation of a predicate. It means there is no

other p'_e such that *(*s_e*,* p_e*,* o_e*)* is semantically equivalent to *(*s_e*,* p'_e*,* o_e*)*. For example, we would like to map *(Statue of Liberty, located in, NYC)* to *(The Statue of Liberty, AtLocation, New York City)*, where "The Statue of Liberty" represents only the famous monument in New York City, "New York City" represents the American city unambiguously, and "AtLocation" is a predicate used to give the location of the subject.

NP canonicalization is more studied than RP canonicalization, but the task is generally treated as a clusterization problem [12]. It is essential to notice that an NP or an RP does not necessarily belong to a single cluster, as this cluster may depend on the context. For example, in *(Obama, be, president of the US)*, "Obama" refers to the entity "Barack Obama", whereas in *(Obama, wrote, Becoming)*, "Obama" refers to "Michele Obama". Also, we must notice that canonicalization does not have a target: *The transformation does not try to imitate the schema of an existing knowledge base.* The main goal is to reduce redundancy, but the number of predicates (and entities) might remain high.

Entity Linking. Entity Linking is the task of mapping an entity name to an entity in a knowledge base. For example, we would like to map *Paris* in *(Paris, be, city of love)* to Q90 in Wikidata, the entity that represents the capital of France. In the triple *(Paris, be, a hero)*, *Paris* should be mapped to Q167646 in Wikidata, the entity that represents the son of Priam. When mapping an open KB to a closed KB, most systems first perform entity linking before processing the predicate [6,44]. This supposes that the subject and the object remain unchanged during the mapping. This is a problem when we want to map to ConceptNet as this KB is not canonicalized, and the subject and object might be modified.

Knowledge Base Construction. Knowledge base construction can be done manually by asking humans to fill in the KB [15,31] or automatically using pattern matching [1,33] or OpenIE [18,19,28]. In general, manual approaches have higher accuracy but struggle to scale. Translating an open KB to a closed KB can be seen as an additional stage in an OpenIE extraction pipeline like Quasimodo or Ascent++. By doing so, we make the KB match a predefined schema. The same result would be possible directly from the corpus using traditional IE techniques. However, this approach is more human-labor intensive, depends on the domain, and does the scale [48].

Ontology Matching. Ontology matching is the task of mapping one structured schema into another [9]. This task has a long history in databases and semantic web research. However, due to the input being of little variance in predicates, it is typically approached as a structured graph alignment problem [5,8]. We cannot simply map one predicate to another in the present problem, as textual predicates are generally ambiguous. The mapping may differ for different s-o-pairs with the same p.

2.3 Existing Systems

In this paper, we are interested in a task that was barely tackled by previous works: We want to map an *entire* open KB to the schema of an existing closed KB. In the Ascent++ paper [19], the authors noticed that using an open KB in practice was difficult due to the lack of existing frameworks. Therefore, they proposed to map Ascent++ to ConceptNet's schema. However, they did a straightforward manual mapping that involved translating as many relations as possible manually. This approach is simplistic and does not yield good results, as we will see later. KBPearl [17] did a variation of the manual mapping in which they used the existing labels of entities and predicates, which greatly limits the system.

When we look at similar tasks, we find two main ideas to transition between an open KB and a closed KB. First, some authors approached this problem via rule mining, a generalization of the manual mapping of predicates. Previous systems [6,29,30] often use a rule mining system (automatic or manual) that relies on the type of subject and object and keywords in the triple. They often return a confidence score. The main issues with these frameworks are that they generalize poorly (particularly to unseen predicates) and require significant human work.

The second way to see our problem is as a classification task: Given an open triple *(s, p, o)*, we want to predict a semantically equivalent/related triple *(s, p', o)* that would be in the considered closed KB. OpenKI [44] used neighbor relations as input of their classifier. Later [41], word embeddings were included to represent the predicate and help with the generalization. However, their training and testing dataset is constructed using an open KB and a closed KB with entities already aligned by humans (ReVerb [10] and Freebase in the original paper). This is not generally the case in practice. Besides, this approach considers that the subject and object remain the same, thus ignoring modification of the subject and object, inverse relations, or closely related entities.

In [24], the authors propose a method to compute the similarity between a triple in an open KB and a triple in a closed KB. This differs from our approach because we do not know potential candidates in the closed KB in advance. Indeed, the closed KB is often incomplete, and we want to generate new triples thanks to the open KB. Therefore, we focus more on the generation rather than the comparison. However, it is essential to notice that this approach integrates word embeddings for comparison. Besides, the authors use the distant supervision approach to create a dataset automatically: Given a close triple, they find sentences (in a different corpus) containing both entities from the triple. Then, they apply an OpenIE algorithm to obtain an open triple. This triple is used as a ground truth. In our case, we do not have this additional textual source: The inputs are the open KB and the closed KB.

T-REx [7] aligns Wikipedia abstracts with the Wikidata triples using a rule-based system. However, it comes with several limitations. First, it takes as input text and not open triples. Even if we were to take the documents used for constructing Ascent++ (a web crawl), the computation time would be much longer because of the difference in scale. Second, T-REx needs to perform named-entity

recognition which does not apply to commonsense. Third, there is a strong dependency between Wikipedia and Wikidata. Some pages are even created automatically from Wikidata. Despite these limitations, we can consider the rule-based alignment presented in Sect. 4.1 as a generalization of their AllEnt aligner. T-REx was used for evaluating language models in a zero-shot fashion [23,38], or for OpenIE [37].

In [13], the authors introduce a methodology to manually evaluate the alignment of triples from an open KB with a closed KB. Besides, they studied how much an open KB (OPIEC [14] in their case) can be expressed by a closed KB (DBpedia [1]). They found that the open triples can often be aligned to DBpedia facts, but they are generally more specific. Also, one can usually express an OpenIE fact in the DBpedia schema. Still, this expressivity is limited if we consider only a single relation rather than a conjunction (or even a more expressive logical formula).

3 Problem Formulation

An open triple t consists of a subject s, a predicate p, and an object o. An open knowledge base $\mathcal{K}_O$ is a set of open triples. A closed schema $\mathcal{R}_C$ is a set of relations $\{R_1, \ldots, R_n\}$. A triple mapping m is a function that takes an open triple t and a closed schema $\mathcal{R}_C$ and produces a set of triples with predicates from $\mathcal{R}_C$.

Note that m is not defined as producing a single output triple per input triple - depending on the closed schema's structure, some open triples may give rise to several closed triples. Besides, the subject and object are not guaranteed to remain the same.

Problem. Given an open KB $\mathcal{K}_O$ and a closed set of relations $\mathcal{R}_C$, the task is to find a mapping m that enables to build a closed KB $\mathcal{K}_C = m(\mathcal{K}_O, \mathcal{R}_C)$, with the following properties:

1. **Preserves source recall**. In other words, ensure that as many triples as possible are mapped to a nonempty set, maximizing $|\{t_O \in \mathcal{K}_O \mid m(t_O, \mathcal{R}_C) \neq \emptyset)\}|$.
2. **Remains source-faithful**. In other words, ensure that each triple in the output stems from one or several semantically similar statements in the input, that is, that for each $t_c \in \mathcal{K}_C$, $m^{-1}(t_C, \mathcal{R}_C)$ is semantically similar to t_C.
3. **Corrects errors**. In other words, the goal is to minimize the set of triples in $\mathcal{K}_C$ that are factually wrong.

The definition above hinges on the concept of semantic similarity. In line with previous work [13], we specifically refer to semantic equivalence or entailment. In summary, the truth of t_O should be a sufficient condition for the truth of t_C. However, our method does not desire the opposite direction, i.e. producing t_C statements that are sufficient conditions for t_O.

4 Methodology

As we saw in Sect. 2 and will describe in more detail in Sect. 5.2, previous works propose to tackle the open KB mapping task in three different ways: manual mapping, rule mining mapping, or classifier mapping. However, these methods all come with challenges: They require much human work, cannot modify the subject and the object, cannot cover all cases, and, as we will see, have low performance. Therefore, we introduce here a new methodology to tackle these issues. We present in Fig. 1 our approach. It is composed of four steps:

1. *Alignment*: We automatically create a dataset of alignments using the open KB and the closed KB.
2. *Finetuning*: We use this dataset to finetune a generative language model (GPT-2 here) to generate alignments.
3. *Generation*: We generate one or several mappings for each triple in the open KB.
4. *Ranking*: Using the score in the original KB, the generation score, and the rank of the generated alignment, we create a final score for each closed triple.

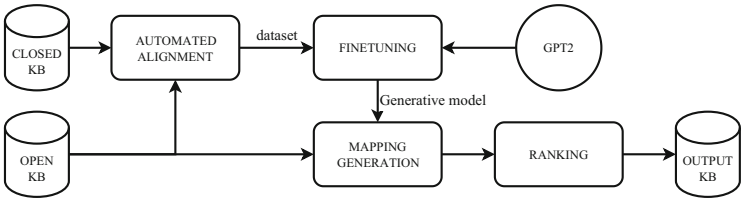

Fig. 1. The Steps of Our Methodology.

4.1 Creating Weakly-Labelled Training Data

The generative translation mapping and existing classification and rule mining approaches require a training dataset of alignments, that is, pairs of open triples and semantically equivalent or entailed closed triples. Creating this at scale manually is hardly feasible. Therefore, we decided to adopt two automatic approaches to generate a broader dataset, even if they contain some noise.

Rule-Based Alignment. The first approach we consider is based on rules. To align an open KB with ConceptNet, we used the following algorithm:

1. Lemmatization and stopwords removal.
2. For each triple (s, p, o) in our open KB, we create one or several alignments (we used _ as a general placeholder):

- s, _, o is in Conceptnet (standard alignment)
- o, _, s is in Conceptnet (reverse alignement)
- s, _, p + o is in Conceptnet (predicate in the object)
- p + o, _, s is in Conceptnet (reverse predicate in the object)

This approach has the advantage of not creating divergence in the alignment: We have hard constraints (words) that do not allow us to align statements that are too different. However, this is not true with the following technique.

LM-Based Alignment. Here, we propose an entirely unsupervised method. First, we compute the embeddings of each triple in both KBs and then align each open triple with the nearest close triple. For computing the triple embeddings, we used a sentence embeddings neural network fed with the subject, the predicate, and the object, separated by a comma. The Python library SBert provides a MiniLM [39] model finetuned on a paraphrasing task. Then, we used Scikit-learn [22] K-nearest neighbor algorithm to find the nearest neighbor in a closed KB for each triple in an open KB (or the opposite, marked as INV later) and the distance between the two triples. Finally, as considering all the alignments might introduce noise, we assess several scenarios in which we only take the top 1k, 10k, and 100k alignments according to the distance score. As generating the mapping is expensive, we will not finetune this parameter more.

With this technique, we might have alignments that are not related. However, compared to the previous method, we will be able to have a larger dataset, and we might get semantic relatedness coming from using different wording (e.g., with synonyms) that was not captured before.

4.2 Generative Translation Mapping

The second step consists in finetuning a generative language model to generate the mappings. This can be seen as a translation problem, similar to machine translation. We formatted our input by separating the OpenIE triple and its aligned ConceptNet triple with a *[SEP]* token. In our experiments, we used the GPT-2 model [25] and the script provided by HuggingFace to finetune GPT-2. Unfortunately, GPT-3 [3] is not publicly available. We also tried T5 [26], but we did not obtain better results (see T5-GenT in Table 1). We also accessed a very large language model, LLaMa [35], following Alpaca [34,45]. However, this model failed to adapt to the structure of the closed KB, even when given various explicit prompts. We hypothesize that such a model lost flexibility as it better understood natural language. Succeeding in reintroducing structured information in very large LM can lead to exciting future works. Another disadvantage of very large LMs is their computation cost at training and during inference.

The third step is the actual generation. We used a beam search for the generation part to obtain the top K results for each statement in our knowledge base. We filter the results to keep only well-formed triples and triples so that the subject and object differ. Considering more than one alignment per triple

can help in many ways. First, a triple can have several translations. Second, the system learned to generate related statements that might help rank the final statements.

Finally, once we have all the translations to ConceptNet triples, we compute a score for each triple based on the frequency at which it appeared (several OpenIE triples generally generate the same closed triple) and the inverse rank among the predictions. More formally, we obtain the score of a triple t using:

$$\text{FinalScore}(t) = \sum_{\text{t' generates t}} \frac{\text{score}(t')}{\text{rank}(t',t)+1}$$

We will also consider two other scores in Sect. 6. The first only considers the open KB score part of the previous formula (a sum of scores), while the second only considers the ranks (a sum of reciprocal ranks). Here, it is essential to notice that the score of an open triple is provided by the open KB. Therefore, if the open KB is not good at scoring triples, we will inherit negative signals that we hope to compensate with the ranks.

In the end, we can generate a ranking for all our statements. Moreover, using a generative LM allows for having friendly properties missing in previous works. For example, it can adapt the subject and the object to match the new predicate. Besides, it can also correct the original statement if it contains a mistake (spelling or truth). Furthermore, it can inverse the subject and the object without additional help. Finally, it can generate multiple outputs from one input, bringing value to the end KB. We will demonstrate these properties in Sect. 6.

5 Experiment Setup

5.1 Evaluation

Automatic Global Metrics. To get a general understanding of the generated KB after the mapping, we compute the size of the KB. However, as the size is insufficient to evaluate the recall [28], we consider that ConceptNet is our gold standard as humans filled it. Then, we measure the number of triples from ConceptNet we can generate. We call it the *automatic recall*. Likewise, we create the *automatic precision*:

$$R_a(KB_{trans}) = \frac{|KB_{trans} \cap KB_{target}|}{|KB_{target}|} \qquad P_a(KB_{trans}) = \frac{|KB_{trans} \cap KB_{target}|}{|KB_{trans}|}$$

As a part of the target KB can be used in the training dataset, we also define $\bar{R}_a$ as:

$$\overline{R}_a(KB_{trans}) = \frac{|KB_{trans} \cap KB_{target} - D_{train}|}{|KB_{target} - D_{train}|}$$

We also define $\overline{P_a}$ following the same rationale as $\overline{R}_a$. However, this metric does not capture the ranking of our statements. The ranking is crucial in open KBs as these KBs are often noisy. Ideally, we want to have correct and important

statements with a high score. We introduce metrics to measure that property. First, we will also use an *automatic precision at K* where, instead of considering the entire KB_{trans}, we will only consider its top K statements. However, these metrics do not consider the entire KB. Therefore, we introduce a generalized mean reciprocal rank of the final ranked KB as follows:

$$MRR(KB_{trans}) = \frac{\sum_{KB_{trans}[i] \in KB_{target}} \frac{1}{i}}{|KB_{target}|} \qquad \overline{MRR}(KB_{trans}) = \frac{\sum_{KB_{trans}[i] \in KB_{target} - D_{train}} \frac{1}{i}}{|KB_{target} - D_{train}|}$$

These metrics allow us to measure the recall, but it gives more weight to correct high-ranked statements.

All the metrics presented depend on the quality and coverage of the original open knowledge base. Therefore, when considering a translated KB, we prefer relative metrics where the metric is divided by the metric computed for the open knowledge base, ignoring the relations.

Automatic Triple Alignment Metrics. In Sect. 4.1, we suggested methods to align an open KB with a closed KB. These techniques generate a dataset of alignments that can be split into a training and a testing set used to evaluate the MRR, the precision (@K), and the recall (@K).

Manual Metrics. The automatic metrics we presented above are cheap to run but give a coarse approximation of the quality of the resulting knowledge. Therefore, we introduce manual metrics here. They are more expensive to run as they require human work but will provide a more precise evaluation.

Manual Triple Metrics. Inspired by [13], we would like to evaluate the quality of the triple mapping according to three parameters:

- *Correct mapping*: Is the generated triple a correct mapping of the open triple, i.e., is it semantically equivalent/related to the original triple?
- *Correct prediction*: Is the resulting triple true? Independently of whether the mapping is correct, we would like to know if the resulting triple is accurate. This can be useful for several reasons. First, even if the mapping is incorrect, we would prefer that it does not hurt the quality of the knowledge base we construct next. Second, as the input triple may be noisy and incorrect, we would prefer that the system generates a correct statement rather than a correct mapping. Finally, if such a property holds, it will prove that the system has some cleaning properties that will help improve the quality of the open KB.
- *Correct open triple*: Is the original open triple correct? This information will help evaluate what the system predicts depending on the quality of the input triple (see the point above).

Knowledge Base Level Metrics. Precision and recall are crude automated heuristics w.r.t. another data source. To evaluate the quality of novel CSK resources meaningfully, we rely on the *typicality* notion of previous works [20,28]: We ask humans how often a statement holds for a given subject. Possible answers are: Invalid (the statement makes no sense) or Never / Rarely / Sometimes / Often / Always. Each answer has a score between 0 and 4 to compute a mean.

5.2 Baselines

Manual Mapping. For this baseline, we manually map the relations in an open KB to relations in ConceptNet. It is inspired by an idea from [19]. Given a predicate p in an open knowledge base, we ask humans to turn it into a predicate p' in ConceptNet (including inverse relations). There are many relations in an open KB, so we only mapped the top relations. We also notice that, in many cases, a triples *(s, p, o)* can directly be mapped to the triple *(s, CapableOf, p + o)*. For example, *(elephant, live in, Africa)* could be mapped to *(elephant, CapableOf, live in Africa)*. If we cannot find a better translation, we default to this translation. This approach is a simple rule system. In our case, we annotated 100 predicates for Quasimodo and Ascent++. By doing so, we cover 82% of triples in Quasimodo and 57% of triples in Ascent++.

Rule Mining. Given that no previous work made their implementation public, we propose a rule mining approach inspired by previous works [6,29,30]. Our method requires a training dataset of mappings. In our case, this dataset was constructed automatically (see Sect. 4.1) using the rule-based alignment method. The LM-based alignment is inappropriate as the subject and object must remain unchanged with the rule mining approach. Like in previous works, we also used WordNet to get the type of the concepts.

We use AMIE [16] to mine Horn rules of the form $B \Rightarrow r(x, y)$. The PCA confidence proposed in AMIE yields poor results. Therefore, we used the standard confidence. Ultimately, we only keep rules with a confidence score greater than 0.5. An advantage of this method is that it provides high interpretability: For each final generation, we can see which open triples were used to generate it and which rules were applied.

Classification Task. For this baseline inspired by OpenKI [44], we want to use a classifier to predict the ConceptNet relation of an open triple. Given a triple *(s, p, o)* in an open KB, we want to predict a relation p' (including inverse relations) in ConceptNet such that (s, p', o) would be in ConceptNet. To do so, we used a classifier based on BERT [4] and trained it with a dataset created automatically (see Sect. 4.1). Building this dataset by hand would be possible, but it would take much time, and we would get problems getting enough examples for each predicate. Besides, we will use the same training dataset with the translation models.

5.3 Implementation

We implemented the baselines using Python3 (except for AMIE, written in Java). For the generative LM, we used GPT-2-large given by Huggingface. We ran our code on machines with NVIDIA Quadro RTX 8000 GPUs. Finetuning a language model required a single machine for a maximum of two days. We used three training epochs in our experiments. However, mapping an open KB to ConceptNet was much longer and took up to 30 days on a single GPU. Nevertheless, the computations can easily be parallelized on several GPUs by splitting the input data, which allows us to speed up the process. In our experiments, we used Quasimodo and ASCENT++ as open KBs and mapped them to ConceptNet commonsense relations. The code (github.com/Aunsiels/GenT) and data (julienromero.fr/data/GenT) are available. .

6 Results and Discussion

6.1 Comparison with Baselines

Table 1 shows the results of the automated metrics for all baselines. The first thing to notice is that the metrics seem "low". We recall that they are in fact relative to the open KB with the relations ignored, as mentioned in Sect. 5.1. Therefore, they only have a relative interpretation. Even with the generous evaluation of the open KB, many metrics have a value of more than one, showing a significant improvement, particularly for the recall. For precision, a value less than one mainly comes with the growth of the KB size.

Our proposed approach clearly outperforms the various baselines. The basic models are not flexible and do not tackle the challenges we mentioned earlier. For manual mapping, the annotation process depends on humans and is not trivial, as translating a predicate often depends on the context. The classifier model performs better than the two other baselines when we look at the recall. Still, we observe problems to generalize as $\overline{R_a}$ is low.

Table 1. Automatic (Relative) Recall And Precision (* ignores the predicates).

KB	Method	**Training data**	R_a	$\overline{R_a}$	P_a	$\overline{P_a}$	P_a@10	P_a@100	P_a@1000	P_a@10k	$\overline{P_a}$@10k	MRR	$\overline{MRR}$	Size
ConceptNet	KB itself	–	–	–	–	–	–	–	–	–	–	–	–	232,532
Quasimodo	KB itself	–	2.54%*	–	0.271%*	–	10%*	17%*	11%*	4.79%*	–	$4.63e^{-6}$ *	–	5,930,628
Ascent++	KB itself	–	1.63%*	–	0.430%*	–	10%*	10%*	6.6%*	3.13%*	–	$3.56e^{-6}$ *	–	1,967,126
KB	**Method**	**Training data**	$R_{a,rel}$	$\overline{R_{a,rel}}$	$P_{a,rel}$	$\overline{P_{a,rel}}$	$P_{a,rel}$@10	$P_{a,rel}$@100	$P_{a,rel}$@1000	$P_{a,rel}$@10k	$\overline{P_{a,rel}}$@10k	MRR_{rel}	$\overline{MRR_{rel}}$	**Size**
Quasimodo	Manual Mapping [19]	–	0.231	–	0.103	–	1.000	0.471	0.455	0.315	–	0.592	–	4,925,792
	Rule Mining [6,29,30]	Rule-based	0.161	0.006	0.509	0.020	3.000	1.235	0.645	0.365	0.004	1.259	0.002	689,146
	Classifier [44]	Rule-based	0.752	0.042	0.299	0.016	2.000	1.000	0.855	0.672	0.002	1.419	0.001	5,478,028
	GenT@1	Rule-based	1.465	0.425	**0.771**	0.217	8.000	4.471	4.127	3.361	0.201	4.816	0.098	4,135,349
	GenT@10	Rule-based	2.563	1.319	0.176	0.085	9.000	4.588	4.227	**3.612**	0.234	**4.968**	0.097	33,425,732
	GenT@10	LM-based@10k	2.370	1.677	0.347	**0.235**	2.000	2.294	3.227	2.777	0.357	2.505	0.069	15,647,853
	GenT@10	LM-based@10k-INV	**2.787**	**1.933**	0.241	0.162	1.000	1.059	1.391	1.939	**0.660**	1.333	**0.216**	25,798,594
	T5-GenT@10	LM-based@10k-INV	1.843	1.020	0.123	0.065	–	–	–	1.094	0.236	0.670	0.070	33,874,204
Ascent++	Manual Mapping [19]	–	0.287	–	0.205	–	0.000	0.500	0.485	0.415	–	0.351	–	1,228,001
	Rule Mining [6,29,30]	Rule-based	0.223	0.060	0.705	0.190	2.000	0.600	0.394	0.511	0.045	1.306	0.034	277,835
	Classifier [44]	Rule-based	0.663	0.180	0.340	0.105	2.000	1.200	0.652	0.649	0.026	0.784	0.016	1,722,441
	GenT@1	Rule-based	1.706	0.785	**1.147**	**0.523**	4.000	4.800	3.091	2.722	0.396	2.949	0.278	1,277,065
	GenT@10	Rule-based	3.055	1.933	0.260	0.160	5.000	4.800	3.803	3.073	0.454	3.989	0.500	10,193,040
	GenT@10	LM-based@10k	3.497	2.546	0.444	0.319	6.000	4.700	3.803	**3.450**	1.096	**4.494**	0.216	7,000,135
	GenT@10	LM-based@10k-INV	**4.000**	**2.613**	0.428	0.272	4.000	2.000	2.727	**3.450**	**1.326**	2.736	**0.556**	8,305,861

Table 2. Manual Alignment Evaluation On Quasimodo (left) and Typicality (right).

Dataset	Sem. Rel.	Open Correct	Close Correct	Both Correct
Rule-based	53.0%	85.3%	69.3%	64.8%
GenT@10k	45.7%	85.3%	75.7%	68.0%
GenT@10k-INV	**55.3%**	85.3%	**77.3%**	**69.7%**

KB	Alignment	Typicality
ConceptNet	-	**3.18**
Quasimodo	-	2.70
Quasimodo	Rule-based	2.91
Quasimodo	GenT@10k-INV	2.88
Ascent++	-	2.31
Ascent++	Rule-based	2.68
Ascent++	GenT@10k-INV	2.88

Table 3. Automatic Triple Alignment MRR, Recall And Precision (as usually defined).

KB	Method	Dataset	MRR	R@1	R@5	R@10	P@1	P@5	P@10
Quasimodo	Manual	Manual	$1.56e^{-2}$	1.51%	–	–	1.56%	–	–
	Rule Mining	Rule-based	$5.96e^{-2}$	5.55%	17.8%	24.8%	5.68%	3.63%	2.52%
	Classifier	Rule-based	0.194	19.0%	–	–	19.4%	–	–
	GenT@10	Rule-based	**0.381**	**31.6%**	**46.7%**	**49.5%**	**31.1%**	**9.67%**	**5.12%**
	GenT@10	LM-based@10k	0.279	23.1%	34.5%	36.9%	23.1%	6.91%	3.69%
	GenT@10	LM-based@100k	0.319	27.5%	38.0%	39.8%	27.5%	7.60%	3.98%
	GenT@10	LM-based@1k-INV	0.211	15.4%	26.9%	34.6%	15.4%	5.38%	3.46%
	GenT@10	LM-based@10k-INV	0.123	8.48%	17.1%	20.4%	8.77%	3.51%	2.09%
	T5-GenT@10	LM-based@10k-INV	0.129	10.0%	16.6%	19.9%	10.1%	3.40%	2.05%

Table 4. SO Conservation For Quasimodo.

	First gen.			At least one gen.			All gens.		
Alignment	S	O	SO	S	O	SO	S	O	SO
Rule-based	36.8%	48.5%	**26.6%**	**57.9%**	76.2%	**48.3%**	25.0%	**35.3%**	**12.4%**
LM-based@1k	27.5%	21.0%	7.37%	45.3%	41.7%	17.9%	22.3%	12.3%	2.37%
LM-based@1k-INV	**38.7%**	**53.6%**	22.4%	55.2%	**77.5%**	42.0%	**27.0%**	31.9%	5.84%

6.2 What Is the Best Alignment Method?

In Sect. 4.1, we presented two automatic alignment methods. The first is based on a rule system, whereas the second aligns with the closest triples in a latent space using embeddings. We refer to the first as Rule-based and the second as LM-based@K(-INV) when we used top K statements of the complete dataset obtained by aligning each open triple with a close triple (INV means we align each close triple with an open triple).

Do They Allow the Model to Generate Accurate Alignments? Table 3 gives the performance of the model on a test dataset derived from the complete dataset. Therefore, it is not the same for all models and depends on the alignment method. Still, it gives us some valuable insights. We can see that the Rule-based alignment is the easiest to learn. This is due to the strong correlation created by the rules between the open and the closed triples. According to the metrics,

the INV methods perform worse than non-INV ones. A reason might be that the INV alignment has more diversity: A triple from ConceptNet can appear only once in the dataset (we align each close triple with a single open triple). Therefore, it might be harder to learn.

Table 4 shows the conservation of the subject S and object O during the generation phase. We want to observe if they remain the same for the first generation, for at least one generation, or for all generations. The rule-based system encodes these constraints and should therefore outperform the other baselines. However, interestingly, we observe that the INV methods have excellent conservation, competing with the rule-based system (except for SO conservation), and largely beating non-INV alignments. This is surprising as it contains no prior constraint. It is a property that we expect from a good alignment method as we do not want the generated close triples to diverge from the original triples.

All these evaluations are automatic and only approximate the model's capabilities. We additionally performed manual annotations of the generations to check if the generated close triples are correct alignments (according to semantic relatedness, as discussed in Sect. 5.1). We sampled 300 triples from the top 10k triples in Quasimodo and looked at the first generation for three models. The results are presented in Table 2. We observed that the rule-based and INV alignments have similar performances for generating related close triples. Only the non-INV model underperforms, which matches what we noticed for SO conservation. Here, semantic relatedness is relatively low because it is quite constraining. However, we observe that the generated triples share most of the time part of the subject or object with the original triple.

What Is the Impact of the Training Dataset Size? We sampled the top-K samples with $K \in \{1k, 10k, 100k\}$ in the training dataset and picked the best size. We observed that the model performs best for 10k samples. Note that finding the optimal size would take too much time as, with these metrics, we need to generate the entire mapping. The testing dataset used in Table 3 gives a faster heuristic for finding a good K.

6.3 What Are the Properties of Our Model?

In Sect. 1 and Sect. 3, we described properties we want our new system to have, such as high recall and precision, flexibility thanks to the LM, and good generalization. We will investigate these advantages here in more detail.

What Are the Advantages of Using a Generative Translation Model? When we use traditional models like manual mapping, rule mining, or classifiers, we encounter problems coming from their lack of flexibility: The subject and object often remain unchanged, the mapping is often unique, mistakes are not corrected, etc. With a generative translation model, we can tackle some of these challenges. In Table 5, we present examples of nice and unique properties we observed in the final results. This shows that using LMs with open KBs lets us

get the best of both worlds. LMs are more flexible and contain knowledge that is not easily extractable [23]. The open KB helps guide the LM.

Can We Improve the Quality of an Open KB with a Generative Translation Model? To evaluate the evolution of the quality of an open KB, we asked humans to annotate the typicality of statements. We sampled 300 statements out of the top 10k statements for each KB and then computed the mean typicality. The results are reported in Table 2. As we can see, the generative translation methods significantly improve the quality of the statements. The best-performing alignment method seems to depend on the open KB. As expected, ConceptNet still outperforms our approach as it was manually generated. However, it does not have the same scaling capabilities.

Table 5. Examples of Mappings from GenT.

Property	Open triple	Generated closed triple
The mapping depends on the context	(elephant, has_property, decorated)	(elephant, ReceivesAction, decorated)
Adapt the subject/object	(elephant, be in, africa killed) (doctor, write, prescription) (doctor, be in, training) (doctor, keep, from getting sick)	(elephant, AtLocation, africa) (doctor, CapableOf, prescribe) (doctor, AtLocation, medical school) (doctor, CapableOf, keep patient healthy)
Inversion subject/object	(elephant, have, tusks)	(tusk, PartOf, elephant)
Correct the subject/object	(elephant, have, ivory tusks answers)	(elephant, HasA, ivory tusk)
Put the predicate in the object	(doctor, wear, coat)	(doctor, CapableOf, wear coat)
Good triples from nonsense	(doctor, has_property, as a career)	(doctor, HasProperty, well respected)
Generate related triples	(doctor, wear, mask)	(doctor, CapableOf, wear lab coat)
Turn s/o into standard form	(apartment, be in, nyc)	(apartment, AtLocation, new york city)

Can GenT Generalize Across Open KBs? Table 6 shows the results of models trained on one open KB, Quasimodo or Ascent++, and used to generate a closed KB from triples in another open KB. We chose the LM-based@10k-INV alignment. In most cases, the original model trained with the same open KB outperforms the foreign model. This is understandable as the data sources and processing steps used to generate the open KBs differ, and therefore the style of the open triples is different. So, the model might have difficulties adapting. Still, the new results are close to the original ones, showing that we can have the reusability of our models with entirely new data. Finally, some metrics seem less impacted by the change of the original open KBs. From what we can see, the ranking capabilities, expressed through P_a and MRR, vary but not necessary for the worst. It shows that the generation and the scoring stage allow selecting good close triples, whatever the new data is.

Table 6. Performances when evaluating with a model trained for another KB.

KB	Method	$R_{a,rel}$	$\overline{R}_{a,rel}$	$P_{a,rel}$	$\overline{P_{a,rel}}$	$P_{a,rel}$@10	$P_{a,rel}$@100	$P_{a,rel}$@1000	$P_{a,rel}$@10k	$\overline{P_{a,rel}@10k}$	MRR_{rel}	$\overline{MRR_{rel}}$
Quasimodo	GenT@1	1.35	0.81	0.77	0.46	70%	67%	37.3%	0.73	0.76	4.28	0.98
Quasimodo	GenT@10	2.44	1.77	0.18	0.13	70%	71%	38.6%	2.92	0.84	4.41	0.96
Ascent++	GenT@1	1.95	0.77	1.28	0.50	80%	56%	25.2%	3.17	0.38	4.69	0.08
Ascent++	GenT@10	3.62	1.96	0.30	0.16	80%	65%	32.4%	3.87	0.44	5.37	0.10

Generalization to Sentences. In Table 7, we took sentences or paragraphs from several sources (Wikipedia, New York Times, GenericsKB [2]) and used our model trained on Quasimodo with the LM-based@10k-INV alignment method. Surprisingly, the model can correctly extract knowledge from sentences. This could lead to several interesting future works: Information extraction directly from sentences, aligning sentences rather than open triples, or commonsense inference.

Table 7. Examples of Generations From Sentences.

Source	First Generation
Elephants are the largest existing land animals	(elephants, DefinedAs, largest land animal)
A lawyer or attorney is a person who practices law	(lawyer, CapableOf, represent client)
Elon Musk Races to Secure Financing for Twitter Bid	(elon musk, CapableOf, bid for twitter)
South Africa's Government Shifts to Rebuilding After Disastrous Flooding. Nearly 4,000 homes have been destroyed and more than twice as many damaged in the Durban area after a week of punishing rains and mudslides. The death toll is now 448, with about four dozen people unaccounted for	(people, CapableOf, die from flooding)
Some air pollutants fall to earth in the form of acid rain	(air pollution, CapableOf, cause acid rain)

6.4 How Does GenT Compare with Direct LM Generation Methods?

As previous works like LAMA [23] suggested, a powerful language model could serve as a knowledge base. Then, aligning this "knowledge base" with a target knowledge base requires finetuning the language model. COMET [15] finetunes GPT-2 [25] to generate triples in ConceptNet. Here, we consider two kinds of input: A subject alone (denoted as COMET S) or a subject/predicate pair (designated as COMET SP). COMET initially accepted only subject/predicates pairs. However, it makes the generation of relevant triples harder as it is not always possible to associate all subjects to all predicates (for example, "elephant" and "HasSubEvent"). Then, we generate ten candidate statements for each subject or subject/predicate pair in ConceptNet. They all come with a generation score that we use for an overall ranking. In addition to the raw COMET, we used the translation models described above to generate a KB (GenT COMET). The

inputs are the same as COMET. We additionally parse the output to keep the triple on the right of the *[SEP]* token.

It turns out that our translation model is a clever scheme in between traditional IE-based KB construction and a general COMET-style generation. It overcomes the limitation of IE that requires a text as input (it can generate more triples without requiring that each is seen in input text). It also tackles some COMET challenges by providing more robust guidance on what to generate based on the input triples.

In Table 8, we observe that GenT consistently outperforms COMET in all metrics but R_a. This is easily understandable: As we do not require alignments, we can get 5 to 10 times more training data than the translation models. These data points are guaranteed to represent different ConceptNet triples (not necessarily the case for the translation models). However, if we look at $\bar{R}_a$, the translation models generalize better. Besides, we have a ranking capability lacking in the original COMET. This could be explained by the fact that the translation models first try to generate an open triple closer to natural language and then map this triple to ConceptNet. Therefore, it can better leverage its prior knowledge to focus on what is essential.

Table 8. Direct Generation Comparison, non-relative metrics

KB	Method	Dataset	R_a	$\overline{R_a}$	P_a	$\overline{P_a}$	P_a@10	P_a@100	P_a@1k	P_a@10k	$\overline{P_a@10k}$	MRR	$\overline{MRR}$
Quasimodo	GenT COMET S	Rule-based	1.09%	0.222%	**3.72%**	0.137%	**100%**	**83%**	42.2%	**13.3%**	1.66%	$\mathbf{2.59e^{-5}}$	$2.96e^{-7}$
Quasimodo	GenT COMET SP	Rule-based	2.33%	0.803%	0.403%	0.782%	30%	24%	18.6%	11.3%	1.24%	$8.15e^{-6}$	$2.66e^{-7}$
Quasimodo	GenT COMET S	LM-based@10-INV	0.657%	0.285%	2.20%	0.975%	20%	17%	15.8%	7.14%	2.86%	$9.59e^{-6}$	$5.20e^{-6}$
Quasimodo	GenT COMET SP	LM-based@10-INV	1.71%	1.07%	0.261%	0.163%	20%	21%	15.5%	6.46%	3.12%	$7.16e^{-6}$	$3.61e^{-7}$
Ascent++	GenT COMET S	Rule-based	0.977%	0.358%	3.10%	**1.15%**	**100%**	81%	**43%**	12.5%	3.90%	$2.63e^{-5}$	$2.68e^{-6}$
Ascent++	GenT COMET SP	Rule-based	2.15%	**1.11%**	0.345%	0.177%	100%	74%	38.7%	12.6%	4.14%	$1.11e^{-5}$	$\mathbf{6.77e^{-6}}$
Ascent++	GenT COMET S	LM-based@10-INV	0.825%	0.326%	2.74%	0.326%	20%	26%	28.8%	8.94%	3.50%	$6.44e^{-6}$	$1.26e^{-6}$
Ascent++	GenT COMET SP	LM-based@10-INV	2.09%	1.10%	0.340%	0.178%	40%	24%	19.6%	10.7%	**4.69%**	$8.74e^{-6}$	$3.52e^{-6}$
–	Comet S	ConceptNet	1.11%	0.144%	2.96%	0.401%	20%	25%	20.1%	7.65%	0.891%	$6.30e^{-6}$	$3.04e^{-7}$
–	Comet SP	ConceptNet	**2.87%**	0.504%	0.179%	0.504%	10%	2%	3.2%	3.36%	0.510	$1.43e^{-6}$	$1.28e^{-7}$

7 Conclusion

We studied the problem of mapping an open commonsense knowledge base to a fixed schema. We proposed a generative translation approach that carries novel properties such as flexibility and cleaning ability. In the process, we compared different ways to create training data and analyzed their advantages and disadvantages. Finally, we experimentally verified the strengths of the proposed approach both in automated and manual evaluation.

We provided the first solution for the mapping task, and there is still room for improvement. For example, we could study how to adapt state-of-the-art translation models. We could also check how the output of the generative model can be constrained to provide closed triples that are not too far from the original triples. Also, as we observed that LM-based models have cleaning capabilities, we could include a negative sample in the training dataset to predict cases where a triple has no translation (e.g. because it is incorrect).

Supplemental Material Statement: We provide mappings to ConceptNet of Quasimodo and Ascent++ as additional resources (julienromero.fr/data/GenT), in addition to the code (github.com/Aunsiels/GenT) and input data. We hope they will help improve tasks such as commonsense question answering that currently use ConceptNet, which can sometimes be problematic as some of these datasets are constructed from ConceptNet (e.g., CommonsenseQA [32]).

Limitations. Our work is based on the GPT-2 model, which is not considered state-of-the-art when this paper is released. Using newer and bigger language models raises additional challenges: Computation time (currently approx. one month on a single GPU with a beam search), training complexity (bigger models have more weights, do not fit on a single GPU, and take more space during training), or hyperparameters hard to finetune. In this paper, we laid the foundations of a new task, and we believe more complex architectures can improve upon our baselines.

In this article, we studied the case of commonsense knowledge bases that generally contain uncanonicalized entities. Our study could be extended to other kinds of knowledge base.

Acknowledgements. This work was partially funded by the Hi!Paris center, and the experiments were performed using HPC resources from GENCI-IDRIS (Grant 2022-AD010613537).

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: a nucleus for a web of open data. In: ISWC (2007)
2. Bhakthavatsalam, S., Anastasiades, C., Clark, P.: Genericskb: a knowledge base of generic statements. arXiv preprint (2020)
3. Brown, T., et al.: Language models are few-shot learners. In: NeurIPS (2020)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
5. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Ontology matching: a machine learning approach. In: Staab, S., Studer, R. (eds.) Handbook on ontologies. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24750-0_19
6. Dutta, A., Meilicke, C., Stuckenschmidt, H.: Semantifying triples from open information extraction systems. In: STAIRS (2014)
7. Elsahar, H., et al.: T-rex: a large scale alignment of natural language with knowledge base triples. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018) (2018)
8. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: six years of experience. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22630-4_6
9. Euzenat, J., Shvaiko, P., et al.: Ontology Matching. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-642-38721-0

10. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: EMNLP (2011)
11. Feng, Y., Chen, X., Lin, B.Y., Wang, P., Yan, J., Ren, X.: Scalable multi-hop relational reasoning for knowledge-aware question answering. EMNLP (2020)
12. Galárraga, L., Heitz, G., Murphy, K., Suchanek, F.M.: Canonicalizing open knowledge bases. In: CIKM (2014)
13. Gashteovski, K., Gemulla, R., Kotnis, B., Hertling, S., Meilicke, C.: On aligning openie extractions with knowledge bases: a case study. In: Eval4NLP (2020)
14. Gashteovski, K., Wanner, S., Hertling, S., Broscheit, S., Gemulla, R.: Opiec: an open information extraction corpus. AKBC (2019)
15. Hwang, J.D., et al.: (Comet-)Atomic 2020: On symbolic and neural commonsense knowledge graphs. In: AAAI (2021)
16. Lajus, J., Galárraga, L., Suchanek, F.: Fast and exact rule mining with amie 3. In: ESWC (2020)
17. Lin, X., Li, H., Xin, H., Li, Z., Chen, L.: Kbpearl: a knowledge base population system supported by joint entity and relation linking. VLDB (2020)
18. Mishra, B.D., Tandon, N., Clark, P.: Domain-targeted, high precision knowledge extraction. TACL (2017)
19. Nguyen, T.P., Razniewski, S., Romero, J., Weikum, G.: Refined commonsense knowledge from large-scale web contents. arXiv (2021)
20. Nguyen, T.P., Razniewski, S., Weikum, G.: Advanced semantics for commonsense knowledge extraction. In: WWW (2021)
21. Nguyen, T.P., Razniewski, S., Weikum, G.: Inside ascent: exploring a deep commonsense knowledge base and its usage in question answering. ACL (2021)
22. Pedregosa, F., et al.: Scikit-learn: machine learning in python. JMLR (2011)
23. Petroni, F., et al.: Language models as knowledge bases? In: EMNLP (2019)
24. Putri, R.A., Hong, G., Myaeng, S.H.: Aligning OpenIE relations and KB relations using a SIAMESE network based on word embedding. In: IWCS (2019)
25. Radford, A., et al.: Language models are unsupervised multitask learners. OpenAI blog (2019)
26. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
27. Romero, J., Razniewski, S.: Inside quasimodo: exploring construction and usage of commonsense knowledge. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. CIKM 2020, New York, NY, USA, pp. 3445–3448. Association for Computing Machinery (2020). https://doi.org/10.1145/3340531.3417416
28. Romero, J., Razniewski, S., Pal, K., Z. Pan, J., Sakhadeo, A., Weikum, G.: Commonsense properties from query logs and question answering forums. In: CIKM (2019)
29. Soderland, S., Gilmer, J., Bart, R., Etzioni, O., Weld, D.S.: Open information extraction to KBP relations in 3 hours. In: TAC (2013)
30. Soderland, S., Roof, B., Qin, B., Xu, S., Etzioni, O., et al.: Adapting open information extraction to domain-specific relations. AI magazine (2010)
31. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. In: AAAI (2017)
32. Talmor, A., Herzig, J., Lourie, N., Berant, J.: Commonsenseqa: A question answering challenge targeting commonsense knowledge. NAACL (2019)
33. Tanon, T.P., Weikum, G., Suchanek, F.: Yago 4: a reason-able knowledge base. In: ESWC (2020)

34. Taori, R., et al.: Stanford alpaca: an instruction-following llama model (2023). https://github.com/tatsu-lab/stanford_alpaca
35. Touvron, H., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
36. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM (2014)
37. Wang, C., Liu, X., Chen, Z., Hong, H., Tang, J., Song, D.: Zero-shot information extraction as a unified text-to-triple translation. arXiv preprint arXiv:2109.11171 (2021)
38. Wang, C., Liu, X., Chen, Z., Hong, H., Tang, J., Song, D.: Deepstruct: pretraining of language models for structure prediction. arXiv preprint arXiv:2205.10475 (2022)
39. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: NeurIPS (2020)
40. Wang, Y., et al.: Multi-label classification with label graph superimposing. In: AAAI (2020)
41. Wood, I., Johnson, M., Wan, S.: Integrating lexical information into entity neighbourhood representations for relation prediction. In: NAACL (2021)
42. Yasunaga, M., Ren, H., Bosselut, A., Liang, P., Leskovec, J.: QA-GNN: reasoning with language models and knowledge graphs for question answering. In: NAACL (2021)
43. Yu, W., et al.: A survey of knowledge-enhanced text generation. ACM Comput. Surv. (2022)
44. Zhang, D., Mukherjee, S., Lockard, C., Dong, X.L., McCallum, A.: Openki: Integrating open information extraction and knowledge bases with relation inference. NAACL (2019)
45. Zhang, R., et al.: Llama-adapter: efficient fine-tuning of language models with zero-init attention. arXiv preprint arXiv:2303.16199 (2023)
46. Zhong, P., Wang, D., Miao, C.: Knowledge-enriched transformer for emotion detection in textual conversations. In: EMNLP. Hong Kong, China (2019)
47. Zhou, K., Zhao, W.X., Bian, S., Zhou, Y., Wen, J., Yu, J.: Improving conversational recommender systems via knowledge graph based semantic fusion. In: KDD (2020)
48. Zhou, S., Yu, B., Sun, A., Long, C., Li, J., Sun, J.: A survey on neural open information extraction: Current status and future directions. arXiv preprint arXiv:2205.11725 (2022)