Enhanced Knowledge Graphs Using Typed Entailment Graphs

Anonymous ACL submission

Abstract

Constructing knowledge graphs from open-domain corpora is a crucial stage in question answering. Most previous works are based on open information extraction methods, which extract relations by parsing sentences into triples $<e_1, r, e_2>$. These methods lack inference ability and are limited by corpus. When the query is different from the relations in the text-based knowledge graph, it is hard to return correct answers. In this paper, we propose a method to enhance knowledge graphs by using typed entailment graphs to add missing links. We construct the enhanced knowledge graph in both dynamical and offline ways. The experiment shows that our method outperforms the pre-trained language models in zero-shot cloze-style question answering. Furthermore, we find entailment graphs can significantly improve the recall and F-score of knowledge graphs.

1 Introduction

Recently, Knowledge graphs are widely used in question answering and information querying. Building knowledge graphs from unstructured text is a crucial task in Natural Language Processing, which aims at extracting (subject, relation, object) triples such as (Google, buy, YouTube) to construct knowledge graphs.

Supervised methods mainly concentrate on classifying relational facts into pre-defined relation types (Mintz et al., 2009; Su et al., 2018). However, these methods require collecting and annotating labeled data, which is time-consuming and human-intensive for practical applications. Open-domain knowledge graphs can be constructed from corpora by applying unsupervised open information extraction methods. Open information extraction methods are mainly based on semantic parsing, which is fast to deal with large corpora but lacks inference ability. If the corresponding triple was not found in the text by parsers, the relation edges will be missing in the knowledge graph. Petroni et al. (2019) prove that language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) may also be storing relational knowledge present in the training corpus too. These models could work as knowledge graphs to answer queries structured as "fill-in-the-blank" cloze statements.

In this paper, we propose a method for enhancing open-domain knowledge graphs by using entailment graphs as a plug-in. Typed entailment graphs (Berant et al., 2010) are used to store the entailment between predicates, which could be an excellent fit for alleviating the above problem. We construct knowledge graphs from large corpora in low time-consuming by semantic analysis. We propose two ways to enhance the knowledge graph: enhancing the knowledge graph offline and using the entailment graph dynamically. The experiments show entailment graphs can improve F-score significantly in question answering tasks. Compared with the state-of-art language models, the enhanced knowledge graph also achieves higher F-score in context-free situation. In addition, we analyze the effects of entailment graphs based on different corpora and score functions.

2 Related Work

Knowledge is expressed as a collection of "facts", represented in the form (subject, relation, object) triples, where subject and object are entities and relations between those entities. Open-domain knowledge graphs aim at extracting these facts from large open-domain corpora.

In Petroni et al. (2019) work, language models could extract relational knowledge present in texts and perform well in cloze-style question answering. Language models encode the sentence between entities. They are optimized to either predict the next words in a sequence or some masked words anywhere in a given sequence. Ali et al. (2021) proposed a method for facts extraction based on BERT, using the BERT sentence-encoding algorithm on a corpus already annotated for named entities(NE). Petroni et al. (2020) find the context information could improve BERT's zero-shot cloze-style question-answering performance.

Another approach is using open-domain information extraction (Etzioni et al., 2011), which is based on semantic parsing. Harrington and Clark (2007) propose an efficient pipeline to extract facts by using a localized update algorithm. Each sentence will be transferred into a syntax structure and added in knowledge graphs incrementally. However, these knowledge graphs lack inference ability and the quality is limited by training corpus. Etzioni et al. (2011) use link prediction to add relation edges that are missing from the graph because the corresponding triple was not found in the text. For example, by semantic analysis, we extract a fact *(Google, buy, YouTube)* from the sentence *"Ten years ago this week, Google bought YouTube for 1.65 billion dollars."*. However, when we query *"Which company owns YouTube now?"*, the knowledge graph can't get the correct answer, because there is no sentence like *"Google owns YouTube"* in our training corpus. It will limit the practical value of the knowledge graph.

In this paper, we propose to use contextual entailment to solve the problem. We construct typed entailment graphs to enhance the knowledge graph in both dynamical and offline ways. After being combined with the entailment graphs, the experiments show that our methods significantly improve the F-score in cloze-style question-answering tasks. Compared with the pre-trained language models, our enhanced knowledge graph performs better than the context-free BERT.

3 Method

In this section, we present our method for building knowledge graphs and entailment graphs on text. The method contains two parts: building a knowledge graph in 3.1 and 3.2 is building entailment graphs to enhance our knowledge graph. There are two methods for constructing knowledge graphs and enhancing knowledge graphs: offline method and dynamic method.

3.1 Knowledge Graph

3.1.1 Build knowledge graph offline:

We extract facts from sentences by semantic parsing. In order to improve semantic parsing precision, we use the Lee et al., 2018 proposed coreference resolution tools to filter sentences. After pre-processing the text, we use the Graph Parser (Reddy et al., 2014) to extract binary relations from documents. Graph parser converts sentences to semantic graphs using combinatory categorial grammar (CCG) (Clark and Curran, 2007) and subsequently grounds them to Freebase. We only extract the triples contain binary relations, the triples are represented in the form $< e_i, r, e_j >$, e_i, e_j means entity i,j and r means the relation name. With extracted facts, we build a directed graph as the knowledge graph. Our offline knowledge graph is trained on whole Wikipedia corpus. To construct the knowledge graph on large corpora in low time-consuming, we use Aidalight (Hoffart et al., 2011) tools to link the extracted mentions to named entities in Wikipedia, which have little ambiguity. Figure 1 shows an example of changes from raw sentences to knowledge. The knowledge graph

sentence: Diego, a miniature painter. He was born at Toledo in 1498.

extracted facts: <Diego, (bear.1, bear.in.2), Toledo>
<Diego, (is.1, is.2), painter>

knowledge in KG: <Diego de Arroyo, (bear.1, bear.in.2), Toledo>
<Diego de Arroyo, (is.1, is.2), painter>

Figure 1: Example of changes from raw sentence to knowledge.

is constituted by a set of facts. The nodes in the knowledge graph are labeled with entity names from the original document. The edges are labeled with extracted relation names. Due to the tokens having been transferred into entity names in the Wikipedia namespace. It means querying the entity and relations in the knowledge graph just needs to search the keys in Hash maps of entities. Not like ASKNet (Harrington and Clark, 2007) calculating nouns similarity matrix, our method is more efficient and accurate. When adding a new fact *(*e_i*, r,* e_j*)*, we only query the knowledge graph and see if the knowledge graph already has the facts *(*e_i*, r,* e_j*)*. If the entities or relations are already in the knowledge graph, we just need to update the frequency of the relation.

3.1.2 Build knowledge graph dynamically:

We also create a pipeline to build the knowledge graph dynamically. We trained a documents retriever based on the DrQA (Chen et al., 2017). Each query, like *(Google, buy, [MASK])*, can be transferred into a natural sentence to retrieve related documents. We build the knowledge graph on the retrieved documents dynamically. Not like the large pre-trained offline knowledge graph, it doesn't need large memories and runs faster for a query.

3.2 Build Typed Entailment Graph

3.2.1 Typed entailment graph:

Textual entailment between predicates is common in natural language. The typed entailment graphs aim at learning entailment rules between typed predicates. For example, the sentence *"Google bought YouTube for 1.65 billion dollars."* entails *"Google owns YouTube"*. With arguments of entity-pair types *(Company, Company)*, the predicate *"buy"* entails "own".

Entailment needs to calculate a directed similarity score function between the typed predicates based on the distributional inclusion hypothesis, which states that a predicate p entails another predicate q if in any context that p can be used, q can be used in the same place (Geffet and Dagan, 2005). Fig 2b shows an example of a simple typed entailment graph. An entailment graph defines a score function between the typed predicates. The similarity score function is used to describe how likely a predicate entails other predicates. The local score function is used to compute local distributional similarity scores to learn entailments between predicates with typed arguments. Previous work compute local similarity scores (both symmetric and directional) between typed predicates: Weeds similarities (Weeds and Weir, 2003), Lin similarities (Lin et al., 2016) and Balanced Inclusion (BInc) similarities (Szpektor and Dagan, 2008). Hosseini et al. (2021) propose a model, named Contextualized and Non-Contextualized Embeddings (CNCE). The model uses contextual link prediction to calculate a new relation entailment score, which could be used to produce high-quality entailment graphs.

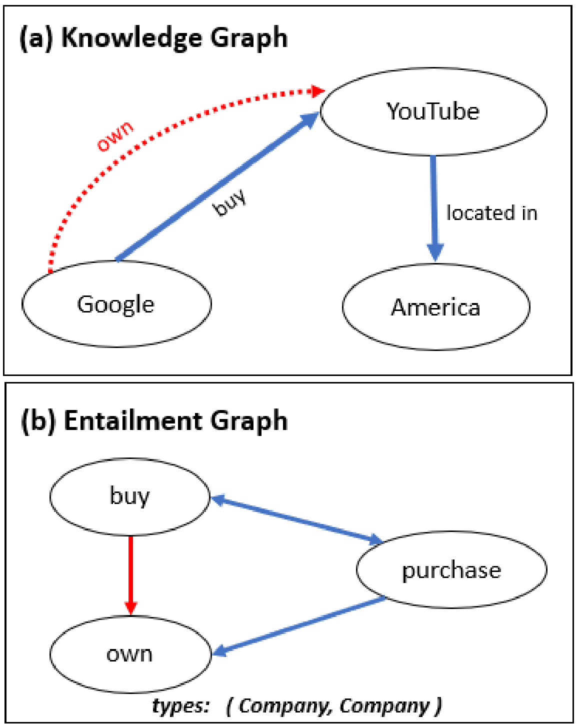

Figure 2: (a) shows an example of knowledge graphs, which relation *"own"* is missing. The missing edge could be added by using the entailment *"buy" entails "own"*. (b) shows an examples of typed entailment graphs for arguments of types *(company, company)*.

Based on the local score, Javad Hosseini et al. (2018) propose a novel method to build high-quality entailment graphs, the global score, based on new soft constraints that consider both the structures across typed entailment graphs and inside each graph. This method performs well in large corpora, we use the global score function to build entailment graphs.

3.2.2 Enhance the knowledge graphs:

The entailment graph is used to predict missing relations in the knowledge graph. We add latent facts in the knowledge graph for querying. The knowledge graph is enhanced in two ways: enhancing the knowledge graph offline or using the entailment graph dynamically. In the offline method, if the predicate p entails the predicate q, it means in any fact (e_i, p, e_j) in the knowledge graph, q can be used in the same place. We will add a new fact (e_i, q, e_j) in the knowledge graph. Fig 2 shows an example, the predicate *"buy"* entails *"own"* with arguments of types *(company, company)* in our entailment graph. There is a fact *(Google, buy, YouTube)* stored in our raw knowledge graph, we could add fact *(Google, own, YouTube)* in our knowledge graph for querying. If the query is *"Which company owns YouTube?"*, it will return a correct answer, even if there is no fact *(Google, own, YouTube)* in the raw knowledge graph.

We also use the entailment graph dynamically during querying. When querying, we run both knowledge graph and typed entailment graph at the same time. If query *(entity, p, [target entity])* returns *"Not found"* in the knowledge graph (p means a predicate), we query the typed entailment graph and get a candidate list of predicates $\mathbf{q} = [q_1,$

q_2, ... , q_n], q_i ($1 \leq i \leq n$) means a predicate in candidate list **q**. $\forall q_i \in$ **q**, q_i entails **p**. We rank the list **q** by global score. Then we choose the predicate in $\boldsymbol{q}$ to replace predicate p for the query. The new query will be *(entity, q_i, [target entity])*. In our experiments, we found the entailment graph can improve F-score in question answering tasks.

4 Experiment

In this section, we present our experimental settings for comparing the baselines in the LAMA probe dataset (Petroni et al., 2019). We present the datasets in 4.1 and the baselines in 4.2. The details of the evaluation settings are presented in 4.3.

4.1 Data

We use the Wikipedia corpus, NewsSpike corpus, and NewsCrawl corpus as the training dataset to build the knowledge graph and entailment graphs. The LAMA probe dataset requires the language model to answer cloze-style questions about relational facts under a zero-shot setting. We focus on Google-RE and T-REx in LAMA for evaluation, which are aimed at factual knowledge.

4.1.1 training data

Wikipedia: We use the Wikipedia corpus to train the knowledge graph and entailment graphs. To include all wiki entities in the training set, we use the whole Wikipedia corpus to train the knowledge graph. The Wikipedia data is extracted from Wikipedia articles and the corpus contains 5.4M documents. We extract about 158M binary relations using the semantic parser of (Reddy et al., 2014), Graph Parser. For the same type of entities, the documents of wikipedia describs similar events. For example, most documents of type *person* in Wikipedia contain a description of the person's birth event, it means the corpus contains many different descriptions of the same event. It works for training entailment graphs. In our experiment, we find that with the size of the entailment graph growth, the rate of increase in retrieval F-score will gradually decrease. Considering the millions of documents in Wikipedia, it will take up a lot of impractical calculations for training entailment graphs on the whole Wikipedia corpus. Our entailment graph is trained on 33 % Wikipedia corpus.

NewsSpike: To analyze the effects of entailment graphs from different corpus sources and make a fair comparison. We use the multiple-source NewsSpike (Zhang and Weld, 2013) corpus to train the entailment graph. In NewsSpike, the corpus was deliberately built to include different articles from different sources describing identical news events. It contains RSS news and linked news to full stories collected through a Web search. The corpus contains 550K articles (20M sentences). We extracted 29M binary relations using the same semantic parser, Graph Parser. We use the NewsSpike corpus to train an entailment graph and compare the results with entailment graphs based on the Wikipedia corpus.

NewsCrawl: The NewsCrawl (Bojar et al., 2017) extracts newspaper articles from multiple sources, in order to obtain separately authored descriptions of the same events. The NewsCrawl is much larger than the NewsSpike. It contains 160M sentences.

4.1.2 test data

Google-RE: Google-RE corpus is used in the LAMA probe (Petroni et al., 2019) for evaluation. The Google-RE corpus is manually extracted from Wikipedia. It covers five relations and three of them are used in the LAMA probe. To fair compare the results of LAMA (Petroni et al., 2019, 2020), we use relations *"Place-of-Birth"*, *"Date-of-Birth"*, *"Place-of-Death"* in our experiment.

In the LAMA probe, the Google-RE contains 5.5K facts and the relation template is defined manually. e.g., "[S] was born in [O]" for "Place-of-Birth". Each fact in the Google-RE is, by design, manually aligned to short pieces of Wikipedia text supporting it. We run the experiment on it with the three relations above.

T-REx: The T-REx knowledge source is a subset of Wikidata triples. It is derived from the T-REx dataset (Elsahar et al., 2018) and is much larger than Google-RE with a broader set of relations. There are 41 relations and subsample at most 1000 facts per relation in our test dataset. In contrast to the Google-RE knowledge source, which is defined manually, the facts in T-REx were automatically aligned to Wikipedia.

YAGO3-10: In order to compare with the language models without context, we add the test set of YAGO3-10 (Rebele et al., 2016) in our experiments. YAGO3-10 is a large semantic knowledge base, derived from Wikipedia, WordNet, GeoNames, and other data sources. YAGO3-10 knows more than 123K entities and contains 37 relations about these entities. To language models, we construct the natural language by facts in YAGO3-10

Corpus	Relation	Statistics	
		Facts	Rel
Google-RE	Place-of-Birth	2937	1
	Date-of-Birth	1852	1
	Place-of-Death	796	1
	Total	5527	3
T-REx	Total	31051	41
YAGO3-10	Total	5000	37

Table 1: Statistics for the test data

and mask the object tokens. For example, when querying the fact *(Kobe Bryant, plays.for, Los Angeles Lakers)*, it will be transformed into sentence *"Kobe Bryant plays for [MASK]"* for language models. The statistics for the test data of our experiment are listed in Table 1.

4.2 Baseline

LAMA (Petroni et al., 2019) defines a cloze-style question answering task that answer queries structured as "fill-in-the-blank" cloze statements. LAMA presents an in-depth analysis of the relational knowledge already present (without fine-tuning) in a wide range of state-of-art pre-trained language models. To compare with the results in LAMA, we consider the following baselines.

Freq: For a subject and relation pair, this baseline ranks words based on how frequently they appear as an entity of the object argument for the given relation in the test data.

RE: For the relation-based knowledge sources, we consider the pre-trained Information Extraction (IE) model of Sorokin and Gurevych (2017). This model was trained on a subcorpus of Wikipedia annotated with Wikidata relations. It extracts relation triples from a given sentence using an LSTM based encoder and an attention mechanism. Based on the alignment information from the knowledge sources, we provide the relation extractor to constructs a knowledge graph of triples. At test time, we query this graph by finding the subject entity and then rank all entities in the correct relation based on the confidence scores returned by RE.

DrQA: Chen et al. (2017) introduce DrQA, a popular system for open-domain question answering. There are two parts in the DrQA pipeline, the retriever and the reader. The retriever finds top K related documents and the neural reading comprehension model then extracts answers. The reader is a machine comprehension component, which is trained with supervision on SQuAD (Rajpurkar et al., 2016).

BERT-based models: In LAMA, the bidirectional language model outperforms in query. BERT proposes to sample positions in the input sequence randomly and to learn to fill the word at the masked position. They employ a Transformer architecture and train it on the BookCorpus (Zhu et al., 2015) as well as a crawl of English Wikipedia. Besides the largest BERT model (BERT-large) from (Devlin et al., 2019), Petroni et al. (2020) propose using retrieved paragraphs to predict the masked words in cloze-style question answering. It uses the retriever of DrQA to search related documents from the Wikipedia, and concat the retrieved context with queries to predict masked tokens. Compared with BERT-large, this method contains more contextual information when predicting masked tokens. It dramatically improves BERT's zero-shot cloze-style question-answering performance. RoBERTa (Liu et al., 2019) propose a replication study of BERT pretraining that carefully measures the impact of many key hyperparameters and training data size. We take the three BERT-based methods as baselines.

4.3 Evaluation

4.3.1 Evaluation on Knowledge Graph

To build the knowledge graph, we extract binary relations with the Graph Parser from the Wikipedia corpus. AIDALight (Hoffart et al., 2011) is used as the entity linking tool in our experiment. We only take the binary relations with name entities that are linked to Wikipedia. The nodes in the knowledge graph are labeled with entity names. The edges are labeled with relation names, which are extracted predicates from sentences. There are 15M nodes and 138M edges in our knowledge graph.

In our evaluation, we use the Graph Parser to extract triples from the cloze-style questions. For example, an instance in evaluation corpus is *"Steve Jobs was born in [MASK]"*, we extract *(Steve Jobs, bear.in, [Target])* and query it in our knowledge graph. Our knowledge graph is trained on the whole Wikipedia corpus. We also construct a knowledge graph on retrieved documents to compare the results of dynamical methods.

4.3.2 Evaluation on Entailment Graph

In order to compare different entailment graphs, we construct distinct entailment graphs based on different corpora and methods in our evaluation.

The entailment graphs built by Javad Hosseini et al. (2018) method contain BInc scores. Another entailment graphs are constructed by CNCE (Hosseini et al., 2021) model. Both methods contain local scores and global scores.

In the Google-RE corpus, there are three relations, *"Place-of-Birth"* (PoB), *"Date-of-Birth"* (DoB) and *"Place-of-Death"* (PoD), so we search for predicates in the entailment graph with two entity-pair types: *(person, location)* and *(person, time)*. In evaluation, if there is no fact $(e_i,\ p,\ e_j)$ in the knowledge graph, where *e* means entity and *p* means predicate, then we query the typed entailment graph and get a candidate predicates list $\boldsymbol{q}$. Every predicate in $\boldsymbol{q}$ entails predicate *p*. The $\boldsymbol{q}$ contains n predicates ranked by score. In order to compare the local score and the global score, we created two entailment graphs separately. We choose the predicate in $\boldsymbol{q}$ to replace *p* for querying. The new query will be $(e_i,\ q_k,\ e_j)$, 1 ≤k≤ n.

For example, when there is no fact *(Tim Cook, study.in, [MASK])* stored in our knowledge graph, we search the entailment graph. In entailment graph, the predicate *(accept.offer.from)* and predicate *(graduate.from)* entails *(study.in)*. We add *(accept.offer.from)* and predicate *(graduate.from)* in a candidate list and rank it by global/local scores. First, we replace the *(study.in)* by *(accept.offer.from)*, the query will be *(Tim Cook, accept.offer.from, [MASK])*. If it still has no facts in the knowledge graph, we query the knowledge graph with the next predicate, *(Tim Cook, graduate.from, [MASK])*, and it will return an answer *"Duke University"*.

We also consider types order when choosing the predicates. For example, The typed predicate *"die.at"* with types order *(person, location)* entails *"dead.found"* with types order *(location, person)*, they have different orders so the *"dead.found"* will not be added candidate for querying.

5 Results

We summarize the main result in Table 2, which shows the F-score for different models across the set of corpora. In the remainder of this section, we discuss the results and analyze errors.

In Table 2, the knowledge graphs built by semantic parsing outperform the BERT-large and RoBERTa. The language models based on retrieved paragraph, DrQA performs better in Google-RE. The language models on retrieved contexts (DrQA and BERT-ret) outperforms in Google-RE. After adding the entailment graph, the F-score of enhanced knowledge graphs improve significantly. In T-REx dataset, the enhanced knowledge graph performs better than DrQA, reaching performance that is on par with BERT-ret. Compared with the knowledge graph on the whole Wikipedia corpus, the dynamical knowledge graph trained on retrieved documents occupies less memory with little F-score down.

To analyze the effect of contexts, we compare the results on YAGO3-10 in Table 3. Not like Google-RE and T-REx contain contextual tokens in sentences, the YAGO3-10 only contains facts from a knowledge base. We calculate the hit@5 of different methods and find our methods perform best. In cloze-style question answering, the language models can get contextual information of the sentence. If query facts, the results of BERT-based models will be limited by lacking contexts. The experiments show our methods outperform in this context-free situation.

In Table 4, We compare the entailment graphs trained on different corpora. The results show that the knowledge graphs outperform in mean average precision at one but the recall is low. The entailment graphs can be used to predict latent facts and improve the recall and F-score. The entailment graphs on NewsSpike outperform other corpora in F-score. Compared with the Wikipedia corpus, the predicates in the NewsSpike have stronger relevance. The articles in the NewsSpike corpus mainly describe news events by multiple authors so it is efficient to extract entailment between predicates. It may explain why EG_{ns} performs better. It also means we don't need to train knowledge graphs and entailment graphs on the same corpus, we could use the pre-trained entailment graphs as a plug-in. We also construct the entailment graphs by different score functions (BInc and CNCE), the results are shown in appendix B. Javad Hosseini et al. (2018) propose global scores and local scores for entailment graphs. To compare the entailment graph based on different scores, we train the entailment graphs on both local and global scores. Table 6 in appendix B shows the result of different scores.

To analyze the recall changes with different entailment graph sizes, we run experiments on Google-RE with different sizes of entailment graphs. The result is shown in Figure 3. The recall increase with the entailment graph size. When

Corpus	Relation	Freq	KB RE	LM BERT-large	LM RoBERTa	LM on retrieved doc DrQA	LM on retrieved doc BERT-ret	Our Method KG	Our Method KG_{ret}	Our Method KG + EG_{ns}	Our Method KG_{ret} + EG_{ns}
Google-RE	birth-place	4.6	13.8	16.1	-	**48.6**	43.5	19.9	20.1	27.7	31.5
	birth-date	1.9	1.9	1.4	-	42.9	**43.1**	7.7	7.8	8.5	8.5
	death-place	6.8	7.2	14.0	-	**38.4**	35.8	14.6	13.5	26.0	24.0
	Total	4.4	7.6	10.5	4.8	**43.3**	40.8	14.0	13.8	20.7	21.3
T-REx	Total	22.0	33.8	32.3	27.1	25.8	**43.1**	33.3	32.0	39.0	37.7

Table 2: F-score for a frequency basline, a information extraction with entity linking (RE), BERT-large, RoBERTa. the BERT-ret is BERT based on retrieved context. The knowledge graph (KG) is built on whole wikipedia corpus and KG_{ret} means KG trained on retrieved documents. The entailment graph (EG_{ns}) is trained on NewsSpike.

	BERT-large	BERT-ret	DrQA	KG	KG+EG
Hit @ 5	38.0	59.5	45.3	41.2	**62.7**

Table 3: Hit@5 in YAGO3-10 dataset.

	P@1	R	F
KG	**58.8**	8.5	14.0
KG+EG_{wiki}	43.8	12.3	17.4
KG+EG_{ns}	41.7	15.0	**20.7**
KG+EG_{nc}	42.6	14.6	19.6

Table 4: Results of different entailment graphs. This table shows the mean average precision at one (P@1), recall and F-score of Google-RE. The result shows the average per number of relations in Google-RE. In this table, EG means entailment graph and the subscripts *wk*, *ns* and *nc* means the entailment graphs are trained on Wikipedia, NewsSpike and NewsCrawl.

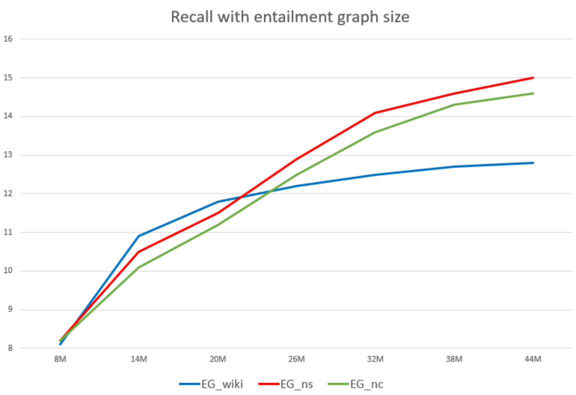

Figure 3: The recall changed with entailment graph size

the training corpus is small, the entailment graph based on Wikipedia corpus performs better. But with the entailment graph size gradually increasing, the recall of entailment graphs based on NewsSpike corpus will perform better. The Google-RE is extracted from Wikipedia, the EG_{wiki} contains more predicates for events in Wikipedia than EG_{ns}. With the size of entailment graphs increasing, the EG_{ns} will perform better because it contains more multiple documents for the same event.

The errors analysis is shown in Table 5. We manually analyze 100 queries. About 40% of them are caused by graph parser and are cases where it returns wrong relations from text. Most of them are caused by non-standard sentences in Wikipedia documents. For example, *"Normand MacLeod (c. 1731 – 1796) was a British army officer, merchant, and official of the British Indian Department."*, the parser can't extract fact *(Normand MacLeod, bear.in, 1731)* from the sentence because the parser can't analyze the "(c. 1731 – 1796)".

31% of the errors in the knowledge graph are due to entity linking in evaluation, it may be caused by ambiguity in Google-RE. For example, a sentence in Googe-RE is *"Jason then continued to Sparta, where he died and was buried"* and the fact in Google-RE is *(Jason, death-place, Sparta)*. But in evaluation, the *"Jason"* is linked to *"Jason Hu"*, who is a modern politician.

About 11% of errors are caused by the mismatch between train corpus and test corpus, e.g. fact in Google-RE is *(Arthur Kinnaird, bear.in, Kensington)*, the output of the object from KG is *"London"*, both of them should be correct.

The errors in the entailment graph are mainly caused by the ambiguity of some high-frequency predicates. For example, predicate *(bear.2,bear.in.2)* entails predicate *(from.1,from.2)*. These predicates, like *(from.1,from.2)*, are common in sentences. If the relation of the query contains these predicates, the knowledge graph

Error-Type	Example	Describe	Rate
semantic parsing error	*"Normand MacLeod (c. 1731 – 1796) was a British army officer, merchant, and official of the British Indian Department."*	parser extracts wrong relations because of not standard sentences structure	40%
Entity linking errors	*"Jason"* is mapped to entity *"Jason Hu"*, which should be *"Jason Mraz"*	It is caused by aidalight wrong outputs	31%
Mismatch	Truth in test dataset: Kensington Knowledge graph output: London	It is casued by mismatch between training dataset and test dataset	11%

Table 5: The errors in knowledge graph

will return wrong answers easily. When we use the *(from.1,from.2)* for querying in the knowledge graph, it will return false results because the predicate has too many meanings. e.g. In the sentence *"Shane Doan is from Arizona"* may mean *"Shane comes from Arizona"*, not the birth-place. In our experiment, some entailment graphs errors are caused by the content of documents. For example, there are many documents in Wikipedia like *"Steve Jobs was born on February 24, 1955, in California, ..., Jobs died at his Palo Alto, California home around 3 p.m."*. From these sentences, we may extract facts like *(Steve Jobs, (bear.1,bear.in.2), California)* and *(Steve Jobs, (die.1,die.in.2), California)*. These predicates link the same entities. It is likely to incorrectly give the entailment relationship between the two predicates.

6 Conclusion

This paper has demonstrated a method to construct a large knowledge graph by semantic network and use the entailment graph to enhance the knowledge graph. Parsing technology can extract relations from unstructured text to build open-domain knowledge graphs in an efficient method. Traditional knowledge graph based on semantic parsing is limited, they can not infer a result if there is no matching relations in the training corpus. We use the entailment graph to extract more latent relations between entities. The entailment graphs dramatically improve the recall and F-score in cloze-style question answering and outperform the BERT-large. When a query lacks context information, the enhanced knowledge graphs perform better than the methods based on retrieved documents, like BERT-ret and DrQA. In future work, we plan to test our method on more corpus sources, such as documents in tweets. The Wikipedia corpus has more actual events but some events in other sources are only discussed or not happened. It will be challenging.

References

Manzoor Ali, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. 2021. Unsupervised relation extraction using sentence encoding. In *ESWC2021 Poster and Demo Track*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2010. Global learning of focused entailment graphs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1220–1229.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. 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)*.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: the second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 3–10.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the*

Association for Computational Linguistics (ACL'05), pages 107–114.

Brian Harrington and Stephen Clark. 2007. Asknet: Automated semantic knowledge network. In *AAAI*, pages 1862–1863.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792.

Mohammad Javad Hosseini, Shay B Cohen, Mark Johnson, and Mark Steedman. 2021. Open-domain contextual link prediction and its complementarity with entailment graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2790–2802.

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R Holt, Shay B Cohen, Mark Johnson, and Mark Steedman. 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133.

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. Yago: a multilingual knowledge base from wikipedia, wordnet, and geonames. In *The 15th International Semantic Web Conferenece*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1784–1789.

Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gür, Huan Sun, and Xifeng Yan. 2018. Global relation embedding for relation 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 820–830.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 849–856, Manchester, UK. Coling 2008 Organizing Committee.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 81–88.

Congle Zhang and Daniel S Weld. 2013. Harvesting parallel news streams to generate paraphrases of event relations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1776–1786.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

	KG_{ret}	$KG + EG_{wk\_binc}$		$KG + EG_{ns\_binc}$		$KG + EG_{nc\_binc}$	
		local	global	local	global	local	global
P@1	**58.8**	43.2	43.8	42.0	41.7	41.7	42.6
R	8.5	12.3	12.3	13.7	**15.0**	14.3	14.6
F	14.0	16.9	17.4	18.0	**20.7**	19.1	19.6

Table 6: knowledge graph combined with different entailment graphs. global means the entailment graph is based on global BInc score, local means the entaiment graph with local BInc score.

A Results of different Entailment Graphs

To compare the entailment graph based on different scores functions, we train the entailment graphs on both local and global scores. Table 6 shows the result of different scores. We aslo compare the entailment graphs based on different score functions (BInc and CNCE), the result is shown in Table 7. Our knowledge graphs outperform BERT-large, BERT-ret, and DrQA in mean average precision at one. The knowledge graph on retrieved documents reach higher precision than the knowledge graph on the whole wiki corpus. However, the recall of knowledge graph is low. The results show entailment graphs can predict latent facts, these latent facts enhance the raw knowledge graph by adding missed edges. The entailment graphs dramatically improve the recall and F-score. The entailment graphs on CCNE perform better than the BInc score. The entailment graphs on NewsSpike outperform other corpora in recall and F-score.

B Samples of predicates in entailment graph

When query with *"Place-of-Birth"*, *"Date-of-Birth"*, *"Place-of-Death"*, we choose *(bear.2,bear.in.2)* and *(die.1,die.in.2)* as the target predicates. When ranked by the BInc score, the top five predicates in the entailment graphs are shown in Table 8.

		P@1	R	F
KG	KG	58.8	8.5	14.0
	KG+EG$_{wk_binc}$	43.8	12.3	17.4
	KG+EG$_{ns_binc}$	41.7	15.0	20.7
	KG+EG$_{ns_cnce}$	40.7	**16.2**	21.0
	KG+EG$_{nc_binc}$	42.6	14.6	19.6
	KG+EG$_{nc_cnce}$	44.9	15.1	20.7
KG$_{ret}$	KG$_{ret}$	**68.3**	7.9	13.8
	KG$_{ret}$+EG$_{wk_binc}$	46.2	13.4	17.1
	KG$_{ret}$+EG$_{ns_binc}$	53.0	15.3	21.3
	KG$_{ret}$+EG$_{ns_cnce}$	56.0	13.1	21.6
	KG$_{ret}$+EG$_{nc_binc}$	47.2	11.9	16.2
	KG$_{ret}$+EG$_{nc_cnce}$	53.7	11.6	17.2
LM	BERT-large	10.5	-	10.5
	BERT-ret	40.8	-	40.8
	DrQA	43.3	-	**43.3**

Table 7: Results of different entailment graphs on Google-RE. This table shows the mean average precision at one (P@1), recall and F-score of Google-RE. The result shows the average per number of relations in Google-RE. In this table, EG means entailment graph and the subscripts *wk*, *ns* and *nc* means the entailment graphs are trained on Wikipedia, NewsSpike amd NewsCrawl corpus. Subscripts *binc* amd *cnce* means the entailment graphs are trained on BInc score and CNCE.

Target Predicates	Types	Top 5 predicate in EG, ranked by global score	Top 5 predicate in EG, ranked by local score
(bear.2,bear.in.2)	**person-location**	(bear.2,bear.in.2) (bear.1,bear.in.2) (in.1,in.2) (be.1,be.in.2) (live.1,live.in.2)	(bear.2,bear.in.2) (native.of.1,native.of.2) (grow.1,grow.in.2) (in.1,in.2) (raise.2,raise.in.2)
(bear.2,bear.in.2)	**person-time**	(bear.2,bear.in.2) (name.1,name.in.2) (address.1,address.in.2) (have.1,have.in.2) (in.1,in.2)	(bear.2,bear.in.2) (give.in.2,give.to.2) (in.1,in.2) (be.1,be.in.2) (live.1,live.in.2)
(die.1,die.in.2)	**person-location**	(die.1,die.in.2) (die.1,die.at.home.in.2) (dead.found.1,dead.in.2) (die.1,die.at.2) (dead.1,dead.in.2)	(die.1,die.in.2) (kill.2,kill.in.2) (in.1,in.2) (dead.1,dead.in.2) (dead.found.1,dead.in.2)

Table 8: Top 5 predicates in entailment graphs