Neurocomputing 613 (2025) 128726

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

SF-GPT: A training-free method to enhance capabilities for knowledge graph construction in LLMs

Lizhuang Sun, Peng Zhang *, Fang Gao, Yuan An, Zhixing Li, Yuanwei Zhao

Chang Guang Satellite Technology Co. Ltd., Changchun, 130102, Jilin, China

ARTICLE INFO

Communicated by M. Gallo

Keywords:
Knowledge graph
Triple extraction
Large language model
Knowledge fusion

ABSTRACT

Knowledge graphs (KGs) are constructed by extracting knowledge triples from text and fusing knowledge, enhancing information retrieval efficiency. Current methods for knowledge triple extraction include "Pretrain and Fine-tuning" and Large Language Models (LLMs). The former shifts effort from manual extraction to dataset annotation and suffers from performance degradation with different test and training set distributions. LLMs-based methods face errors and incompleteness in extraction. We introduce SF-GPT, a training-free method to address these issues. Firstly, we propose the Entity Extraction Filter (EEF) module to filter triple generation results, addressing evaluation and cleansing challenges. Secondly, we introduce a training-free Entity Alignment Module based on Entity Alias Generation (EAG), tackling semantic richness and interpretability issues in LLM-based knowledge fusion. Finally, our Self-Fusion Subgraph strategy uses multi-response self-fusion and a common entity list to filter triple results, reducing noise from LLMs' multi-responses. In experiments, SF-GPT showed a 55.5% increase in recall and a 32.6% increase in F1 score on the BDNC dataset compared to the UniRel model trained on the NYT dataset and achieved a 5% improvement in F1 score compared to GPT-4+EEF baseline on the WebNLG dataset in the case of a fusion round of three. SF-GPT offers a promising way to extract knowledge from unstructured information.

1. Introduction

Knowledge Graphs (KGs) offer an efficient means of structuring knowledge, primarily by depicting the relationships between entities [1,2]. These entities can be individuals, places, and events, and their interconnecting relationships encompass actions, affiliations, geographies. [3,4]. Knowledge graphs use a graph structure to present intricate relationships, providing a clear and structured view of the intricate connections between complex entities [5]. Due to their outstanding interpretability and capability to integrate heterogeneous data sources, knowledge graphs have been widely applied in sectors such as healthcare [6,7], science [8–10], finance [11,12], and industry [13,14]. Nevertheless, inherent challenges like incomplete and outdated knowledge and the complexity of construction have shifted research focus towards the automation of KG construction [15–17].

Against this background, triple extraction is the linchpin of KG construction. Its objective is to accurately identify and extract entities and their relationships from natural language text. Researchers have typically relied on manually created rules or templates to perform this task, demanding extensive human effort and offering limited generalization capabilities [18]. With the rise of deep learning in NLP, neural network models like RNNs and LSTMs have also been employed for triple extraction [19,20]. However, these early models needed help understanding the implied semantics of the text. For instance, the implicit relationship between "Beijing, capital, China" and "He lives in Beijing, China" was elusive for them.

In 2018, the introduction of Google's BERT model marked a paradigm shift towards the "Pre-training and Fine-tuning" models [21]. Infusing external knowledge from large-scale unsupervised pre-training led to significant triple extraction performance improvements. Models like TPLinker [22] and UniRel [23] quickly achieved F1 scores over 90% on open-source datasets like NYT [24] and WebNLG [25], leading many to mistakenly believe that the challenge of triple extraction had been thoroughly addressed, with some researchers even claiming the results were "on par with humans".

However, was this the case? As illustrated in Fig. 1, using the best-performing UniRel model as an example, the observed outcomes were not due to word-level triple extraction. Instead, the model learned specific "sentence-level expressions" to determine the existence of particular relationships. Such "sentence-level expressions" are also invisible to humans, making mere sample collection challenging and pointless. For these "Pre-training and Fine-tuning" models reliant on vast annotated data and necessitating rich "sentence-level expressions", there is no

* Corresponding author.
E-mail address: zp920306a@163.com (P. Zhang).

https://doi.org/10.1016/j.neucom.2024.128726
Received 6 February 2024; Received in revised form 22 July 2024; Accepted 12 October 2024
Available online 21 October 2024

Fig. 1. Using the UniRel model, which demonstrates the best performance among the Pre-training and Fine-tuning models, we conducted an extraction experiment. After extensively fine-tuning the model using the NYT dataset, triples were extracted for four specific relations: BirthPlace, DeathPlace, Capital, and Nationality.

guarantee that they can genuinely realize the task of triple extraction in the sentences they must predict. Increasing the number of samples in the supervised training set could lead to a dilemma: If we proceed this way, why not employ manual methods to extract triples to construct a knowledge graph? Thus, applying the proposed method to KG construction tasks is challenging. The issue was that these supervised fine-tuning models shifted the task of manual KG construction to the training set annotation without addressing the substantial manual intervention required in KG construction.

In 2023, Large Language Models (LLMs) have become the focal point of NLP [26]. Their question-answering and generalization capabilities meet the requirements of KG construction [27]. However, LLMs still face many challenges in triple extraction: (1) "Pretrain and Fine-tuning" models achieve high F1 scores but rely on extensive annotated training data, effectively shifting the burden to dataset labeling. These models suffer from performance reduction on test sets of different distributions, posing challenges in real-world applications. (2) LLMs show impressive generalization but lack a mechanism like BERT-like models for output confidence, making it difficult to assess the quality of extracted triples and clean up error results. (3) Current semantic entity alignment methods for knowledge fusion still predominantly require trained models, which constrains the amalgamation of heterogeneous knowledge sources and the enrichment of extracted information. (4) Single-instance triple extraction with LLMs needs to be completed, and while multi-response approaches can mitigate this, they introduce some noise with each iteration, cumulatively impacting knowledge completion quality.

Given the challenges above, the primary contributions of this research are:

(1) **Training-Free Knowledge Graph Triple Extraction:** We adopt Large Language Models (LLMs) as a baseline to extract knowledge triples and build knowledge graphs without relying on training datasets. The proposed method tackles the issues inherent in practical applications of LLMs to improve their real-world usability.

(2) **Entity Extraction Filter (EEF) for Error Triple Reduction:** This technique filters out incorrect knowledge triples by validating the presence of head and tail entities in the entity extraction list. We also enhance the extraction accuracy using secondary prompting, reduce redundant outputs, and apply relationship constraints.

(3) **Entity Alias Generation (EAG) for Entity Alignment:** We leverage LLMs to generate aliases for entities, which helps identify entities with similar origins. Due to its semantic richness, the proposed method is particularly effective and interpretable in unsupervised settings.

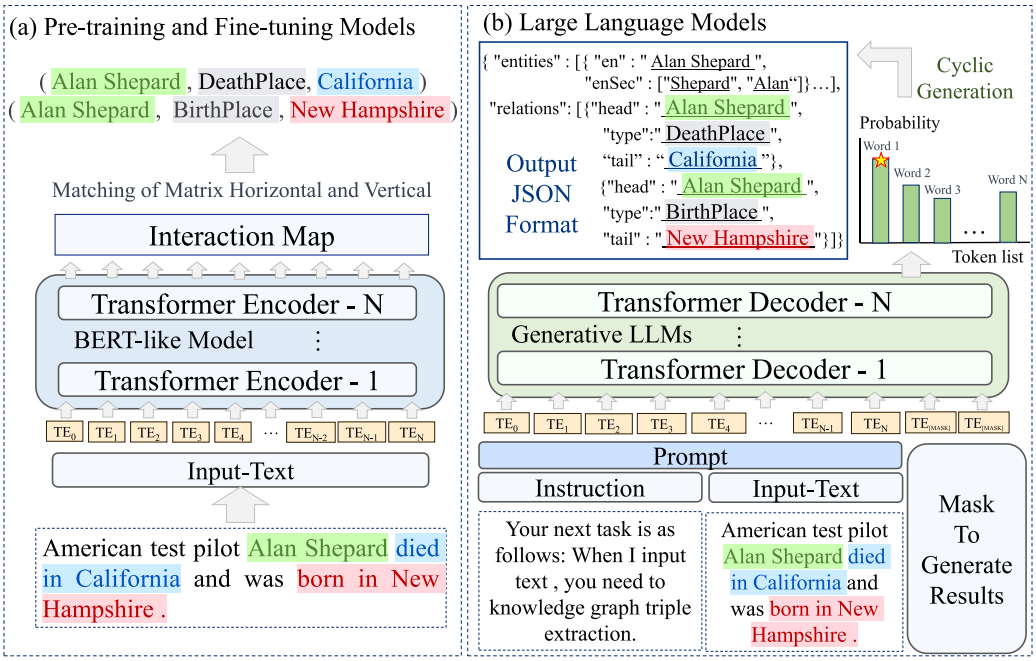


Fig. 2. (a) Pre-training and Fine-tuning models; (b) Large Language Models.

(4) **Self-Fusion Subgraph for Minimizing Triple Noise:** Our SF-GPT method minimizes noise from multiple responses in KG augmentation using generative self-fusion subgraphs. The proposed method merges knowledge from repeated LLM inputs while maintaining a consistent entity list, improving knowledge extraction quality.

SF-GPT achieves triple extraction without relying on extensive annotated datasets, significantly reducing the need for manual annotations. It offers invaluable tools for constructing knowledge graphs in specialized domains and real-time data-based KGs.

2. Related works

2.1. Natural Language Processing (NLP)

Natural Language Processing (NLP) attempts to give computers the ability to understand and generate natural language [28]. Knowledge Graphs (KG) offer a structured representation of knowledge, and this content is typically distilled from natural language text; thus, the advancement of KG research is closely related to NLP [29]. Traditional NLP techniques are predominantly rule-based and require manual creation and maintenance. Subsequently, statistical machine learning methods, with their data-driven edge and vast corpora, began to dominate the NLP realm. As we stepped into the 21st century, deep learning technologies like Word2Vec [30], RNN, LSTM [31], and Transformer [32] started showcasing their performance in sequential text processing, achieving remarkable results. Notably, the "Pre-training and Fine-tuning" models such as BERT [33] and GPT [34] set new records across various NLP tasks. By 2023, the advent of massive pre-trained models like ChatGPT and GPT-4 not only unified the tasks of understanding and generating in NLP but also attracted considerable academic interest due to their exceptional generalization capabilities and in-context learning attributes [35]. These developments have paved new research avenues for triple extraction in KG.

2.2. Knowledge graph triple extraction

In the early stages of triple extraction research, the task was primarily approached by first recognizing entities in natural language texts and then conducting relationship extraction to link the identified entities. Methods at that time were primarily grounded in traditional machine learning techniques and often overlooked the intricate semantic interplays between entities and relations. However, with the rapid rise of deep learning in NLP, neural network-driven joint triple extraction models have emerged, allowing direct triple extraction from texts without intricate feature engineering. With the success of RNNs, LSTMs, and Transformers in tasks like machine translation, researchers began integrating sequence generation techniques into triple extraction [19,20]. Although "Pre-training and Fine-tuning" models like BERT have achieved significant success in NLP tasks and have demonstrated human-comparable performance in triple extraction, they still rely heavily on high-quality annotated data [21]. The BERT-like models only shift the manual building process to data annotation, which does not alleviate the labor-intensive problem of KG construction. As shown in Fig. 2, knowledge graph triple extraction is based on "Pre-training and Fine-tuning" models and based on LLM models. With the development of large language models (LLMs), recent studies, such as ChatIE [36], have used LLMs for knowledge extraction tasks, achieving the triple extraction of knowledge triples without the need for annotations. However, knowledge extraction based on LLMs lacks confidence score output similar to BERT-like models, leading to erroneous and incomplete triple generation. This paper introduces the SF-GPT model, which attempts to enhance the performance of LLMs in knowledge extraction tasks in a training-free manner.

Our study introduces LLMs to address this challenge and proposes an SF-GPT triple extraction method for KGs based on LLMs and generative Self-Fusion subgraphs. This approach achieves triple extraction without the need for annotated training sets, significantly reducing manual effort, and matches the performance of "Pre-training and Fine-tuning" models on specific datasets.

3. Details of the SF-GPT model

3.1. Problem definition

The knowledge graph triple extraction task is a process in which an input text X, segmented by words as the smallest units, is fed into an algorithm model. The goal is to identify and extract triples as accurately

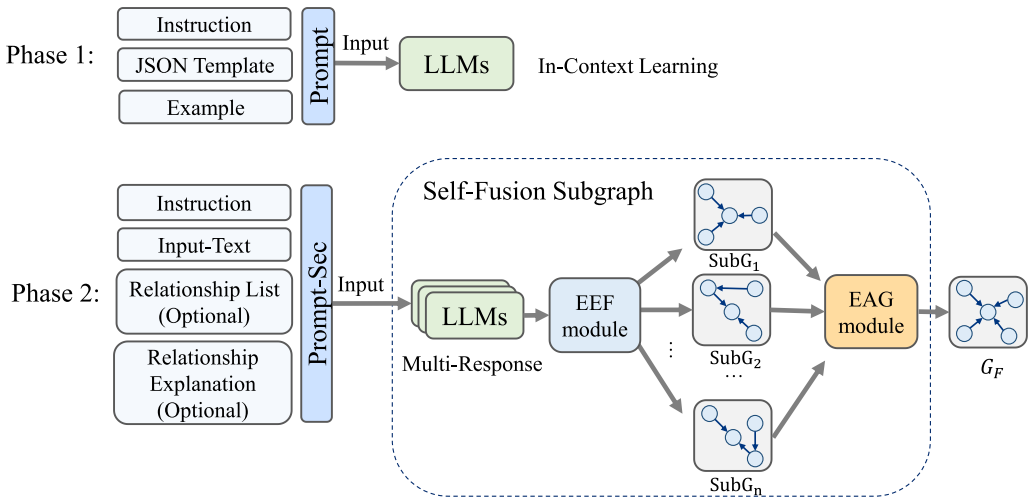


Fig. 3. Overview of SF-GPT Model.

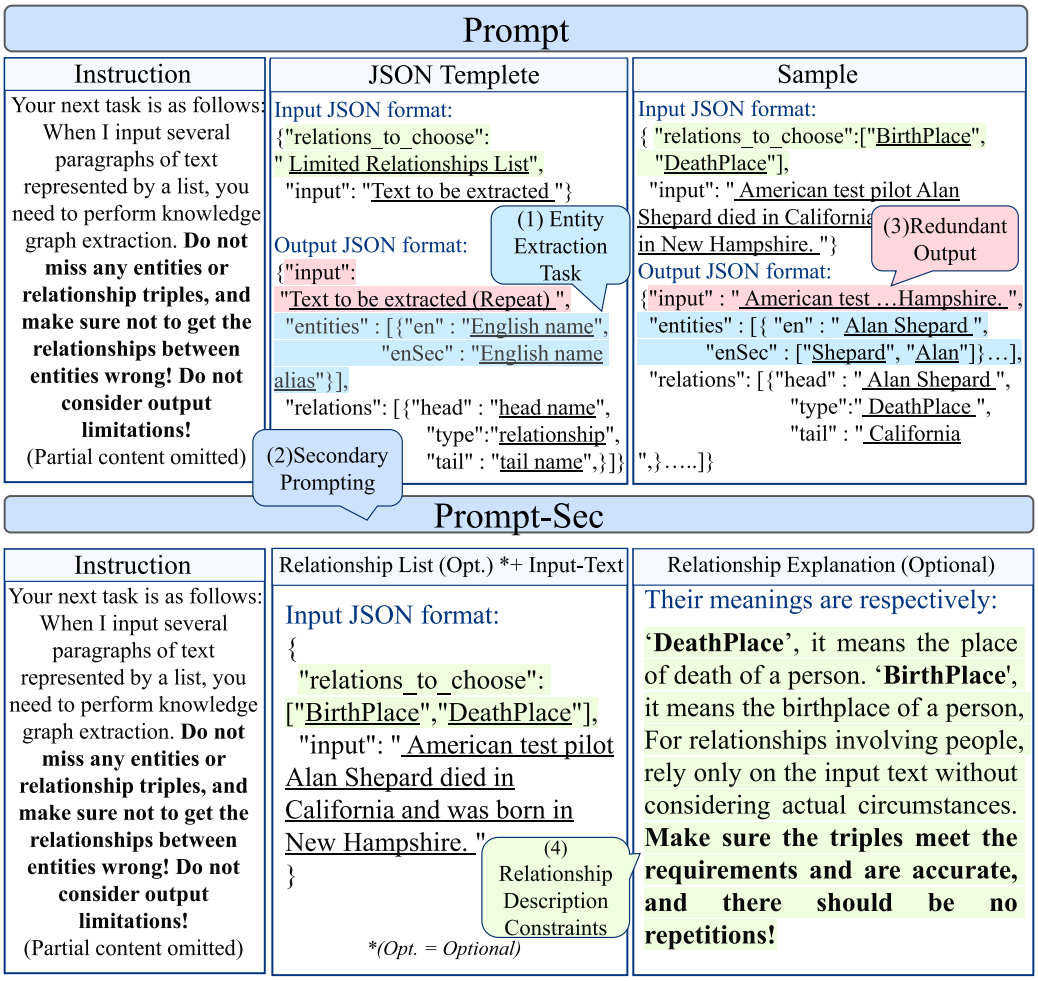


Fig. 4. Details of Prompt Engineering.

as possible, denoted as $T = \left[(h_l, r_l, t_l)\right]_{l=1}^{c}$. Here, L represents the total number of triples identified and extracted from a single input text. The symbols h, r, and t respectively stand for the head, relation, and tail entities, respectively. Both h_l and t_l originate from the entity set derived from the input text X, expressed as $E = \{e_1, e_2, \ldots, e_M\}$, where M denotes the maximum number of recognized entities. Each entity e_m in the set comprises continuous text substrings from the input text.

The exact nature of r_l is task-dependent and can be categorized into two types:

(1) **Constrained Relationship Extraction Task:** In this, r_l must be selected from a given set of predefined relations, $R = \{r_1, r_2, \ldots, r_K\}$. Here, K is the number of predefined relations.

(2) **Open Relationship Extraction Task:** In this case, r_l is selected from the relationship set extracted based on the input text X, denoted as $R' = \{r'_1, r'_2, \ldots, r'_O\}$, where O represents the maximum number of relationships that can be identified. The essential distinction between the two types is that while the former relies on predefined relations, the latter is dynamic and extracts relationships directly from the input text.

In this study, we extracted knowledge triples to construct knowledge graphs based on a Training-Free approach utilizing Large Language Models (LLMs), as depicted in Eq. (1), Eq. (2), and Eq. (3). This formulation represents the generation of results using the probabilistic properties of LLMs in a dialog context. Here, W represents the query vector matrix, while w_i denotes the embedding vector form of each word in the sentence. Through the deployment of an n-layer Transformer decoder $n * \mathbf{TransformerDecoder}()$, a self-attention vector matrix W' is generated. Subsequently, the softmax function $\mathbf{Softmax}()$ is applied to predict the probability matrix P of the next word w_{n+1}. The highest probability token was selected as the predicted content of the next word.

$$W = [w_1, w_2, \ldots, w_N] \tag{1}$$
$$W' = [w'_1, w'_2, \ldots, w'_N] = n * \mathbf{TransformerDecoder}(W) \tag{2}$$
$$P(w_{N+1}|w_1, w_2, \ldots, w_N) = \mathbf{Softmax}(W') \tag{3}$$

In Eq. (4), Eq. (5) and Eq. (6), we detail the current popular methodology of extracting knowledge triples and constructing them using $\mathbf{LLMs}()$, as inspired by methodologies discussed in studies such as ChatIE [36]. These Formulas expand upon the process outlined in Eq. (1), Eq. (2) and Eq. (3). Here, I represents the input command, S signifies the knowledge extraction sample, and $\mathbf{LLMs}'()$ denotes the language model that has been fine-tuned to extract knowledge triples. I' is the manually adjusted input command, which, in combination with the input text X, is fed into the fine-tuned model. This process extracts the entity extraction results E and relation extraction results R. These are then processed using the knowledge graph construction function $g()$ and the final knowledge graph G.

$$\mathbf{LLM}' \leftarrow \mathbf{LLM}(I + S) \tag{4}$$
$$E, R = \mathbf{LLM}'(I' + X) \tag{5}$$
$$G = g(E, R) \tag{6}$$

3.2. Overview of the SF-GPT model

As shown in Fig. 3, the SF-GPT model proposed in this paper is mainly composed of two phases.

(1) **In-Context Learning Phase:** This is the initial step. Instructions are given to generate triples for the knowledge graph, machine-readable JSON template code, and example samples for in-context learning. Together, these form the first Prompt. When this Prompt is input to a large language model, the model gains the capability to extract triples based on the input text X. Here, referencing Eq. (4), describes the process of generating a fine-tuned Large Language Model $\mathbf{LLMs}'()$ through the use of commands and samples. The refined model was specifically tailored to extract knowledge triples.

(2) **Knowledge Graph Triple Extraction Phase:** The extraction instruction is taken together with the input text for this phase. Here, we introduce a training-free method to enhance knowledge triple extraction based on subgraph fusion called SF-GPT. This approach significantly improves the performance of traditional knowledge triple extraction using Large Language Models (LLMs). The Subgraphs Fusion module primarily comprises two modules: the Entity Extraction Filter (EEF) and Entity Alias Generation (EAG); the specific design details are elaborated in Sections 3.3, 3.4, and 3.5.

In Eq. (7), we set several iterations i, representing the generation of multiple entities E and relations R using the fine-tuned Large Language Model $\mathbf{LLMs}'()$, ensuring that the answers do not reference each other. Eq. (8) focuses on the ith iteration entities $E_{(i)}$ and relations $R_{(i)}$. After processing through the Entity Extraction Filter (EEF) module $\mathbf{EEF}()$, the entity results are used to filter and cleanse the relation generation results, yielding entities $E'_{(i)}$ and relations $R'_{(i)}$. Eq. (9), involving the Entity Alias Generation (EAG) module $\mathbf{EAG}()$, fuses the results from M iterations to produce the final entity results E_F and relation results R_F. Finally, Eq. (10) uses the function $g()$ to construct the final knowledge graph G_F.

$$E_{(i)}, R_{(i)} = \mathbf{LLM}'_{(i)}(I' + X) \tag{7}$$
$$E'_{(i)}, R'_{(i)} = \mathbf{EEF}(E_{(i)}, R_{(i)}) \tag{8}$$
$$E_F, R_F = \mathbf{EAG}(\bigcap_i^M SubG_i) = \mathbf{EAG}\left\{\bigcap_i^M g(E'_{(i)}, R'_{(i)})\right\} \tag{9}$$
$$G_F = g(E_F, R_F) \tag{10}$$

3.3. Entity extraction filter module and prompt engineering

With the continuous advancement of LLMs, their outstanding generalization ability allows them to handle various natural language processing tasks without needing a pre-labeled training set. GPT-4 addresses many challenges related to natural language understanding from a natural language generation perspective. However, this generalization ability is not without its costs: the probabilistic and random nature of the generated results. Given the structural characteristics of the decoder-only model, its output, to some extent, depends on the prior input, meaning that once an error occurs in the input or memory, which gradually decays, the subsequent output may significantly deviate. As shown in Fig. 4, To minimize the adverse effects of randomness, this paper adopts an Entity Extraction Filter (EEF) module and prompt engineering to standardize knowledge graph triple extraction based on LLMs as follows:

(1) **Entity Extraction Filter Module:** The foremost and central task in enhancing LLMs' triple extraction performance is assessing the generated triples' quality. The quality of the triples extracted using LLMs can be assessed based on whether the head and tail entities in the triple extraction results appear in the entity extraction list. As shown in Eq. (11) and Eq. (12), the EEF module consists of two parts: Entity Extraction Task and Filter Task. As shown in Fig. 4, we employ prompt engineering to enable the model to perform entity extraction and obtain an entity list. Then, we filter out triples for which head and tail entities are absent in the entity list. When performing only the triple extraction task without the EEF module, errors often occurred due to generation randomness, which led to incorrectly long entities. Additionally, in the entity extraction process, we generate aliases for each entity to support subsequent Self-Fusion entity alignment. $E_{(i)}$ and $R_{(i)}$ are the entities list and triple list for the ith extraction. The $\mathbf{EntitySet}()$ function can obtain the set list of head and tail entities by inputting relational triples. K represents the length of the triple list $R_{(i)}$. r_k represents the kth triple in the triple list $R_{(i)}$. $R'_{(i)}$ is a list of triples filtered by entity extraction.

$$E'_{(i)} = \mathbf{EntitySet}(R_{(i)}) \cap E_{(i)} \tag{11}$$
$$R'_{(i)} = \bigcap_k^K \left\{ r_k | \mathbf{EntitySet}(r_k) \in E'_{(i)} \ and \ r_k \in R_{(i)} \right\} \tag{12}$$

(2) **Secondary Prompting:** The knowledge graph triple extraction process is divided into two stages. The first stage is the In-Context Learning Phase, allowing the LLMs to preliminarily have the ability to extract triples based on the input text X . The first stage Prompt is the same for each sample. The second stage is the Knowledge Graph Triple Extraction Phase. Unlike

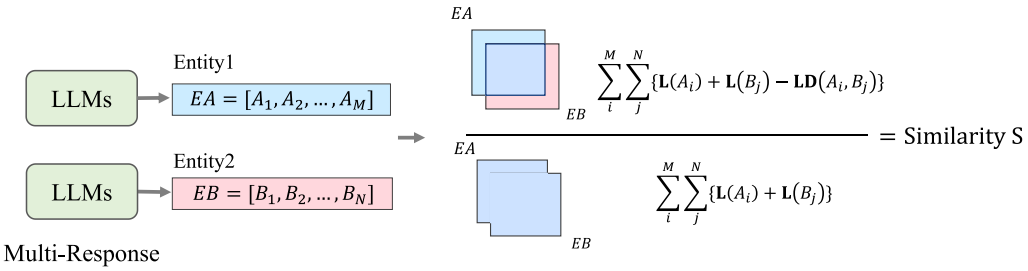


Fig. 5. Entity Alignment Algorithm Based on Entity Alias Generation.

the first prompt engineering, the second stage focuses more on assembling Prompt-Sec for the sample text to be predicted, with only the instruction part being the same as in the first stage. Also, in the instructions, asking the large language model to "not omit", "not consider output limitations", and other such commands is crucial.

(3) **Redundant Output:** The primary idea of redundant output is to enhance the output quality of the large language model by adding redundant output content in the instruction. For example, directing the model to present the input text within a JSON framework again during its output and to re-output predefined relationships in specific relationship extraction tasks mitigates the problem of memory decay for the decoder-only model.

(4) **Relationship Description Constraints:** When giving instructions, there is an additional emphasis on describing specific relationships to ensure their semantics are clear and accurate, and it avoids misunderstandings caused by the ambiguity of relational vocabulary itself. For instance, the term "contains" could imply composition in some contexts, but in certain datasets, it might only refer to a geographical containment relationship in the NYT dataset.

These methods, when combined, provide a more standardized and robust framework for extracting knowledge graph triples from unstructured text using LLMs.Prompt details of these methods can be viewed in Appendix.

3.4. Entity alignment based on entity alias generation module

While extracting knowledge graph triples from natural language text using LLMs, due to generation randomness, entities that are nominally different but refer to the same entity are often produced. The entity alignment task determines whether two input entities are identical. Implementing the entity alignment algorithm is critical for dynamically updating and maintaining the knowledge graph and completing triples in the knowledge graph. Traditional entity alignment methods include methods based on character similarity and language model discrimination [37,38]. Although the former is efficient and does not require labeling, it lacks deep semantic recognition, which can result in incorrect alignments in tasks based on semantic knowledge. In methods based on language model discrimination, when two entity pieces of information are input into a similarity calculation discrimination model like BERT, the similarity between entities can be effectively calculated. However, the proposed method requires significant supervised labeling and may not be suitable for the current task. The two entities are input as prompts in the GPT model to determine whether they are the same entity that lacks interpretability.

As shown in Fig. 5, we propose a training-free entity alignment module based on entity alias generation. In the judgment of entity alignment, it can be observed through examination that alignable entities typically have the same aliases. Based on the generation capabilities of LLMs, different entity aliases can be generated using these entity alias lists to determine whether they are the same entity. This article proposes an entity alignment algorithm based on entity alias generation. The Prompt instructs the extraction result of the entity to output a list of alias information for the entity. For two entities, Entity1 and Entity2, which must be verified for alignment, their entity alias lists are EA and EB, respectively. In the lists of entity aliases EA and EB, containing aliases for corresponding entities represented as A_i and B_j respectively, $A_i[0, m]$ denotes the substring from position 0 to m, while M and N represent the total number of generated entity aliases. Eq. (13) expresses the similarity between two aliases A_i and B_j using the Levenshtein distance, denoted as $\mathbf{LD}(A_i, B_j)$, which signifies the minimum number of insertions, deletions, or substitutions required to transform A_i into B_j. $\mathbf{c}()$ indicates the length of a string, and m and n are the character lengths corresponding to alias A_i and alias B_j. The position on the two-dimensional matrix $D[m][n]$ stores the minimum Levenshtein distance between a string of length m and another of length n.

$$m = \mathbf{c}(A_i), n = \mathbf{c}(B_j) \tag{13}$$

$$\mathbf{LD}(A_i, B_j) = D[m][n] = \begin{cases} m, \text{ if } n = 0 \\ n, \text{ if } m = 0 \\ D[m-1][n-1], \text{ if } A_i[0,m] = B_j[0,n] \\ \min(D[m-1][n]+1, D[m][n-1]+1, \\ \quad D[m-1][n-1]+1), \text{ if } A_i[0,m] \neq B_j[0,n] \end{cases}$$

Eq. (14) shows the similarity calculation formula for entity alignment based on entity alias generation, $\mathbf{c}(A_i) + \mathbf{c}(B_j)$ represents the maximum Levenshtein distance between two substrings. M and N represent the total number of generated entity aliases EA and EB. Using the formula below, the similarity between two entities from two entity lists can be calculated based on their Levenshtein distance.

$$S = \frac{\sum_i^M \sum_j^N \{\mathbf{c}(A_i) + \mathbf{c}(B_j) - \mathbf{LD}(A_i, B_j)\}}{\sum_i^M \sum_j^N \{\mathbf{c}(A_i) + \mathbf{c}(B_j)\}} \tag{14}$$

Where S is the similarity of entity alignment, the calculation method is the number of intersections of the two entity list aliases divided by the number of unions of the two list aliases.

$$\text{isSameEntity} = \begin{cases} \text{False} & \text{if } S < s \\ \text{True} & \text{if } S \geq s \end{cases} \tag{15}$$

As shown in Eq. (15), finally, a threshold s is set based on the actual situation. When the entity alignment similarity S is more significant than s, they are considered similar entities. When knowledge graph fusion across multiple input text sources is not considered, s can typically be set to 0, meaning that entities are deemed identical if they have the same alias in a set. However, when there is a rich data source and fusion of knowledge graphs across different texts to ensure strict and accurate entity alignment, the value of s can be adjusted based on the specific task and data circumstances. As shown in Fig. 3, The EAG module consists of three parts: Entity Fusion, remove Noise Relationships, remove Discrete Entity. In the "Entity Fusion" methodology, when the alignment similarity between entity1 and entity2 exceeds a threshold s, elements from entity2 are integrated into entity1. In the "remove Noise Relationships" methodology, we filter out triples where either the head

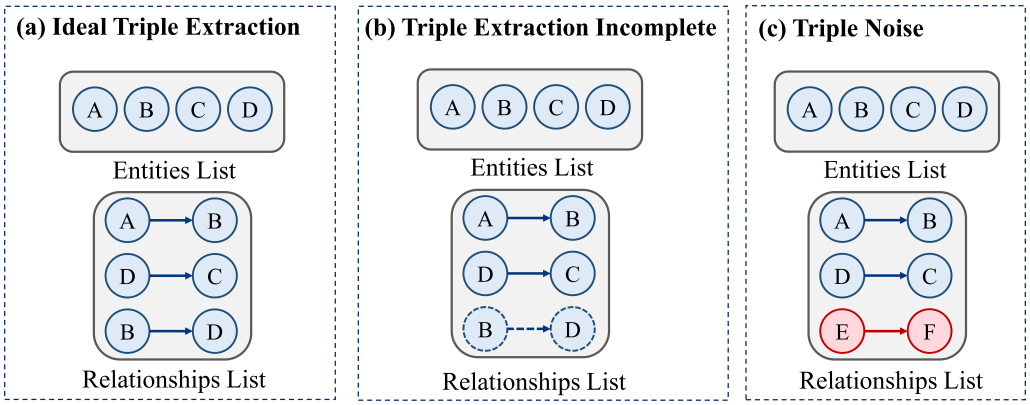


Fig. 6. (a) Ideal Triple Extraction; (b) Triple Extraction Incomplete; (c) Triple Noise.

or tail entities are not found within the common entity list. Lastly, in the "remove Discrete Entity" methodology, we exclude entities from the entity list that have not appeared in either head or tail entities. By generating a series of aliases for entities through the large language model and calculating the entity similarity between aliases, it features unsupervised learning, high efficiency, semantic interpretation, and strong interpretability.

3.5. Knowledge graph enhancement algorithm based on generative self-fusion subgraph

While we can implement knowledge graph triple extraction tasks using LLMs as described in Section 3.3, and to some extent ensure the quality of the generated triples, there are still issues of incomplete triple extraction and triple noise. As shown in Fig. 6, incomplete triple extraction refers to a situation where, for an input text X, the total number of triples extracted by the algorithm is less than the potential total number of triples. Triple noise refers to triples that contain incorrect entities or incorrect relationships. This problem tends to become increasingly severe with multiple responses to completion.

To address the issues above, namely the completion of triple extraction results through multiple responses of fusion and the minimization of the impact of triple noise issues, this paper proposes a knowledge graph enhancement algorithm based on a generative Self-Fusion subgraph, termed a "Self-Fusion Subgraph". As shown in Fig. 7, the Self-Fusion Subgraph refers to processing input text through multiple responses using a large language model, viewing each output as a subgraph representing the ideal knowledge graph extraction result. These subgraphs are then merged using a knowledge graph fusion algorithm, iteratively refining the knowledge graph towards completion. The specific algorithm for the Self-Fusion Subgraph is outlined in Algorithm 1. The entity alignment algorithm adopted is from Section 3.4, which is based on the generation of entity aliases.

Through multiple extraction and completion responses, potential triples can be effectively mined. By aligning entities, a unified entity extraction list is generated. By comparing the entities in the triples with the entities in the entity extraction list, the standard entity extraction results are used to regularize the triple extraction results, mitigating, to some extent, the noise impact from multi-response fusion.

4. Experiments

4.1. Dataset and evaluation

In the dataset section, we have chosen two widely-used datasets for the knowledge graph triple extraction task: NYT [24] and WebNLG [25], as well as the Common Relationship BDNC dataset from them used for generalization performance experiments, and the new dataset CCKS-LIFE [39] introduced in supplementary experiments.

Table 1
Statistics of the Knowledge Graph Triple Datasets.

Datasets	Train	Test	Relationship
NYT	56 195	5000	19
WebNLG	5019	703	171
Common Rel.BDNC	–	99	4
CCKS-LIFE7	42 335	248	7

(1) **NYT dataset:** The NYT dataset is primarily generated from New York Times articles using distant supervision methods. It contains 19 relationships.
(2) **WebNLG dataset:** The WebNLG dataset is constructed in reverse from a natural language generation task. It should be noted that the input texts for both NYT and WebNLG are in English.
(3) **Common Relationship BDNC dataset:** The Common Relationship BDNC dataset is selected from the test set of WebNLG, covering place of birth, place of death, nationality, and capital — four general relationship types that also appear in the NYT dataset.
(4) **CCKS-LIFE7 dataset:** The CCKS-LIFE7 dataset originates from the CCKS competition in 2022. As an emerging dataset, it focuses on Chinese knowledge graph triples for person relationship extraction.

Table 1 presents the detailed statistics of these datasets. Among them, the "Train" column represents the data volume of the training set. In our proposed SF-GPT model, we did not utilize the training set data, indicating a substantial reduction in associated human annotation costs. The "Test" column denotes the data volume of the test set, while the "Relationship" column represents the number of relationship categories.

For evaluation, we employed standard metrics for knowledge graph triple evaluation: Precision, Recall, and F1-Score [40].

Precision – This metric is typically employed to evaluate the Precision of a model in the task of triple extraction. A higher value of this metric indicates that the model's predictions are more accurate. Eq. (16) is calculated as the ratio of the number of triples correctly extracted and identified (True Positives, TP) to the total number of triples the model extracts (Sum of True Positives and False Positives, TP+FP). In subsequent tables, this will be abbreviated as Prec.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

Recall – This metric is generally used to assess a model's capability in mining triples from the triple extraction task. A higher value indicates that the model can extract as many triples as possible from a text. Eq. (17) is determined by the ratio of triples correctly extracted and identified (True Positives, TP) to the total number of potential triples

Algorithm 1： Self-Fusion Subgraph Algorithm

Input：

P：Prompt for in-context learning.

Ps：Prompt-Sec for knowledge graph extraction.

t：Natural language text for knowledge graph extraction.

n：Number of subgraphs.

$subG_i$: Number of subgraphs.

s: Entity alignment threshold.

Output：

G：Final output knowledge graph.

Function:

// *1.The main function of extracting knowledge graph G via Self-Fusion Subgraph Algorithm.*

Self-Fusion():

```
Initialize G as an empty knowledge graph.
Initialize P,Ps,t,n, subG_i,s based on inputs.
LLM() ← LLM(P)   // in-context learning of LLM via P.
for i=0 to n do
    subG_i ← LLM(Ps, t)   // Generating subgraphs subG_i via Ps and t.
    if i=0 then
        G ← subG_i
    else
        G ← GraphFusion(G,subG_i,s) // Fusing G and subG_i with a threshold of s.
```

// *2.The function of fusing G1 and G2 based on entity alias similarity s.*

GraphFusion(G_1, G_2, s)**:**

```
// Get the EntitiesList and RelationList generated by LLM via the get function.
EntitiesList1← getEntitiesList(G_1) ; EntitiesList2← getEntitiesList(G_2);
RelationsList1← getRelationsList(G_1) ; RelationsList2← getRelationsList(G_2);
for each entity1 of EntitiesList1 do
    for each entity2 of EntitiesList2 do
        // EAS(entity1,entity2) is the Entity Alias Similarity of entity1 and entity2.
        if entity1,entity2 have the same name or EAS(entity1,entity2)>s then
            Change entity2 name of head or tail entity in RelationsList2 as entity1 name.
            entity1← EntityFusion(entity1,entity2) // Default to select entity1 directly.
            EntitiesList2← EntitiesList2-{entity2}
EntitiesList3← EntitiesList1+ EntitiesList2
// removeNoiseRelations() is used to remove Triple Noise.
RelationsList3← removeNoiseRelations(RelationsList1, RelationsList2)
// removeDiscreteEntity() remove entities that do not appear in the RelationList1.
EntitiesList, RelationsList← removeDiscreteEntity (EntitiesList3, RelationsList3)
G' ←{ EntitiesList, RelationsList}
return G'
```

(Sum of True Positives and False Negatives, TP+FN). In subsequent tables, this will be abbreviated as Rec.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{17}$$

F1-Score - This metric takes into account both Precision and Recall to reflect, to a certain extent, the model's overall performance. Eq. (18) is computed as the harmonic mean of Precision and Recall. In subsequent tables, this will be abbreviated as F1.

$$F1\text{-Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{18}$$

This research used a generative large language model based on GPT-4 as a baseline. Considering the diverse relationship types in the dataset, to reduce interference between samples and relationships, we

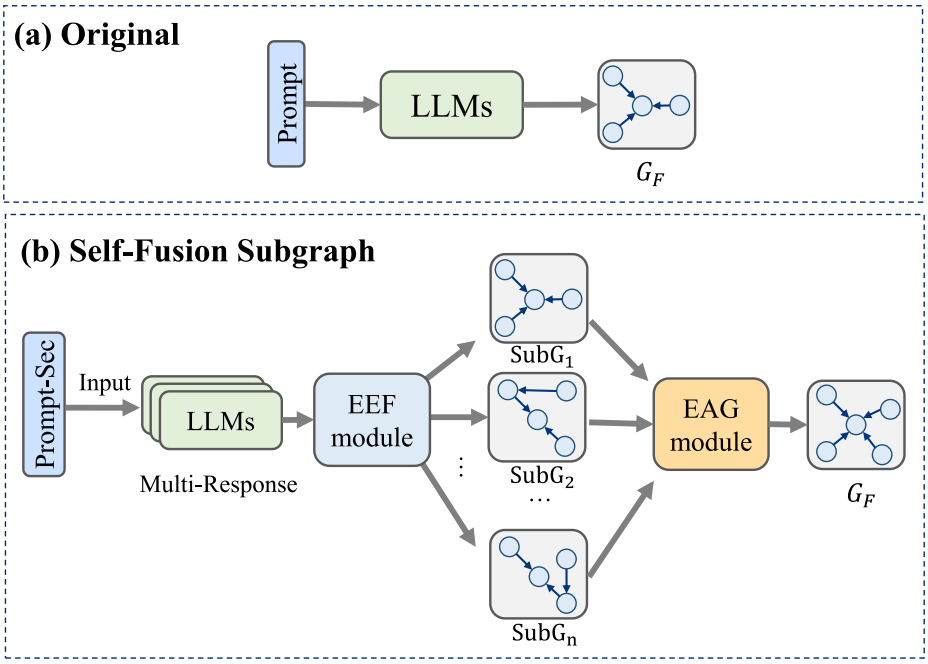


Fig. 7. (a) Original Knowledge Graph Triple Extraction Base on LLMs; (b) Knowledge Graph Enhancement Algorithm Based on Generative Self-Fusion Subgraph.

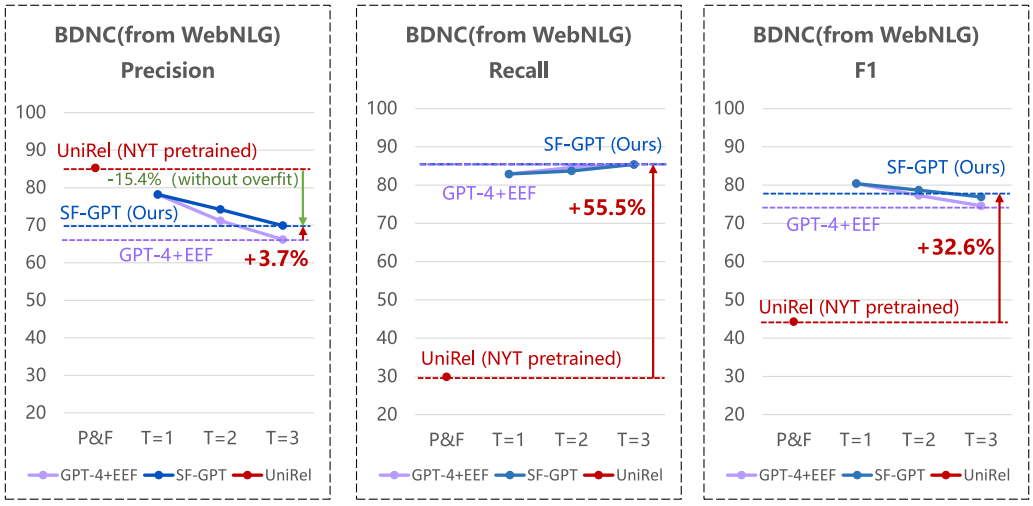


Fig. 8. Results of Knowledge Graph Triple Extraction Generalization Experiment.

decided to input samples individually and combine them with various specified relationships. We can also categorize and predict relationships based on semantic similarity to enhance efficiency when defining relationships. For instance, relationships related to individuals, such as "place of birth", "place of death", and "nationality", can be grouped. However, we propose constraining one type of relationship at a time, maximally reducing contextual interference. In terms of experimental design, we conducted the following experiments to evaluate the efficacy of SF-GPT comprehensively:

(1) **Generalization Experiment:** Aimed at validating SF-GPT's capability to execute knowledge graph triple extraction without relying on specific input texts and ensuring robust performance. We used the Common Relationship BDNC dataset. More details are provided in Section 4.2.
(2) **Ablation Experiment:** This experiment mainly assesses the performance of multi-response fusion Self-Fusion in eliminating triple noise. We tested using the NYT and WebNLG datasets. More details are provided in Section 4.3.
(3) **Comparative Experiment:** This test aims to compare the performance of SF-GPT with other triple extraction models that rely on training sets. We employed the WebNLG dataset. More details are provided in Section 4.4.
(4) **Supplementary Experiment:** On a relatively new knowledge graph triple extraction dataset, we tested the generalization capability of SF-GPT. The dataset used for this experiment is CCKS-LIFE7. More details are provided in Section 4.5.

In this section, GPT-4 represents the knowledge extraction model based on the GPT-4 base model, with ChatIE [36] as the fine-tuning model. The Subgraphs Fusion primarily comprises two modules: the Entity Extraction Filter (EEF) and the Entity Alias Generation, with Self-Fusion Subgraph Algorithm in experiments (EAG), whose specific design details will be elaborated in Sections 3.3, 3.4, and 3.5. SF-GPT is equivalent to GPT-4+EEF+EAG.

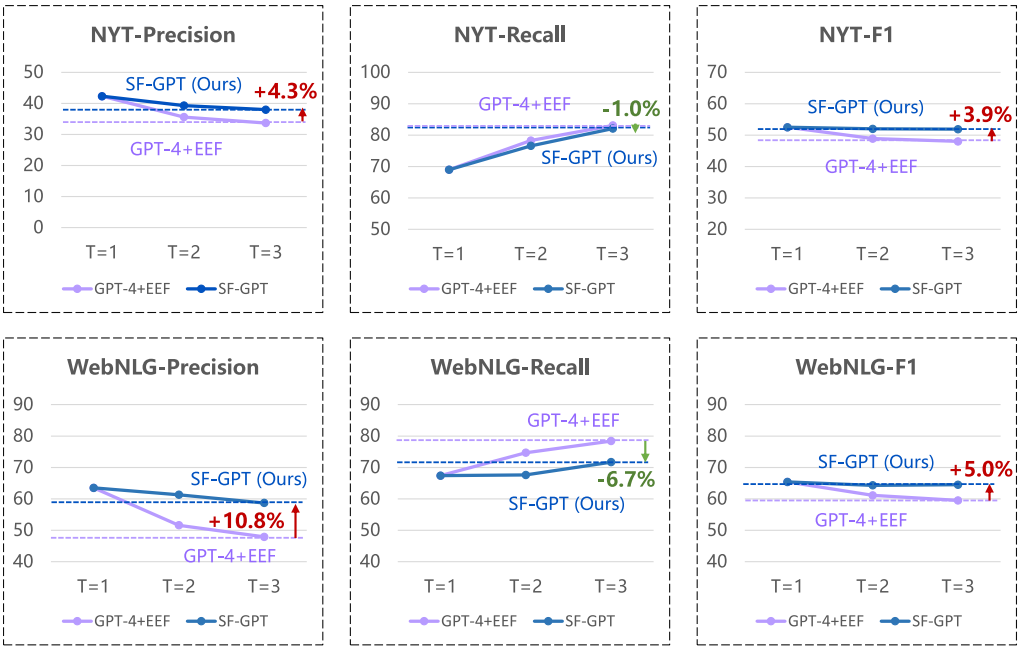


Fig. 9. Results of Knowledge Graph Triple Extraction Ablation Experiment.

Table 2
Results of Knowledge Graph Triple Extraction Generalization Experiment.

Model	Common Relationship BDNC (From WebNLG)		
	Prec.	Rec.	F1
UniRel$_{(NYTtrained)}$	85.3	29.9	44.3
GPT-4 (ChatIE-like)	73.1$_{(-12.2\%)}$	82.9$_{(+53.0\%)}$	77.6$_{(+33.3\%)}$
GPT-4+EEF	78.2$_{(-7.1\%)}$	82.9$_{(+53.0\%)}$	80.4$_{(+36.1\%)}$
GPT-4+EEF $_{T=2}$	71.2$_{(-14.1\%)}$	84.6$_{(+54.7\%)}$	77.3$_{(+33.0\%)}$
SF-GPT $_{T=2}$	74.2$_{(-11.1\%)}$	83.7$_{(+53.8\%)}$	78.7$_{(+34.4\%)}$
GPT-4+EEF $_{T=3}$	66.2$_{(-19.1\%)}$	85.4$_{(+55.5\%)}$	74.6$_{(+30.3\%)}$
SF-GPT $_{T=3}$	69.9$_{(-15.4\%)}$	85.4$_{(+55.5\%)}$	76.9$_{(+32.6\%)}$

Table 3
Results of Knowledge Graph Triple Extraction Ablation Experiment - NYT.

Model	NYT		
	Prec.	Rec.	F1
GPT-4 (ChatIE-like)	41.8	69.0	52.1
GPT-4+EEF	42.3$_{(+0.5\%)}$	69.0	52.5$_{(+0.4\%)}$
GPT-4+EEF $_{T=2}$	35.6	78.3	48.9
GPT-4+EEF+EAG $_{T=2}$	39.3$_{(+3.7\%)}$	76.6$_{(-1.7\%)}$	52.0$_{(+3.1\%)}$
GPT-4+EEF $_{T=3}$	33.7	83.1	48.0
GPT-4+EEF+EAG $_{T=3}$	38.0$_{(+4.3\%)}$	82.1$_{(-1.0\%)}$	51.9$_{(+3.9\%)}$

Table 4
Results of Knowledge Graph Triple Extraction Ablation Experiment - WebNLG.

Model	WebNLG		
	Prec.	Rec.	F1
GPT-4 (ChatIE-like)	56.9	67.4	61.7
GPT-4+EEF	63.5$_{(+6.6\%)}$	67.4	65.4$_{(+3.7\%)}$
GPT-4+EEF $_{T=2}$	51.6	74.7	61.1
GPT-4+EEF+EAG $_{T=2}$	61.3$_{(+9.7\%)}$	67.6$_{(-7.1\%)}$	64.3$_{(+3.2\%)}$
GPT-4+EEF $_{T=3}$	47.9	78.4	59.5
GPT-4+EEF+EAG $_{T=3}$	58.7$_{(+10.8\%)}$	71.7$_{(-6.7\%)}$	64.5$_{(+5.0\%)}$

4.2. Generalization experiment results dataset and evaluation

In 2018, the BERT model was introduced by Google and was widely adopted in natural language processing, spearheading the trend of the "Pre-training and Fine-tuning" model. BERT-like models resulted in a significant increase in the performance of triple extraction tasks in the knowledge graph. Subsequently, models based on this paradigm, such as TPlinker and UniRel, achieved F1 scores surpassing 90% on open-source datasets like NYT and WebNLG, leading many researchers to commend their performance in triple extraction tasks as "on par with humans". However, such "Pre-training and Fine-tuning" models remain highly dependent on the input texts at the core. They do not extract triples based on the occurrence of relevant lexemes; instead, they are confined to the context of the training set. In this section, we describe a generalization experiment in which we conjectured that the "Pre-training and Fine-tuning" model, such as the UniRel model, which is currently the best-performing model, actually overfits a single dataset. We found that the NYT and WebNLG datasets share some of the same relationship types, such as place of birth, place of death, nationality, and capital. If we train the UniRel model with the NYT dataset and then extract the relationships in the WebNLG dataset as place of birth, place of death, nationality, and capital, and then validate the extraction performance metrics on this basis, we can validate the performance of triple extraction on different data distributions.

As shown in Fig. 8 and Table 2, T represents the number of query response rounds, when T is set to 1, the performance of GPT-4+EEF and SF-GPT is consistent. P&F in the legend stands for "Pre-training and Fine-tuning" models. To systematically evaluate the generalization capabilities of "Pre-training and Fine-tuning" models versus LLMs in the task of triple extraction, we use the top-performing "Pre-training and Fine-tuning" model, UniRel, as a baseline and trained it on the NYT dataset. Red values indicate performance improvements compared to UniRel(NYT train), while green represents a decline in performance. Our large language model experiment employed the GPT-4 model that underwent in-context learning for triple extraction, while SF-GPT introduced the Self-Fusion Subgraph algorithm on top of it.

The experimental results demonstrate that the baseline model UniRel achieved a high precision of 85.3%, a recall rate of 29.9% and an F1 score of only 44.3%. This shows that when some data samples in the BDNC dataset are distributed similarly to NYT coincidentally, the UniRel model can accurately extract triples with a high precision rate. However, when the data distribution is inconsistent, it seriously interferes with the model's triple extraction ability, and the omission of knowledge graphs is more expensive than the manual correction cost

Table 5
Results of Knowledge Graph Triple Extraction Comparative Experiment.

	Model	WebNLG			NYT		
		Prec.	Rec.	F1	Prec.	Rec.	F1
Train Set Required	NovelTagging [41]	52.5	19.3	28.3	62.4	31.7	42.0
	CopyRE [42]	37.7	36.4	37.1	61.0	56.6	58.7
	GraphRel [43]	44.7	41.1	42.9	63.9	60.0	61.9
	OrderCopyRE [40]	63.3	59.9	61.6	77.9	67.2	72.1
	$CasRel_{BERT}$ [44]	93.4	90.1	91.8	89.7	89.5	89.6
	$TPlinker_{BERT}$ [22]	91.7	92.0	91.9	91.3	92.5	91.9
	$PRGC_{BERT}$ [45]	94.0	92.1	93.0	93.9	91.9	92.6
	$OneRel_{BERT}$ [46]	94.1	94.4	94.3	92.8	92.9	92.8
	$UniRel_{BERT}$ [23]	94.8	94.6	94.7	93.5	94.0	93.7
No Train Set Required	GPT-4(ChatIE-like)	56.9	67.4	61.7	41.8	69.0	52.1
	GPT-4+EEF $_{T=3}$	47.9	78.4	59.5	33.7	83.1	48.0
	SF-GPT $_{T=3}$	58.7	71.7	64.5	38.0	82.1	51.9

Table 6
Results of Knowledge Graph Triple Extraction Supplementary Experiment (see [47]).

	Model	CCKS-LIFE7		
		Prec.	Rec.	F1
Train Set Required	UIE_{BERT} [47]	-	-	72.5
No Train Set Required	GPT-4 (ChatIE-like)	58.1	62.1	60.1
	GPT-4+EEF	61.0 (+2.9%)	62.1	61.5 (+1.4%)
	GPT-4+EEF $_{T=2}$	52.4	70.3	60.0
	SF-GPT $_{T=2}$	57.0 (+4.6%)	68.4 (−1.9%)	62.2 (+2.2%)
	GPT-4+EEF $_{T=3}$	48.7	73.0	58.5
	SF-GPT $_{T=3}$	54.0 (+5.3%)	70.4 (−2.6%)	61.1 (+2.6%)

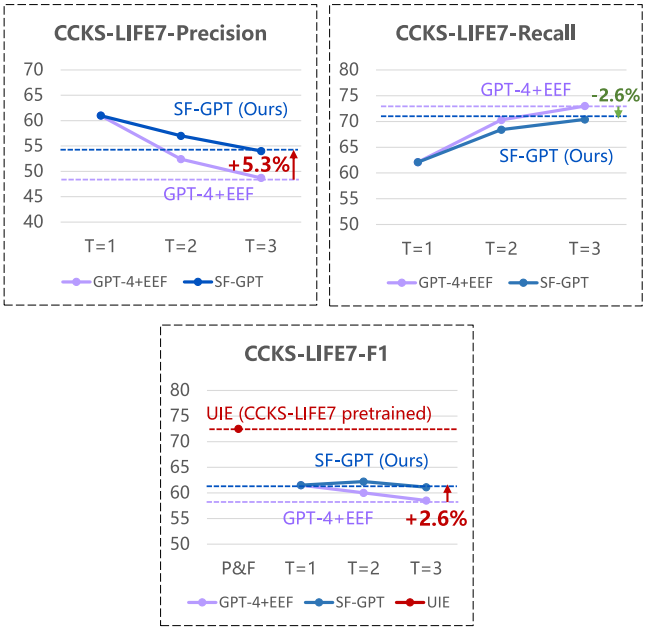


Fig. 10. Results of Knowledge Graph Triple Extraction Supplementary Experiment.

caused by the error in the extraction results in practical engineering applications. Based on the LMMs model, GPT-4, which has the strongest performance at present, can realize the triple extraction task without training by only relying on the context learning of ChatIE, which is represented by GPT-4 (ChatIE-like) in the Table 2. The precision rate of GPT-4 is 73.1%, which is 12.2% lower than UniRel; however, this results from the precision rate in the case of no overfitting. GPT-4 has a precision of 82.9%, which is 53% better than UniRel, indicating that UniRel has an extraction omission problem due to overfitting of a particular dataset. The combined precision and recall scores also resulted in a 33.3% higher F1 score for GPT-4 than for UniRel. With the same number of response rounds, using GPT-4 as the baseline model, we improved the precision rate and F1 by the addition of EEF and EAG modules, and an analysis of this part of the improvement will be presented in subsequent sections.

4.3. Ablation experiment results

Using LLMs for triple extraction in knowledge graphs indeed faces challenges of extraction incompleteness and noise introduction. To address the incompleteness of triple extraction, we adopted a strategy of generating subgraphs through a multi-response query for completion in this study. We conducted an ablation experiment to compare the performance of multiple models generated by different combinations of modules at different numbers of query rounds on the NYT and WebNLG triple extraction datasets.

As shown in Table 3, Table 4, and Fig. 9, T represents the number of query response rounds when T is set to 1, the performance of GPT-4+EEF and SF-GPT(GPT-4+EEF+EAG) is consistent, the results show that compared to a single-response query with T=1, multi-response queries with$T = 2$ and$T = 3$ achieved significant recall improvements on both baseline GPT-4 (ChatIE-like) and SF-GPT models. Taking the NYT dataset as an example, the recall of GPT-4+EEF is 69% for T=1. The recall of SF-GPT was 76.6% for T=2. The recall of SF-GPT was 82.1% for T=3. Recall improved at different rounds, and the GPT-4 baseline model was the same. This paper argues that each LLM can target the same text from a different perspective through multiple rounds of fusion. Although there is a certain amount of random omission in a single extraction, this omission is reduced by fusing the results.

However, the multi-response fusion strategy introduced a new problem: as the number of query rounds increased, noise in the triples gradually accumulated. We introduced the EAG module and the Self-Fusion Subgraph method to address this issue. With the EAG module and Self-Fusion Subgraph method, SF-GPT typically demonstrated better performance in Precision and F1 scores under the same number of rounds. Taking the WebNLG dataset as an example, at T=2, the Precision of GPT-4+EEF was 51.6%, whereas SF-GPT showed 9.7% improvement and reached 61.3% Precision. At T=3, GPT-4+EEF had 47.9% Precision, whereas SF-GPT showed 10.8% improvement and reached 58.7%, with a similar improvement in the F1 score. In this paper, it is argued that the core role of the EAG module and Self-Fusion Subgraph method is to maintain a list of familiar entities between different graphs through which noise tests across graphs can be effectively realized. In the case of multiple rounds of generation, correct content is more likely to be generated, and the generation of incorrect content is accidental. This common entity list represents this collected from those entities that are repeatedly generated, and then noise triples are removed after Self-Fusion by multiple graphs generated for the exact text to enhance Precision and F1.

When the query rounds are $T = 2$ and T=3, using the Self-Fusion Subgraph can noticeably enhance Precision and F1 scores, but it also

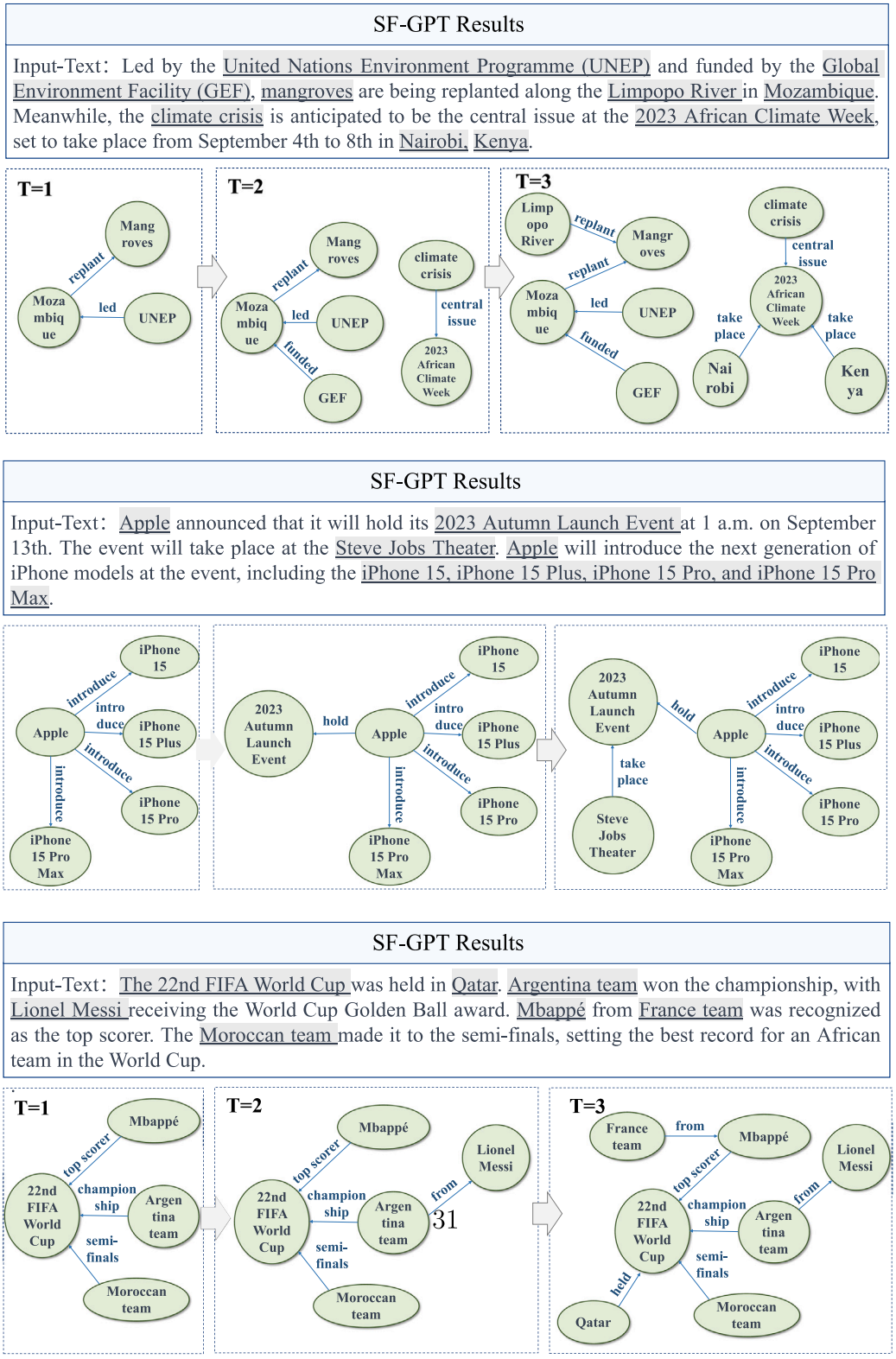


Fig. 11. Results of Knowledge Graph Open Relationship Extraction Task Experiment.

reduces recall, and this is because the core of the Self-Fusion strategy aims to improve Precision by reducing the number of triples. A few correct triples might get mistakenly omitted without a prior supervised signal. On the WebNLG dataset It can be mainly attributed to the complex relationships and numerous numerical extraction tasks in the WebNLG dataset, such as "number Of Members", "was a crew member of", and "LCCN number". In traditional knowledge graph construction, such numerical values are often considered entity attributes rather than independent entities. However, the self-fusion method primarily cleans triples through a common entity list from multi-response subgraphs, causing specific numerical values not to be recognized as entities in the model. Thus, for the same number of rounds, the Self-Fusion Subgraph

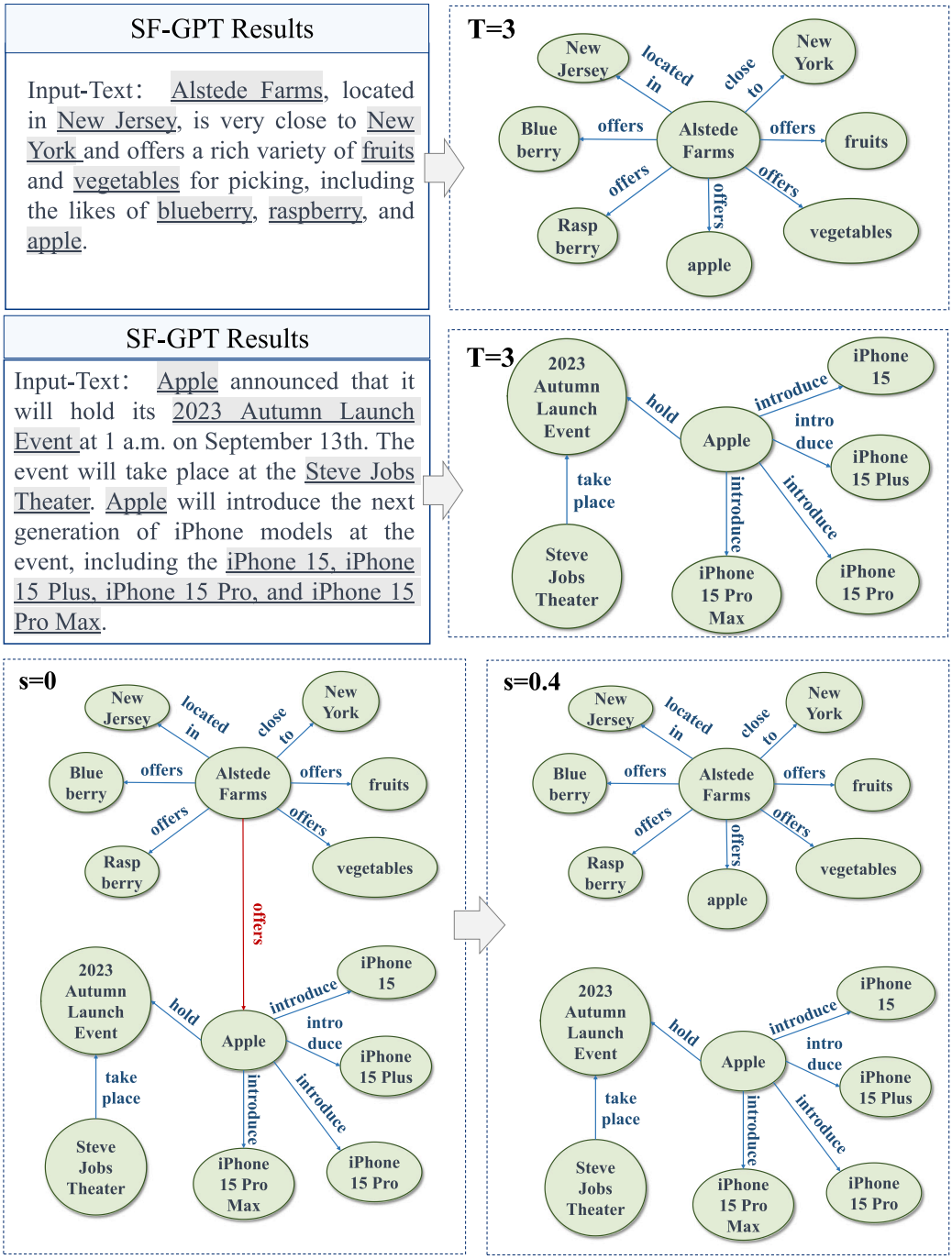


Fig. 12. Results of Knowledge Graphs Fusion from Different Texts.

causes the Recall score to drop more in WebNLG compared to NYT, e.g., for T=3, the Recall score for SF-GPT on the NYT dataset drops by only 1% compared to GPT-4+EEF, but drops by 6.7% on WebNLG. To ensure model consistency and completeness, we did not apply specialized optimization.

4.4. Comparative experiment results

In this paper, a comparative experiment is conducted to compare the performance of LLM methods, such as GPT-4 and SF-GPT, with that of the currently dominant triple extraction model. The following comparison results are presented.

T represents the number of query response rounds, and *BERT* means that this type of model uses BERT as the base model. From Table 5, in the comparative experiment for knowledge graph triple extraction, by the time the number of query rounds reached T=3, the GPT-4 and SF-GPT, which do not rely on training datasets, have already surpassed traditional neural network models in triple extraction for WebNLG dataset. For example, at T=3, the Recall and F1 scores of SF-GPT on the WebNLG dataset were 71.7% and 64.5%, respectively, which are higher than those of 59.9% and 61.6% for OrderCopyRE. Given that "Pre-training and Fine-tuning" models suffer from overfitting to a single dataset, it is conceivable that our model's performance may exhibit

Table A.7
Details of Prompt in In-Context Learning Phase.

Instruction	Your next task is as follows: When I input several paragraphs of text represented by a list, you need to perform knowledge graph extraction. Identify the entities within the text and store them in the ″entities″ field. Then, using the entities identified in the ″entities″ field, recognize the relationship triples and store them in the ″relations″ field.The results should be outputted in JSON format. I will provide an example for you.Below is the format of the JSON, where the values indicate the filling requirements There should be at least 3 or more in the ″enSec″ field.The entities in the ″head″ and ″tail″ fields must appear in the ″en″ field.All entities in the ″entities″ field labeled as ″en″ must appear in the ″head″ or ″tail″ ofthe ″relations″.No isolated entities are allowed in the knowledge graph!Do not miss any entities or relation -ship triples, and make sure not to get the relationships between entities wrong!

Json Template	Input JSON format:
	{
	″relations_to_choose″: ″A limited list of relationships. Only these types of relationships should be extracted. Do not miss any of these relationships.″
	″input″: ″Text to be extracted″
	}
	Output JSON format: {
	″input″: ″ Text to be extracted ″ ,
	″entities″: [
	{
	″id″: ″Unique identifier number″ ,
	″en″: ″Identified entity's English name (can be a country, city, person, number, year, etc.)″ ,
	″enSec″: [″Non-standard English name, often used as an alias or abbreviation for the identified entity's English name″] ,

(*continued on next page*)

numerical disparities when compared to ″Pre-training and Fine-tuning″ models.

However, it is imperative to underscore that our approach obviates the need for labor-intensive data labeling, resulting in significant resource savings when juxtaposed with ″Pre-training and Fine-tuning″ models. Moreover, our method yields enhanced generalization capabilities. The human costs thus conserved can be judiciously redirected towards the manual refinement of outcomes, facilitating the seamless execution of the knowledge graph construction task.

4.5. Supplementary experimental results

Some opinions suggest that during the pre-training process, due to the extensive use of the NYT and WebNLG datasets, these two datasets have been employed for pre-training sample collection, leading to data leakage. The CCKS-LIFE dataset was proposed in the CCKS2022 general information extraction competition in April 2022. From interactions with GPT-4, it is known that GPT-4's data was updated until September 2021. Supplementary experiments are conducted in this paper to demonstrate the performance gains that can be achieved using SF-GPT on new datasets.

Table A.7 (*continued*).

Json Template	}
	],
	"relations": [
	{
	"head": "Head entity name (must appear under the 'en' field in 'entities')" ,
	"type": "Type of relationship" ,
	"tail": "Tail entity name (must appear under the 'en' field in 'entities')" ,
	}
	]
	}

Example	Input: [{
	"input":"But that spasm ofirritation by a master intimidator was minor compared with what Bobby Fischer,the erratic former world chess champion , dished out in March at a news conference in Reykjavik , Iceland ."
	"relations _to_choose":["capital","nationality","deathPlace"] ,
	}]
	Output: [
	{
	"input":"But that spasm ofirritation by a master intimidator was minor compared with what Bobby Fischer, the erratic former world chess champion,dished out in March at a news conference in Reykjavik,Iceland." ,
	"relations _to_choose":["capital","nationality","deathPlace"] ,
	"entities": [
	{
	"id": "1_",
	"en": "Iceland ",
	"enSec": ["Iceland ","the Republic of Iceland ","iceland "]
	}, {
	"id": "2_",
	"en":" Bobby Fischer ",
	"enSec": ["Fischer ","Bobby Fischer ","bobby fischer "]
	}, {
	"id": "3_",
	"en": "Reykjavik ",
	"enSec": ["Reykjavik ","reykjavik "]
	}
	],

(*continued on next page*)

T represents the number of query response rounds. When T is set to 1, the performance of GPT-4+EEF and SF-GPT is consistent. From Table 6 and Fig. 10, it is evident that GPT-4 and SF-GPT are capable of executing the task of knowledge graph triple extraction even on data post-2022. In the CCKS2022 competition, the UIE model served as the official baseline, achieving an F1-Score of 72.5%. Meanwhile, the SF-GPT model, with a query round of T=3, also attained a commendable F1-Score of 61.1% without any dependency on training datasets. Similar to the NYT and WebNLG datasets, GPT-4+EEF showed an increase in Precision and a slight decrease in recall compared with SF-GPT under the same number of iterations, but the F1 score still improved. It is shown that even with a new dataset, SF-GPT still delivers performance gains for extracting triples using LLMs.

4.6. Knowledge graph open relationship extraction task experimental results

All the experiments presented above were performed under predefined relationship types, termed the constrained relationship extraction task. T represents the number of query response rounds. When T is set to 1, the performance of GPT-4+EEF and SF-GPT is consistent.

Table A.7 (*continued*).

Example	"relations":[
	{
	"head": "Iceland ",
	"type": "capital ",
	"tail": "Reykjavik "
	},{
	"head": "Bobby Fischer ",
	"type": "nationality ",
	"tail": "Iceland "
	},{
	"head": "Bobby Fischer ",
	"type": "deathPlace ",
	"tail": "Reykjavik "
	}
	]
	}
	]

As illustrated in Fig. 11, the proposed SF-GPT model in this study is also capable of effectively handling the open relationship extraction task. Utilizing the Multi-response Self-Fusion subgraph, the model can continuously enrich the knowledge graph. We selected three sets of input texts from diverse sources to demonstrate the iterative completion capability. The experimental results demonstrate that continuously increasing the number of rounds makes it possible to constantly mine the potential triples of a specific text, thereby making the knowledge graph complete. Thus, the proposed SF-GPT can accurately and comprehensively accomplish the knowledge graph construction task.

As illustrated in Fig. 12 and Eq. (15), When performing a fusion of knowledge graphs from different textual sources, it is essential to adjust the thresholds of the entity alignment algorithm based on entity alias generation. Utilizing a consistent threshold of s=0 for entities derived from the same textual source can result in misalignments. For instance, "Apple" could refer to a corporation or be understood as a type of fruit from a farm. Through experimentation, it was observed that when s=0.4, multiple aliases must coincide to be considered the same entity.

When threshold s is adjusted higher, determining whether it is the same entity requires not only syntactic similarity at the character level but also the similarity of the name of the alias list. For example, when Apple is regarded as a company, it generates the aliases Apple Inc. and Apple Company, which enable the model to determine whether it is the same entity from deeper semantics and reduce the fusion of errors.

5. Conclusion and future

5.1. Summary

In this study, we address the generalization issue of the "Pre-training and Fine-tuning" model in the knowledge graph triple extraction task. We recognize the challenges posed by the probabilistic generation of LLMs, which occasionally results in errors during triple extraction. We employ the LLMs model to achieve knowledge graph triple extraction without relying on a specific training set. We further propose an efficient module to filter the results of triple generation, based on entity extraction, named EEF, thus addressing the challenge of evaluating and cleansing triple generation. To accomplish the fusion of triple extraction results that balances both semantics and interpretability from the perspective of LLMs, we introduce a training-free entity alignment module based on entity alias generation, named EAG, tackling the lack of semantic richness and interpretability in knowledge fusion methods as seen by LLMs. In addressing the issues of incomplete triple extraction due to LLM's Prompt output limitations and the accumulation of triple noise from multi-response results fusions, we adopt a multi-response Self-Fusion subgraph strategy. In experiments, SF-GPT showed a 55.5% increase in recall and a 32.6% increase in F1 score on the BDNC dataset compared to a BERT-like model trained on the NYT dataset and achieved a 5% improvement in F1 score compared to the original GPT-4+EEF model on the WebNLG dataset in the case of a fusion round of three. SF-GPT offers a promising pathway for extracting knowledge from unstructured information and constructing professional knowledge graphs.

5.2. Future

While our method has made some progress, there are still certain limitations and challenges:

(1) **Model Dependency:** The current work is highly dependent on the underlying performance of the GPT-4 model. Many LLMs weaker than GPT-4 cannot perform a single extraction task. With the release of more advanced models in the future, we aim to enhance extraction performance further.
(2) **Trade-off between Precision and Recall:** The use of the Self-Fusion subgraph method indeed reduces noise and improves precision, but it might also impact recall. Finding a balance while enhancing both precision and recall remains a crucial challenge.
(3) **Time Efficiency:** To minimize interference between outputs and relationships, we process only one question at a time and include some constrained relationships. This approach results in considerable time consumption. In the future, by introducing more potent language models and optimization strategies, we aim to address these interference issues effectively, thereby increasing processing speed.

In conclusion, although we have made some progress, employing LLMs to achieve knowledge graph triple extraction without a training set remains a research area worth further exploring and is challenging.

CRediT authorship contribution statement

Lizhuang Sun: Writing – original draft, Validation, Methodology, Conceptualization. **Peng Zhang:** Writing – review & editing, Investigation. **Fang Gao:** Writing – review & editing, Investigation. **Yuan An:**

Table A.8
Details of Prompt in Knowledge Graph Triple Extraction Phase.

Instruct	Your next task is as follows: When I input several paragraphs of text represented by a list, you need to perform knowledge graph extraction. Identify the entities within the text and store them in the ″entities″ field. Then, using the entities identified in the ″entities″ field, recognize the relationship triples and store them in the ″relations″ field. The results should be outputted in JSON format. I will provide an example for you. Below is the format of the JSON, where the values indicate the filling requirements. There should be at least 3 or more in the ″enSec″ field. The entities in the ″head″ and ″tail″ fields must appear in the ″en″ field. All entities in the ″entities″ field labeled as ″en″ must appear in the ″head″ or ″tail″ of the ″relations″. No isolated entities are allowed in the knowledge graph!Do not miss any entities or relationship triples,and make sure not to get the relationships between entities wrong!
Input-Text and Relationship List	The ″type″ in ″relations″ can only be chosen from the following options: ['deathPlace','birthPlace ','nationality ','capital ']. [{ ″input″: 'Elliot See died in St Louis which is part of the Kingdom of France. ', ″relations _to_choose″: ['DeathPlace ','BirthPlace ','nationality ', 'capital '] }]
Relationship Explanation (Optional)	Their meanings are respectively: 'deathPlace', it means the place of death of a person. 'birthPlace',it means the birthplace of a person, 'nationality', it means the nationality of a person, and 'capital', it means the capital of acountry. For relationships involving people, rely only on the input text without considering the actual circumstances. For the capital of a country,base it on factual information! Make sure the triples meet the requirements and are accurate, and there should be no repetitions!

Writing – review & editing, Investigation. **Zhixing Li:** Writing – review & editing, Validation, Software, Resources, Methodology. **Yuanwei Zhao:** Writing – review & editing, Validation, Investigation, Formal analysis.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Peng Zhang reports financial support was provided by the Jilin Scientific and Technological Development Program under Grant 20220201017GX. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Jilin Scientific and Technological Development Program, China under Grant 20220201017GX.

Appendix. Details of prompt

See Tables A.7 and A.8.

Data availability

Data will be made available on request.

References

[1] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert Syst. Appl. 141 (2020) 112948.
[2] A. Hogan, et al., Knowledge graphs, ACM Comput. Surv. 54 (4) (2021) 1–37.
[3] S. Ji, et al., A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Trans. Neural Netw. Learn. Syst. 33 (2) (2021) 494–514.
[4] S. Deng, H. Rangwala, Y. Ning, Dynamic knowledge graph based multi-event forecasting, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020.
[5] T. Shen, F. Zhang, J. Cheng, A comprehensive overview of knowledge graph completion, Knowl.-Based Syst. 255 (2022) 109597.
[6] K.M. Malik, et al., Automated domain-specific healthcare knowledge graph curation framework: Subarachnoid hemorrhage as phenotype, Expert Syst. Appl. 145 (2020) 113120.
[7] M. Rotmensch, et al., Learning a health knowledge graph from electronic medical records, Sci. Rep. 7 (1) (2017) 5994.
[8] B. Abu-Salih, et al., Healthcare knowledge graph construction: A systematic review of the state-of-the-art, open issues, and opportunities, J. Big Data 10 (1) (2023) 81.
[9] M.D.L. Tosi, J.C.D. Reis, SciKGraph: A knowledge graph approach to structure a scientific field, J. Informetrics 15 (1) (2021) 101109.
[10] D. Mrdjenovich, et al., Propnet: a knowledge graph for materials science, Matter 2 (2) (2020) 464–480.
[11] S. Elhammadi, et al., A high precision pipeline for financial knowledge graph construction, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020.
[12] D. Cheng, et al., Knowledge graph-based event embedding framework for financial quantitative investments, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.
[13] P. Zheng, et al., Towards Self-X cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach, J. Manuf. Syst. 61 (2021) 16–26.
[14] H. Han, et al., Construction and evolution of fault diagnosis knowledge graph in industrial process, IEEE Trans. Instrum. Meas. 71 (2022) 1–12.
[15] X. Ma, Knowledge graph construction and application in geosciences: A review, Comput. Geosci. 161 (2022) 105082.
[16] L. Li, et al., Real-world data medical knowledge graph: construction and applications, Artif. Intell. Med. 103 (2020) 101817.
[17] X. Zhu, et al., Multi-modal knowledge graph construction and application: A survey, IEEE Trans. Knowl. Data Eng. 36 (2) (2022) 715–735.
[18] H. Ko, et al., Machine learning and knowledge graph based design rule construction for additive manufacturing, Addit. Manuf. 37 (2021) 101620.
[19] N. Kertkeidkachorn, R. Ichise, T2kg: An end-to-end system for creating knowledge graph from unstructured text, in: Workshops At the Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[20] X. Li, et al., PSDRNN: An efficient and effective HAR scheme based on feature extraction and deep learning, IEEE Trans. Ind. Inform. 16 (10) (2020) 6703–6713.
[21] B. Qiao, et al., A joint model for entity and relation extraction based on BERT, Neural Comput. Appl. (2022) 1–11.
[22] Y. Wang, et al., TPLinker: Single-stage joint extraction of entities and relations through token pair linking, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020.
[23] W. Tang, et al., UniRel: Unified representation and interaction for joint relational triple extraction, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 2022.
[24] S. Riedel, L. Yao, A. McCallum, Modeling relations and their mentions without labeled text, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III, vol. 21, Springer Berlin Heidelberg, 2010.
[25] C. Gardent, et al., Creating training corpora for nlg micro-planning, in: 55th Annual Meeting of the Association for Computational Linguistics, ACL, 2017.
[26] H. Zhao, et al., Explainability for large language models: A survey, ACM Trans. Intell. Syst. Technol. 15 (2) (2024) 1–38.
[27] H. Ye, et al., Contrastive triple extraction with generative transformer, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 16, 2021.
[28] D.W. Otter, J.R. Medina, J.K. Kalita, A survey of the usages of deep learning for natural language processing, IEEE Trans. Neural Netw. Learn. Syst. 32 (2) (2020) 604–624.
[29] M. Treviso, et al., Efficient methods for natural language processing: A survey, Trans. Assoc. Comput. Linguist. 11 (2023) 826–860.
[30] K.W. Church, Word2Vec, Nat. Lang. Eng. 23 (1) (2017) 155–162.
[31] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[32] A. Vaswani, et al., Attention is all you need, in: Advances in Neural Information Processing Systems 30, 2017.
[33] J.D.M.-W.C. Kenton, L.K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL-HLT, 2019.
[34] T. Wu, et al., A brief overview of ChatGPT:The history, status quo and potential future development, IEEE/CAA J. Autom. Sin. 10 (5) (2023) 1122–1136.
[35] J. Kocoń, et al., ChatGPT: Jack of all trades, master of none, Inf. Fusion 99 (2023) 101861.
[36] T. Xie, et al., Empirical study of zero-shot NER with ChatGPT, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023.
[37] D. Tam, et al., Optimal transport-based alignment of learned character representations for string similarity, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.
[38] X. Zhao, et al., An experimental study of state-of-the-art entity alignment approaches, IEEE Trans. Knowl. Data Eng. 34 (6) (2020) 2610–2625.
[39] S. Chen, et al., DASH: An agile knowledge graph system disentangling demands, algorithms, data resources, and humans, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022.
[40] X. Zeng, et al., Learning the extraction order of multiple relational facts in a sentence with reinforcement learning, 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, 2019.
[41] Z. Yan, et al., A partition filter network for joint entity and relation extraction, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021.
[42] X. Zeng, et al., Extracting relational facts by an end-to-end neural model with copy mechanism, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018.
[43] T.-J. Fu, P.-H. Li, W.-Y. Ma, Graphrel: Modeling text as relational graphs for joint entity and relation extraction, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.
[44] Z. Wei, et al., A novel cascade binary tagging framework for relational triple extraction, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.
[45] H. Zheng, et al., PRGC: Potential relation and global correspondence based joint relational triple extraction, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021.
[46] Y.-M. Shang, H. Huang, X. Mao, Onerel: Joint entity and relation extraction with one module in one step, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 10, 2022.
[47] Y. Lu, et al., Unified structure generation for universal information extraction, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022.

Lizhuang Sun received the M.S. degree from Peking University, Beijing, China, in 2023, and the B.E. degree from Dalian University of Technology, Dalian, China, in 2020.

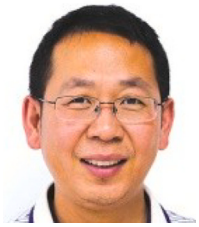
Since 2023, he has been a Research Assistant with Chang Guang Satellite Technology Company Ltd. His research interests include knowledge graphs, large language models, remote sensing image interpretation and computer vision.

Peng Zhang received the bachelor's degree in software engineering and the master's degree in computer application technology from Jilin University, Changchun, China, in 2014 and 2017, respectively.

Since 2017, he has been engaged in research on remote sensing image processing and applications, including remote sensing image modulation transfer function compensation, quality enhancement, image fusion, remote sensing image interpretation, and so on.

Fang Gao received the B.Sc. and Ph.D. degrees from Jilin University, Changchun, China, in 2010 and 2016, respectively.

Since 2016, he has been a Research Assistant with the Key Laboratory of Satellite Remote Sensing Application Technology of Jilin Province and Chang Guang Satellite Technology Company Ltd. His research interests include remote sensing image processing and artificial intelligence.



Yuan An received the B.Sc. degree from Jilin University, Jilin, China, in 2002, and the Ph.D. degree from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun, China, in 2006.

Since 2005, he has been working with Chang Guang Satellite Technology Company Ltd. From 2007 to 2014, he worked as a Professor with the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences. His research interests include the design of optical remote sensors and remote sensing applications.

Zhixing Li received the B.E. and M.S. degrees from Dalian University of Technology, Dalian, China, in 2020 and 2023, respectively.

Since 2023, he has been a Research Assistant with Chang Guang Satellite Technology Company Ltd. His research interests include remote sensing image processing, knowledge graphs, large language models, remote sensing applications, computer vision and artificial intelligence.

Yuanwei Zhao received the bachelor's and master's degrees from Jilin University, Changchun, China, in 2019 and 2022, respectively.

Since 2022, he has been a Research Assistant with Chang Guang Satellite Technology Company Ltd. His research interests include knowledge graphs, large language models, nature language processing, computer vision and artificial intelligence.