

Information Extraction from Biographies Using NLP (SpaCy) and LLM (GPT-4) to Populate a Knowledge Graph: Lessons Learnt

Christopher S.G. Khoo and Siam-Gek Ng
Wee Kim Wee School of Communication & Information
Nanyang Technological University, Singapore

Abstract

This preliminary report describes on-going effort to develop an information extraction method to extract entities and relations from nearly 450 biographies in the Singapore Infopedia to populate SingPioneers.sg—a knowledge graph of personalities significant in the history and development of modern Singapore. Two approaches are explored: a natural language processing approach (using the SpaCy Python package) and an LLM approach (using GPT-4o). We had initially planned to explore a third approach—developing a Transformer/BERT model to perform the task—but abandoned the idea when we found that using GPT-4o is much easier and likely to be more effective. Thus, this paper focuses on lessons learnt in “prompt engineering” to instruct GPT-4o to extract the desired entities and relations from the biography texts. We compare this with the NLP approach of constructing linear token patterns and graphical syntactic patterns (or dependency tree patterns) for entity-relation extraction using SpaCy.

Introduction

This paper reports on our effort to populate SingPioneers.sg—a knowledge graph of personalities significant in the history and development of modern Singapore, based on nearly 450 biographies in the Singapore Infopedia, an electronic encyclopedia of the Singapore National Library Board. A prototype knowledge graph application and visualization interface have been implemented at <https://SingPioneers.sg> using a Neo4j graph database management system as backend database, a Node.js Web API as middleware, and Cytoscape.js JavaScript library to implement a graph visualization Web interface. The prototype knowledge graph was handcrafted from a manual analysis of the biographies of 20 selected SingPioneers.

In the second phase of the project, we are developing an information extraction method to extract entity-relation information from the biographies to complete the knowledge graph. This paper reports on our ongoing work and lessons learnt in exploring two approaches to information extraction—using a natural language processing approach (using the SpaCy Python package) and an LLM approach (using GPT-4o). We had initially planned to explore a third approach—developing a Transformer/BERT model to perform the task—but abandoned the idea when we found that using GPT-4o is much easier and likely to be more effective. Thus, this paper focuses on lessons learnt in “prompt engineering” to instruct GPT-4o to extract the desired entities and relations from the biography texts. We compare this with the NLP approach of constructing linguistic patterns and dependency tree patterns for entity-relation extraction using SpaCy.

SingPioneers.sg Knowledge Graph

A graph is a network comprising nodes (or vertices) and links between nodes, called edges (or arcs). The links are often “directed”, in which case they are represented by arrows. In a knowledge graph, the nodes are assigned “meaning”: they represent persons, entities and concepts. In the SingPioneers knowledge graph, the entities represent persons, organizations, schools, hobbies, achievements, awards, creative works (e.g., songs, movies, etc.), economic sectors, and more. The links represent specific relations between the persons/entities/concepts. The types of relations in the knowledge graph include kinship and other types of interpersonal relations (e.g., teacher of, teammate of, friend of, coached by, mentor of). Other relation types include alumni of (a school), has business in (a particular commodity), composer of (a song), and well known for (some accomplishment). These conceptual relations between

entities/concepts are an important feature of knowledge graphs that make it different from other information representations.

Nodes can store attributes relating to the entity or concept represented (sometimes referred to as metadata), as well as text, hyperlinks and hyperlinks to images and files. Similarly, links can store attributes and information regarding the relation that it represents (e.g., the date range when the relation holds). In the SingPioneers visualisation interface, the attributes stored in the nodes and links are displayed in a pop-up info box.

The SingPioneers knowledge graph is basically a social network of persons (SingPioneers and significant persons in their lives), enhanced with additional information provided in the biographies. A biography in the Singapore Infopedia typically lists immediate family members, including wives, husbands, children, parents but also more distant relationships (e.g., ancestor, uncle, grandchild) to well-known persons. A biography will also list the interpersonal relations: school-related relations (classmates, teachers, students, talent-scouts), professional relations (employer, colleague, contemporary, team-mate in a sports team), and associates (fellow members of an association). Figure 1 is a screenshot of the SingPioneers interpersonal network (based on 20 SingPioneers). A graph focused on pioneer artist Chen Wen Hsi is illustrated in Figure 2.

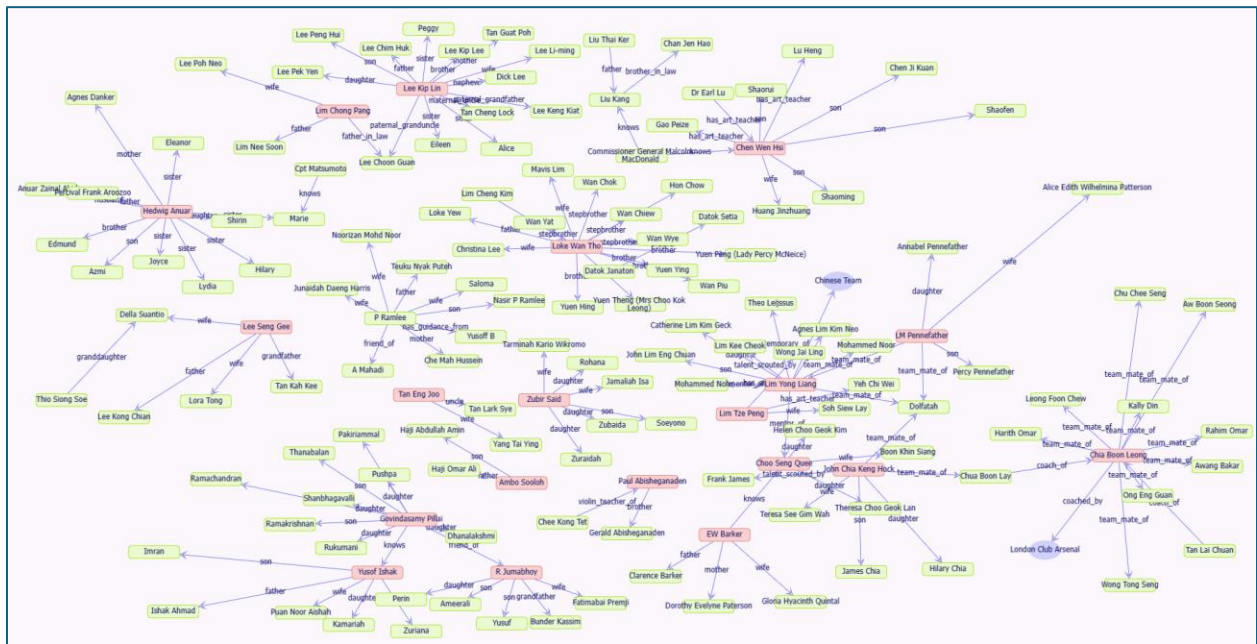


Figure 1. Interpersonal network for 20 SingPioneers

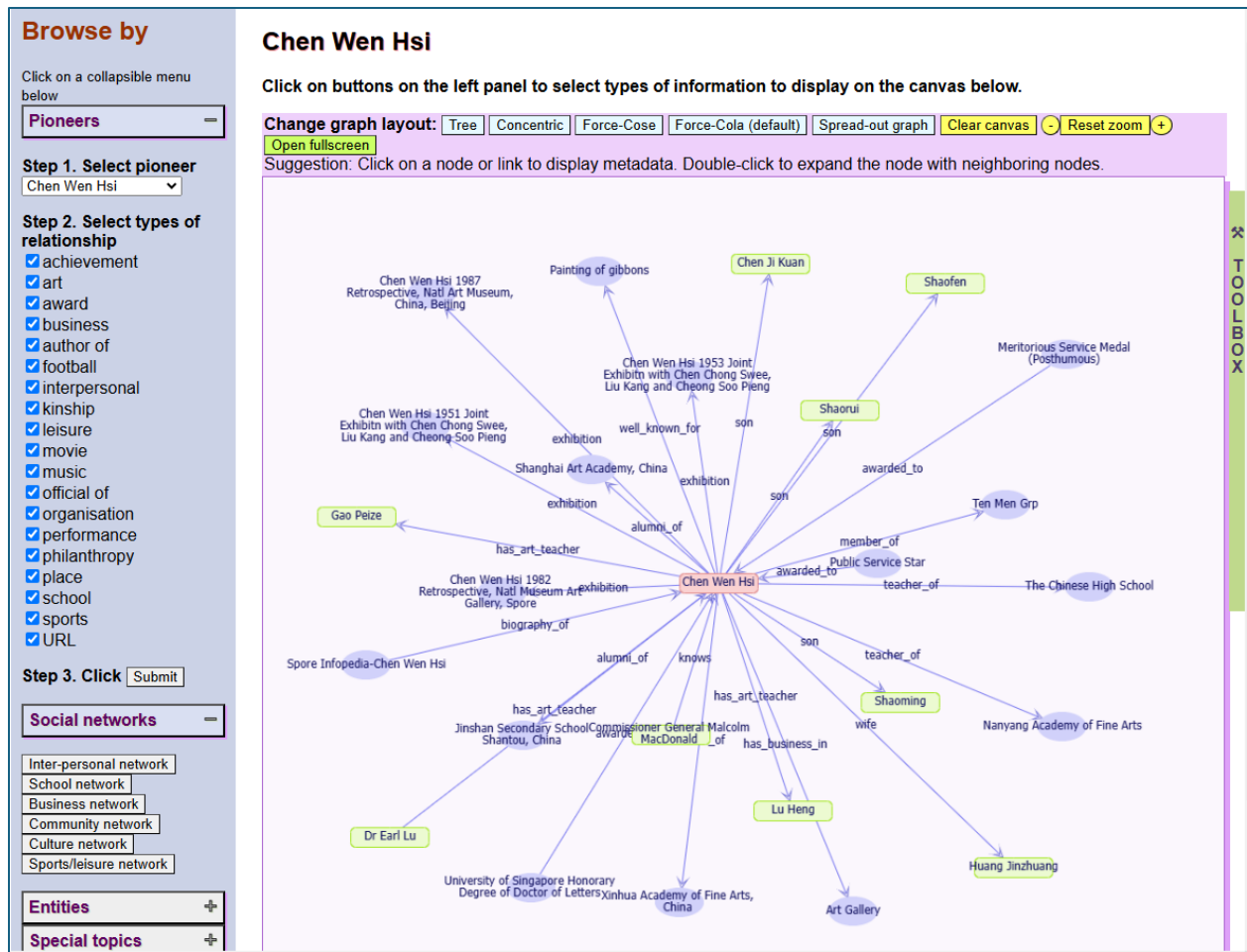


Figure 2. A graph focused on pioneer artist Chen Wen Hsi

Two Information Extractions Approaches: Linguistic Pattern Construction for SpaCy Versus Prompt Engineering for GPT-4o

We now describe the two information extraction approaches that we explored to complete the SingPioneers knowledge graph with entity-relations from nearly 450 biography texts.

Linguistic patterns for SpaCy

SpaCy is a powerful NLP package that offers two approaches to information extraction:

1. Using linear token patterns to identify the textual context (within a sentence) where an embedded text span (or text chunk) is expected to represent the desired entity or relation of interest;
2. Using graphical syntactic patterns to identify where in the syntactic parse tree of a sentence (represented as a dependency tree) the desired entity or relation is expected to be represented as a node or a subtree of the dependency tree.

In the former case (linear token patterns), the identified text span is extracted. Alternatively, the subtree of a text token can be extracted. In the latter case (graphical syntactic pattern), the text token occupying a node in the dependency tree is extracted. Alternatively, the text span representing the subtree of the token

can be extracted. In other words, the linear token representation and the dependency tree representation are linked, which allows for a combination of the two approaches to be used.

Consider the following two example sentences:

Sentence 1: Chen Wen Hsi (b. 9 September 1906, Guangdong, China–d. 17 December 1991, Singapore) was one of Singapore’s pioneer artists.

Sentence 2: Haji Ambo Sooloh (b. 1891–d. 1963) was a prominent Malay businessman of Bugis descent.

A simple token pattern for extracting the date of birth is:

```
Pattern1: [{'LOWER': 'b.'}, {'ENT_TYPE': 'DATE', 'OP': '{1,4}' }, {'IS_PUNCT': True}]
```

This pattern (in JSON format) lists three tokens, each token encapsulated in curly brackets `{}`. The first token indicates a text token converted to lowercase “b.”. The second token indicates the entity type of DATE, which can be repeated up to three times—that is, there can be one to four tokens tagged with the entity type DATE. The third token in the pattern indicates any punctuation mark. Thus, for each token, one can specify a combination of criteria (text string, lemma or root word, entity type, part-of-speech, and other linguistic features such as punctuation mark) to match.

When a match is found in the text for this pattern, additional code associated with the pattern specifies what text token or text span to extract. For this pattern, the text span tagged with the entity type DATE is extracted:

```
birthDate|9 September 1906  
birthDate|1891
```

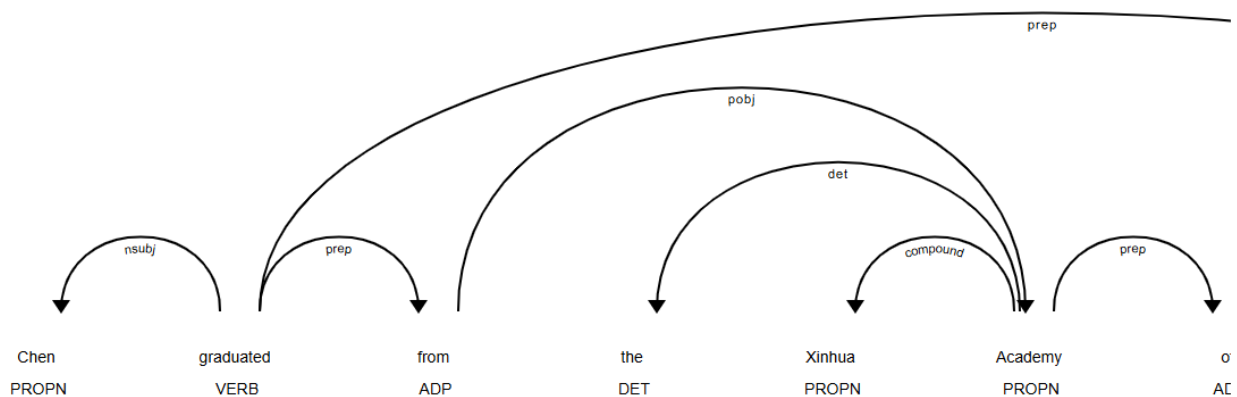
We have found the linear token patterns to be useful only for simple facts expressed in standard ways, without much variation in the text expression or overall sentence construction. Although it is possible to specify wildcard tokens in the token pattern, too many wildcard tokens make it difficult to precisely identify the desired text span to extract. In general, graphical syntactic patterns (i.e., dependency tree patterns) can identify the desired text span more precisely.

Consider the following sentence:

Sentence 3: Chen graduated from the Xinhua Academy of Fine Arts in 1929.

The syntactic dependency tree generated by SpaCy can be represented diagrammatically in this way:

Diagram 1:



A possible graphical pattern for extracting the school Chen was a student_at is as follows:

Pattern2:

```
[ {'RIGHT_ID': 'graduate', 'RIGHT_ATTRS': {'LEMMA': 'graduate'}},  
  {'LEFT_ID': 'graduate', 'REL_OP': '>', 'RIGHT_ID': 'prep', 'RIGHT_ATTRS': {'DEP': 'prep'}},  
  {'LEFT_ID': 'prep', 'REL_OP': '>', 'RIGHT_ID': 'school', 'RIGHT_ATTRS': {'ENT_TYPE':  
    'ORG'}}  
]
```

1. The first token in the pattern indicates the lemma (or root word) “graduate”, which matches with the second token in Sentence 3.
2. The second token in the pattern indicates the dependency relation:
graduate –prep→ <some token>
This will match with “graduated –prep→ from” in Diagram 1.
3. The third token indicates the dependency relation:
<some token> —> entity type: ORG
This will match with “from —> Academy” in Diagram 1.

After the “Academy” node in the dependency tree is identified, the code associated with this graphical pattern instructs the program to extract the subtree under the “Academy” node. Thus the text span “the Xinhua Academy of Fine Arts” is extracted.

It is clear that token patterns and dependency tree patterns are difficult to construct, requiring careful examination of example token tagging and dependency trees output by SpaCy, and meticulous pattern construction to give high precision and recall.

Prompt engineering for GPT-4o

Entity and relation extraction using ChatGPT and GPT-4o starts with “prompt engineering” to construct clear instructions to perform the information extraction, followed by a small number of example input/output to address errors (referred to as *few-shot learning*). Additional fine-tuning is possible using a training set of example input/output to create an additional refinement layer above GPT-4o.

Our prompts begin by specifying the overall purpose or task, followed by more specific task specifications that indicate subtypes of the task or example keywords. Then, the output format is specified. Our prompt to extract the birthdate and birthplace from Sentence 1 and 2 is as follows:

```
Your task is to carry out information extraction from the text given in triple quotes.  
Specifically, extract a person and his/her birthdate or birthyear, and birthplace if available.  
Output the result in the format: birth|person|birthdate|optional birthplace
```

This prompt is obviously easier to construct than linguistic patterns as it is in natural language and is closer to task instructions we might give to another human. Also, related pieces of information are specified together for extraction, including the person’s name and optionally the birthplace.

The following two prompts illustrate how relation subtypes can be specified. The second example requests for any date, year or year range associated with the relation.

```
Your task is to carry out entity and relation extraction from the text given in triple quotes.  
Specifically, extract kinship relations between persons.  
Kinship relations include: ancestor, brother, brother_in_law, daughter, father, father_in_law,  
granddaughter, grandfather, grandson, husband, maternal uncle, mother, nephew, paternal  
grandfather, paternal granduncle, sister, son, son_in_law, stepbrother, uncle, wife, child,  
grandchild, sibling, aunt, nephew, niece, granduncle, grandaunt  
Output the result in the format: has_relation|person1|person2
```

Your task is to carry out information extraction from the text given in triple quotes. Specifically, extract a person and his/her achievements and recognitions gained. Types of achievement include contribution, well_known_for, expertise, specialty, developer_of, inventor_of, donor_of, composer_of, creator_of, author_of. Types of recognition include award, named_after, appointment, honours. Also extract date, year or year range associated with the achievement or recognition, if available. Output the result in the format: type of achievement or recognition|person|achievement or recognition|optional date

If there are extraction errors in the GPT output, then a few example input/output can be provided to enable few-shot learning:

Your task is to carry out information extraction from the text given in triple quotes. Specifically, extract a person and his/her occupation or profession. Output the result in the format: Person-Occupation|person|occupation or profession

Here are a few example input and output:

Input: ""Sculptor Zhang Chenbo and ink painter Pan Tianshou were among his teachers.""
Output: Person-Occupation|Zhang Chenbo|sculptor
Output: Person-Occupation|Pan Tianshou|ink painter

Input: ""Chen also gave private art lessons, and among his students was Earl Lu, who became a prominent artist in his own right and an avid art collector.""
Output: Person-Occupation|Earl Lu|artist
Output: Person-Occupation|Earl Lu|art collector

As we have several types of relations to extract from the biography texts, we grouped the individual prompts into three multi-step prompts, so that we need only three passes to complete the entity-relation extraction task for SingPioneers.sg. An example of a multi-step prompt is provided in the Appendix.

Advantages of Using GPT-4o for Information Extraction

Our exploration of ChatGPT and GPT-4o for entity-relation extraction suggests that using LLM is the recommended approach to take in entity-relation extraction. In addition to ease of use and high accuracy of extraction, we found GPT-4o to have the following capabilities that are difficult to emulate using token pattern and graphical pattern matching:

1. Reference resolution across sentences: it can identify the full name of a person referred to by a pronoun or surname in a sentence. The full name may be given only at the beginning of the biography text.

Sentence: A prolific painter who worked in a range of styles, Chen won acclaim for combining Western art with Chinese brush strokes in his paintings.
Output: well_known_for|Chen Wen Hsi|combining Western art with Chinese brush strokes

Sentence: In 1948, Chen embarked on a trip to exhibit his works in Hanoi, Vietnam, and Singapore. He arrived in Singapore in 1949, and staged a one-man exhibition at the Chinese Chamber of Commerce with the help of the China Society of Singapore.
Output: arrival_in_Singapore|Chen Wen Hsi|1949|Vietnam

GPT-4o can resolve quite complex reference expressions:

Sentence: Ambo Sooloh had two wives and 10 children. One of them, Haji Abdullah Amin, was a Malay literary pioneer.

has_child|Haji Ambo Sooloh|Haji Abdullah Amin

Sentence: Haji Ambo is best remembered as one of the founders of the Malay newspaper Utusan Melayu, which he started together with Yusof Ishak (who later became the first president of Singapore) in 1939.

Output: Person-Occupation|Yusof Ishak|founder of Utusan Melayu

GPT-4o is able of infer the person's name not explicitly given in a table:

Table row: 1980: Gold medal, National Museum of History, Taipei, Taiwan.

Output: awarded|Chen Wen Hsi|Gold medal, National Museum of History, Taipei, Taiwan|1980

Table row: Wife: Huang Jingzhuang (Ng Kang Chan).

Table row: Children: Sons Shaorui, Shaoming and Shaofen.

Output: has_wife|Chen Wen Hsi|Huang Jingzhuang

Output: has_child|Chen Wen Hsi|Shaorui

Output: has_child|Chen Wen Hsi|Shaoming

Output: has_child|Chen Wen Hsi|Shaofen

2. Identify the scope of conjunctions: it can accurately extract individual entities linked by conjunctions and output them as separate relations. This is tedious to specify using token patterns and graphical patterns.

Sentence: In Singapore, Chen was employed as an art teacher at The Chinese High School from 1949 to 1968, and at NAFA from 1950 to 1959.

Output: teacher_at|Chen Wen Hsi|The Chinese High School|1949-1968

Output: teacher_at|Chen Wen Hsi|Nanyang Academy of Fine Arts|1950-1959

Sentence: Sculptor Zhang Chenbo and ink painter Pan Tianshou were among his teachers.

Output: has_teacher|Chen Wen Hsi|Zhang Chenbo

Output: has_teacher|Chen Wen Hsi|Pan Tianshou

Sentence: He opened the Old and New Gallery at Tanglin Shopping Centre in 1972, and the Chen Wen-Hsi Gallery at the Singapore Handicraft Centre in 1976.

founder_of|Chen Wen Hsi|Old and New Gallery at Tanglin Shopping Centre|1972

founder_of|Chen Wen Hsi|Chen Wen-Hsi Gallery at the Singapore Handicraft Centre|1976

3. Date extraction: for an entity-relation extracted, GPT-4o can identify the associated date, year or year range. Dates are, of course, easy to identify using NLP, but they are tedious to include in linguistic patterns as they can occur at many syntactic locations in a sentence. It is also difficult to determine which relation the date applies to, if there are multiple relations in a sentence. The following examples illustrate reference resolution and date extraction across three sentences:

Sentence: Subsequently, a group of teachers left the academy and set up a private art college known as **Xinhua Academy of Fine Arts**. **Chen transferred to this school** and became

classmates with Liu Kang, Chan Jen Hao and Chen Chong Swee, all of whom would later become Singapore’s pioneer artists. Chen **graduated from the academy in 1929**.
Output: student_at|Chen Wen Hsi|Xinhua Academy of Fine Arts|1929

Some Issues Encountered

In this section, we describe two issues that we encountered in the entity-relation extraction.

In the case when a relation involves two entities of the same type (e.g., kinship relation between two persons), GPT-4o was not consistent in representing the direction of the relation. For example,

son|Person1|Person2

can indicate “Person1 has son Person2” or “Person1 is the son of Person2”. To help GPT-4o handle this issue, we provided a few example input and output. This seemed to reduce the errors but GPT-4o was still occasionally confused. We amended the output format from “son|Person1|Person2” to “has_son|Person1|Person2”. Specifying a relation with a clear direction appears to have solved the problem.

A second issue is that when several instances of a relation type occur in a text, GPT-4o may not output all the relation instances. Adding the instruction “Provide comprehensive results.” to the end of the prompt did not make GPT-4o output more comprehensive results. However, a subsequent prompt of “more” or “try harder” may elicit more extractions. Thus, whenever we expect several instances of a relation to be extracted, we would submit a subsequent “more” after the initial output.

A formal evaluation of entity-relation extraction from the biographies using GPT-4o is in progress, and further improvements to the prompts will be made. GPT-4o supports fine-tuning of the model based on a training set. The examples above suggest that a fine-tuning layer may not be necessary: entity-relation extraction by GPT-4o from biography texts is surprisingly good.

Further Work to Extract Cause-Effect Information from Social Science Research Articles

The biography texts in Singapore Infopedia use mainly everyday language and concepts, and is meant to be readily understood by the general public. The entity and relation types that are the target of the information extraction effort are also everyday concepts and relations. We wondered whether GPT-4o can handle more specialized or abstract relation types. So, we carried out preliminary exploration for our next entity-relation extraction project—to tag and extract cause-effect information from social science research papers.

We constructed a simple prompt to ask GPT-4o to “identify cause-effect relations found in the text given in triple quotes, and to tag text spans with the following cause-effect roles: cause_concept, effect_concept, modality (e.g. probably, possibly, negation), polarity (e.g., positive, negative), effect_size (e.g., strong, weak), moderator, mediator, qualifier, effect_type.” Here is an example output:

This study examines the underlying dimensions of co-creation of an experience in the context of tourism and its effects on <effect_concept>behavioral consequences such as tourists' satisfaction with the co-creation of an experience, subjective well-being, and loyalty to the service provider</effect_concept>.

The purpose of the study is achieved by showing that <cause_concept>tourists' co-creation of an experience</cause_concept> <polarity>positively</polarity> affects the <effect_concept>vacation experience</effect_concept> and <effect_concept>loyalty to the service provider</effect_concept>.

<effect_concept>Satisfaction with the vacation experience</effect_concept> <polarity>positively</polarity> influences <effect_concept>overall life satisfaction</effect_concept>.

The results of this study should help <moderator>service providers</moderator> <effect_type>change strategies</effect_type> and implement a platform for creating unique

<qualifier>co-creation of experiences</qualifier>, allowing tourists to become more
<effect_size>physically and emotionally engaged</effect_size> in the <cause_concept>planning
of their vacations</cause_concept>.

The cause-effect tagging by GPT-4o is better than we expected, but improvements in the prompt and example output are needed. Fine-tuning with a training corpus may also be needed. This task is clearly more difficult than entity-relation extraction from biography text.

APPENDIX. Example multi-step prompt for extracting multiple sets of relations

Your task is to carry out entity and relation extraction from the text given in triple quotes.
The task is divided into the following steps.

Step 1 - Extract the relation between a person and a school.

Relations of interest include: alumni_of, principal_of, sport_master_of, teacher_at, primary_schooling_at, secondary_schooling_at, tertiary_education_at, student_at.

Also extract date, year or year range associated with the relation, if available.

Output the result in the format: type of relation|person|school|optional date

Step 2 - Extract the relation between a person and an organization or group of people.

Relations of interest include: chairman_of, co_founder_of, director_of, founder_of, honorary_secretary_of, leader_of, member_of, president_of, singer_with, violinist_with, worked_for, vice_president_of, editor_of, co_editor_of, owner_of, manager_of, treasurer_of, secretary_of, adviser_to, captain_of, partner_at, board_member_of, committee_member_of, founder_member_of, sportsperson_with, conductor_of, coach_of, played_for.

Also extract date, year or year range associated with the relation, if available.

Exclude the relations extracted in earlier steps.

Output the result in the format: type of relation|person|organization|optional date

Step 3 - Extract a person and his/her achievements and recognitions gained.

Types of achievement include contribution, well_known_for, expertise, specialty, developer_of, inventor_of, donor_of, author_of, composer_of, creator_of, won.

Types of recognition include award, named_after, appointment, honours.

Also extract date, year or year range associated with the achievement or recognition, if available.

Exclude the relations extracted in earlier steps.

Output the result in the format: type of achievement or recognition|person|achievement or recognition|optional date