Assignment 2: Knowledge Graph Population

Task 1 (50%)

Using an LLM of your choice, implement a prompt-based relation classifier that takes as input a sentence and two entities, and determines if and which of the above four relations are true in the sentence. If the input sentence does not express any of these relations the system should return "Unknown". Use the provided dataset relation_extraction_dataset.tsv (available here) to evaluate the precision and recall of your classifier for each relation. In addition, inspect the errors the system makes and see if you can identify any systematic error patterns.

Answer:

We load the data

```
import pandas as pd
data = pd.read_csv('assignment3_dataset.tsv', sep='\t')
data
```

✓ 0.2s

	text	subject	object	relation_in_sentence
0	All John wants is to live in Palo Alto.	John	Palo Alto	unknown
1	Although Jane was married, John was not.	John	Jane	unknown
2	Apple does not employ John.	John	Apple	unknown
3	Apple extended a job offer to John.	John	Apple	unknown
4	Apple gave John an offer for a job.	John	Apple	unknown
...	...	...	...	...
315	John was married to Jane	John	Jane	spouse
316	John worked for Apple.	John	Apple	employee_of
317	John works for Apple.	John	Apple	employee_of
318	John's Alma Mater is Stanford University.	John	Stanford University	schools_attended
319	John's wife is Jane.	John	Jane	spouse

320 rows × 4 columns

Across the whole exercise we will use local models:

Models tried are:

- Llama3.2:3b-instruct-fp16

FIrst prompt tried:

```
"""Analyze the sentence to identify if there is a clear, explicitly
stated relation between the subject and object entities. Return ONLY
a single JSON key-value pair with "relation" as the key and the
matched relation or "Unknown" as the value.
```

```
Input:
- Sentence: {row['text']}
- Subject: {row['subject']}
- Object: {row['object']}

Consider these relations ONLY if directly and unambiguously stated in
the sentence:
- cities_of_residence: relates a person to cities where their
physical residence is explicitly mentioned in a factual way
- employee_of: relates a person to organizations where their
employment status is explicitly mentioned through clear terms like
"works for", "is employed by", "joined", etc.
- schools_attended: relates a person to educational institutions
where their student status is explicitly mentioned through clear
terms like "studies at", "attended", "graduated from", etc.
- spouse: relates a person to persons where their marriage status is
explicitly mentioned through clear terms like "married to", "wed",
etc.

Return "Unknown" if ANY of these conditions exist:
1. The relation is implied but not explicitly stated
2. Any temporal ambiguity exists about when the relation occurred
3. The sentence contains qualifiers, modalities, or uncertainties
4. The relation direction is not 100% clear
5. Multiple interpretations of the relationship are possible
6. The sentence uses future tense or hypotheticals
7. The relationship is mentioned in passing or as background
information
8. Any nuanced context that requires interpretation
9. The sentence describes wishes, plans, or intentions
10. The relation is negated or questioned

Rules
- Do not add any introduction or conclusion to the response

Response"""
```

Evaluation:

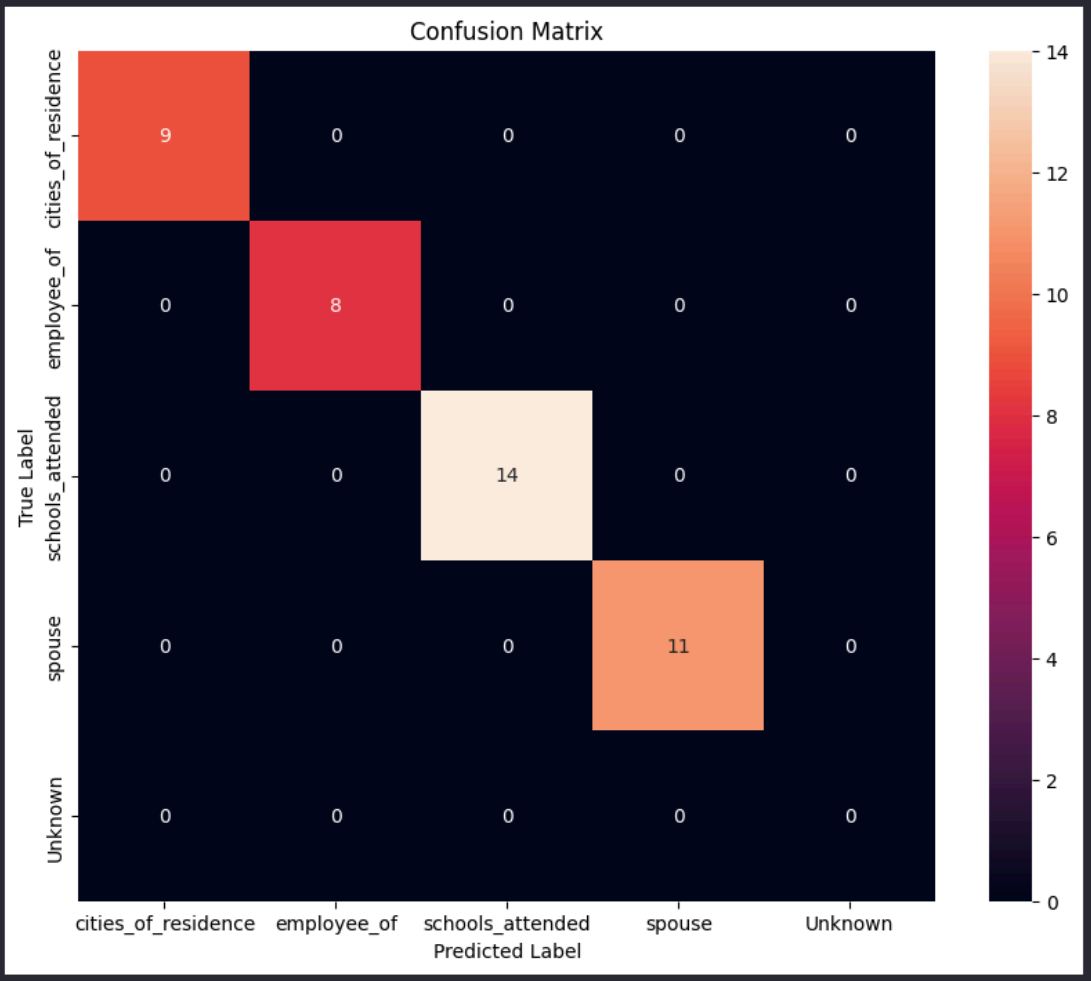
```
=== Relation Extraction Evaluation Report ===

Per-Class Metrics:
           Relation  Precision  Recall       F1  Support
cities_of_residence   1.000000     0.9 0.947368       10
        employee_of   1.000000     1.0 1.000000        8
   schools_attended   1.000000     1.0 1.000000       14
            spouse   0.785714     1.0 0.880000       11
           Unknown   0.000000     0.0 0.000000        0

Overall Accuracy: 98.75%

Per-Class Accuracy:
cities_of_residence: 100.00%
employee_of: 100.00%
schools_attended: 100.00%
spouse: 100.00%
Unknown: nan%

Class Distribution:
unknown: 277 samples (86.6%)
schools_attended: 14 samples (4.4%)
spouse: 11 samples (3.4%)
cities_of_residence: 10 samples (3.1%)
employee_of: 8 samples (2.5%)
/tmp/ipykernel_14154/2363292951.py:42: RuntimeWarning: invalid value encountered in divide
  class_accuracy = cm.diagonal() / cm.sum(axis=1)
```



Adding few shots

```
Analyze the sentence to identify if there is a clear, explicitly
stated relation between the subject and object entities. Return ONLY
a single JSON key-value pair with "relation" as the key and the
matched relation or "Unknown" as the value.

Input:
- Sentence: {row['text']}
- Subject: {row['subject']}
- Object: {row['object']}

Consider these relations ONLY if directly and unambiguously stated in
the sentence:
- cities_of_residence: relates a person to cities where their
physical residence is explicitly mentioned in a factual way
- employee_of: relates a person to organizations where their
employment status is explicitly mentioned through clear terms like
"works for", "is employed by", "joined", etc.
- schools_attended: relates a person to educational institutions
where their student status is explicitly mentioned through clear
terms like "studies at", "attended", "graduated from", etc.
- spouse: relates a person to persons where their marriage status is
explicitly mentioned through clear terms like "married to", "wed",
etc.

Return "Unknown" if ANY of these conditions exist:
1. The relation is implied but not explicitly stated
2. Any temporal ambiguity exists about when the relation occurred
3. The sentence contains qualifiers, modalities, or uncertainties
4. The relation direction is not 100% clear
5. Multiple interpretations of the relationship are possible
6. The sentence uses future tense or hypotheticals
7. The relationship is mentioned in passing or as background
information
8. Any nuanced context that requires interpretation
9. The sentence describes wishes, plans, or intentions
10. The relation is negated or questioned

Examples:
Elizabeth wants to live in New York. -> Unknown
```

```
Elizabeth lives in New York. -> cities_of_residence
Elizabeth is employed by Google. -> employee_of
Elizabeth has been studying at Harvard. -> Unknown
Tesla would be lucky to have John as an employee -> Unknown
Jill and Jack are getting married. -> Unknown
Jill and Jack are married. -> spouse

Rules
- Do not add any introduction or conclusion to the response

Response:
```

Evaluation:

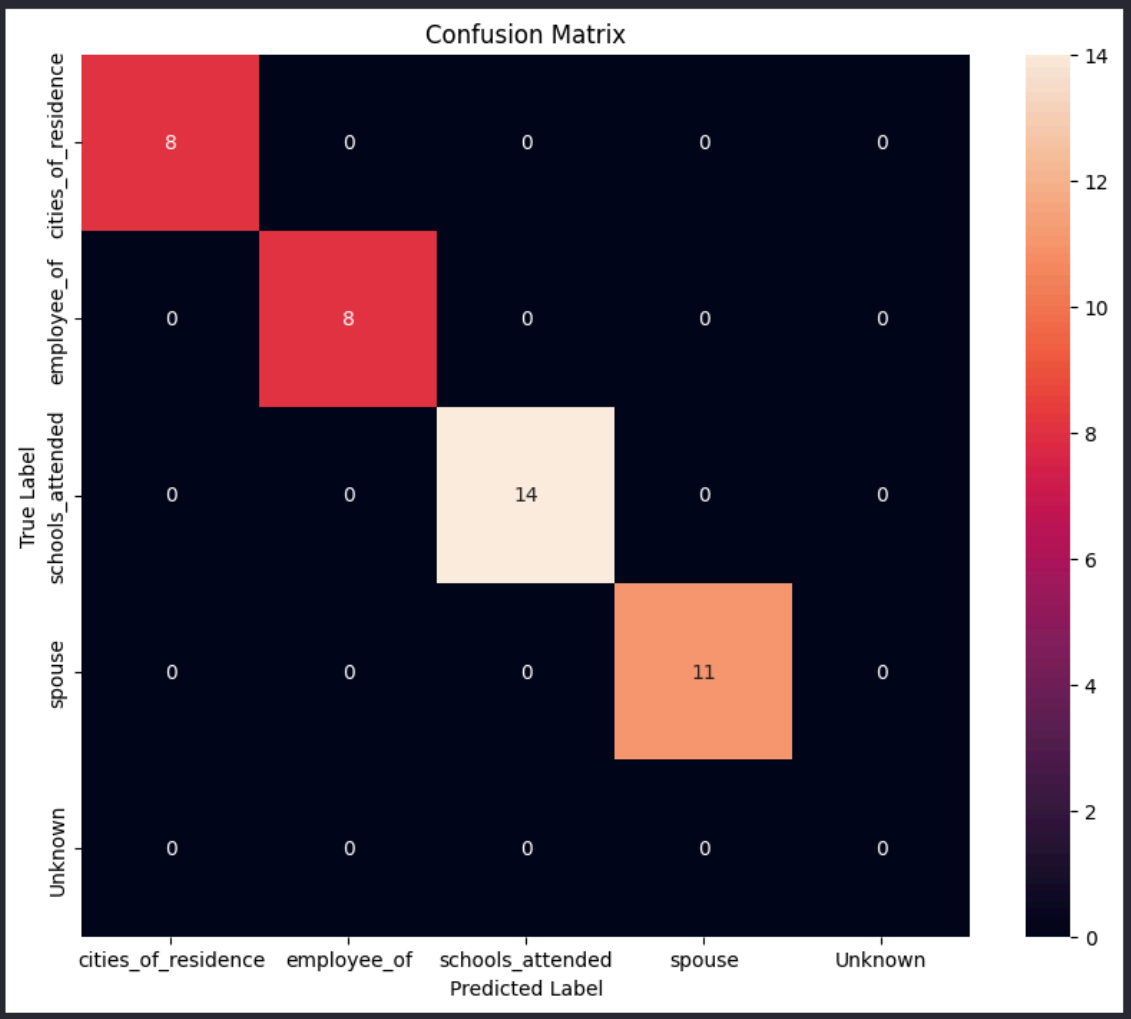
```
=== Relation Extraction Evaluation Report ===

Per-Class Metrics:
           Relation  Precision  Recall       F1  Support
cities_of_residence   1.000000     0.8 0.888889       10
        employee_of   1.000000     1.0 1.000000        8
   schools_attended   1.000000     1.0 1.000000       14
            spouse   0.916667     1.0 0.956522       11
           Unknown   0.000000     0.0 0.000000        0

Overall Accuracy: 99.06%

Per-Class Accuracy:
cities_of_residence: 100.00%
employee_of: 100.00%
schools_attended: 100.00%
spouse: 100.00%
Unknown: nan%

Class Distribution:
unknown: 277 samples (86.6%)
schools_attended: 14 samples (4.4%)
spouse: 11 samples (3.4%)
cities_of_residence: 10 samples (3.1%)
employee_of: 8 samples (2.5%)
/tmp/ipykernel_14154/2363292951.py:42: RuntimeWarning: invalid value encountered in divide
  class_accuracy = cm.diagonal() / cm.sum(axis=1)
```



Task 2 (50%)

Answer

Creating the dataset

```
def generate_test_dataset():
    """
    Generate test dataset for uncertainty and advice/wish scenarios
    """
    # Test cases for uncertainty
    uncertainty_cases = [
        # cities_of_residence uncertainty
        ("It's possible that Maria lives in Seattle", "Maria",
"Seattle", "cities_of_residence", "unknown"),
        ("There's a chance John moved to Boston last year", "John",
"Boston", "cities_of_residence", "unknown"),
        ("Sarah might be residing in London now", "Sarah", "London",
"cities_of_residence", "unknown"),
        ("Reports suggest that Tom could be living in Paris", "Tom",
"Paris", "cities_of_residence", "unknown"),
        ("I heard James may have moved to Tokyo", "James", "Tokyo",
"cities_of_residence", "unknown"),

        # employee_of uncertainty
        ("Emma might work at Google", "Emma", "Google", "employee_of",
"unknown"),
        ("There are rumors that David is employed by Microsoft",
"David", "Microsoft", "employee_of", "unknown"),
        ("It's possible that Lisa joined Amazon recently", "Lisa",
"Amazon", "employee_of", "unknown"),
        ("Sources suggest Alex could be working at Tesla", "Alex",
"Tesla", "employee_of", "unknown"),
        ("I think Robert might be employed at Facebook", "Robert",
"Facebook", "employee_of", "unknown"),

        # schools_attended uncertainty
        ("Kevin might be studying at Stanford", "Kevin", "Stanford",
"schools_attended", "unknown"),
        ("There's a possibility that Anna enrolled at Harvard",
"Anna", "Harvard", "schools_attended", "unknown"),
        ("I heard Peter could be attending MIT", "Peter", "MIT",
```

```
"schools_attended", "unknown"),
        ("Reports indicate that Rachel might be at Yale", "Rachel",
"Yale", "schools_attended", "unknown"),
        ("Sophie possibly studied at Oxford", "Sophie", "Oxford",
"schools_attended", "unknown"),

        # spouse uncertainty
        ("There are rumors that Mike and Jessica might be married",
"Mike", "Jessica", "spouse", "unknown"),
        ("It's possible that Eric and Diana got married", "Eric",
"Diana", "spouse", "unknown"),
        ("People say Chris and Laura could be married", "Chris",
"Laura", "spouse", "unknown"),
        ("I heard that Paul and Linda might have tied the knot",
"Paul", "Linda", "spouse", "unknown"),
        ("There's speculation that Mark and Amy are married", "Mark",
"Amy", "spouse", "unknown"),
    ]

    # Test cases for advice/wish
    advice_wish_cases = [
        # cities_of_residence advice/wish
        ("Jake should move to Chicago", "Jake", "Chicago",
"cities_of_residence", "unknown"),
        ("I wish Emily would live in San Francisco", "Emily", "San
Francisco", "cities_of_residence", "unknown"),
        ("It would be great if Daniel moved to Miami", "Daniel",
"Miami", "cities_of_residence", "unknown"),
        ("Helen ought to consider living in Vancouver", "Helen",
"Vancouver", "cities_of_residence", "unknown"),
        ("I hope Nathan relocates to Austin", "Nathan", "Austin",
"cities_of_residence", "unknown"),

        # employee_of advice/wish
        ("Susan should apply to work at IBM", "Susan", "IBM",
"employee_of", "unknown"),
        ("I wish Brian would join Apple", "Brian", "Apple",
"employee_of", "unknown"),
        ("It would be nice if Karen worked at Netflix", "Karen",
"Netflix", "employee_of", "unknown"),
        ("Tim ought to consider working at Intel", "Tim", "Intel",
"employee_of", "unknown"),
```

```
        ("I hope Michelle gets a job at Twitter", "Michelle",
"Twitter", "employee_of", "unknown"),

        # schools_attended advice/wish
        ("Andrew should attend Princeton", "Andrew", "Princeton",
"schools_attended", "unknown"),
        ("I wish Julia would study at Columbia", "Julia", "Columbia",
"schools_attended", "unknown"),
        ("It would be great if Rick enrolled at Berkeley", "Rick",
"Berkeley", "schools_attended", "unknown"),
        ("Emma ought to consider attending UCLA", "Emma", "UCLA",
"schools_attended", "unknown"),
        ("I hope Patrick goes to Cambridge", "Patrick", "Cambridge",
"schools_attended", "unknown"),

        # spouse advice/wish
        ("Steve and Mary should get married", "Steve", "Mary",
"spouse", "unknown"),
        ("I wish Jack and Kate would tie the knot", "Jack", "Kate",
"spouse", "unknown"),
        ("It would be nice if Tom and Sarah got married", "Tom",
"Sarah", "spouse", "unknown"),
        ("Dave and Lisa ought to consider marriage", "Dave", "Lisa",
"spouse", "unknown"),
        ("I hope Bill and Nancy get married someday", "Bill", "Nancy",
"spouse", "unknown"),
    ]

    # Combine all test cases
    all_cases = uncertainty_cases + advice_wish_cases

    # Create DataFrame
    df = pd.DataFrame(all_cases, columns=['text', 'subject', 'object',
'relation_type', 'relation_in_sentence'])

    return df
```

Testing with the previous best prompt

```
=== Relation Extraction Evaluation Report ===

Per-Class Metrics:
Relation  Precision  Recall  F1  Support
 unknown        1.0     1.0 1.0       40

Overall Accuracy: 100.00%

Per-Class Accuracy:
unknown: 100.00%

Class Distribution:
unknown: 40 samples (100.0%)
/home/jason/Documents/coding_repos/chatbot_app/.venv/lib/python3.11/site-packages/sklearn/metrics/_class
  warnings.warn(
```