Deep Generalization based on Types

B659 - Project Output Description

Paventhan Vivekanandan

Indiana University Bloomington

Output Description

Generating Graph

The program has two function calls in the main function: grow_type_tree() and generate_query(). Run one of them at a time (comment the other). If we run grow_type_tree(), then we get a knowledge graph as given in fig. 1. Here, we have six types "t1" to "t6". Type "t1" is the super type for "t2" and "t3". Type "t5" is the super type for "t4" and "t6". Any inferred edges or nodes or properties will start with string "inf_". Red edges denotes super type to sub type relationship. Blue denotes *inferred* super type to sub type relationship. Brown denotes relationship based on direct evidence and yellow denotes *inferred* relationships.

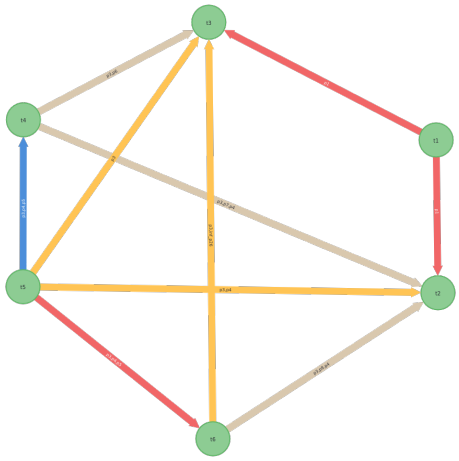

Fig. 1: Explanation Generation

Generating Explanations

The following are the examples of queries and generated explanations for fig. 1.

Query 1 Here, a set of properties is send as input and the most relevant type is returned. Also, the system explains its decision by listing the properties of the returned type and giving a relevance score. Also, the system returns a list of the next best matches. This is similar to a conversational case-based reasoner which returns the the list of most relevant questions or answers to queries based on relevance scores.

Listing 1.1: Python output for Query 1

```
# for call anwser_query_with_explanation(["p2", "p3", "p4"], "", "", 1)
$ python3 tbr.py
Query Type 1:

The best matching type is t4 with properties ['p2', 'p3', 'p4', 'p5']
and score 3

The next best matches based on scores:
type: t5, properties: ['p3', 'p4', 'p5'], score: 2
type: t6, properties: ['p3', 'p4', 'p9'], score: 2
=======================================

$
```

In listing. 1.1, the list of properties [”p2”, ”p3”, ”p4”] is send as input and the system returns type ”t4” (see fig. 1) as the most relevant type with similarity score 3. The next best matches are also returned by the system.

Listing 1.2: Python output for Query 2

```
# for call anwser_query_with_explanation([], "t6", "t3", 2)
$ python3 tbr.py
Query Type 2:

t6 relates to t3 based on properties ['p3', 'inf_p26']

I am inferring the edge because of the following reasons:
The properties ['p3'] of t6 is common with sibling t4
The inferred property ['inf_p26'] of t6 relates to ['p6'] of t4
=======================================

$
```

Query 2 In query 2, we send two types as input and the system retrieves the most similar properties. Also, it explains its choice by giving the closest

matching sibling. After that, if the returned type has inferred properties, it gives information on what that properties might be. In listing. 1.2, the types "t6" and "t3" are send as input and the system returns the properties ["p3", "inf_26"] (see fig. 1). It explains that the sibling "t4" has the closest match. The system also relates the inferred property "inf_26" to "p6".

Query 3 Query 3 is a more generalized version of query 2. Here, we send a type and a list of properties as input and the system returns the most relevant type. In addition to the best match, the system also returns the next best matches with relevance scores. Again, the system explains its choices for inferences.

Listing 1.3: Python output for Query 3

```
# for call anwser_query_with_explanation(["p3", "p4"], "t6", "", 3)
$ python3 tbr.py
Query Type 3:

The best match to type t6 is type t2 with properties ['p3', 'p8', 'p4']
and score 2

The next best matches based on scores:
type: t3, properties: ['p3', 'inf_p26'], score: 1

t6 relates to t3 based on properties ['p3', 'inf_p26']

I am inferring the edge because of the following reasons:
The properties ['p3'] of t6 is common with sibling t4
The inferred property ['inf_p26'] of t6 relates to ['p6'] of t4
=====================================

$
```

In listing. 1.3, we send list of properties ["p3", "p4"] and type "t6" as input and the system returns the most relevant type "t2" (see fig. 1). Since the properties ["p3", "p8", "p4"] common to input "t6" and output "t2" are not inferred (they are from data evidence), the system does not make any inferences. The system also returns the next best matches. For the next best match, we have an inferred property for which the system explains the reason for making the inference.