Ontology Learning with FCA

Paletto Andrea, Tuninetti André

September 12, 2023

Contents

1 Introduction

In this document, we present the implementation of an Ontology Learning system using Formal Concept Analysis (FCA) with Python.

- Your choice (both the "concepts" and the features).
  - Example features: semantic properties, related words, synsets, occurrence in documents, etc.
- Example (useful but not limited to):
  - Given two input languages (e.g., Italian and English):
    - Concepts: terms (from both languages).
    - Features: membership synsets.

2 Methodology

2.1 Functions

In our implementation, we use two different types of data: two lists of words (English and Italian), and a topic retrieved from Wikipedia using the API. We have implemented the following functions:

- **get_info(syns)**

  This function takes a list of WordNet synsets (Word Senses) as input and extracts information about their parts of speech and names. It does the following:

  1. Initializes an empty set, `pos_set`, to track unique parts of speech.
  2. Initializes an empty list, `info`, to store information about the synsets.
  3. Iterates through the synsets:
     - Retrieves the part of speech using `syn.pos()`.
     - Checks if the part of speech has been encountered before. If yes, it skips to the next iteration.
     - Adds the part of speech to `pos_set` to track unique occurrences.
     - Appends a corresponding string to `info` ('noun' for 'n', 'adjective' for 'a', 'verb' for 'v') based on the part of speech.
     - If there are synsets (not an empty list), appends the name of the first synset to `info`.
  4. Returns the `info` list containing part of speech and synset name information.

- **get_data()**

  The `get_data` function sets up a user agent and uses the Wikipedia API to retrieve summaries for specific Wikipedia pages, which are defined in the 'titles' list. These summaries are then joined together and returned as a single string.

- **extract_concepts(content)**

  The `extract_concepts` function processes the content by tokenizing it into words, removing stopwords and non-alphabetic tokens, and extracting nouns. These nouns are considered as concepts.

- **create_definitions(words)**

  The `create_definitions` function generates definitions for the given words using WordNet synsets. It does the following:

  1. Initializes a `Definition` object named `d` to store word definitions.
  2. Iterates through each word in the combined list of English and Italian.
     - Determines the language of the word ('eng' for English and 'ita' for Italian).
     - Retrieves the synsets of the word from WordNet using `wn.synsets()`.
     - Calls the `get_info()` function to extract information about the synsets (part of speech and name).
     - Adds the word and its associated information to the `d` object using `d.add_object()`.
  3. Creates a `Context` named `c` using the populated `Definition` object `d`.

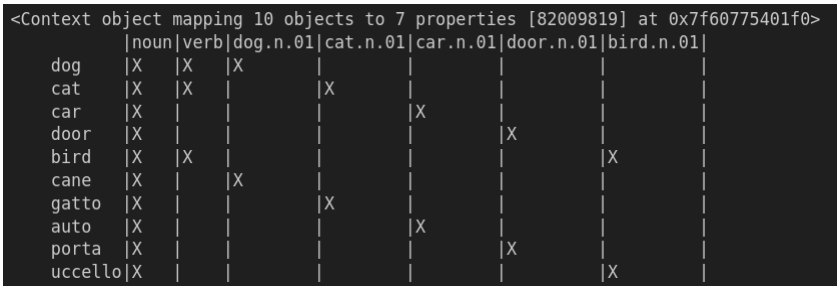
2.2 Workflow

The workflow of the code involves the following steps:

1. Declare English and Italian words.
2. Retrieve data from Wikipedia using the `get_data()` function.
3. Extract concepts from the retrieved data using the `extract_concepts()` function.
4. Create word definitions for the extracted concepts using the `create_definitions()` function.
5. Visualize the lattice of the context and print the context itself.

3 Conclusion

Here are the concept-feature matrices and lattice representations for both scenarios:
First versin with english and italian word:



```
<Context object mapping 10 objects to 7 properties [82009819] at 0x7f60775401f0>
          |noun|verb|dog.n.01|cat.n.01|car.n.01|door.n.01|bird.n.01|
    dog   |X   |X   |X       |        |         |         |         |
    cat   |X   |X   |        |X       |         |         |         |
    car   |X   |    |        |        |X        |         |         |
    door  |X   |    |        |        |         |X        |         |
    bird  |X   |X   |        |        |         |         |X        |
    cane  |X   |    |X       |        |         |         |         |
    gatto |X   |    |        |X       |         |         |         |
    auto  |X   |    |        |        |X        |         |         |
    porta |X   |    |        |        |         |X        |         |
    uccello|X  |    |        |        |         |         |X        |
```

Figure 1: table

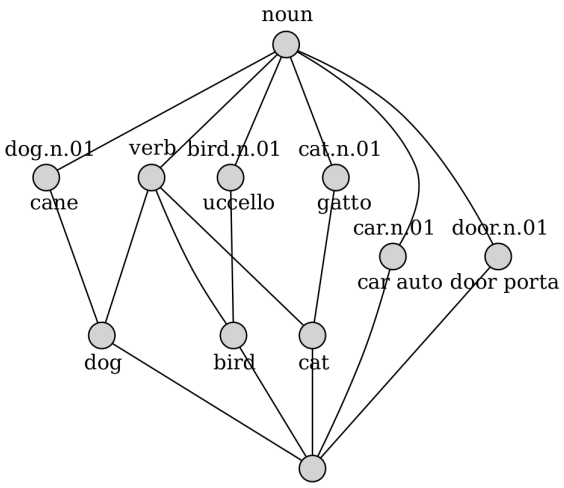


Figure 2: lattice

Second version with text retrive using wikipedia api:

```
<Context object mapping 9 objects to 10 properties [65422e3e] at 0x7f60775e0130>
           |noun|rivera.n.01|baseball.n.01|pitcher.n.01|verb|season.n.01|league.n.01|york.n.01|yankee.n.01|career.n.01|
   mariano |    |           |             |            |    |           |           |         |           |           |
   rivera  |X   |X          |             |            |    |           |           |         |           |           |
   baseball|X   |           |X            |            |    |           |           |         |           |           |
   pitcher |X   |           |             |X           |    |           |           |         |           |           |
   seasons |X   |           |             |            |X   |X          |           |         |           |           |
   league  |X   |           |             |            |X   |           |X          |         |           |           |
   york    |X   |           |             |            |    |           |           |X        |           |           |
   yankees |X   |           |             |            |    |           |           |         |X          |           |
   career  |X   |           |             |            |X   |           |           |         |           |X          |
```

Figure 3: table

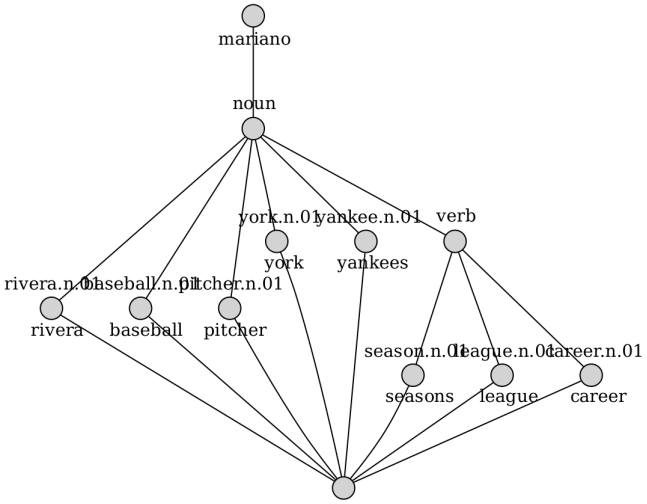


Figure 4: lattice