Stereotype analysis with Prolog

Ester Molinari

August 26, 2024

Abstract

Stereotypes are generalized belief about a particular category of people and are used as expectation that people might have about every person of a particular group. They are often overgeneralized, inaccurate, and resistant to new information and they can be positive, neutral, or negative. The main idea of this project is to analyze and extract classic explicit stereotypes from everyday language through logic programming. The full system code can be found on GitHub.

1 Introduction

The term stereotype derives from the Greek words *stereos*, which means 'firm, solid' and *typos*, which means 'impression', hence 'solid impression on one or more ideas or theories'. Thus, from this concept comes the idea of representing some of the most commonly used gender stereotypes as facts of a knowledge base made with Prolog.

This project focuses on role-based stereotypes, such as jobs, but also on trait-based stereotypes, such as epithets related to males and females people. A generative grammar will be used to create sentences following the classical stereotyped structure on which will be applied techniques to extract subject-object or subject-adjective pairs that lead back to stereotypes.

2 Related Work

Gender stereotypes have been studied over the years bringing out concordant results among various researches. The information from the following studies was crucial to create the system's knowledge base.

As shown in this study [Ell18], gender stereotypes implicitly impact the expectations we have about the qualities, priorities, and needs of individual men and women, as well as the standards.

A classic example is when women become parents. We tend to assume that caring for their children will be their first priority and should make them less committed and ambitious at work. However, when men become fathers, this does not impact negatively on their perceived suitability as workers. Men tend to be underrepresented in occupational and family roles that emphasize communality and care while women tend to be underrepresented in leading roles due to their high sensitivity. A short list of these expectations can be found in Table 1.

In another study [KRLZ+16], is proven that more men are studying in science fields than women due to the prevalence of gender stereotypes. This analysis shows also that there are more men than women depicted with science profession and that more women than men were depicted as teachers, which is a true reflection of the gender distribution in works.

Gender stereotypes	Male	Female
Stereotypical domain	Agency	Communality
Relevant behavior	Individual task performance	Care for others
Anticipated priorities	Work	Family
Perceived qualities	Competence	Warmth
Neglected needs	Interpersonal connection	Personal achievement

Table 1: Gender stereotypes and gendered expectations

Common gender stereotypes associated with women are submissive, emotional, quiet, neat, clean, artsy, housewife and childrearing. Hence, common gender stereotypes associated with men are aggressive, no emotions, loud, messy, athletic, math and science oriented and money maker. These examples are provided in an interesting study [Aks05] focused on recognizing gender stereotypes with children.

3 Materials

3.1 Data Sources

The system consists of a knowledge base built using WordNet to search for hyperonyms, hyponyms and synonyms of the words *female* and *male*. In addition to these information, are defined 10 female traits, 10 male traits, 10 female roles, and 10 male roles.

3.2 Tools and Technologies

A Python script has been used to speed up the knowledge base creation by using NLTK's WordNet corpus but Prolog is the main language used for the project, which is made with SWI-Prolog.

4 Methods

4.1 Methodology

The system is composed of several distinct modules operating on the same knowledge base:

1. **stereotypes.pl** which is the knowledge base,
2. **extract.pl** which extract subject-object and subject-adjective pairs to recognize stereotypes,
3. **grammar.pl** which generates three type of sentences with a defined structure,
4. **dfs.pl** a Depth-First Search algorithm for female and male words,
5. **bfs.pl** a Best-First Search algorithm for female and male words,
6. **menu.pl** a basic interface to test the system.

Each module will be discussed in more detail in the next sections of this document.

4.2 Implementation

4.2.1 Knowledge base

The knowledge base contains `role/2`, `trait/2`, `hypernym/2` and `synonym/2`.
The first two are dynamic and they can be updated through a CLI interface.

The rule `role/2` is defined as `role(Gender, Role)` where `Gender` can only be female or male and `Role` depends on `Gender`. All roles are nouns. A code snippet is shown below.

```prolog
    % Female roles
    % role(female, FemaleRole) means that FemaleRole is a stereotypical female role
    role(female, nurse).
    role(female, teacher).
    ...

    % Male roles
    % role(male, MaleRole) means that MaleRole is a stereotypical male role
    role(male, engineer).
    role(male, leader).
    ...
```

The rule `trait/2` works the same as `role/2` but instead of roles which are nouns, it uses adjectives. It is defined as `role(Gender, Trait)`. A code snippet is shown below.

```prolog
    % Female traits
    % trait(female, FemaleTrait) means that FemaleTrait is a stereotypical female
    trait
    trait(female, emotional).
    trait(female, nurturing).
    ...

    % Male traits
    % trait(male, MaleTrait) means that MaleTrait is a stereotypical male trait
    trait(male, strong).
    trait(male, assertive).
    ...
```

The rule `synonym/2` is defined as `synonym(Word1, Word2)` where `Word1` is different from `Word2` such that `Word1` is a synonym for `Word2` and vice versa. A general rule has been used for this rule: only `Word1` can be find in `hypernym/2` rules.

```prolog
    % Female synonyms of hypernyms and hyponyms from WordNet
    % synonym(A,B) means that A is a synonym for B and vice versa
    synonym(female, female_person).
    synonym(female, she).
    ...

    % Male synonyms of hypernyms and hyponyms from WordNet
    % synonym(A,B) means that A is a synonym for B and vice versa
    synonym(male, male_person).
    synonym(male, he).
    ...
```

The rule `hypernym/2` is defined as `hypernym(Hypernym, Hyponym)` where `Hypernym` is the hypernym for `Hyponym` and `Hyponym` is the hyponym of `Hypernym`.

```prolog
    % Female hypernyms and hyponyms from WordNet
    % hypernym(A,B) means that A is a hypernym of B and B is a hyponym of A
    hypernym(female, female_child).
    hypernym(female, female_offspring).
    ...

    % Male hypernyms and hyponyms from WordNet
    % hypernym(A,B) means that A is a hypernym of B and B is a hyponym of A
    hypernym(male, boy_wonder).
    hypernym(male, foster-brother).
    ...
```

4.2.2 Pairs extraction

This module is made of an actual pair extractor and a stereotyped pair recognition.

The aim of the pair extractor is to obtain `(Subject, Object)` and `(Subject, Adjective)` pairs where `Subject` is female or male. To make sure that a word is a subject, an object or an adjective, it checks their position in the sentence by using a rule of thumb: the subject precedes the verb, the object and the adjective succeeds the subject. The code snippet is shown below.

```prolog
% Extract subject-object pairs, checking if the object matches a stereotypical role or
     trait using matches_trait_or_role
extract_subject_object_pairs(Tokens, SubjectObjectPairs) :-
    findall((Subject, Object),
        ( member((Subject, n), Tokens),
          member((Object, n), Tokens),
          Subject \= Object,
          nth0(VerbPos, Tokens, (_, v)),
          nth0(SubjectPos, Tokens, (Subject, n)),
          nth0(ObjectPos, Tokens, (Object, n)),
          SubjectPos < VerbPos,
          VerbPos < ObjectPos,
          matches_trait_or_role(Subject, Object) % Verify if the object is a valid
    role or trait
        ),
        SubjectObjectPairs).

```

```
% Extract subject-adjective pairs, checking if the adjective matches a stereotypical
    trait using matches_trait_or_role
extract_subject_adjective_pairs(Tokens, SubjectAdjectivePairs) :-
    findall((Subject, Adjective),
        (
        ...
        ),
        SubjectAdjectivePairs).
```

The next step is to recognize a `role(Subject, Object)` and/or a `trait(Subject, Adjective)` from these pairs. Firstly, we need to check if we have a `role(male, Object)` or a `role(female, Object)` or a `trait(male, Adjective)` or a `trait(female, Adjective)`. To do so, we take advantage of being able to define if-then-else constructs in Prolog using the `->` operator.

```
% Check if the term is a stereotypical trait or role for the given subject
matches_trait_or_role(Subject, Object) :-
    (   is_hyponym_of_male(Subject)
    ->  ( trait(male, Object)
        -> format('Stereotypical trait found! Males are ~w.~n', [Object])
        ...
        )
    ;   is_hyponym_of_female(Subject)
    ->  ( trait(female, Object)
        -> format('Stereotypical trait found! Females are ~w.~n', [Object])
        ...
        )
    ;   format('No correspondence found for ~w.~n', [Subject])
    ).
```

To also cover cases where stereotypes are implicit, we use negative examples obtained by substituting `female` instead of `male` and vice versa to detect hidden traits or roles.

```
% Check if the term is a stereotypical trait or role for the given subject
matches_trait_or_role(Subject, Object) :-
    (   is_hyponym_of_male(Subject)
    ->  ( trait(male, Object)
        ...
        ;   trait(female, Object)
        -> format('Negative stereotypical trait found! Females are ~w, not males.~n',
    [Object])
        ...
        )
    ;   is_hyponym_of_female(Subject)
    ->  ( trait(female, Object)
        ...
        ;   trait(male, Object)
        -> format('Negative stereotypical trait found! Males are ~w, not females.~n',
    [Object])
        ...
    ;   format('No correspondence found for ~w.~n', [Subject])
    ).
```

The final result is carried out by `extract_and_print/1` which shows which stereotypes are present in the sentence and the extracted pairs.

4.2.3 Generating sentences

Prolog's DCG rules have been used to generate sentences based on three types of structure:

1. Somebody —s Adjective
2. Somebody —s too Adjective to INFINITIVE Noun
3. Somebody —s not suited for Noun

The main idea behind the creation of these structures comes from WordNet Verb Frames. Each verb synset contains a list of generic sentence frames illustrating the types of simple sentences in which the verbs in the synset can be used.

An example of DCG for `[Subject, is, Adjective]` sentences is the following:

```
% DCG for "subject be adjective"
sentence(subject_be_adjective(Sentence)) -->
    subject(Subject),
    [is],
    adjective_for_subject(Subject, Adjective),
    { Sentence = [Subject, is, Adjective] }.
```

More interesting DCGs are the following:

```
% DCG for subject (hyponyms of male or female)
subject(Subject) -->
    [Subject],
    { (is_hyponym_of_male(Subject); is_hyponym_of_female(Subject)) }.

% DCG for adjective (traits) - only traits valid for the subject's gender
adjective_for_subject(Subject, Adjective) -->
    [Adjective],
    { find_gender(Subject, Gender), trait(Gender, Adjective) }.

% DCG for role - only roles valid for the opposite gender
role_for_subject(Subject, Role) -->
    [Role],
    { (is_hyponym_of_female(Subject) -> role(male, Role) ; role(female, Role)) }.

% DCG for opposite role - only roles valid for the opposite gender
role_opposite_for_subject(Subject, Role) -->
    [Role],
    { (is_hyponym_of_female(Subject) -> role(male, Role) ; role(female, Role)) }.
```

The knowledge base is heavily used in sentence generation because stereotyped sentences have a peculiar structure. Sentences like `[Subject, be, too, Adjective, to be, Role]` must use an `Adjective` related to the `Subject` and a `Role` related to the opposite gender of the `Subject` in order to work. The same happens with `[Subject, is not, suited, for, Role]` sentences where the `Role` has to be related to the opposite gender of the `Subject`.

The resulting sentences will be used by the pair extraction module, so they are provided with WordNet POS tagging to find the position of every element needed for the stereotype recognition.

4.2.4 Depth-First Search for gender

The first search implemented for the system is a Depth-First Search which explores a tree with `hypernym(Hypernym, Word)` a root and tries to reach `hypernym(female, Hyponym)` or `hypernym(male, Hyponym)` where `Hyponym` is an hypernym of `Word` in an unspecified depth of this tree. It starts by trying to directly reach `female` or `male` and, if it fails, it search for synonyms and retries until it get to `female` or `male`. Otherwise, it fails if the input word gender cannot be found. A drawing of the knowledge base representation is reported in this image 1.

The code snippet of the Depth-First Search is the following:

```
% Find the gender of a word
find_word_gender_dfs(Word, Gender) :-
    ...

    (   is_hyponym_of_female(Word)
    ->  Gender = female
    ;   is_hyponym_of_male(Word)
    ->  Gender = male
    ;   synonym(Synonym, Word)
    ->  find_gender(Synonym, Gender)
    ;   Gender = unknown
    ).
```

This search uses `is_hyponym_of_female/1` and `is_hyponym_of_male/1` to explore the hypernym tree as a depth search. If these searches fail, the DFS algorithm will start to check the word synonyms and process them again in depth until it fails or it reaches `female` or `male`. The code snippet is the following:

```
% Check if a word is a hyponym of "female" (direct or indirect)
is_hyponym_of_female(Word) :-
    ...
```

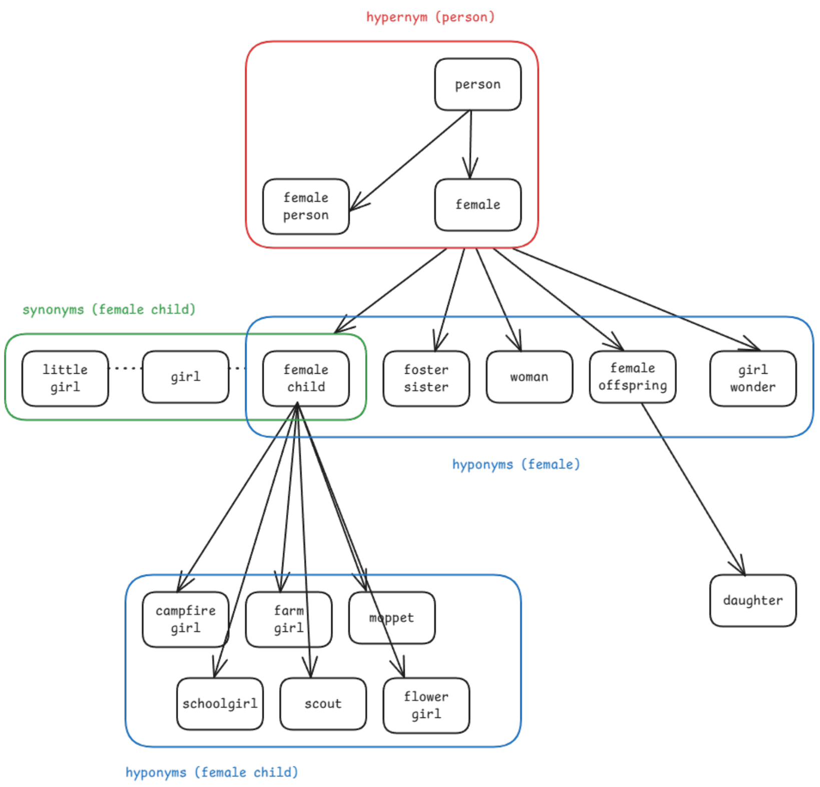

Figure 1: Hypernyms, Hyponyms and Synonyms example

```prolog
    hypernym(female, Word).
is_hyponym_of_female(Word) :-
    ...
    hypernym(Intermediate, Word),
    is_hyponym_of_female(Intermediate).
```

One consideration about this algorithm is that it always check first for `female` words and then for `male` words. One way to make this search more efficient could be to introduce a root for the `hypernym/2` rule such as `hypernym(person, female)` and `hypernym(person, male)` have the same hypernym and the whole tree would be explored un reaching one of them.

4.2.5 Best-First Search for gender

A second search algorithm has been implemented and checked if could lead to some improvements although the tree to explore is not so wide so it was not strictly needed.

```prolog
% Find the gender of the word
find_word_gender_bfs(Word, Gender) :-
    ...

    % Perform the search for "female"
    bfs_hyponym_of_gender_tracked(Word, female, FemalePath),
    (   FemalePath \= []
    -> Gender = female
    ;   % Perform the search for "male"
        bfs_hyponym_of_gender_tracked(Word, male, MalePath),
           MalePath \= []
        ->    Gender = male
        ;   % No gender found
            Gender = unknown
    ).
```

By using a Best-First Search, the gender of the input word will depend on the list that is used during the search. If the list during the gender search is empty, the gender will be to opposite of the one searched.

Also in this implementation we have again the lack of a common root for `female` and `male` words, so it could be optimized further.

For this search, `bfs_hyponym_of_gender_tracked/3` is used.

Firstly, the BFS starts from `Word` and tries to find a path that links it to `Gender` which could be `female` or `male`. The search starts with a list with only `Word` invoking `bfs_tracked/3`.

```
 % Use the tracked predicates in your BFS search
bfs_hyponym_of_gender_tracked(Word, Gender, Path) :-
    bfs_tracked([[Word]], Gender, Path).
```

If the head of the path list contains an actual path, it will return the complete path. The cut `!` operator helps in stopping Prolog to explore other paths once it has been found.

```
% Tracked BFS search
bfs_tracked([[Gender | Path] | _], Gender, [Gender | Path]) :-
    !.
```

The recursive step explodes next nodes using the `findall/3` operator with `tracked_hypernym/2` and `tracked_synonym/2` adding every new node to the current path in `Neighbors`. Using these nodes, new paths can be made starting from them and can be explored.

```
bfs_tracked([[Current | Path] | Rest], Gender, Result) :-
    findall([Next, Current | Path],
            (tracked_hypernym(Next, Current); tracked_synonym(Next, Current)),
            Neighbors),
    append(Rest, Neighbors, NewPaths),
    bfs_tracked(NewPaths, Gender, Result).
```

If no path is found, the cut `!` operator will stop the search returning an empty list.

```
bfs_tracked([], _, []) :-
    !.
```

With respect to a classic graphsearch which uses two lists, `OPEN` and `CLOSED`, in this case we are only using the `OPEN` one because this specific graph, by construction, is acyclic.

4.2.6 Basic interface

All these operations are easily tested through a basic CLI menu made with Prolog.

```
Menu
1. Find word gender with DFS
2. Find word gender with BFS
3. Find traits and roles for female and male
4. Generate a sentence and show POS tags
100. Add a role (dev)
200. Add a trait (dev)
0. Exit

Enter your choice: |:
```

The word gender search can be done with both algorithm and it shows also some statistics about the number of operation needed for the search.

```
Enter the word to find its gender: |: boy.
The gender of boy is male using DFS.
Number of operations: 10

...

Enter the word to find its gender: |: boy.
The gender of boy is male using BFS.
Number of operations: 24
```

The number of operations shown by the DFS search are actually inaccurate as it only shows the number of operation carried out by one of the gender searches, while the BFS statistics includes both gender searches.

Find traits and roles for female and male options queries the knowledge base in order to show every trait and role related to females and males.

```
Finding traits and roles for female and male...

Traits and roles for female:

Trait for female: [emotional,nurturing,patient,empathetic,delicate,gentle,supportive,
    caring,kind,understanding]
Role for female: [nurse,teacher,caregiver,homemaker,secretary,psychologist,journalist,
    writer,therapist,babysitter]

Traits and roles for male:

...
```

The last option prompts a submenu which allows to choose what kind of sentences to generate and explain.

```
Generate a sentence based on the following frames:
1. Subject be adjective
2. Subject be too adjective be role
3. Subject is not suited for role
Select a frame (1-3): |:
```

The following is an example of output from the whole process of this option:

```
Generated Sentence: [old_boy,is,not,suited,for,homemaker]

POS Tags: [(old_boy,n),(be,v),(not,r),(suited,v),(for,r),(homemaker,n)]
Negative stereotypical role found! Females are homemaker, not males.
Negative stereotypical role found! Females are homemaker, not males.

Subject-Object Pairs: [(old_boy,homemaker)]
Subject-Adjective Pairs: []
```

Sometimes, The sentence generator will not generate any sentence returning the message *Invalid choice. Please try again.* or *Please retry, something went wrong* due to a yet unknown issue in the module.

There are also two more options with a `(dev)` tag to update traits and roles so they can be used for generating sentences. These options uses `asserz/1` and the new rules are not permanently added to the knowledge base.

A simple example of use is the following:

```
Enter the gender (female or male): |: female.
Enter the trait: |: smart.
Added trait: (female, smart)

...

Generated Sentence: [dame,is,smart]
POS Tags: [(dame,n),(be,v),(smart,a)]
Stereotypical trait found! Females are smart.
```

5 Results

The system is able to generate stereotyped sentences and explain the common stereotype used for the generation, as it has been shown in the previous sections. It also provides different tools for gender analysis and extraction by using WordNet, DCGs and POS tagging, showing a good standalone capability of handling these tasks.

6 Conclusions

In its simplicity, the system shows how the representation of stereotypes is easily traced with facts in Prolog. Building ad hoc knowledge bases for this type of problem can be a great support for stereotype detection in sentences using modern NLP techniques available today.

As is reported in other sections, an improvement on the searches can be made by changing the representation of hypernyms with the single root `person` for both `female` and `male` hypernyms.

A full WordNet implementation could really make this a standalone system but it requires also some lemmatization procedures in order to get WordNet informations. The WordNet for Prolog package was initially used for this system but has been removed due to an high complexity in use.

References

[Aks05] Bengü Aksu. Barbie against superman: Gender stereotypes and gender equity in the classroom. *Journal of Language and Linguistic studies*, 1(1):12–21, 2005.

[Ell18] Naomi Ellemers. Gender stereotypes. *Annual review of psychology*, 69(1):275–298, 2018.

[KRLZ$^+$16] Anne H Kerkhoven, Pedro Russo, Anne M Land-Zandstra, Aayush Saxena, and Frans J Rodenburg. Gender stereotypes in science education resources: A visual content analysis. *PloS one*, 11(11):e0165037, 2016.