Birthday Gift Advisor

Nirmallya Kundu
nxk161830

Objective:

Developing a common-sense reasoning application using the s(ASP) system that will recommend a "Birthday Gift" for friends and relatives, based on knowledge about the person, knowledge about the age group of the person, knowledge about the gender of the person, knowledge about the hobbies of the person, knowledge about the relationship with the person, knowledge about liking of the person in general, knowledge about the disliking of the person in general, etc.

Solution:

Based on the information obtained about the user, who is planning for the gift and the person for whom the gift is being planned, there are a number of pre-defined rules in the application which we evaluate to recommend the best possible gift, taking into consideration all the possible user constraints.

Features and Rules:

The application is designed in a way to recommend the best the best gift to the based on the information received from the user about the user as well the person to be gifted.
There a number of pre-defined rules which are discussed below:

1. **Users**

The information about the user is in the file "users.lp".
The structure for user is defined below:

```
user (from_name('Nirmallya','Kundu'),
        from_gender(male),
        form_age(26),
        from_finance(mid),
        to_name('John', 'Naas'),
        to_gender(male),
        to_age(30),
        to_relation(friend),
        to_hobby(reading),
        to_likes([notebook, pen]),
        to_dislikes([pencil, eraser])
        )
```

1.1. from_name('FirstName', 'LastName')
This represents the first name and last name of the person or the user who is planning for the gift.

1.2. from_gender(male)
This represents the gender of the person or the user who is planning for the gift.

1.3. form_age(26)
This represents the age of the person or the user who is planning for the gift, so that we can calculate the age-group of the person who is planning for the gift.
Age-group can be of four types:

- 1.3.1. Child
- 1.3.2. Teenager
- 1.3.3. Adult
- 1.3.4. Senior

1.4. from_finance(mid)
This represents the financial-budget of the person or the user who is planning for the gift.
Financial-budget can be of three types:

- 1.4.1. Low
- 1.4.2. Mid
- 1.4.3. High

1.5. to_name('John', 'Naas')
This represents the first name and last name of the person for whom the gift is being planned/ recommended.

1.6. to_gender(male)
This represents the gender of the person for whom the gift is being planned/ recommended.

1.7. to_age(30)
This represents the age of the person for whom the gift is being planned/ recommended. So that we can calculate the age-group (child/ teenager/ adult/ senior) of the person for whom the gift is being planned/ recommended.

1.8. to_relation(friend)
This represents the relationship of the user or the person who is planning for the gift with the person for whom the gift is being planned/ recommended.
Relation can be of five types:

- 1.8.1. Sibling
- 1.8.2. Parent
- 1.8.3. Grand_parent
- 1.8.4. Friend
- 1.8.5. Couple

1.9. to_hobby(reading)
   This represents the hobby of the person for whom the gift is being planned/ recommended.
   Hobby can be of five types:
   1.9.1. Reading
   1.9.2. Sports
   1.9.3. Travel
   1.9.4. Music

1.10. to_likes([notebook, pen])
   This represents the likes of the person for whom the gift is being planned/ recommended.

1.11. to_dislikes([pencil, eraser])
   This represents the dislikes of the person for whom the gift is being planned/ recommended

2. Gifts

The information about the gifts are in the file “gifts.lp”.
The structure for gifts are defined below:

2.1. _gift(category_id, gift_id, gift_name)
   This represents a single gift with gift_id and gift_name, listed under the category with category_id.
   Example: _gift(1, 2, pen). This gift structure represents a gift “pen” with gift_id 2, listed under the category_id 1.

2.2. _category(categoryNumberList)
   This represents the category structure with a list of category ids.
   Example: _category([1,2,3,4,5,6,7,8]). This category structure represents the list of categories 1,2,3,4,5,6,7,8 which are possible or which has gifts listed under it.

2.3. _gift_id(Category_id, Gift_id_NumberList)
   This represents the gift_ids under a category.
   Example: _gift_id(1, [1,2,3,4]). Here gifts with gift_id 1,2,3,4 are listed under category_id 1.

2.4. category_based_on_age_group(age_group, categoryList)

This represents the list of category_ids based on the age-group(child/ teenager/ adult/ senior).
The rules are defined below:

2.4.1. If the age-group is “child” then we can only recommend gifts in the category that are most suitable for the children.

2.4.2. If the age-group is “teenager” then we can only recommend gifts in the category that are most suitable for the teenager.

2.4.3. If the age-group is “adult” then we can only recommend gifts in the category that are most suitable for the adults.

2.4.4. If the age-group is “senior” then we can only recommend gifts in the category that are most suitable for the seniors.

Example: category_based_on_age_group(child, [1, 2, 3]). This represents that the gifts under the category_id 1,2,3 can be presented to a person within the child age-group.

2.5. category_based_on_finance(finance_level, categoryList)

This represents the list of category_ids based on the financial-budget (low/ mid/ high) of the person who is planning for the gift.
The rules are defined below:

2.5.1. If the financial-budget is “low” then we can only recommend gifts in the category that are most feasible for the low budget cost.

2.5.2. If the financial-budget is “mid” then we can only recommend gifts in the category that are most feasible for the medium budget cost.

2.5.3. If the financial-budget is “high” then we can only recommend gifts in the category that are most feasible for the high budget cost.

Example: category_based_on_finance(mid, [2, 3, 4]). This represents that the gifts under the category_id 2,3,4 can be presented if the financial budget is mid (medium).

2.6. category_based_on_gender(gender, categoryList)

This represents the list of category_ids based on the gender (male/ female) of the person who is planning for the gift as well as the person for whom the gift is being planned.
The rules are defined below:

2.6.1. If the gender is "male" then we can only recommend gifts in the category that are most feasible for the male person.

2.6.2. If the gender is "female" then we can only recommend gifts in the category that are most feasible for the female person.

Example: category_based_on_gender(male, [1,2, 3, 6,7]). This represents that the gifts under the category_id 1,2,3,6,7 can be presented if the gender of the person is male for whom the gift is being planned.

2.7. category_based_on_relation(relation, categoryList)

This represents the list of category_ids based on the relation (sibling/ parent/ grand_parent/ friend/ couple) of the person who is planning for the gift with the person for whom the gift is being planned.
The rules are defined below:

2.7.1. If the relation is "sibling" then we can only recommend gifts in the category that are most suitable for the siblings.

2.7.2. If the relation is "parent" then we can only recommend gifts in the category that are most suitable for the father and mother gifting to their children or children gifting to their parents.

2.7.3. If the relation is "grand_parent" then we can only recommend gifts in the category that are most suitable for the grand-children gifting to their grand-parents or grand-parents gifting to their grand-children.

2.7.4. If the relation is "friend" then we can only recommend gifts in the category that are most suitable for friends.

2.7.5. If the relation is "couple" then we can only recommend gifts in the category that are most suitable for husband gifting to wife or wife gifting to husband.

Example: category_based_on_relation(parent, [1,3,5,7]). This represents that the gifts under the category_id 1,3,5,7 can be presented if the relation of the person who is planning for the gift with the person for whom the gift is being planned is parent.

2.8. category_based_on_hobby(hobby, categoryList)
This represents the list of category_ids based on the hobby (reading/ sports/ travel/ music) of the person for whom the gift is being planned.
The rules are defined below:

2.8.1. If the hobby is "reading" then we can only recommend gifts in the category that are related to reading, like books, magazine, etc.

2.8.2. If the hobby is "sports" then we can only recommend gifts in the category that are related to sports, like bat, ball, etc.

2.8.3. If the hobby is "travel" then we can only recommend gifts in the category that are related to travelling, like tickets to some places, hotel bookings, etc.

2.8.4. If the hobby is "music" then we can only recommend gifts in the category that are related to music, like music albums, tickets to music concert, etc.

Example: category_based_on_hobby(reading, [1,2,5]). This represents that the gifts under the category_id 1,2,5 can be presented if the hobby of the person for whom the gift is being planned is reading.

2.9. category_default(DefaultCategoryIdList)
This represents the list of default category_ids that are by default age-group, financial budget, gender, relation, hobby neutral. If some information about the user or the person to whom the gift is being gifted is missing or we do not have some concrete information, we can suggest a gift listed under the default category_ids.
Example: category_default([0]).

3. The rules for the "gift advisor" is in the file "advisor_rules.lp"
The rules are defined below:

3.1. update_category_based_on_hobby(hobby, oldCategoryList, newCategoryList)
This predicate update the category list based on the following rules:

3.1.1. If the person hobby is "reading", he might not like "sports", thereby updating the new list by removing any category that has sports related gifts in it.

3.1.2. If the person hobby is "sports", he might not like "reading", thereby updating the new list by removing any category that has reading related gifts in it.

3.1.3. If the person hobby is "music", he might not like "reading", thereby updating the new list by removing any category that has reading related gifts in it.

3.1.4. If the person hobby is "reading", he might not like "music", thereby updating the new list by removing any category that has music related gifts in it.

3.1.5. If the person hobby is “sports”, he might like “music” too, thereby updating the new list by adding any category that has music related gifts in it.

3.1.6. If the person hobby is “travel”, he might like “music” too, thereby updating the new list by adding any category that has music related gifts in it.

3.1.7. If the person hobby is “travel”, he might like “reading” too, thereby updating the new list by adding any category that has reading related gifts in it.

3.2. age_group(age, age-group). –
Based on the age of the user the age-group is returned.
3.2.1. 00 < age < 12 – child.
3.2.2. 13 < age < 20 – teenager.
3.2.3. 20 < age < 50 – adult.
3.2.4. 50 < age < 99 – senior.

3.3. gifts(CategoryList, Gifts)
This predicate returns the list of the all the gifts (gift structure) under each category_id present in the category list.
Example: gifts([3,6], Gifts).
This will return all the gifts (gift structure) in each category_id 3 and 6.
Gifts =
[gift(3,4,pen),gift(3,3,pencil),gift(3,2,eraser),gift(3,1,sharpener),gift(6,4,ball),gift(6,3,bat),gift(6,2,wicket),gift(6,1,gloves)].

3.4.get_like_gifts(Likes_list, Gifts)
This predicate returns the list of the all the gifts (gift structure) that the person may like (based on the Likes_List) and if the gift is present in the gift dataset (facts).
Example: get_like_gifts([pen, bat, perfume], Gifts).
Gifts = [gift(2,1,pen),gift(4,2,bat)]
This will return all the gifts (gift structure) in the Likes_list based on if that gift is present in the dataset (facts). Since perfume is not present in the dataset (facts), Gifts only returned the two gifts (gift structure).

3.5.filter_dislike(Gifts_all, DislikeGiftsList, Gifts)
This predicate filters out the Gifts present in the Gifts_all list based on the DislikeGiftList and return the filtered list of gifts in Gifts.
Example: filter_dislike([gift(1,2,pen), gift(3, 1, eraser), gift(4, 2, pencil)], [pen, pencil], Gifts).
This will return: Gifts= [gift(3,1, eraser)].

4. The main file “advisor.lp” that includes the main predicate and all other files.

   This file also includes the step by step procedures/rules to find the best recommendation of gifts.

   4.1. Included files are mentioned below:
   4.1.1. #include ‘gifts.lp’
   4.1.2. #include ‘users.lp’
   4.1.3. #include ‘helper.lp’
   4.1.4. #include ‘advisor_rules.lp’

   4.2. There are two main predicates in the file for recommending the best possible gift.

   4.2.1. advise_gift_random(GiftName)
   This predicate will recommend a random gift from the gift dataset (facts). When we do not have any information about the user of the person to be gifted, we can use this predicate to recommend a random gift.

   4.2.2. advise_gift(UserId, GiftName)
   This predicate will recommend a gift from the gift dataset (facts), based on the information received with the UserId from the user dataset (facts).

   The rules are defined below:

   4.2.2.1. Firstly, the “user” information is fetched with the user_id.

   4.2.2.2. Secondly, the list of default category_ids are fetched and stored in a list, C0.

   4.2.2.3. Then, the age-group of the user as well as person to be gifted is calculated, to fetch the correct set of category_ids, and stored in a list C1.

   4.2.2.4. Then, the gender of the user as well as person to be gifted is fetched, to fetch the correct set of category_ids, and stored in a list C2.

   4.2.2.5. Then, the relation of the user with the person to be gifted is fetched, to fetch the correct set of category_ids, and stored in a list C3.

   4.2.2.6. Then, the hobby of the person to be gifted is fetched, to fetch the correct set of category_ids, and stored in a list C4.

   4.2.2.7. Then, the union of the category_ids C0, C1, C2, C3, C4 is calculated and stored in a list C04.

4.2.2.8. Then, the category list C04 is updated based on the rules mentioned in 3.1. (hobby) and stored in a new list C05.

4.2.2.9. Then, based on the Likes of the user a new Category list is calculated and stored in a list C6.

4.2.2.10. Then, the union of the category_ids C05, C6 is calculated and stored in a list C06.

4.2.2.11. Then, based on the Dislikes of the user a new Category list is calculated and stored in a list C7.

4.2.2.12. Then, the list minus operation is performed on C06 and C7 and stored in C07.

4.2.2.13. Then, the financial-budget of the user is fetched, to fetch the correct set of category_ids, and stored in a list C8.

4.2.2.14. Then, the intersection of the C07 and C8 is calculated to fetch the category of gifts that are within the financial-budget of the user and stored in C09.

4.2.2.15. Now, a gift listed under the category C09 is recommended.

Lessons Learned:

1. Developing a fully functional application in s(ASP).
2. Creative thinking of rules which constitutes the major part of the project.

References:

1. Dr. Gopal Gupta, for helping with each and every part of the project. Also, for most of the creative rules in the project.
2. Sample s(ASP) project on https://gitlab.com/saikiran1096/gradaudit/