HumHub custom specifications

Francesco Bailo francesco.bailo@sydney.edu.au

December 12, 2018
v2

1 Changes

Added Pseudocode for Algorithm 1 and Algorithm 2

Added more details in Section 4.3

2 Introduction

I will use the "Social Network Kit" HumHub to run an online experiment. The experiment will provide behavioural data to measure the effect on exposure to political content representing specific political views.

3 Definitions and abbreviations

These definitions are based on (my understanding of) the documentation of HumHub and on the requirements for the project.

Social networking site (SNS) is the online platform based on the HubHub framework where the experiment will take place. It will be hosted on servers administered by the Nectar research cloud (NRC) and on servers administered by the University of Sydney (USYD).

Preliminary survey Each user will complete a survey before signing up with the SNS. The survey will be used to quantify users' political views. The online survey will be administered externally with RedCap.

User Each user will need to be characterised according to his/her *political orientation* and *topic interest*. *Political orientation* will be captured by variables coded after the completion of a preliminary surveys. There will be a general political orientation variable and variables quantifying political opinion for each topic (see Table 1 for a list of topics). For each defined topic, a user will also have a variable defining his/her interest. These variables will be coded based on the observed behaviour of the user (see Table 2 for a list of user variable types).

Variable A variable is stored in a database field: $variable == field$.

Topic A topic is a tag associated to a content entry. It is defined by the author of the content entry (either a user or an administrator). The complete list of topics is presented in Table 1.

Content entry A content entry contains text, images or links to some resource (internal or external). It is published by a user or by an administrator. Each content entry will be quantified by at least one topic. Each content entry will be qualified

Stream A stream is an ordered series of content entries.

Dashboard The dashboard is the timeline of a user. It must be always the first visible tab when a user log in.

Abbreviation	Description
`abo`	Abortion
`imm`	Immigration
`gay`	LGBT rights
`eco`	Economic governance
`cli`	Climate change

Table 1: List of topics: abbreviations and description

Abbreviation	Description	Data type	Scope
`pol_op`	Political opinion	double	It can be followed by a topic label. Alone, it refers to *general* political opinion.
`int_sur`	Political interest as captured by survey responses	double	Always followed by a topic label
`int_obs`	Political interest as captured by SNS observation	double	Always followed by a topic label

Table 2: Types of variables: abbreviations and description

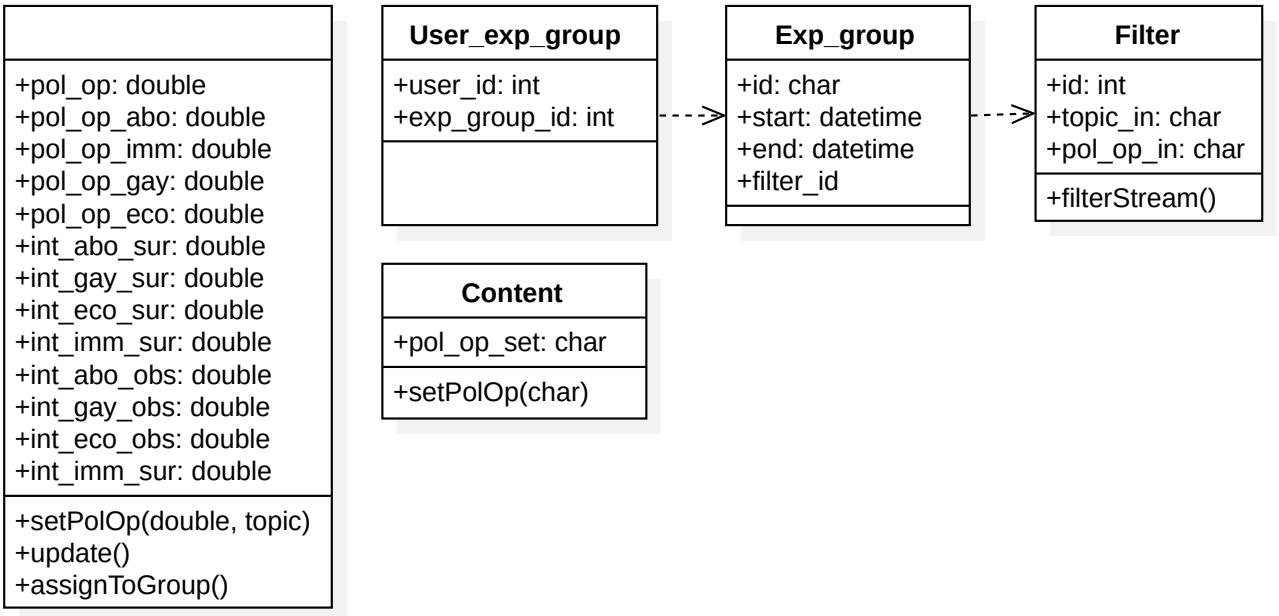

Figure 1: New or modified classes

4 Processes

The class `user` will be modified based on what described in Figure 1. All the new variables with prefix `pol_op` and suffix `sur` are `set()` after the completion of the survey. All the new variables with suffix `obs` are updated following the activity of the user.

Note: `set()` and `assignToGroup()` can be performed externally. That is, the developer doesn't need to write functions or create a web interface. I just need to know which SQL commands I can *safely* execute on the SNS database.

To do: `update()` will need to be implemented by the developer.

To do: During an experiment `obs` variables are not updated.

4.1 Content

Each content is described by an object of class `topic` and by one new variable capturing the political opinion as defined *directly* by the administrator when posting the content. `pol_op_set` that takes two values (`left` and `right`).

Note: The manipulation of the `content` objects can be performed externally. Although it would be useful if the administrator could select `left` or `right` when creating a new content.

4.2 Experiment set up

When an experiment is set up, two objects of class `exp_group` are (manually) created and stored in the corresponding table under the same `id`, one for the *treatment group* and one for the *control group*. The table `exp_group` primary key consists of two fields: `id` and `group_type`. The field `group_type` can only take two values: `treatment` and `control`.

When a user is (manually) assigned to an experiment, a corresponding object of class `user_id` is stored into the table `user_exp_group`.

Note: The manipulation of objects of class `user_exp_group`, `exp_group` and `filter` can be performed externally.

4.3 Filtering

Each user assigned to a group (`exp_group`) of an ongoing experiment, will have his/her dashboard filtered by the function `filterStream()` according to what set in the corresponding object of class `filter`. An object of class `filter` is characterised by a variable `topic_in`, defining the single topic that must be given preference. `topic_in` can assume as value any of the topic labels defined in Table 1 and the is nullable if no preference is to be given to a specific topic. `pol_op_in` can assume two values: `left` and `right` and it is nullable if no preference is to be given on the political opinion of the content.

4.3.1 filterStream()

The function `filterStream`is the core function for the administration of the experiment. It takes two objects: an object `this_filter` of the class `filter` and an array `stream` of objects of class `content`. The function determines what a user participating to an experiment see on his/her Dashboard.

The function `filterStream` calls these other functions that, if not otherwise specified, consist of a simple SQL query:

- `getTopicIn()` accesses the `Filter` table.
- `getPolOpIn()` accesses the `Filter` table.
- `getContentTopic()` accesses the `Topic` table.
- `getContentPolOpSet()` accesses the `Topic` table.
- `getAuthor()` accesses the `Topic` table and return the author id.
- `getAuthorPolOp()` accesses the `User` table.
- `getAuthorPolOpTopic()` accesses the `User` table.
- `getContentLikers()` access the `Like` table.

Algorithm 1 Stream filtering

```
1: procedure filterStream(stream, this_filter)
2:     filter_topic_in ← getTopicIn(this_filter)
3:     filter_pol_op_in ← getPolOpIn(this_filter)
4:     filtered_stream ← INITIATED AS ORDERED EMPTY ARRAY
5:     for each content in stream do
6:         content_topic ← getContentTopic(content)
7:         if filter_topic_in IS NOT NULL then
8:             if filter_topic_in IS NOT content_topic then
9:                 NEXT
10:        if filter_pol_op_in IS NOT NULL then
11:            content_pol_op_set ← getContentPolOpSet(content)
12:            content_author ← getAuthor(content)
13:            author_pol_op ← getAuthorPolOp(content_author)
14:            author_pol_op_topic ← getAuthorPolOpTopic(content_author, content_topic)
15:            content_likers ← getContentLikers(content)
16:            likers_pol_op ← INITIATED AS EMPTY ARRAY
17:            likers_pol_op_topic ← INITIATED AS EMPTY ARRAY
18:            if length of content_likers > 0 then
19:                for each user in content_likers do
20:                    likers_pol_op.append(getAuthorPolOp(user))
21:                    likers_pol_op_topic.append(getAuthorPolOpTopic(user, content_topic))
22:            content_pol_op_inferred ← inferContentPolOp(
           content_pol_op_set, author_pol_op, author_pol_op_topic, likers_pol_op, likers_pol_op_topic)
23:            if filter_pol_op_in == "left" AND content_pol_op_inferred > 0 then
24:                NEXT
25:        filtered_stream.append(content)
26:    filtered_stream.order(decreasing=TRUE)
27:    RETURN filtered_stream
```

4.3.2 inferContentPolOp()

This function allows to infer the political opinion of some content based on three sources of information: the administrator creating the content (`content_pol_op_set`), the user creating the content (`author_pol_op` and `author_pol_op_topic`) and the users liking the content (`likers_pol_op` and `likers_pol_op_topic`). Importantly, `content_pol_op_set` can be NULL and `likers_pol_op` (and consequentially also `likers_pol_op_topic`) can be of length 0 if no user liked that specific content. But `content_pol_op_set` and `author_pol_op` cannot be both NULL, since the content was either created by the administrator of by a user.

The function `inferContentPolOp`calls two functions: `length` and `mean`, which respectively return the number of items in an array and the simple arithmetic mean of the values contained in an array.

Algorithm 2 Infer political opinion of content from user behaviour

```
1: procedure inferContentPolOp(
       content_pol_op_set, author_pol_op, author_pol_op_topic, likers_pol_op, likers_pol_op_topic)
2:     if content_pol_op_set IS NOT NULL then
3:         if content_pol_op_set == "left" then
4:             RETURN -10
5:         if content_pol_op_set == "right" then
6:             RETURN 10
7:         else
8:             CONTINUE
9:     bundled_author_pol_op ← (author_pol_op × 1 + author_pol_op_topic × 3) × 1/4
10:    if length(likers_pol_op) IS NOT NULL then
11:        bundled_likers_pol_op ← (mean(likers_pol_op) × 1 + mean(likers_pol_op_topic) × 3) × 1/4
12:        RETURN (bundled_author_pol_op + bundled_likers_pol_op) × 1/2
13:    else
14:        RETURN bundled_author_pol_op
```