University of Windsor

School of Computer Science
Faculty of Science

COMP-8730: Natural Language Processing & Understanding
Winter 2021

#	Title	Due Date	Grade Release Date
4	**Proposed Solution (Method)**	March 15, 2021, AoE	March 22, 2021, AoE

This course is research-oriented and project-driven in which a research project should be defined and completed in the field of NLP within one semester. The objectives of the research project are to provide graduate students with:

- An experience with research procedure, in general, and research in NLP, in particular.
- Hands-on experience with NLP.
- Advancing state of the art in NLP while passing a grad course.
- An opportunity to present a research outcome at an international computer science conference
- An opportunity to meet with scholars in the NLP community

In the research project, we propose a solution(s) to a problem by implementing an algorithm like a software project. However, there are differences in some respects. For instance, while a software project may implement an existing algorithm, a research project should propose and implement a *new* algorithm that improves or addresses a particular aspect of a problem that the current algorithms overlook. Roughly, a research project has the following milestones (phases):

1) Proposal
2) Literature Review
3) Proposed Method (Formal + Code)
4) Experiment (Evaluation)
5) Presentation (Paper + Talk)

In this course, a manual is prepared to guide the students through each milestone. The current manual is for the third milestone: Proposed Solution (aka Proposed Method), which further has the following steps:

Formal Definition: Input & Output (Solution)

At a very high abstract level, any proposed method to solve a problem can be defined as a function of inputs that does finite steps of processing and/or calculation on the inputs and results in a solution. In the following, I explain it by example.

For instance, if your proposed method is to cluster people into groups of like-minded people, the input is a bunch of human individuals. The output is groups of people where each individual is a member of one group. This is the *informal* definition which is not exact, and a reader may have different interpretations of what you explained, such as:

1- Is it possible for a person to be a member of *multiple* groups?
2- What do you mean by *like-minded*?
3- What do you mean by a *bunch* of human individuals? Do you probably mean their information?
4- …

To be exact and come up with the same interpretation for all readers, we use the language of logic and mathematics. So, we *formally* define the solution. We already did a similar step in the proposal milestone for problem definition.

Here is an example formal definition for the above solution:

> *Given a set of people* $A = \{a_i\}_{i=1}^{N}$, *our proposed solution is a function f such that f (A) outputs a partition on A that includes subsets of A such that each person* $a_i \in A$ *is a member of one and only one subset (group), and the union of all subsets (groups) is equal to A. Formally,*
>
> $$f: A \rightarrow P(A)$$
> $$f(A) = \{C_j \mid C_j \subseteq A, \ |C_j| > 0, \forall C_j, C_{k \neq j}\ C_i \cap C_j = \emptyset\}$$
>
> *where P is the powerset of A. As an example, if* $A=\{a_1, a_2, a_3\}$, *then* $f(A)=\{\{a_1, a_3\},\{a_2\}\}$.

Well, you may wonder it is difficult sometimes to formalize a method!

Formal Definition: Algorithm

So far, we only formally showed the input and outputs (solution). We have to proceed with more details about our method's processing steps on the input and how it outputs the final solution. Continuing on our example:

> *In our work, we represent each person* $a_i \in A$ *by the documents that she has published within a time period T. So, Given the set of all documents* $D = \{d_{i,t}\}$, *published by all persons in A, where* $d_{i,t}$ *is the document published by the person* a_i *at time t;* $1 \leq t \leq T$, *we calculate the similarity of a pair of person* (a_i, a_j) *by calculating the similarity of their documents as follows:*
>
> $$g(a_i, a_j) = \sum_{t=1}^{T} h(d_{i,t}, d_{j,t})$$
>
> *where h is a function that calculates the similarity of a pair of document based on the number of words that are shared, i.e.,*
>
> $$h(d_1, d_2) = \frac{d_1 \cap d_2}{d_1 \cap d_2}$$
>
> *where* d_i *is a set of words in the document.*

Wait, we are not done yet. We have not formalized how we group similar people. I leave it as an exercise for you!

Formal Definition: Pseudo-code

You can support your algorithm by pseudo-code that performs all the processing steps that you formalized in the previous section like:

Algorithm 2 Finding regions of like-mindedness for time interval t ($\mathcal{R}_t$)

Inputs:
- c, homogeneity condition;
- G_t, multigraph at time interval t;
- U, set of users;
- $\mathcal{Z}$, set of topics of interest;

Output:
- $\mathcal{R}_t$, set of regions of like-mindedness for time interval t

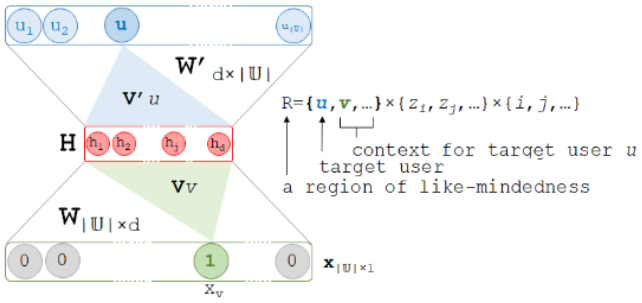
Initialization:
- $\mathcal{R}_t = \emptyset$;
- find_r_t($r = \mathrm{U} \times \emptyset, \mathrm{C} = [z_1, z_1, z_2, z_2, ..., z_{|\mathcal{Z}|}, z_{|\mathcal{Z}|}]$);

```
1:  procedure find_r_t(r = A × B, C)
2:      if (r ⊨ c) ∧ (∄r' ∈ R_t : r ⊂ r') then
3:          ∀r'' ∈ R_t if r'' ⊂ r then R_t ← R_t \ r''
4:          R_t ← R_t ∪ r
5:      for all z_j ∈ Z do
6:          A ← r.A; B ← r.B ∪ z_j; C ← C \ z_j
7:          if r.B = ∅ then find_r_t(A × B, C)
8:          else
9:              for all z_i ∈ r.B do
10:                 for all (z_i → z_j) ∈ U_t do
11:                     A ← r.A ∩ U_{z_i z_j, t}
12:                     find_r_t(A × B, C)
```

A pseudo-code is not precisely your program line by line in a specific programming language! It is, however, the overall flow of your program in a general abstract programming language. You should include the important parts of your code that are essential to provide the final solution.

Illustrative Figure

You can also support your algorithm by flow diagram, data model diagram, sequence diagram, etc. Recall, in the undergrad course Software Engineering, you were taught how to document software. Here, we want to showcase our algorithm via a visual diagram. For instance,



Implementation (Code)

By implementation, aka code, we mean the program you wrote in a programming language and does the processing steps on the inputs and produce the final solution. It is imperative that your code follows the FAIR Guiding Principles[1], published in 2016, to improve the *findability*, *accessibility*, *interoperability* and *reusability* of digital research objects for both humans and machines! Simply, it says that anyone (a human or even a machine) can find your code and run your code without error. So, your code should be:

1- Publically available online on repos like Github,
2- Must be runnable without error
3- Aligned with what you formalize in your research paper, that is, your code does exactly the steps that you explained in your research paper.
4- Accompanied with installation and or running guides or *ReadMe* documents so that anybody can easily run your code by following your guidelines.

Also, your code should be *reproducible* and *replicable*. There is a debate on the exact definition and their differences in the research community. Simply, your code should output the same (or similar) solutions when given the same inputs and using the same settings as you explained in your research paper. For instance, if in your paper you said that *if* $A=\{a_1, a_2, a_3\}$, *then* $f(A)=\{\{a_1, a_3\},\{a_2\}\}$, when somebody downloads your code and run your code on $A=\{a_1, a_2, a_3\}$, she sees the same (or similar) result. Your algorithm may not be *deterministic* and involve some random procedures. For instance, most of the data mining and machine learning methods are iterative and involve random steps. That is why some languages like python provide you with the ability to seed the random process such that every run of the code using the same seed ends up with the same result!

```
import random
random.seed(0)
import torch
torch.manual_seed(0)
import numpy as np
np.random.seed(0)
```

[1] The FAIR Guiding Principles for scientific data management and stewardship: https://www.nature.com/articles/sdata201618

Submission Guidelines

- Submission must be written in English, in the current ACM two-column conference format in LaTeX. Overleaf templates are available from the ACM Website (use the "sigconf" proceedings template).

- Submission must be 2 pages (4 columns) in length, no more not less, including figures, tables, authored by the team members, plus 1 column for references.

- The implementations (code) should be available in an online repo (preferably Github) and the link should be mentioned as a footnote to the report's title. See the example below.

- Submission must be in one single zip file with COMP8730_Proposed_Solution_UWinId1_UWindId2.zip, including:
  1. the LaTeX files
  2. the pdf file

A sample submission has been attached to this manual in Blackboard.

In summary, your submission has (%marking schema for this milestone):

1.1) (20%) Formal Definition
1.2) (10%) Pseudo-code
1.3) (10%) Illustrative Figure
1.4) (60%) Code (FAIR-compliant)