Multi-Factor Duplicate Question Detection in Stack Overflow[1]

Team 8 – Team iSmart

Bharadwaj S (2020122003, ECD)

Sasanka Uppuluri (2019102036, ECE)

Siva Durga (2019102038, ECE)

Viswanadh (2019112011, ECD)

SMAI Course Project
Course Instructor: Anoop N
Mentor TA: Sireesha V

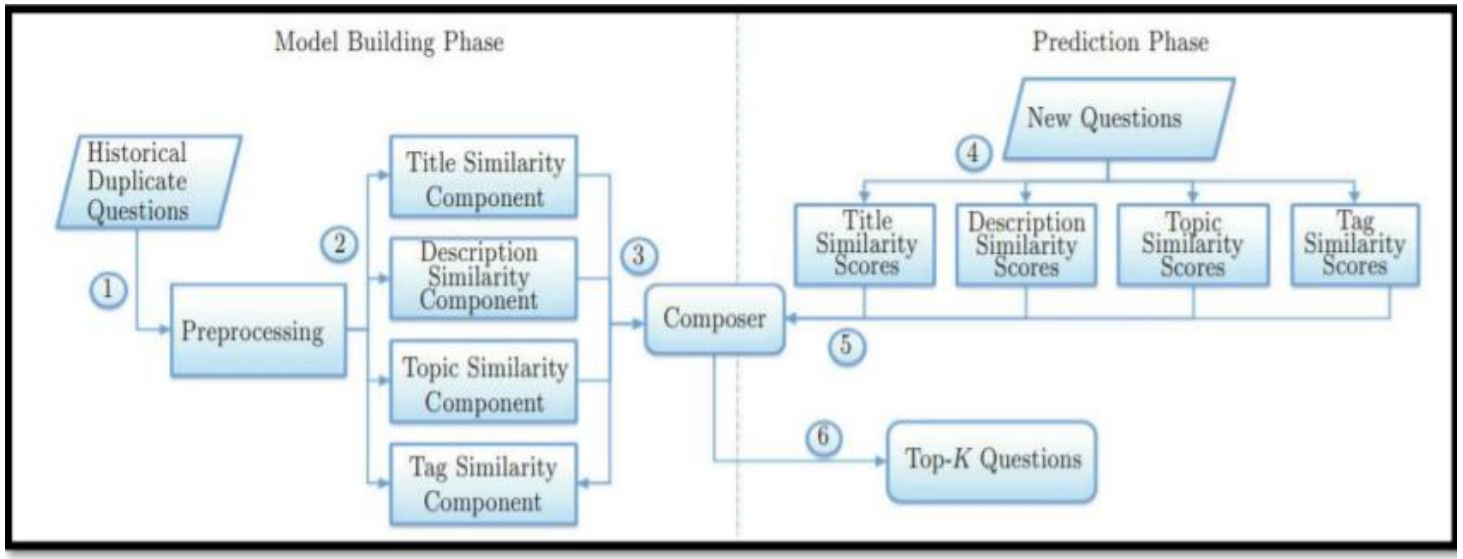
Github Repo: **Click here**

1) Zhang Y, Lo D, Xia X et al. Multi-factor duplicate question detection in Stack Overflow. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(5): 981–997 Sept. 2015. DOI 10.1007/s11390-015-1576-4

Index

Introduction

- The primary goal of our project is to implement DupPredictor which takes a new question as input and returns ‘k’ potentially duplicate/ similar questions as the output.



Dataset

- We collected the Stack overflow question dataset from data.stackexchange.com. It consists of 8000 duplicate and the corresponding original questions.
- Each row of the dataset primarily consists of question Id, Title, description, tags of duplicate and the corresponding original question.

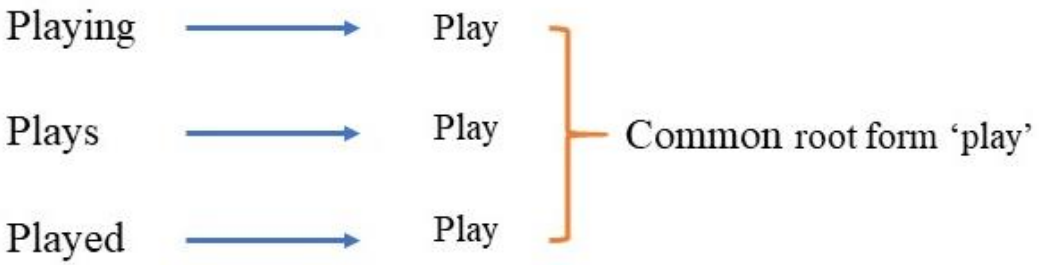
	RelatedPostId	OId	OTitle	OTags	OBody	DId	DTitle	DTags	DBody
0	4	4	How to convert a Decimal to a Double in C#?	<c#><floating-point><type-conversion><double><...	<p>I want to use a <code>Track-Bar</code> to c...	51027658	How to echo a JS variable to php?	<javascript><php>	<p>Is that possible to pass a JS variable to P...

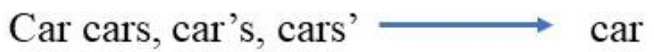
OId represents original question Id

DId represents duplicate question Id

Preprocessing

- We then **tokenize** the text that appears in the title and description of each question, remove common English **stop words**, and perform '**stemming**'.
- Stemming stops the commonly used articles, conjunctions and so on and reduces a word to its root form





Using above mapping a sentence could be normalized as follows:

Similarity scores

- We have 4 components:
  - **Title**
  - **Tag**
  - **Description**
  - **Topic.**
- In the first three initially do **vector space modeling** (VSM), we represent the two titles as two vectors:
- Vecm = (wtm,1, wtm,2, . . wtm,v)
- Vecn = (wtn,1, wtn,2, . . . wtn,v).

$$TitleSim(\boldsymbol{TitleVec}_m, \boldsymbol{TitleVec}_n) = \frac{\boldsymbol{TitleVec}_m \cdot \boldsymbol{TitleVec}_n}{|\ \boldsymbol{TitleVec}_m\ ||\ \boldsymbol{TitleVec}_n\ |}.$$

Latent Dirichlet Allocation (LDA)

- LDA is a topic modelling which takes the title and description and returns the probability distribution of the topics k.
- We first perform LDA and take the probability values to be the vectors and then calculate the similarity scores.

"*Dogs* like to *chew* on *bones* and fetch sticks".
"*Puppies* drink *milk*."
"Both like to *bark*."

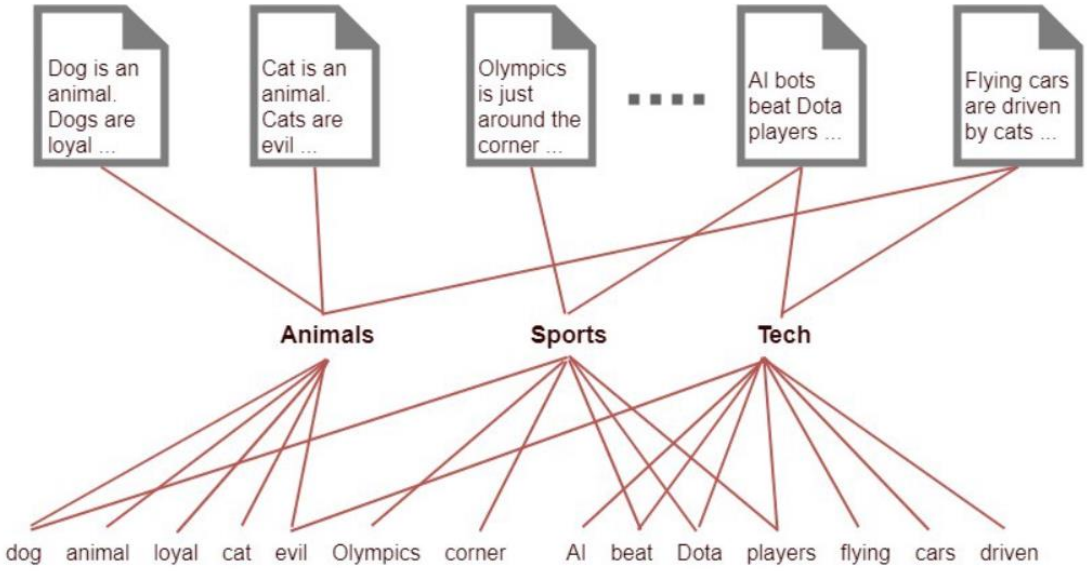


Fig: Deriving topics from documents. (Taken from towardsdatascience)

Composer

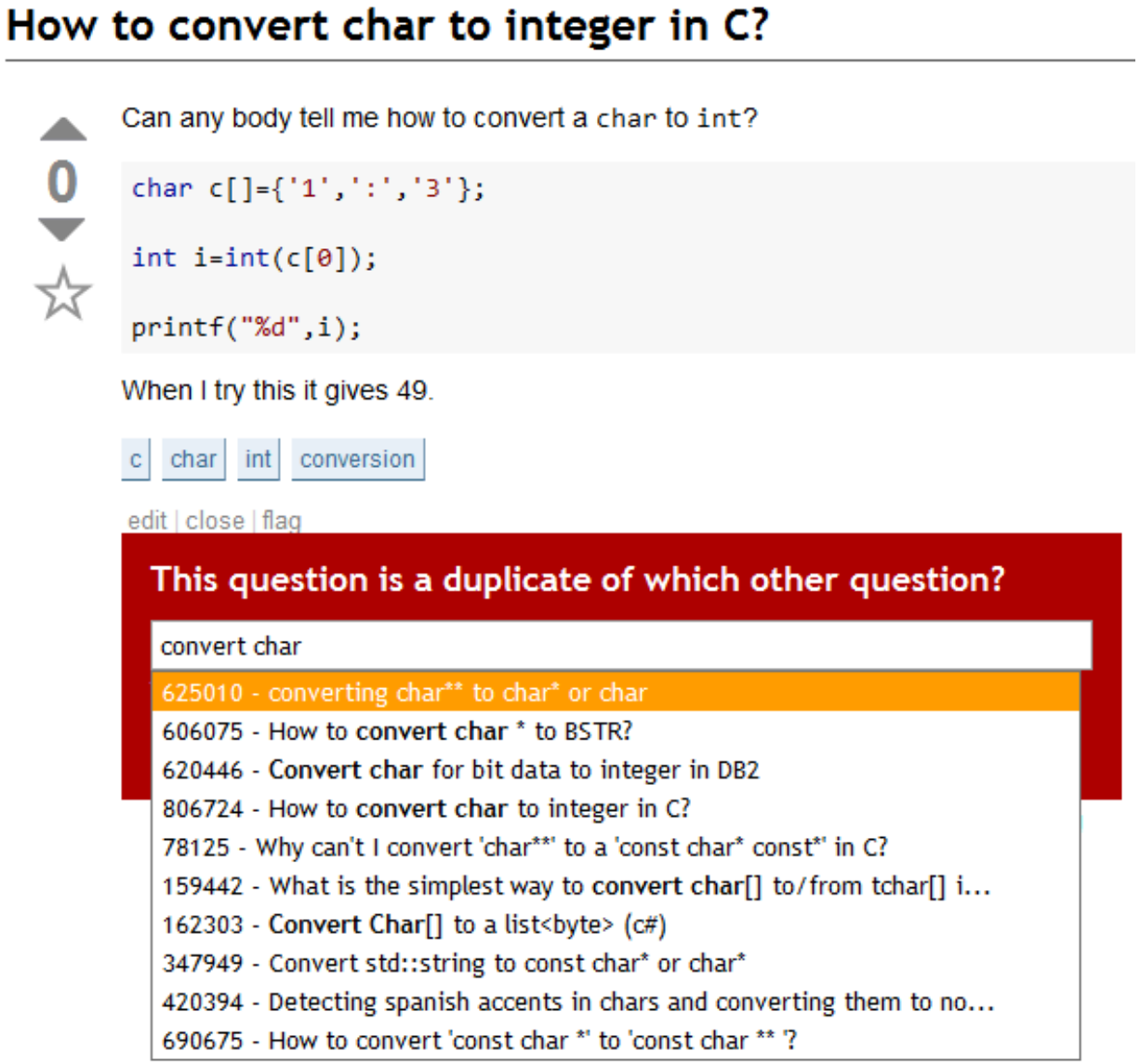
- The composer takes all these similarity scores and calculates

$$\text{CompScore} = \alpha \times \textit{TitleScore} + \beta \times \textit{DesScore} + \gamma \times \textit{TopicScore} + \delta \times \textit{TagScore}$$

- $\alpha, \beta, \gamma, \delta \in [0, 1]$ are calculated using a greedy algorithm approach of time complexity $O(m \times n \log n)$.

Prediction Phase

- Now that we have the values of the coefficients, we can now enter the prediction phase where we calculate the scores with respect to that new question and return the top k answers.



Training & Testing

- To **Train** the *dupPredictor* model, we assumed our original questions as the questions dataset, and out of all the duplicate questions, we used the first 300 questions as train data for training the composer model parameters, (Title, Tag, Body, and Topic similarity scores).
- For **testing** the *dupPredictor* model, we take the first 1200 duplicate questions of the remaining dataset as test data and tested the composer model obtained in the training model.

Results & Evaluation metrics

- Evaluation Metric – **Recall rate@k:** denotes the ratio of number of original questions in the top – k similar questions to the total number of questions.

$$recall\text{-}rate@k = \frac{N_{\text{detected}}}{N_{\text{total}}}.$$

- For computing composer scores, we used pseudo-greedy algorithms, and the optimal composer scores obtained from it are as follows
  - $\alpha = 0.682$, $\beta = 0.64$, $\delta = 0.15$, $\gamma = 0.06$
- After tweaking the scores obtained from the pseudo random algorithm to maximise the accuracy, we get
  - $\alpha = 0.8$, $\beta = 0.51$, $\delta = 0.37$, $\gamma = 0.04$
- The recall-rate obtained with the above composer scores in the testing phase is **Recall-rate@20 = *0.725*** [in the testing phase].

Analysis

- **Contribution of individual components in the dupPredictor model**

Type	Recall-Rate@5	Recall-Rate@10	Recall-Rate@20
Title only	0.30666	0.375	0.445833
Description	0.2525	0.306666	0.37666
Tag only	0.261666	0.331666	0.398333
Combined	0.455	0.585	0.725

Additional data sets

- To test the model, we take a new dataset which contains questions related to physics, taken from physics.stackexchange.
- Unlike, the stackoverflow dataset, which contains both coding blocks and english words, this dataset contains mostly english words.
- We notice that the $\alpha > \beta > \delta > \gamma$ does not hold in this case.
- Instead, we have β to be the **highest value** meaning that it gives more weightage to the description component.

Analysis

- For the dataset with physics questions, we noticed the following

	α	β	δ	γ	Recall-Rate@20
Stack overflow	0.8	0.51	0.37	0.04	0.725
Physics questions	0.4804	0.9271	0.52995	0.00944	0.77

- We can notice that the recall rate for stack overflow is slightly less because it contains mostly coding blocks and not english words, so information is lost in description due to preprocessing the coding block and thus resulting in lesser weightage for description component.

GUI

- A simple python tkinter[1] based GUI was developed where a question is taken as input through multiple fields such as "Title", "Body" and "Tags".
- It outputs top K similar questions to the given question depending on their similarity scores.

1) https://docs.python.org/3/library/tkinter.html

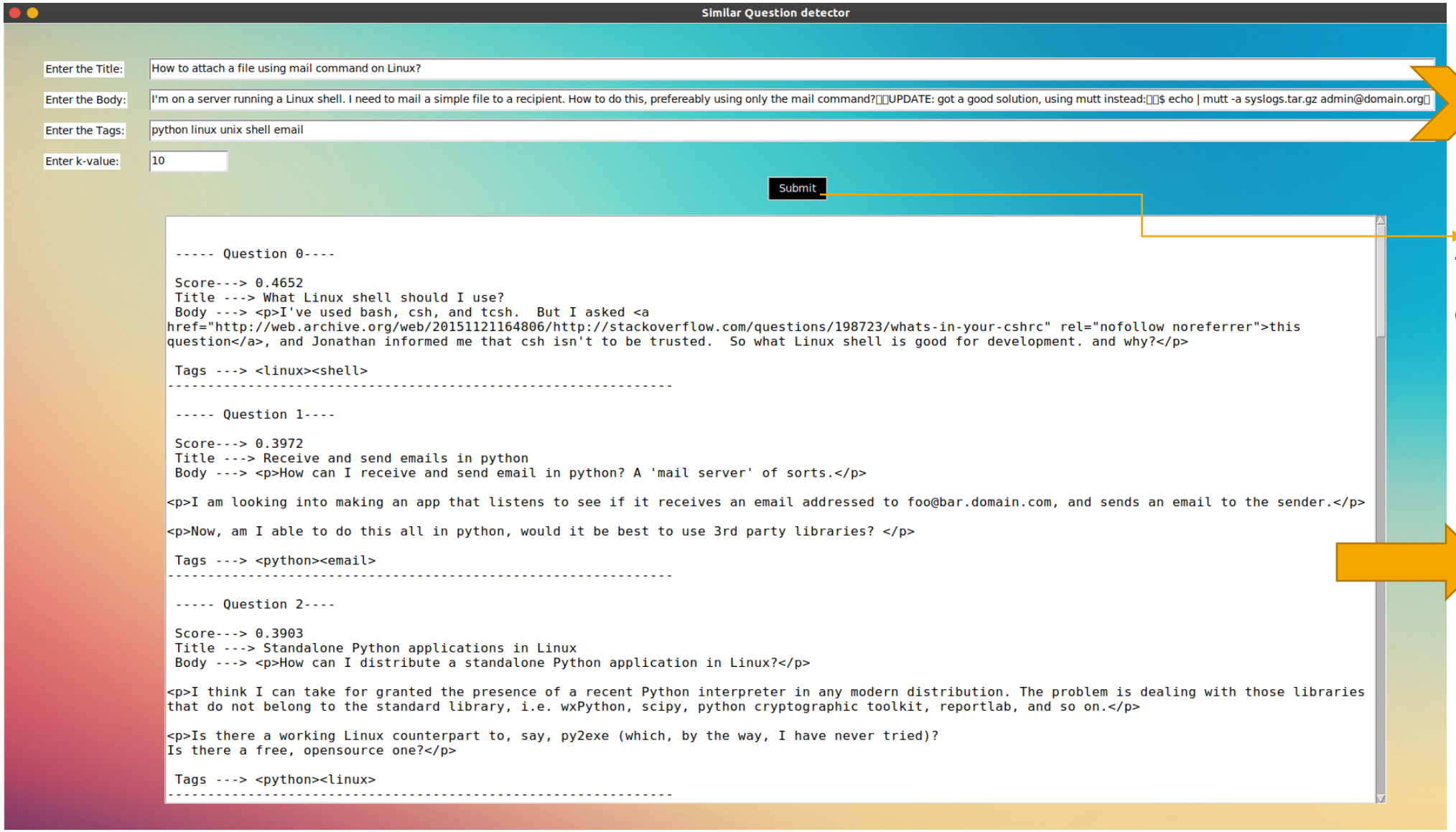
Input Format:

"Enter the title" - text

"Enter the body" - text or HTML body

"Enter the tags" - tags separated by ',' or ' ' (whitespace) or '<>'

"Enter k-value" - A integer value



Input Parameters for a question

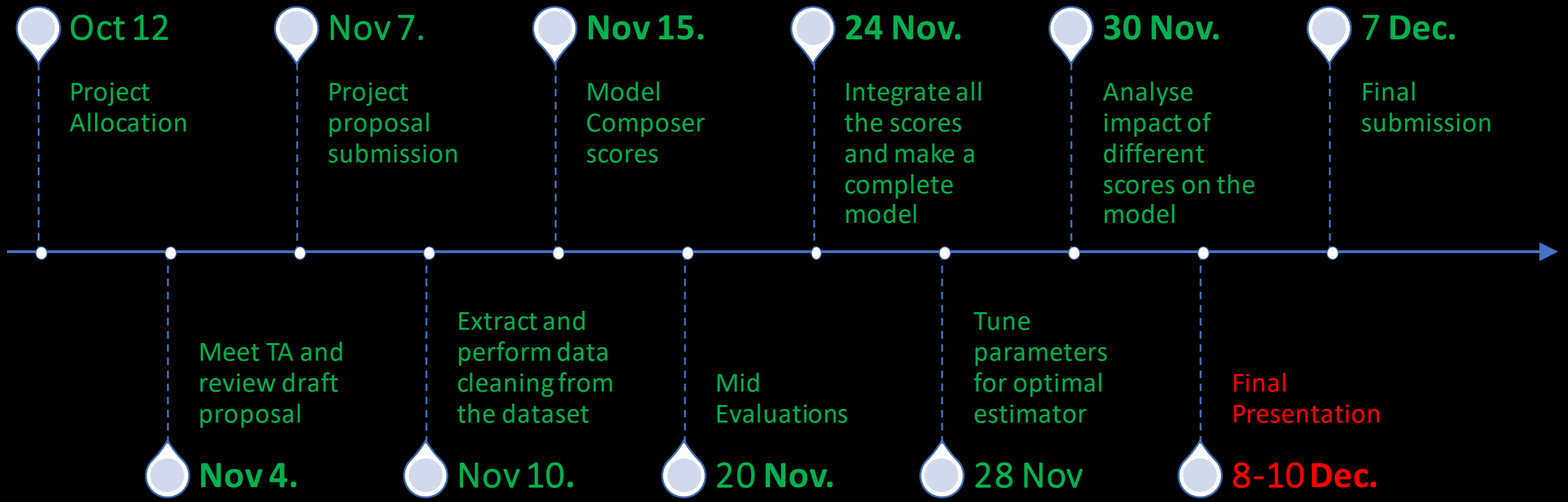
Test the model by clicking Submit

Detected Similar questions

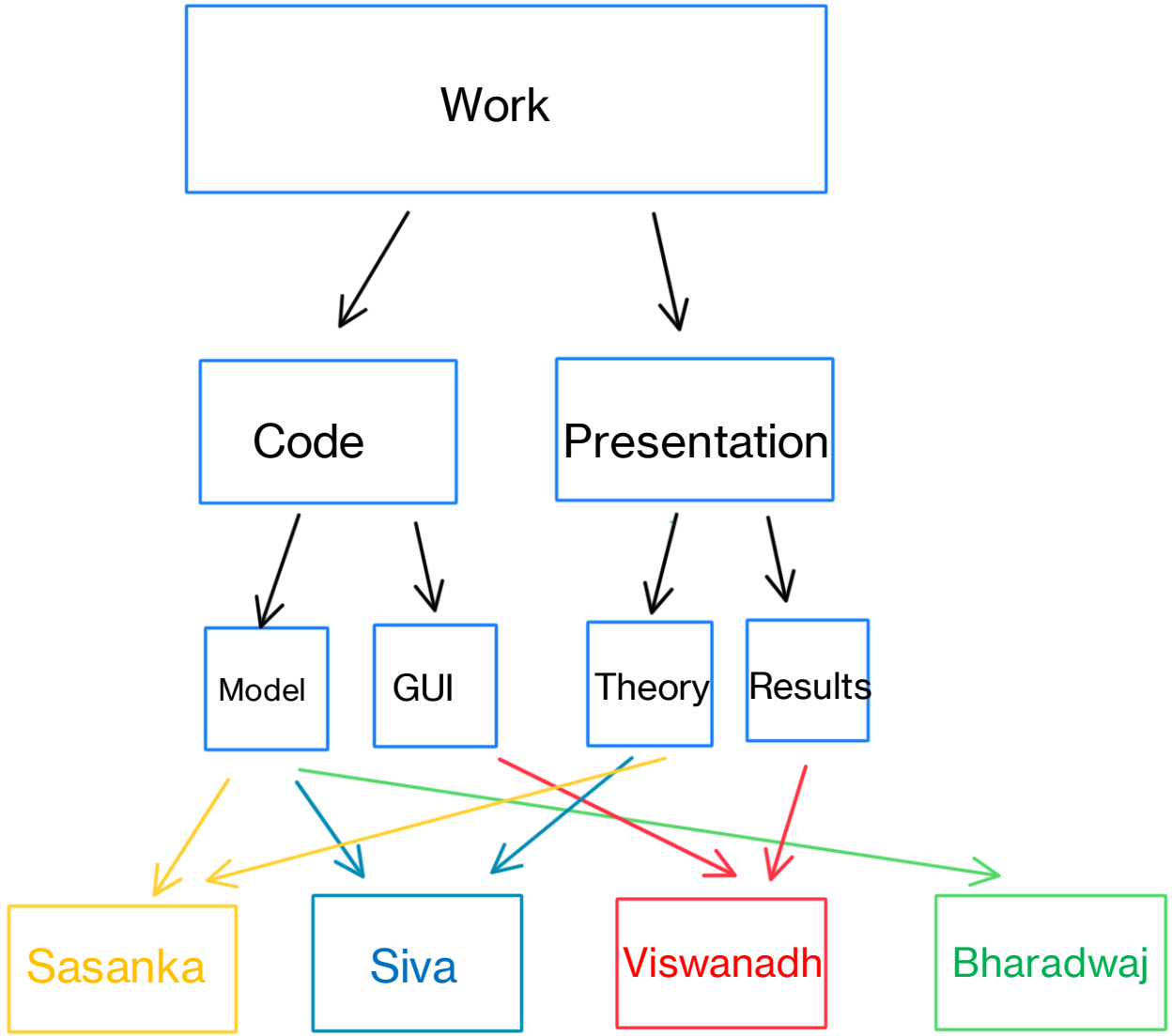
Fig: Preview of GUI

Timeline

Work Completed
Yet to be done



Member responsibilities



Thank You!