Multi-Factor Duplicate Question Detection in Stack Overflow (Team's proposal)

Paper Name: Multi-Factor Duplicate Question Detection in Stack Overflow

Project ID: 26

Team Name: Hyperplane Scavengers

Team members: Gurkirat Singh, Anmol Agarwal, Shrey Gupta, Pratyush Pratap Priyadarshi

Mentor Name: Sireesha

Overview and problem statement

Brief description of the working of StackOverflow

What is StackOverflow: Stack Overflow is a question and answer website for professional and enthusiast programmers and developers. It features questions and answers on a wide range of topics of computer science.

What is the traffic the site generates?

- 14 million registered users
- 21 million questions and 31 million answers already present
- 3700 new questions generated every day

How is the site moderated?

- Users can ask questions, answer questions, upvote/downvote both questions and answers etc
- Users with valued contributions earn reputation
- Users with a reputation above a certain threshold ‘X’ can comment on and even edit other’s questions
- Moderators can flag posts, edit posts and answers, delete questions/answers, mark questions as duplicates of one another etc

Problem the paper wants to solve:

The paper wants to develop an automated system to detect if a newly posted question is a duplicate of any of the earlier asked questions

Why is this an important problem to solve?

- **Manual cross-checking for duplicates by site moderators is not good enough as:**
  - **It is impossible for moderators to cover the entire search space, so a large number of actual duplicates go undetected:** With over 3700 new questions every day and with each of them possibly being a duplicate of any of the earlier posted questions, the search space for potential duplicates is extremely large (21 million).
  - Might take a lot of time for moderators to flag a duplicate leading to:
    - Large waiting time by the person who asked the question even though the question has already been answered earlier
    - Wastage of effort by an answerer when he spends time writing an answer to an earlier question

- The system proposed by the paper (**DupPredictor**) can help identify duplicate questions and provide relevant solutions much faster and more efficiently by suggesting top k questions from the past to which a newly asked question might be a duplicate of. This will help as:
  - The original poster can skim past the duplicate suggestions
  - In case, the question about to be asked was actually a duplicate, the original poster will be able to get an **immediate answer**, hence reducing the waiting time
  - The site moderators will have an **comparatively smaller sample** space to consider candidates for duplicates for a newly asked question Q

Dataset

- The dataset used (published as a part of MSR 2013 mining challenge) spans from 2008 to 2012.
  - It has 20,65,998 questions out of which 1641 have been marked as DUPLICATES of some previous questions by StackOverflow Moderators.
  - The authors claim that they found some questions incorrectly “labelled” as duplicates and hence, on manual inspection, reduced the set of duplicate questions from 1641 to 1528.
- Since the dataset involves posts before 2013, we can assume that all potential edits, “duplicate detection by a StackOverflow admin” etc have been already done and there are unlikely to be any changes for such posts and such questions are considered stable
- The dataset currently has the following files:
  - Badges.xml
  - Comments.xml
  - Posts.xml
  - Users.xml
  - Posthistory.xml
  - Votes.xml

Approach used by the paper

Pre-processing

- The Authors initially preprocess, clean and parse the dataset using **WVTool (or a similar library)** to extract the title, description, and tag for each question.
- The data is then sorted on the basis of time to simulate real usage of temporal constraints: “*Future can’t be used to predict the Present*” and the initial set of 300 duplicate questions are taken as the training sample.

Model

- After the dataset curation, the authors create the individual similarity components for the title, tag and description. Each component creates a bag of word embeddings representation of the text where the weight of each word in the text is represented as:

$$wt_{q,i} = \frac{n_{q,i}}{\sum_v n_{q,v}}$$

After calculation of the text embeddings, the paper implements a cosine similarity measure with all the other questions to calculate similarity b/w each question pair given by the following methodology:

$$TitleSim(\mathbf{TitleVec_m}, \mathbf{TitleVec_n}) = \frac{\mathbf{TitleVec_m} \cdot \mathbf{TitleVec_n}}{|\mathbf{TitleVec_m}||\mathbf{TitleVec_n}|}$$

This process is followed out for the creation of the title, tag and description similarity measures.

- The authors also use a topic similarity component to identify the semantic meaning of the post. The embeddings for this similarity is created using Latent Dirichlet Allocation (LDA) to get a vector of topic probability on which the cosine similarity is measured for each question pair.

- The paper then merges these components together using a greedy composer and the weighted sum of the individual components. They estimate the correct parameters (α, β, γ, δ) which are the weights of each component, which will provide the match score between two questions using the above-mentioned components.

$$\begin{aligned} Composer_{nq}(oq) = \alpha * TitleSim_{nq}(oq) \\ + \beta * DesSim_{nq}(oq) \\ + \gamma * TopicSim_{nq}(oq) \\ + \delta * TagSim_{nq}(oq) \end{aligned}$$

- The top K master questions with the highest scores are taken as plausible duplicates for the new input question.

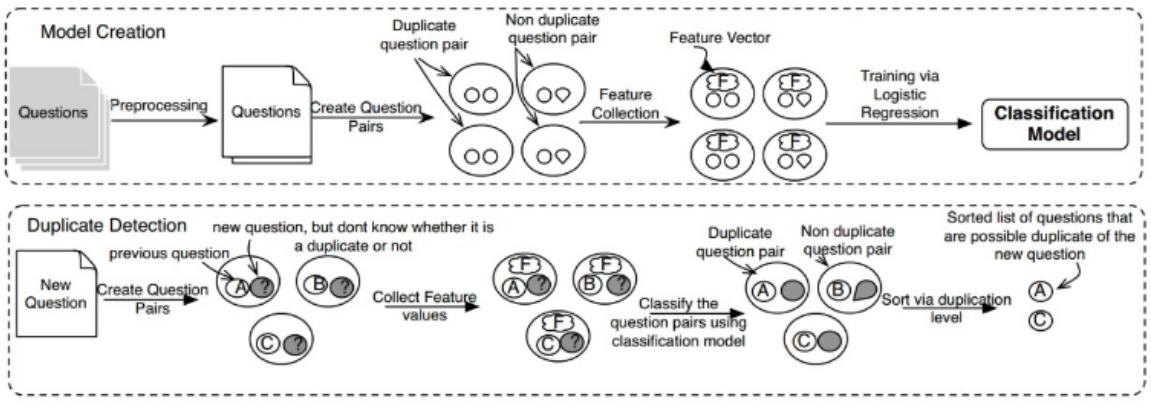
Results and Experiments

- The authors compare the result obtained from **DupPredictor** with the standard search engine of Stack Overflow, and the method proposed in Runeson et al. paper which involves the Cosine Similarity of title, and the description concatenated together according to the metric recall-rate@K, K = 5, 10, 20

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

- DupPredictor is also compared with its four components to show whether the composite method DupPredictor performs better than its constituent components
- The paper attempts to find the effect of varying the number of duplicate questions in the training set on the effectiveness of the **DupPredictor**?
- How varying the weights of the parameters of the composer affect the **DupPredictor** prediction accuracy?
- The paper also proves that the training time and prediction time of the composer is efficient and can be used in a practical setting via timing the model training and prediction times
- It also conducts an Ablation Study on the Topic Similarity Component to find its effectiveness in the whole model

Expected Deliverables



From the implementation of the project, the expected deliverables are:

- **Deliverable 1:** Replication of the methods used in the paper. The findings and the results of the goals and objective section along with the results and experiments would be the main deliverable along with the code of the implemented model from scratch. The current paper does not have the labelled
- **Deliverable 2:** Doing an extensive analysis of the dataset, if time permits [See details below]
- **Deliverable 3:** Expanding the dataset size (using the latest dataset dumps present here), if time permits
- **Deliverable 4:** Improving the accuracy and speed of the model, if time permits [see details below]

Planned data analysis

- Questions of which tags occur most frequently in the dataset
- Questions of which tags are asked most frequently on the dataset
- Which tags have the highest duplicate marked percentage (i.e., the ratio of duplicate questions containing that tag to the total number of questions having that tag)
- The time it generally takes for the post to be marked as a duplicate

- Analyze the profile of users who ask the duplicate question (using attributes like when did the user join the site, what sort of questions had he asked earlier etc)

Our ideas regarding improving speed and accuracy which we would like to explore

Let Q be a new question that has been posted by the user at the time T and Let S be the set of all questions asked in StackOverflow before time T

- We want to improve on **TWO** MAIN aspects:
  - **Time/Speed**
    - Pruning potential candidates for a question Q by imposing restrictions such as questions must have common tags
    - The model used in the paper searches for duplicates for Q among all questions in set S. We define a new search space S' as below. The smaller size of the search space will lead to an increase in speed.

$$S' = \{x \in S | \ S \ has \ a \ tag \ common \ with \ Q \quad and \quad S \quad has \ atleast \ one \ answer\}$$

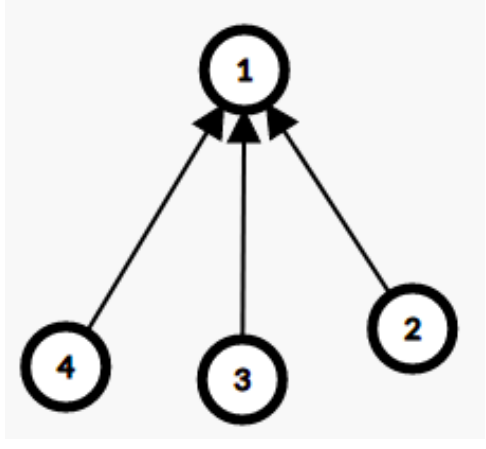
  - **Accuracy**
    - Exploring different similarity measures apart from Cosine Similarity such as WMD etc.
    - Being a basic model, both LDA and Bag of Words are expected to fail to capture nuances of natural language. We plan to improve on this using:-
      - Exploring different methods to form Word representations apart from LDA and BoW:
        1. Word2Vec
        2. Glove
        3. Pre Trained Bert Tokenizer and Embeddings
      - Introducing alternate features like:-
        1. Jaccard Coefficient on the combined title + description + tag text as some words present in the title for one question may be present in another section for another question

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

        2. Jaccard for NER to see technical words similarity
        3. Jaccard for Entity Type to capture similarity b/w questions which contain C++ and Java (which are 2 entities but of a similar type)
      - As a result, our composer would now look like:-

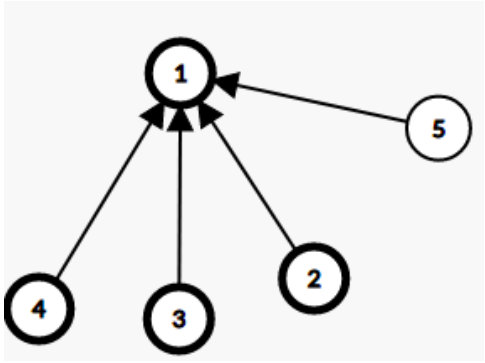
$$Composer_{nq}(oq) = \Sigma_i^n * f_i$$

        where f_i is the i^{th} composite feature of the new proposed model
      - **Modifying the policy to detect when a prediction is valid or not:** Let the questions be represented by nodes 1,2,3 and 4. Let a directed edge from A to B mean that A has been detected as a duplicate of B. Currently, let a new question i.e. node 5 be introduced such that it is a duplicate of node 1 and hence, also a duplicate of nodes 2, 3 and 4.

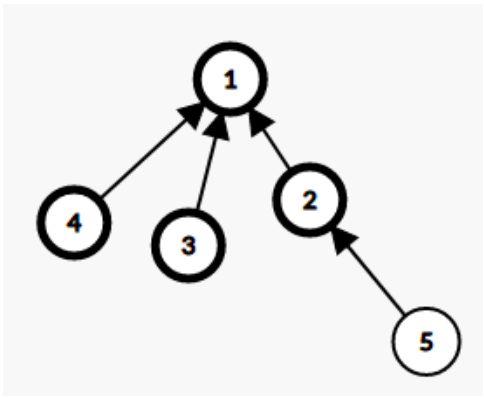


The state of the graph before the new question has been inserted

Now, the current paper considers an accurate prediction only if the model predicts an edge from node 5 to node 1 but does not consider a prediction of an edge between node 5 to any of the nodes 2,3 or 4 as accurate. But we would like to remedy this by detecting a prediction (5 --> node N) as accurate if N is not just equal to 1 but also if N is a child of 1.

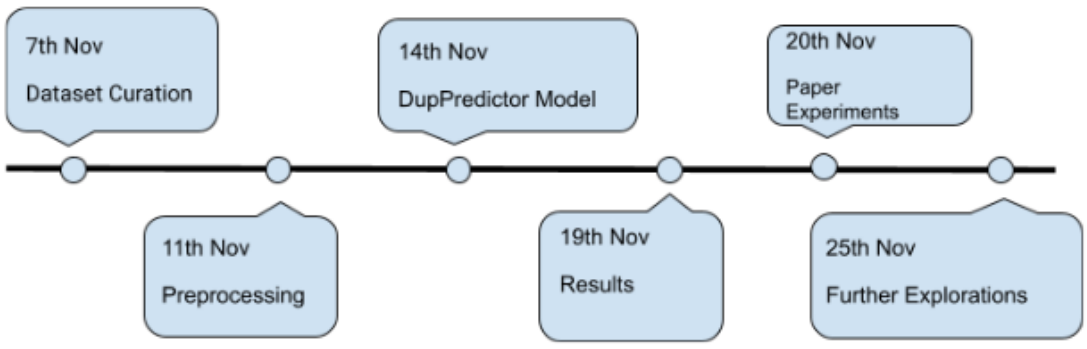


DupePredictor just considers an edge of the above nature as accurate



We want to consider an edge Q and a child of 1 to be an accurate prediction as well

Milestones and Timeline (Start Times have been mentioned)



Work distribution

Aa Module	People
Dataset	Pratyush, Anmol
Pre Processing	Shrey, Gurkirat, Pratyush
DupPredictor Model Creation and Results	Shrey, Gurkirat, Anmol, Pratyush
Paper Experiments	Shrey, Anmol, Gurkirat
Further Explorations	Shrey, Gurkirat, Anmol, Pratyush