THE IBM HACK CHALLENGE

User Query on Stack Overflow

- Github link : StackOverflow_UserQuery
- Video Link
  - Brief Explanation of the code
  - Few more examples

Team Members:

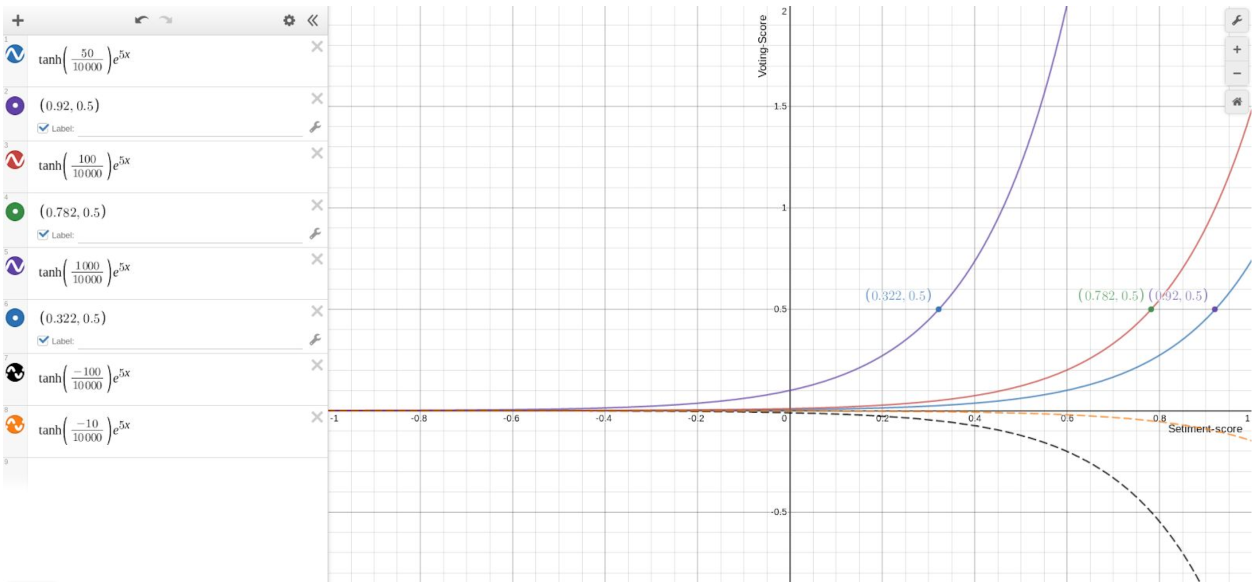
- Samyak Jain
- Sayak Kundu
- Yudhik Agrawal

StackOverflow's search is too rigid. But human language is not. While searching for a solution to an error, it might become very frustrating to describe it in a fixed syntax. Our solution to this problem is that we could make ***use of present infrastructure of StackOVerflow*** by using their APIs and remove it's rigidity. We incorporated features like ***automatic tag extraction, similarity search and sentiment analysis*** which ***increases the quality of questions and answers*** that the user gets from our system.

What is the uniqueness/novelty added by you to the defined problem statement?

- ***Identifying Most relevant questions***
  - We do not just rely on text similarity
    - We use Universal Sentence Encoder (USE) which returns a higher value than Bleu Score 1, 2 based on context and relevance along with text similarity. [current state-of-the-art textual similarity framework]
  - Tag extraction and ranking them with respect to upvotes reduces our search space drastically.
  - We still have the scope of finding related questions from the most relevant questions which would be helpful.

    - **NOTE:** To not use up all the API calls, we fixed the starting time to Wednesday, January 1, 2014 12:00:00 AM and with minimum votes equal to 40. All Questions with these parameters will be fetched.
- ***Identifying Most Relevant Tags from questions***
    - Tag extraction happens using a neural network, which removes the dependency on sequence matching. It reduces the search space which increases efficiency.
- ***Identifying top 'k' solutions***
    - We not only consider the most upvoted answer and the reputation of the person answering, we also consider the feedback of the answer from the feedback by doing ***sentiment analysis*** on the comments and then decide the final order accordingly.
    - **NOTE:** To not use up all the API calls, we fixed the starting time to Wednesday, January 1, 2014 12:00:00 AM. All Answers after this date will be fetched.
- ***Model Deployment***
    - Improved UX by making it faster to resolve a query by loading all the pre-trained models and creating sessions while starting the server.

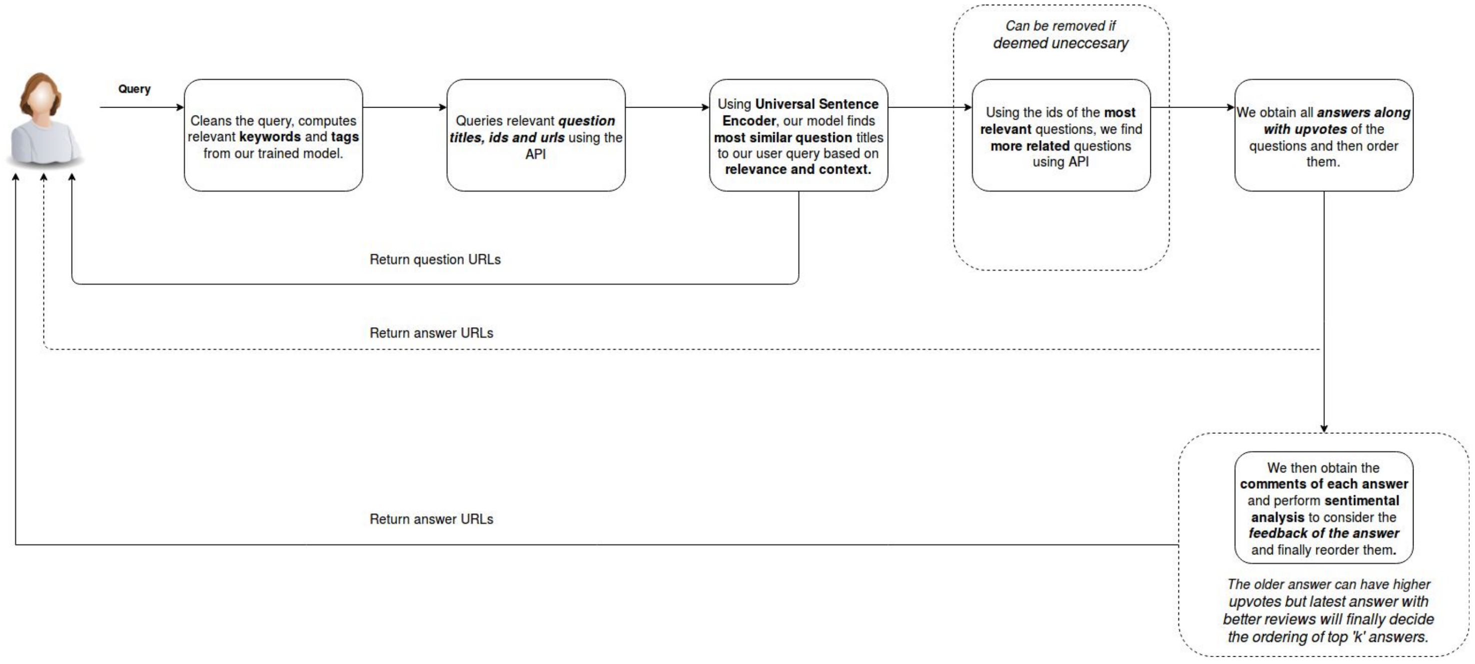


How is the proposed solution impacting the business? How are the business processes simplified or bringing value over the existing process?

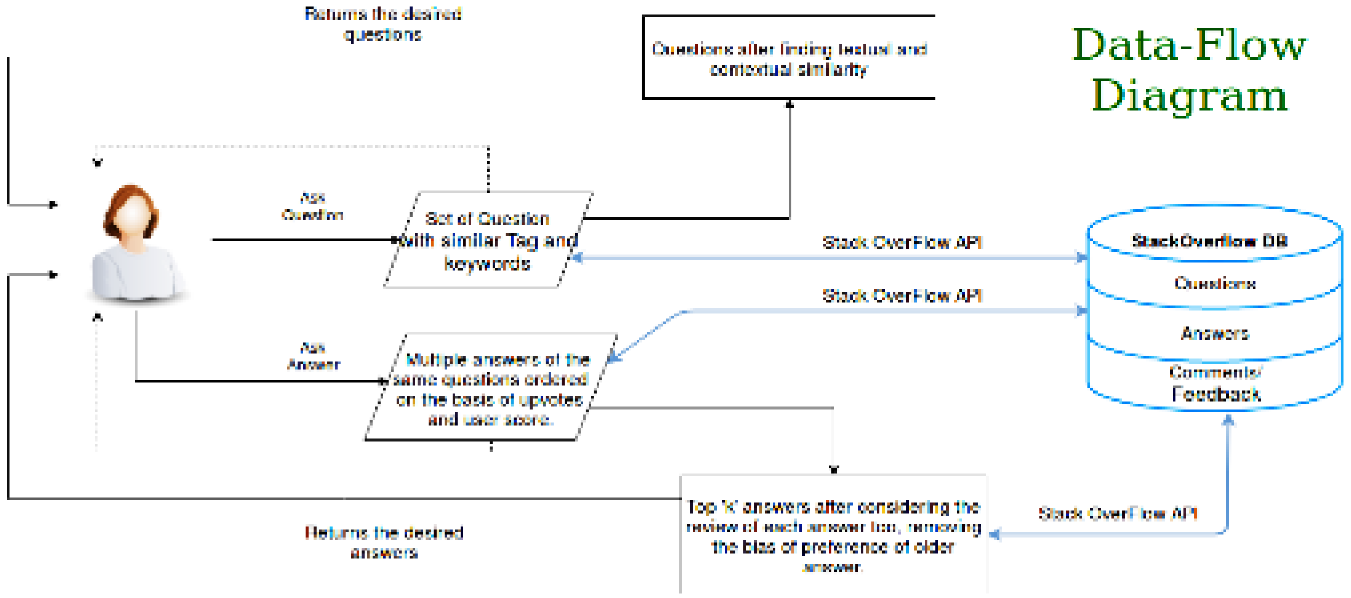
- The proposed solution can identify the very recent answer as the best one even after having less upvotes based on the rating which makes it much easier to bring changes and ***not rely on the old school methods***.
- **Scalability**
    - In our product, we made use of Django and kept it separate from the persistence layer. Django web nodes have no stored state, they scale horizontally - just fire up more of then when you need them. Being able to do this is the essence of good scalability.
    - TensorFlow models are highly scalable, flexible and portable. They can be easily retrained with new or more data and can also be visualized or used as a server itself. Hence it was our choice.
- **How easy it is to deploy**
    - We tried to keep our product deployment steps to bare minimum (less than 10). This keeps it easy to understand and work with.
- **Cost/revenue/Time saving**
    - The cost of our product depends mainly on the following:
        - API costs
        - Scalable Deployment costs
    - These can be optimized according to the need and usage of our product.
    - Time saving is one of the major qualities of our product. Instead of glancing through all of the questions, answers and comments and getting confused on which answers to choose, our product saves time by providing the most sensible, community upvoted answer along with the analysis of the comments.

Architectural flow of the proposed solution, with the mention of technologies to be used in developing the solution.

- **Technologies used:**
  - UI/UX: **Django**
  - Language/Libraries: **Python, Tensorflow**
  - Web Services: **StackOverflowAPI**
  - Data Collection: **StackExchange Data Explorer**



Query
Cleans the query, computes relevant **keywords** and **tags** from our trained model.
Queries relevant ***question titles, ids and urls*** using the API
Using **Universal Sentence Encoder**, our model finds **most similar question** titles to our user query based on **relevance and context.**
Can be removed if deemed uneccesary
Using the ids of the **most relevant** questions, we find **more related** questions using API
We obtain all ***answers along with upvotes*** of the questions and then order them.
We then obtain the **comments of each answer** and perform **sentimental analysis** to consider the ***feedback of the answer*** and finally reorder them.
The older answer can have higher upvotes but latest answer with better reviews will finally decide the ordering of top 'k' answers.
Return question URLs
Return answer URLs
Return answer URLs



Data-Flow Diagram
Returns the desired questions
Questions after finding textual and contextual similarity
Ask Question
Set of Question with similar Tag and keywords
Stack OverFlow API
Stack OverFlow API
StackOverflow DB
Questions
Answers
Comments/ Feedback
Ask Answer
Multiple answers of the same questions ordered on the basis of upvotes and user score.
Top 'k' answers after considering the review of each answer too, removing the bias of preference of older answer.
Stack OverFlow API
Returns the desired answers

Define the scope of work to be implemented in the project with modules etc.

- **Tags Extraction -** This module extracts tags from the user query and helps us in reducing the search space which increases efficiency.
- **Query Similarity -** This module helps in measuring and showing question titles similar to that of user query. This further reduces the search space.
- **Answer Ranking -** This module helps in ranking the answers on the basis of sentiment of the comments left by the user and user upvotes.
- **UI/UX -** This module helps in rendering the results of all the above modules for users to see.