# **Finding the Most Relevant Answer from Discussion Forum using NLP**

*Divjot Dhody, Ajith Balaji Nagarajan, Pooja Ramaswamy, Khang Doan*

*Abstract*— **Discovery of relevant content has attracted increasingly more attention due to its ability to distill the most relevant information from noisy data. The major challenges of inferring true information on text data stem from the multifactorial property of text answers (i.e., an answer may contain multiple key factors) and the diversity of word usages (i.e., different words may have the same semantic meaning). To tackle these challenges, in this paper, the proposed method will jointly groups the keywords extracted from the answers of a specific question into multiple interpretable factors, and infers the trustworthiness of both answer factors and answer providers. After that, the answers to each question can be ranked based on the estimated relevancies of factors. Experiments on three real-world datasets show that the proposed model can accurately select the most relevant answers,** **by using the various factors present in the answers.**

***Keywords*—NLP, Machine Learning, Python, React.js, Stackoverflow.**

## Introduction

In the era of increasing internet usage, information used on the internet is proven to be both beneficial and detrimental to the users. With the democratization and anonymization of data, anyone and everyone can share their thoughts and expertise with a click of a button. Simply put, any users can access and contribute to the vast collective knowledge with unprecedented ease. This brings tremendous value to the field of academia and professional by facilitating the data collection process, letting the users focus on their primary tasks. All the benefits of the collective knowledge are not present without its short coming. Before the data collected are deem to be useful, users must validate what they’ve collected. This proves to be a laborious task especially the task involves finding or troubleshooting technical solution.

One prime example would be StackOverflow, a website that is ubiquitous in the engineering field for both academia and professional settings. StackOverflow remains one of the most used websites when it comes to finding and obtaining solutions to technical problems. Any users on that website has experience with trying to find the most appropriate answer from a long list of comments. This process involves manually looking for keywords, checking the number of up-votes each answer has, and trying out solution that may not be right.

The proposed solution is a system that takes in users’ query and matching it up with the most appropriate answer.

## Architecture

The system will comprise of three main components: front end, and backend. The front end was Bootstrapped with Create React App. Create React App is divided into two packages:

* create-react-app is a global command-line utility that you use to create new projects.
* react-scripts is a development dependency in this project.

There will never be a need to update create-react-app itself; it delegates all the setup to react-scripts. When the create-react-app is run, it always creates the project with the latest version of react-scripts. All the new features and improvements in newly created apps will be obtain automatically. To update an existing project to a new version of react-scripts, open the changelog to find the version that is currently on (check package.json in this folder), and apply the migration instructions for the newer versions. In most cases bumping the react-scripts version in package.json and running *npm* install in this folder should be enough, but it’s good to consult the changelog for potential breaking changes.

The flow of the front-end was intended toward simplicity. The users submit a query and the best answer should show for the users (see *figure 1* and *Figure 2* in appendix).

### Algorithm

The core of this application is the backend python code that does the actual work. There are several factors to determine on what would make an answer the most relevant for a particular question from amongst the pool of other answers. These include: the number of non-stop words, the number of repetitive words present in the answer, the answer upvotes, the answer downvotes, the reliability of the user who wrote the answer, the number of unique words present in the answer, the reputation of the answer, the count of comments written for the answer, the relevancy between answer body, the question body and the question tags and the information provided by the answers. In this model, all these factors are obtained for all the answers for a particular question and that is used to predict the most relevant answer for that particular question.

### Process Flow

The very first step involves cleaning up of the raw data. This involves the removal of any HTML tags present in the data. After that the data is then put in a data frame with each data object under their corresponding attributes. This is then followed by sorting and exporting the cleaned data frame into a separate dataset, which will then be used for the further analysis.

The next step is to extract the features mentioned above from the cleaned-up dataset. For this purpose, a separate class function names “feature\_vectors” was constructed in python. This class has several methods, with each method performing the necessary preprocessing and some in extracting the necessary factors from the data. A method to extract the information from the given dataset is being constructed. The function tries to extract the information from the answer body using Markov chains and entropy. A Markov chain is a mathematical system usually defined as a collection of random variables, that transition from one state to another according to certain probabilistic rules. These set of transition satisfies the Markov Property, which states that the probability of transitioning to any particular state is dependent solely on the current state and time elapsed, and not on the sequence of state that preceded it. This unique characteristic of Markov processes renders them memoryless [1]. The relevancy between the answer body, the question body and the question tags are computed by the getRelavancy method. The properties module in python is used for this purpose, which has functionalities to get relevancy between the textual data. The Uniqueness of the answer factors is also being determined using the unique method in that module. The non-stop words and the subjectivity of the answer body is also being obtained using the corresponding methods in the same module. The answer score, the answer upvotes and the answer downvotes multiplies with a negative value are obtained as separate features. The number of comments for an answer are obtained from another method. The readability of the answer is obtained from the readability index method, which uses the readability module in python.

The reliability of each user is inferred according to the answers they provide. As aforementioned, the answer of a user *u* may merely cover part of the trustworthy answer factors, and at the same time may consist of untrustworthy answer factors. For instance, some users may only provide the factors that they are very confident of. On the contrary, other users may cover a broad collection of answer factors with different trustworthiness in their answers. This naturally motivates us to use a two-fold score to model the reliability of a user. Suppose all the answer factors and their truth labels is known in advance, for all the questions and their answers, TPu and FPu is used to denote the number of trustworthy and untrustworthy answer factors that are covered by the answers from user *u* (i.e., the number of true positive and false positive factors), respectively. Similarly, FNu and TNu was used to denote the number of trustworthy and untrustworthy answer factors that are not covered by the answers from user *u* (i.e., the number of false negative and true negative factors), respectively. Based on these statistics, it is possible to intuitively use the false positive rate (defined as: F Pu/F Pu+T Nu), and the true positive rate (defined as: T Pu/T Pu+FNu) to fully characterize u’s reliability [2]. This methodology is used getRepu method to get the user’s reputation.

The properties module mentioned above was constructed to perform Natural Language processing (NLP) of the input dataset. The subjectivity of the answer is obtained by performing sentiment analysis. The non-stop words, unique words were also obtained from the same module. The spell check is also performed to make the algorithm more powerful. The relevancy is calculated using this module. For this purpose, it extracts the keywords from the questions and answers and calculates the relevancy value.

The tf-IDF function is used to compute the Inverse Document Frequency (IDF) after converting the questions and answers into sparse matrices, which would make it easier for the analysis to perform. The IDF basically tries to reduce the impact of the dominant keywords from affecting the prediction. The model was trained and tested on five different classifiers such as Decision Trees classifier, Support vector machine classifier, k-Nearest Neighbor classifier, Random forest classifier and Naïve Bayes classifier, using a small portion of the labeled data. The training and the test classification scores for each model was obtained as shown in the table below,

|  |  |  |
| --- | --- | --- |
| Model | Training Score | Test Score |
| Decision tree Classifier | 94.3333% | 89.9731% |
| Support Vector Machine Classifier | 91.2482% | 88.7169% |
| K-Nearest Neighbor Classifier | 79.0038% | 76.9877% |
| Random forest Classifier | 83.4731% | 83.0299% |
| Naïve bayes classifier | 75.0944% | 70.1176% |

*Table 1. Comparison of training and test scores of various models.*

From the above analysis, decision trees classifier is being selected for the final analysis. The input question is obtained from the user and the dataset is searched for its corresponding answers to be passed into the properties module and the feature vector module to obtain the answer candidates which is then passed to the decision trees classifier to obtain the most relevant answer for that particular question. If the question typed by the user is not present in the dataset, it searches for the relevant questions in the dataset, takes those answers, repeats the previous step to obtain the most relevant answer for that particular question.

### The Dataset

The dataset was obtained from stackoverflow.com which has over 38,900 question and answers for across various topics. The raw dataset had several HTML tags associated with it, which has to be removed to make it clean. The cleaned up dataset was then used as the database for the application to perform its function.

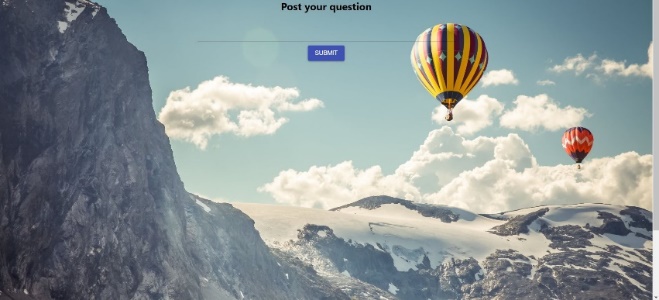
## Conclusion

In conclusion, the implemented system was successful in providing the top answer for each question. The solution that was implemented was a proof of concept that can be extended to real world application and for future enhancement.

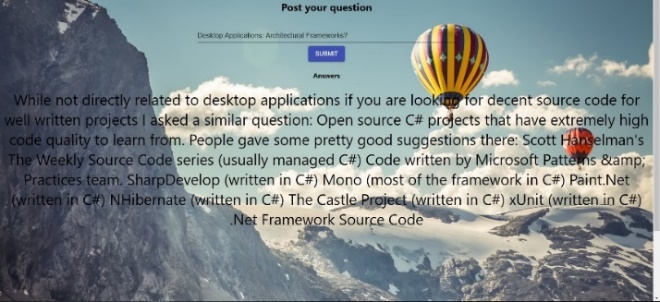
### Future Enhancements

Since the implemented system only works for StackOverflow because of the dataset, having this system works for other website would not be far from reach. By changing the dataset and a few of the weighting factors that the algorithm looks at, this system can be extended to validating reviews, job postings, real estate posting, etc.

## Appendix



*Figure 1. Front end website before query submission*



*Figure 2. Front end website after query submission*

## Acknowledgment

We would like to thank professor Rakesh Ranjan for exposing us to NLP and other industry standard data analysis tools to help us with the project.

Repository: <https://github.com/A-B-N/CMPE272-Project-Group-20>

### References

[1] TextTruth: An Unsupervised Approach to Discover Trustworthy Information from Multi-Sourced Text Data Hengtong Zhang, Yaliang Li, Fenglong Ma, Jing Gao, Lu Su SUNY Buffalo, Buffalo, NY USA Tencent Medical AI Lab, Palo Alto, CA USA

[2] DavidMBlei, AndrewY Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. JMLR 3, Jan (2003), 993–1022.

[3] Mohamed Bouguessa, Benoît Dumoulin, and Shengrui Wang. 2008. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In Proc. of SIGKDD. 866–874.

[4] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation

of observer error-rates using the EM algorithm. Applied statistics (1979), 20–28.

[5] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. 2009. Integrating

conflicting data: the role of source dependence. PVLDB 2, 1 (2009), 550–561.

[6] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. PVLDB 8, 9 (2015), 938–949.

[7] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei

Han. 2014. A confidence-aware approach for truth discovery on long-tail data.

PVLDB 8, 4 (2014), 425–436.

[8] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. Resolving

conflicts in heterogeneous data by truth discovery and source reliability

estimation. In Proc. of SIGMOD. 1187–1198.

[9] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava.

2012. Truth finding on the deep web: is the problem solved? PVLDB 6, 2 (2012),

97–108.

[10] Xian Li, Xin Luna Dong, Kenneth B Lyons, Weiyi Meng, and Divesh Srivastava.

2015. Scaling up copy detection. In Proc. of ICDE. 89–100