Fact Checking in CQA Forums

Tulika Agrawal
IIT Hyderabad
cs17mtech11022
cs17mtech11022@iith.ac.in

Eti Chaudhary
IIT Hyderabad
cs17mtech11029
cs17mtech11029@iith.ac.in

Suryamukhi K
IIT Hyderabad
cs17mtech01002
cs17mtech01002@iith.ac.in

ABSTRACT

Now-a-days, CQA forums have become very popular for sharing information and satisfying various information needs. With the growing popularity of these forums, wrong information is also getting wide-spread. Hence, there is a requirement of some means to effectively check whether the information provided is factually true or not. In our work, we began with using different classifiers in the existing model proposed in [7]. We further extend the basic classification task to rank the factually true answers according to their relevance using SVM-Rank. We also propose and test a new set of features which have not been used in the original work. Lastly, we also provide a rule-based method for summarizing answers of 'where' type questions.

Keywords

Fact Checking; Community Forums; Qatar-Living

1. INTRODUCTION

The first place people go to find answers these days are the Community Question Answering (cQA) forums. The questions may range from 'What is the best grad school for Design in Europe?' to 'How to learn to use command-line in Linux systems?'. Be it Yahoo! answers, StackExchange, StackOverflow or Quora to name a few, there's barely ever a dearth of answers.

The main issue that arises in such a situation is: how to filter out the correct answers? The term *'correct'* (or *'veracious'* as referred in [7]) is too subjective to be addressed as a whole. However, focusing on a smaller subset of these questions, i.e. the *'factual'* ones, this problem can be addressed in a much more elegant & manageable way.

There has been a decent amount of research towards fact checking. However, it's not just been restricted to cQA, but rather similar tasks, namely, Fake News Detection, Rumour Detection, Click Bait Detection to name a few [12]. As a starter in this area, [12] provides a good formalization of the underlying problem, the various related areas and an overview of how it has been addressed so far. A majority of the techniques proposed towards solving the problem assume a Subject-Predicate-Object (a.k.a. triples) representation of the query, thereby providing a structure to the problem. While some approaches such as [11] [10][8][9][1][2] take source credibility into account, others rely on support from external sources, mainly the web, to check for credibility of information [4][9][3]. Almost all of them, also use various text-analysis measures, such as objectivity of the claims, linguistic features including hedges, implicatives , assertives etc.[10][9][1][2].Some works like the ones analyzed in [5] categorize truth-discovery methods into 3 different categories into iterative, probabilistic and optimization based methods, where these methods rely on source weights or answer-votes (also in [6]) for truth computation tasks. However, from our literature review, the problem of checking of factual statements has not been applied to CQA forums before [7][6].

The main reference for the succeeding work was taken to be [7]. The authors deal with the problem of checking of factual-questions as a fourfold model. In order to check the facutality of an answer, the multi-facted model, takes into consideration, who is answering the question (user characteristics), how the question has been answered (answer's content), where it is said (Intra-forum evidence) and external support from the web (i.e. similarity between the current answer and the ones available on the web). Taking the above sets of features as their model, they classify all answers/comments to be either 'factually true' or 'factually false' using SVM as their classification algorithm.

The work done in [7] being the main point of reference, our contributions and extensions for the purpose of this project were as follows:

- Testing different classification algorithms on the model presented in [7]
- Extending from the area of classification, we tried to rank the factually true answers, using SVM-Rank
- We also proposed and tested 3 new sets of features that were not mentioned in the original paper. These features aimed to capture User Expertise, Answer Quality and Readability of answers.
- As an extension to simply classifying factually true answers, we also proposed a rule-based method to summarize answers to one-word answers of 'where' type questions.

2. WORK DONE

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Feature	Acc(%)	P(%)	R(%)	F1	MAP
Intra-Forum Qatar Living	65.46	66.4	66.4	66.4	83.97
Bing External source	63.45	59.58	89.84	71.65	67.71
Linguistic Features	60.64	60.41	67.96	63.97	78.8
Google Embedding	52.61	53.62	57.81	55.63	69.22

Table 1: Support Vector Machine

Methods	K-Fold (%)	Stratified (%)
Combined Best Feature groups	65.46	64.25
Combined Best Features feed forward	70.68	69.47

Table 2: Support Vector Machine Accuracy (Combined Features for Accuracy Optimization)

This section begins by a description of the dataset used for experimentation, followed by the contributions of this project in the subsections that follow.

2.1 DataSet

We have used the dataset and source code[1] used by our main paper[7]. The dataset consists of a number of Question-Answer threads, where each thread is composed of a question followed by the answers for that question. Each question consists of QuestionID, Timestamp, Question Category among the relevant tags. Similarly, each answer also consists of the UserID of the answerer, a Timestamp, CommentID, as well as the ground-truth label of that answer. There are a total of 249 answers out of which 128 are positive (Factually True) and the rest negative (Factual False, Factual-Partially True, Factual-Conditionally True, User-Unsure and Non-Factual).

2.2 Various Classifiers used & their Parameter Tuning

In the base paper[7], they have used Support Vector Machine (SVM) to classify the factually and non-factually true answers. There are two methods used to combine the features to calculate the overall optimized accuracy. First, they have calculated category-wise accuracy and then sorting the feature categories in descending order of accuracy. In the first method, they combined it according to the sorted sequence of features, for ex- In Top-N features, they will combine the top-n categories. In the second method, they have taken features one-by-one in the sorted order and combined it with other top-11 categories one-by-one. The accuracy in top feature categories and optimized accuracy from both methods in case of SVM can be seen in **Table 1** and **Table 2** respectively.We wanted to check whether the accuracy can be improved by using other classifiers. We tried with 4 classifiers, namely : Naive Bayes, Random Forest, XGBoost, Logistic Regression.

[1]https://github.com/qcri/QLFactChecking

Feature	Acc(%)	P(%)	R(%)	F1	MAP
Qatar Living Embedding	59.03	69.11	36.71	47.95	73.50
Bing External source	54.21	53.01	96.09	68.33	64.37
Sentiment	52.20	52.43	75.78	61.98	70.03
Intra-Forum Qatar Living	51.00	51.22	97.65	67.20	64.43

Table 3: Naive Bayes

Methods	K-Fold (%)	Stratified (%)
Combined Best Feature groups	67.87	67.06
Combined Best Features feed forward	67.06	67.46

Table 4: Random Forest Accuracy (Combined Features for Accuracy Optimization)

2.2.1 Naive Bayes classifier

The first classifier which we used is Naive Bayes Classifier. In our case, it was not performing well. We can see the accuracy of top feature categories from this method in **Table 3**.

2.2.2 Random Forest

The second classifier which we used is Random Forest. From this method, category-wise performance is improving in many cases. This method is performing better than SVM in one method. The parameters which we tune are 'n-estimators', 'max-features' and 'max-depth'. We can see it's accuracy in both methods in **Table 4**.

2.2.3 XGBoost Classifier

The third classifier which we used is XGBoost. This method is quite slow. This method is also not performing good on our dataset. The parameters which we tune here are 'colsample bytree', 'n estimators', 'max depth', 'gamma', 'learning rate'. We can see the accuracy of top feature categories from this method in **Table 5**. When we combined features to get the optimized accuracy, it was giving very bad results.

2.2.4 Logistic Regression

The fourth classifier which we used is Logistic Regression. This method is performing almost equivalent to SVM in first method of feature combination (not much decrease). The parameters which we tune are 'C', 'penalty' and 'solver'.

Feature	Acc(%)	P(%)	R(%)	F1	MAP
Intra-Forum Qatar Living	64.65	65.38	66.40	65.89	82.23
Bing External source	61.45	58.0	90.63	70.73	66.19
Linguistic Features	57.03	57.55	62.5	59.93	72.01
Extended Ling. Features	55.02	55.79	60.16	57.89	72.54

Table 5: XGBoost

Methods	K-Fold (%)
Combined Best Feature groups	65.06
Combined Best Features feed forward	63.85

Table 6: Logistic Regression Accuracy (Combined Features for Accuracy Optimization)

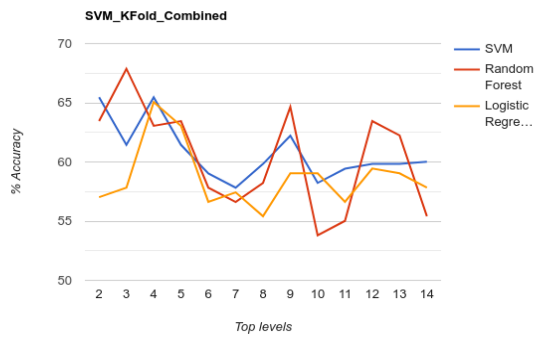

Figure 1: Accuracy Comparison of SVM, Random Forest and Logistic Regression in case of Combined Best Feature Group Method

We can see it's accuracy in **Table 6**.

So, we can see from **Figure 1** that in case of first method of feature combination, Random Forest is giving the highest accuracy from SVM and Logistic Regression. Random Forest is getting highest accuracy in case of Top-3 Feature group while SVM is giving highest accuracy in case of Top-4 Feature group.

2.3 Ranking of 'Factually True' answers using SVM Rank

Rank-SVM is one of the most popular machine learning methods to address the ranking problem. It is a pariwise ranking approach in which the 'learning' task is translated into a classification task (correct / incorrect ordering) for all possible pairs of documents in the input ordering, with the goal of minimizing the missclassified document pairs. With the help of margin maximization, Rank-SVM can have good generalization ability. Further, a number of kernel tricks have also been proposed on top of RankSVM to handle non-linear problems.

This phase of our work focussed on ranking of factually true answers by integrating SVM-Rank to work with the existing code. While the original work conducted their experiments in two phases,i.e. category-wise and combination of top-n categories, we applied Rank-SVM to evaluate the ordering of answers classified to be factually true only on the second phase of experiments. The core reason behind it being that the category-wise experiments were merely to provide an insight as to which categories hold more relevance for the dataset and further for choosing the top-n best performing categories for the second (and actual) phase of experiments.

2.3.1 *Input-Output Format*

An example of input required by SVM-Rank is as follows:

3 qid:1 1:1 2:1 3:0 4:0.2 5:0
2 qid:1 1:0 2:0 3:1 4:0.1 5:1
1 qid:1 1:0 2:1 3:0 4:0.4 5:0
1 qid:2 1:0 2:0 3:1 4:0.2 5:0
2 qid:2 1:1 2:0 3:1 4:0.4 5:0

Each row represents an answer/comment. In every row, the elements of the form <*integer*>:<*float*> represent the feature number and it's corresponding value. The 'qid' tag is used to group answers/comments belonging to a particular question together, such that, SVM-Rank only orders answers having a particular 'qid' together. Hence, the output for SVM-Rank provides different orderings for different sets of answers, where each set contains the answers that had the same 'qid'. SVM-Rank being a supervised learning algorithm, the leftmost values in each row are the ground-truth relevance labels.

2.3.2 *Workflow*

The workflow for the task of obtaining and assessing SVM-Rank orderings, takes place in the following steps:

1. Normal classification procedure is applied, where the feature set is taken to be the Top-N feature groups, with N ranging from 2 to 40.

2. During the classification phase, 30% of the data-set is set as test-data while the rest is used for training of SVM-Rank model.

3. From the classification task, the train-set, test-set and the predicted labels for different answers are extracted.

4. Using the extracted data, syntactically correct input files are created for SVM-Rank. For cross-checking and analysis purposes, we save the input files (for both test and train sets).

5. SVM-Rank is then invoked like a command-line query from within the code (for both train and test data) and the stdout stream is redirected into log files.

6. Parsing the generated logfiles, we cummulate the relevant statistics and present a final analysis in a csv formatted file. For an intuitive analysis, we also generate an output file containing the question-answer threads with the answers ranked according to Rank-SVM ordering.

2.3.3 *Analysis*

As our dataset lacks the relevance labels needed by SVM-Rank, our initial implementation made a naive assumption where the relevance of an answer was decided based upon it's timestamp, with the most recent answer being the most relevant. The results under this assumption were (expectedly) very good, giving a 100% accuracy when Top-N feature groups were considered, for N>4. Such a performance could be easily attributed to the fact that timestamps where also used in the set of features being considered in the model. Our next efforts were then guided towards finding a more meaningful way of assigning ground-truth relevance labels. For this purpose, we used BM25 scores to calculate relevance labels for the answers. With the relevance labels set

Table 7: SVM-Rank Results

Top-N Feature Categories	%of Swapped Pairs
Top 2	25.64
Top 3	19.44
Top 4	16.67

using the above described method, the results obtained are shown in **Table 7**. As one may observe, the accuracy values obtained were pretty high.

On inspecting and analysing the data more, we found a good number of cases for which the BM-25 values were zero. This was not only restricted to some answers in a question-answer thread, but was also seen for a few whole threads i.e. zero BM-25 scores for all the answers in that thread. Due to a small sized dataset, with the added quality-of-content issues, such as, poor quality questions and lack of prominent keywords, both the train and test sets get reduced to smaller size by a good amount, hence, leading to inadequate SVM-Rank results. For the above discussed reasons, we mention finding better relevance judgments measures or labelling them manually, as a plausible future work.

2.4 Feature Set

We use the following 4 sets of features as input representations for developing our model:

- the user profile features
- the language used in the answer
- the context in which the answer is located
- external sources of information

2.4.1 User profile

We extract a set of features indicative of users' profile and expertise on various topics. Some of these features were previously used by [7]. To this set we added few more features. User answering more but in selective categories indicates his expertise in that category. We design an equation as follows to take this into consideration:

$$User\,expertise = \frac{Total\,no.\,of\,answers\,answered}{no.\,of\,distinct\,categories} \quad (1)$$

The above equation gives more weightage for the users who answered more in selective categories.We also include a well known Z-score method which gives an intuitions as the more a person asks the lesser he knows. This is modelled as follows:

$$Z - score = \frac{A - Q}{\sqrt{A + Q}} \quad (2)$$

where, A and Q represent no. of answers and questions respectively. The above equation gives more weightage to the users who answered more questions rather than asked. We also include a feature which tells the more difficult question a user answered the more is his expertise. That is, he is considered to be an expert if he answered a question with few answers. This is modelled as follows:

$$user\,level = \frac{\sum \frac{1}{no.\,of\,answers\,for\,a\,question}}{N} \quad (3)$$

where, N is the total number of answers answered. We also use lexical density of users as feature to indicate the fraction of content bearing words they use in their answers.

2.4.2 Answer content

These features model what the answer says, and how. Some of these features were already used by [7]. To this feature set we add a set of readability features. These include no. of syllables per word, no. of syllables per sentence, no. of polysyllables per word, no. of polysyllables per sentence. These are obtained from Flesch Kincaid grade level formula and SMOG index. We also include automated readability index as a score to indicate how readable an answer is. We also add a set of features which determine the answer quality. These features are given below:

- Vocabulary features: These include no. of words, no. of distinct words in an answer and their ratio.
- Typos: These are the fraction of non dictionary words present in an answer.
- POS tag distribution: This feature gives the probability distribution of the POS tags in an answer.
- Lexical density: This feature gives the fraction of content bearing words present in an answer. This includes no. of nouns, pronouns, proper nouns, adjectives and adverbs.
- Grammatical Errors: We checked correct grammatical usage of the most commonly used words like *your* and *its*. Your can be mistakenly written as you're and its as it's. In order to identify such cases, we extracted the POS of the word used immediately after *your* and *its*. If used correctly, the POS of the succeeding word should not be a verb. For example, verb: *your doing great!* is incorrect, adverb: *its freaking amazing* is incorrect or a determiner: *a* or *the*.
- BLUE n-gram overlap: This approach works by counting matching n-grams in the answer to n-grams in the question, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. The comparison is made regardless of word order.

For determining the answer quality, it is equally important to consider the question quality. The better the quality of the question, the better will be the answer. Let $F1$ be the length of the answer or question normalized by the maximum length, $F2$ be the fraction of typos, and $F3$ be the lexical density of the question or answer. We take weighted average of $F1$, $F2$ and $F3$ as follows:

$$\alpha F1 + \beta F2 + \gamma F3.$$

We take α as 0.2, β and γ as 0.4.Finally take the $F1 - Score$ of the question quality and answer quality.

2.4.3 Intra forum Evidence

These set of features model according to where the answer is being said. These features include calculating similarity of question-answer pairs within same thread or with the answers of the entire forum.These were already used by [7]

2.4.4 External support

These set of features are used to verify whether an answer's claim is true by searching for support in the web. These set of features were already used by [7]

Table 8: Category-Wise rankings and accuracy obtained for proposed feature sets

Category	SVM		Random Forest	
	Rank	Accuracy (%)	Rank	Accuracy(%)
Readability	21	0.4136546185	8	0.5140562248
User Expertise	22	0.4136546185	20	0.4819277108
Answer Quality	15	0.5381526104	4	0.546184738

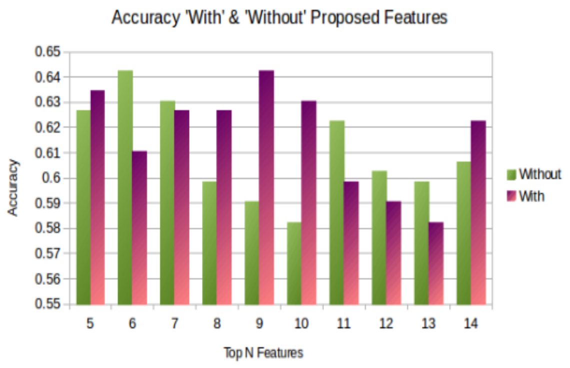

Figure 2: Accuracy of SVM classifier with and without the proposed sets of features

2.4.5 *Experiments*

Table 8 shows category-wise rankings and accuracy obtained for proposed feature sets, using SVM and Random Forest Classifiers.

Figures 8 and 9 show the accuracy of SVM model and Random forest model respectively with and without our proposed set of features. As seen, our proposed features perform quite well.

2.5 Answer Summarization

We also tackled the problem of answer-summarization for 'where' type questions. For this task, we begin by dividing an answer into a number of parts using delimiters such as ',', ';' etc. For each of these parts, we find the sentiment of the sentence making use of Vader, a sentiment analysis. Only the sentences/parts, that result in a 'positive' or 'neutral' sentiment are considered for summarization. For the actual summarization part, we again check the length of the sentence. If the length is <= 5, we simple try to find proper nouns or a word sequence with the first letter of words capitalized (bi-grams till the current implementation). Further for longer sentences, we search for prepositions and within some proximity of a preposition we try to find either proper nouns or a word sequence with the first letter capitalized. As mentioned in the presentation, this work is still at a native stage and was proposed as a furture direction for refinement and handling of other categories of questions such as what-type, when-type etc.

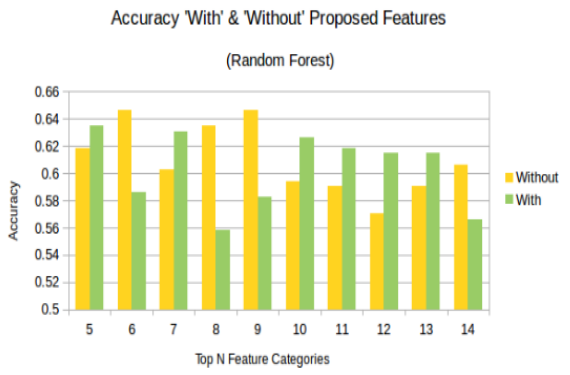

Figure 3: Accuracy of Random forest classifier with and without the proposed sets of features

3. CONCLUSIONS

In this study, we have checked the veracity of forum answers rather than just the credibility. We proposed set of features like readability, user expertise and quality of answer content in our two models namely, SVM and Random forest. The evaluation results have shown very strong performance. On top of Fact-Checking, we ranked the answers according to their relevance. This project can be extended by summarizing the factually true answers to get the pin-pointed answers and then rank those answers.

4. REFERENCES

[1] M. J. Blooma, A. Y.-K. Chua, and D. H.-L. Goh. Selection of the best answer in cqa services. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 534–539. IEEE, 2010.

[2] F. Eskandari, H. Shayestehmanesh, and S. Hashemi. Predicting best answer using sentiment analysis in community question answering systems. In *Signal Processing and Intelligent Systems Conference (SPIS), 2015*, pages 53–57. IEEE, 2015.

[3] G. Karadzhov, P. Nakov, L. Màrquez, A. Barron-Cedeno, and I. Koychev. Fully automated fact checking using external sources. *arXiv preprint arXiv:1710.00341*, 2017.

[4] X. Li, W. Meng, and C. Yu. T-verifier: Verifying truthfulness of fact statements. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 63–74. IEEE, 2011.

[5] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *ACM Sigkdd Explorations Newsletter*, 17(2):1–16, 2016.

[6] X. Lian, X. Yuan, and H. Zhang. Subordinating to the majority: Factoid question answering over cqa sites. *Journal of Computational Information Systems*, 9:6409–6416, 2014.

[7] T. Mihaylova, P. Nakov, L. Marquez, A. Barron-Cedeno, M. Mohtarami, G. Karadzhov, and J. Glass. Fact checking in community forums. *arXiv preprint arXiv:1803.03178*, March 2018.

[8] N. Nakashole and T. M. Mitchell. Language-aware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1009–1019, 2014.

[9] P. Nakov, T. Mihaylova, L. Màrquez, Y. Shiroya, and I. Koychev. Do not trust the trolls: Predicting credibility in community question answering forums. In *Proceedings of the International Conference Recent*

Advances in Natural Language Processing, RANLP 2017, pages 551–560, 2017.

[10] K. Popat, S. Mukherjee, J. Strötgen, and G. Weikum. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2173–2178. ACM, 2016.

[11] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics, 2011.

[12] J. Thorne and A. Vlachos. Automated fact checking: Task formulations, methods and future directions. *arXiv preprint arXiv:1806.07687*, 2018.