PowerCoder: Question Tagging

Abstract:

Question classification or question tagging is the process of identifying to which category a question belongs to. The research work this area has so far focussed on more general categories but not on specific sub-categories. In the paper, we propose a Random Forest algorithm based classifier that has been used to classify coding questions or questions that have to be answered using a high level program based on the data structures that can be used to solve the question. The purpose of identifying the appropriate data structure is that, it can be used to generate algorithms and high level programs that can be used to solve the question. The questions in the data belong to 'array', 'graph' and 'string'. Only three classes have been used because of the lack of a balanced dataset. Most of the standard machine learning classification algorithms have been used out of which Random Forest and Gradient Boosting algorithms performed the best. Since, Gradient Boosting took a longer time to train, Random Forest has been used as the final model in this paper. It achieved an accuracy of 78.77% in the test set without much hyper-parameter tuning. Future works will focus on including more classes and the development of deep learning based architectures for the problem at hand.

1. Introduction:

Tagging a text instance is the process of assigning a particular tag to the instance based on the content of the text and based on the purpose for which the process is carried out. Tagging has been used in various fields like law, research journals, etc. But in these use cases tagging has always covered a broad range of topics, not for the topics within a bigger topic. Questions have also been tagged on a broad range of topics. This research work focuses on tagging coding questions based on the data structures that can be used to solve the questions using a machine learning based approach.

The question tagging dataset was collected with the view of understanding how a machine learning or a deep learning algorithm would understand the coding questions and, in the process, use the algorithms to classify the questions based on the data structures that can be used to solve such questions. The weights of these learned models can then be used for a variety of useful tasks such as question generation, algorithm generation and even the generation of the complete high level program for a question in a suitable programming language.

2. Related Work:

The research carried out in the question classification has more or less been based on general or broad domains in which questions belonging to one domain are clearly separated from questions belonging to another domain.

Prudhvi Raj Dachapally et al. (2017)[1] proposed a two-tier CNN-based deep learning architecture for classifying questions. The two-tier architecture was used to classify the broader domains first using which the subsequent sub-domain were classified. The data used for the training the model was the question classification dataset by University of Illinois, Urbana Campaign(UIUC) and the test datasets used were from the TREC dataset by UIUC and the Quora website. The proposed model used the word2vec vectoriser for generating the vector representation of each word and they were able to achieve an accuracy of 93.4%.

B. A. Upadhya et al. (2019)[2] used two different model architectures - LSTM and CNN - to classify yes/no questions and open-ended questions. They chose such categories to develop an automated question answering system for e-commerce websites like Amazon. The dataset they used to develop the models was the Amazon QA dataset [3] and they concluded that the LSTM based model significantly outperformed the CNN model in all the categories and was able to achieve an accuracy of 91%.

Nirob et al. (2017)[4] used Support Vector Machine(SVM) - a classical machine learning model - to classify questions in Bangla language. They used various kernels like Linear, Sigmoid, Polynomial and RBF kernels out of which the the linear kernel outperformed other kernels and it also outperformed its predecessors by achieving an accuracy of 88.62%. Another similar research work is that of Somnath Banerjee et al. (2019)[5] where they combined the classifiers using techniques like stacking, ensemble bagging, ensemble boosting and voting to classify questions in Bengali language and they were able to achieve an accuracy of 87.07% using the stacking of classifiers.

The research works mentioned and other related works have all focussed on classifying questions from the cover a wide range of topics and to the best of our knowledge research on classifying coding questions has been pretty scarce. Hence in this research work, we propose a Random Forest based machine learning model for classifying coding questions of limited classes, which will set up the foundation for future research and the development of advanced models for the same.

3. Dataset Description:

The question tagging dataset consists of coding questions from two websites namely Leetcode and Codeforces. The questions have been scraped off the websites using tools such as BeautifulSoup and Selenium using the standard HTML formatting of the question's text. Only the question text has been used, not the "sample input" and "sample output", that accompany those questions, since getting them into a proper format would have been too difficult and time-consuming. Along with question's text, the tags of the respective text – data structures, algorithms, and any other relevant tag – have also been collected which form the target classes.

At present, the question tagging dataset consists of 10105 rows and each row consists of a question text and 7 tags. The number of tags for each row varies but they have been either truncated to 7 tags if number of tags is more than 7 or padded with "No Tag" if the number of tags is less than 7. The tag columns have been named as "tag1", "tag2", …, "tag7" and each tag column does not have any definition associated with, such as "tag1" consists of "data structure tags". The dataset is a matrix of dimension 10105 x 8.

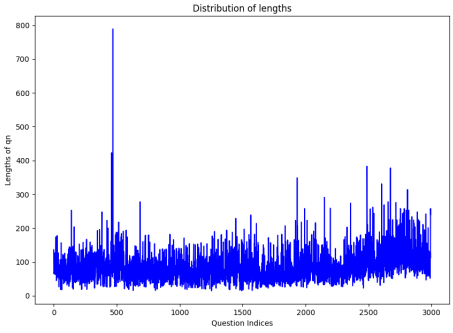

figure i

4. The Process:

The Question Tagging dataset consisting of coding questions, to the best of our knowledge, is first of its kind and it comes with a its own set of problems. This section explains the various steps undertaken to transform the dataset into a suitable format, that can be fed into the algorithm. The workflow of the work is presented in *figure ii.*

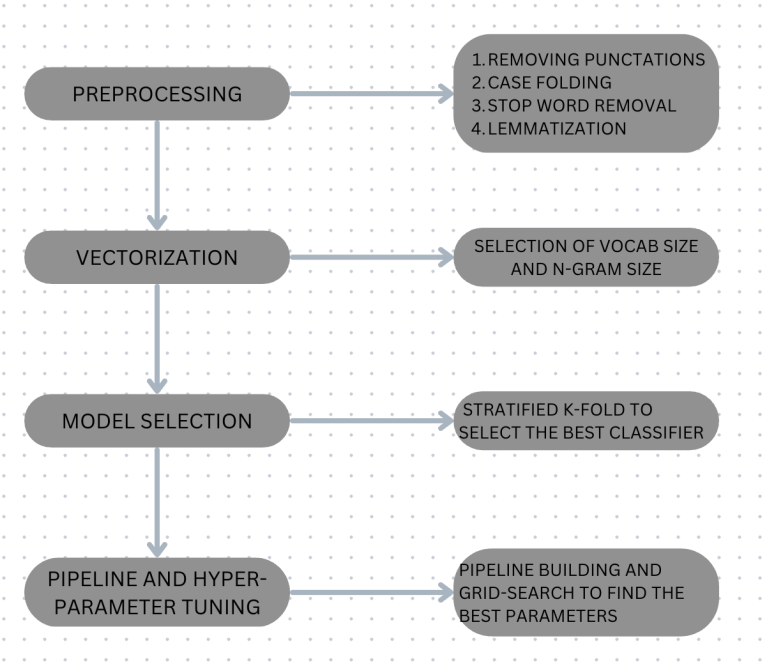

figure ii

4.1 Data Cleaning:

Questions from these websites usually have some names, locations, objects, etc which are not really needed for identifying the tags of the questions. In addition, it might also contain some mathematical expressions in a not so suitable format. The steps followed to remove those noises are discussed in the following sections.

4.1.1 Removal of punctuations:

Punctuations except '[]' have been removed since the objective of the process is to tag these questions based on the words present. '[]' have not been removed as they are the ones used to represent indexing in most of the questions.

4.1.2 Removal of stop words:

Stop words are words that occur most commonly in a particular language and their feature representation often have little to no value during the modelling process. Though the stop words are required for the understandability of a question or a text for humans, they just make the vector representation of a question less meaningful when they are used in a modelling setup. They have been removed so that the machine learning algorithm can just focus on words that fully represent a particular question.

4.1.3 Lowercasing the questions:

Lowercasing as the name suggests, is the process of converting any word in uppercase or even in title case to lowercased words. The questions have been lowercased to reduce the size of the questions dataset vocabulary, as same words appearing in uppercase in one question and lowercase in another question will be treated as two different words.

4.1.4 Lemmatization:

Lemmatization is the process of converting words to their base forms - the ones that appear in the dictionary. Lemmatization has been done on the questions so that, multiple forms of the same word have a single representation in the questions dataset vocabulary.

4.1.5 Separation of Tags:

Each of the tag columns in the questions dataset does not have a proper order as the same tag is present in two different tag columns for different instances. To bring an order to the dataset the following changes are made,

- column 'tag1' contains the algorithms that can be used to solve the question.
- column 'tag2' contains the data structures that can be used to solve the question.

It is also necessary to ensure that when the target classes are selected there is no overlap with the other tags. For example, if "dynamic programming" and "greedy" are being chosen as the target

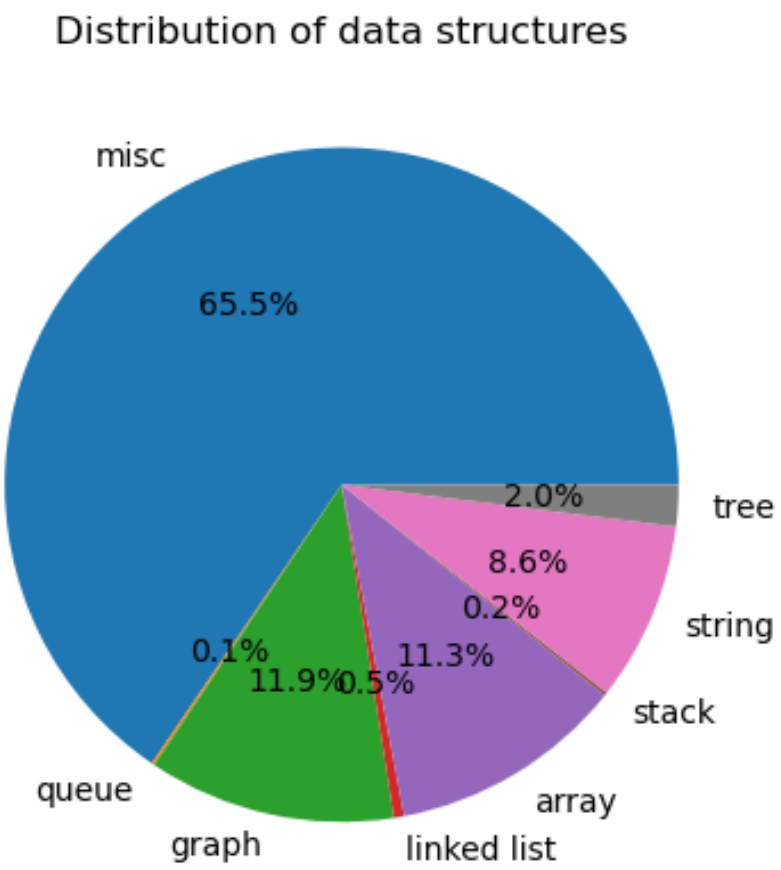

figure iii

classes, the instances the algorithm will be trained on, should have both "dynamic programming" and "greedy" as tags in the columns "tag1" to "tag7".

Removing of this overlap is an important step for the proper training of the model. The instances with the overlap of the tags will result in the model not properly understanding the difference between the tags as those instances belong to both the classes while the training process is done with each instance having a single tag.

For this work, only the questions belonging to the data structures - array, graph and string have been used to train the model because from *figure iii,* a lot of questions have been tagged as 'misc' as we could not find a proper tag for majority of the questions while scraping them. From *table i*, the distribution of array, string and graph tags are almost equal and hence they have been used for training the model.

TAG	COUNT
Array	944
Graph	1225
String	843

table i

4.2 Building the pipeline:

After the questions have gone through the data cleaning stage, each question must be given a numerical representation as machine learning algorithms work with numbers not words. Once the questions dataset is converted into a numeric format, the machine learning algorithms that will be used for the task at hand has to be decided, for which validation methods are used.

4.2.1 Numerical Representation of the questions:

Machine learning algorithms do not understand text data, they understand only numbers. Each question in the questions dataset should be given a numerical representation that captures the meaning of the question or gives more weights to the words that possesses the central meaning of the question.

There are many algorithms for generating the numerical representation of a text data. One such algorithm is the TF-IDF vectorisation, which has been used to convert the text data into feature vectors. The algorithm gives each word in the question a weight based on the count of the word in the question and based on the number of questions that contains the word in the whole dataset.

$$tf - idf(t) = tf(t,d)\ x\ idf(t) - (1)$$

$$idf(t) = \log\frac{1+n}{1+df(d,t)} - (2)$$

The algorithm weights a particular word using the product of the count of the word in a particular document and the log of inverse fraction of the number of documents in which the word appears to the total number of documents. The term 'document' refers to a particular row in a dataset. The algorithm uses equation (1) to weight each term or word in a document. It is the product of $\mathrm{tf}(\mathrm{t}, \mathrm{d})$, known as term-frequency, which is the count of the word in the document and $\mathrm{idf}(\mathrm{t})$, known as

inverse document frequency, which measures the commonality of a particular word in a corpus. If a word occurs in most of the documents of a particular corpus, then the word is common to the vocabulary of the corpus and is likely not to add value as a feature input while building the model. Hence a word which occurs in most of the documents will be weighted less by this algorithm.

4.2.2 Stratified K-Fold to select the best model:

There are a lot of machine learning models that one can train on the questions dataset but to choose the best performing models, a validation method must be used. A validation method checks if the model overfits(memorises the training set and fails to learn the patterns thereby not performing well on the test set) and how it performs on various subsets of the data. Accuracy is chosen as the evaluation metric since there is almost an even distribution of the tags or classes.

The validation method used for selecting the best performing models is the stratified k-fold method. This method operates by splitting the dataset into n number of splits such that the distribution of classes in each split is like that of the original distribution. Out of the n splits, n-1 splits are used for training the model and the left-out split is used for validating the model. Likewise, each of the n splits are used as for validating the model resulting in n-iterations.

4.2.3 Hyperparameter Tuning:

The vectoriser and the algorithms have been trained separately on the questions dataset without much hyper-parameter tuning. They had been trained separately to find the optimal range of hyper-parameters for the vectoriser and to find the best performing base models. But to get the best performing architecture, the steps involved in the training process must be combined and the combination's hyper-parameters have to be tuned.

The optimal hyper-parameters for the pipelined model, consisting of the TF-IDF vectoriser and the RandomForest model is obtained through the grid search mechanism. The grid search mechanism finds the accuracy score for all the combinations of parameters and returns the combination of parameters with the highest accuracy score. The hyper-parameters used for the vectoriser were n-gram range which is the number of consecutive words that will be used for generating the vocabulary and max features which is the number of the most frequent words that the vectoriser will have in its vocabulary. The range of values for the n-gram range was - (1,1) and (1,2) and for the max-features was - [200, 300, 400, 500, 600, 700, 800]. The hyper-parameters tuned for the Random Forest model were maximum depth which is the depth upto which a tree is allowed to grow and the the number of estimators which is nothing but the number of trees in the model. The range of values for maximum depth was - [-1, 8, 9, 10, 11, 12] and for number of estimators was - [100, 200, 300, 400, 500, 600]. -1 for maximum depth indicates that the trees can grow without any constraints.

5. Results:

This section presents the results of the various experiments performed to arrive at the final model. The results are presented in the order of the pipeline described in section 4.

Accuracy scores of the TF-IDF vectoriser for different vocabulary sizes is presented in *figure iv.* Based on the vocabulary size the vectoriser only stores the words with the most frequency and for words outside its vocabulary it assigns an '<UNK>' value. Various vocabulary sizes ranging from 100 to 3000 were tried and the vocabulary size of 500 performs the best.

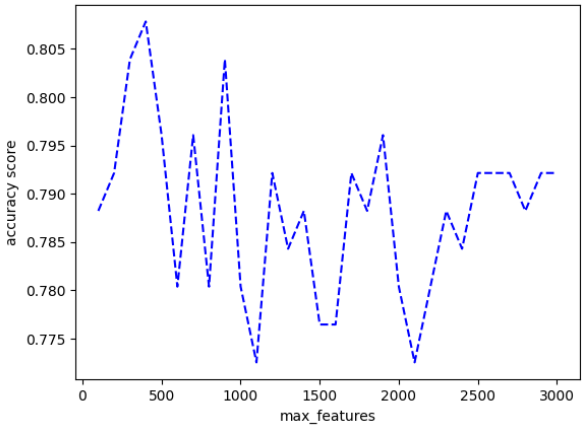

figure iv

The results of the stratified cross validation for 10 folds is presented in *table ii.* The questions dataset was first put through a vectoriser of vocabulary size 500 before using the data for training and validating the models. The accuracy attained by the Random Forest algorithm is clearly better than the other machine learning algorithms used in the experiment.

Models	Mean accuracy	Best accuracy
Logistic Regression	72,0	76,07
Random Forest	77,55	80,0
XGBoost	75,11	79,52
AdaBoost	72,36	77,16
Gradient Boosting	76,84	79,21
Light GBM	75,47	78,34

table ii

Results of the grid-search experiment for the hyper parameters of the random forest algorithm are present in *table iv* and that of the vectoriser is presented in *table iii,* and a vocabulary size of 200 and an n-gram range of (1,1) for the vectoriser and a 600 tree Random Forest model with a maximum depth of 12 gives the highest accuracy score of 78.77%.

	(1,1)	(1,2)
200	78,47	78,39
300	78,39	78,31
400	78,31	78,3
500	78,32	78,14
600	78,33	78,03
700	78,2	78,02
800	78,33	78,0

table iii

	100	200	300	400	500	600
-1	78,21	78,29	78,36	78,39	78,41	78,4
8	77,89	77,9	77,87	77,9	77,91	77,89
9	78,1	78,14	78,15	78,18	78,19	78,18
10	78,27	78,28	78,3	78,3	78,3	78,29
11	78,33	78,37	78,39	78,42	78,4	78,4
12	78,44	78,48	78,48	78,5	78,47	78,48

table iv

6. Future Work:

The work presented in this paper sets up the foundation for understanding how machine learning models learn about coding questions and how the internally represent them. Without carrying out much hyper-parameter tuning an accuracy of 78.77% has been achieved and hence, performing extensive hyper-parameter tuning is only going to improve the model's performance. In the future work, the number of data structures will be increased and algorithms that are used for solving the questions will also be included thereby transforming the problem into a multi-label classification problem. Also in the subsequent work the machine learning models will be replaced by RNN based deep learning models. The questions in the dataset have to be cleaned and a proper structure definition has to be given for the questions, as the more clean the data is, better will be the understanding of the questions by the models and better will be the performance.

7. Conclusion:

The work presented in this paper opens up a range of research ideas to be explored which includes a deep learning model to classify questions with a more wide range of data structures and algorithms, question generation, algorithm generation for the questions and finally generation of high level language programs.

References:

[1] Prudhvi Raj Dachapally and Srikanth Ramanam, "In-depth Question classification using Convolutional Neural Networks". Available: https://doi.org/10.48550/arXiv.1804.00968

[2] B. A. Upadhya, S. Udupa and S. S. Kamath, "Deep Neural Network Models for Question Classification in Community Question-Answering Forums," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944861.

[3] Amazon QA dataset. Available: https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_qa

[4] Nirob, Syed & Nayeem, Md & Islam, Md Saiful. (2017). Question classification using support vector machine with hybrid feature extraction method. 1-6. 10.1109/ICCITECHN.2017.8281790.

[5] Somnath Banerjee, Sudip Kumar Naskar, Paolo Rosso and Shivaji Bandyopadhyay, "Classifier Combination Approach For Question Classification For Bengali Question Answering System". Available: https://doi.org/10.48550/arXiv.2008.13597

[6] A. Sangodiah, R. Ahmad, and W. F. W. Ahmad, "A review in feature extraction approach in question classification using support vector ma- chine," in 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014), Nov 2014, pp. 536–541.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Nov. 2011. Available: http://dl.acm.org/citation.cfm?id=1953048.2078186

[8] K. Hacioglu and W. Ward, "Question classification with support vector machines and error correcting codes," in Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2, NAACL-Short '03, (Stroudsburg, PA, USA), pp. 28–30, Association for Computational Linguistics, 2003.

[9] T. Dodiya and S. Jain, "Question classification for medical domain question answering system," in 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON- ECE), pp. 204–207, Dec 2016.

[10] D. Zhang and W. S. Lee, "Question classification using support vector machines," in Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Infor- mation Retrieval, SIGIR '03, (New York, NY, USA), pp. 26–32, ACM, 2003.

[11] Donald Metzler and W Bruce Croft. 2005. Analysis of statistical question classification for fact-based questions. Information Retrieval 8, 3 (2005), 481–504.

[12] Das, Mamata et al. "A Comparative Study on TF-IDF Feature Weighting Method and Its Analysis Using Unstructured Dataset." *International Conference on Computational Linguistics and Intelligent Systems* (2021).

[13] Code and dataset of this work: https://github.com/SathyaKrishnan1211/question_tagging