In general, the procedures for me to find out good features could be divided into three steps as the following:

1. To begin with, I firstly rationally think of some factors that should influence the correctness of the guesses. For example, I believe the length of run should be a different feature from the length of guess because with shorter part of the question it will be harder to guess, so I also include ‘LengthRun’;

I believe if the current shown contents of a question (run) contains too many noninformative words such as ‘a’, ‘the’ or ‘of’, then it is hard to guess correctly, so I let the number of these useless words to be a feature called ‘UselessInfo’; Similarly, I count the number of words indicating genders like ‘he, she, it, they, his, her, its’ as a feature ‘GenderInfo’ because for question on people’s name these words would help a lot; The year in which the question was asked can reflect how difficult the question was and influence the correctness, and similarly the ‘Difficulty’ and ‘Tournament’ attributes of the question reflects similar information, so I add all of them as features; The ‘Category’ of a question might matter because sometimes it is just harder to guess some genre of questions than the other.

1. After adding these ‘reasonable’ features, I begin to test individually that if having one single feature can I beat the performance of that with no feature.

The baseline is no feature model trained with 500 training samples and its performance on 50 test samples. The result is 167 right out of 407 with Accuracy: 0.73 Buzz ratio: 0.32 Buzz position: -0.073743 Best rate: 0.41.I focused on the right number and best rate.

‘UselessInfo’ gives me best rate of 0.42 and right number as 169 which improves a little bit. The more important point is that I find this feature does vary among different questions as shown in the screenshot below.

A screenshot of a computer

Description automatically generated

The first guess has 10 useless words and the second has 4.

On the opposite, among the 50 test samples the features ‘Year’ are almost the same as shown below.

A screenshot of a computer screen

Description automatically generated

The Best rate and right number are the same as the baseline. The comparison proves my hypothesis that if the features are almost the same for most of the samples, then it cannot really help the classifier.

With such rule, I delete features that don’t vary a lot among different samples such as ‘Year’, ‘Prompt’, ‘GamePlay’ and ‘Tournament’. I also delete ‘Frequency’ because every time it brings all indices down as shown below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

I tried out the combination of ‘Frequency’ and other features but still cannot save these indices, so I finally delete this.

I remain ‘Difficulty’, ‘Length’, ‘LengthRun’, ‘UselessInfo’ and ‘GenderInfo’ because these features are different among different guesses and bring me good indices locally. By the way I thought encoding ‘Difficulty’ into integer might help the classifier better, but by test I find the performance is better if remaining the feature to be in string type.

1. Since I already filtered the pool of features, I can try some combinations of picked features on Gradescope and see their performances. Of course, to include all the features will cause overfit so I only use some subsets. Sometimes a combination will have very high accuracy like 0.774, but medium Best Score 0.414. As the instruction said the Best Score is more important than Acc, then I finally choose a combination with balanced score, which is ['UsefulInfo','Length','GenderInfo','LengthRun']



I beat the baseline a lot and am at a good ranking until the time point when I wrote this analysis. In all I used some reasonable features based on human judgment and it works somehow.