**1) Approach to the problem**

**- high level description of solution**

**- pre-processing**

**- feature extraction**

**- models**

**- alternative approaches**

**2) Evaluation of results**

**- split 80/20**

**- logloss**

**- succeeded/failed tests**

**- other metrics**

Pre-processing

- the text should be cleaned beforehand

- be careful with overcleaning

*The main thing to remember when you are cleaning data is to remove as much 'noise' from it as possible. In this case, noise is anything that doesn't help to prove if pairs of questions are similar or not: punctuation, over common words (the, a, an, of...), limiting the variation of words (What, what, He, he).*

- normalize as much as possible!

- be aware of the risks/drawbacks…

- find a trade-off

About log\_loss

Log-loss is an appropriate performance measure when you're model output is the probability of a binary outcome.

The log-loss measure considers confidence of the prediction when assessing how to penalize incorrect classification. For instance consider two predictions of an outcome P(Y=1|X), where the predictions are 0.51 and 0.99 respectively. In the former case the model is only slightly confident of the class prediction (assuming a 0.5 cutoff), while in the latter it is extremely confident. Since in our case both are wrong, the penalty will be more harsh for the more confident (but incorrect) prediction by employing a log-loss penalty.

Log Loss heavily penalises classifiers that are confident about an incorrect classification. For example, if for a particular observation, the classifier assigns a very small probability to the correct class then the corresponding contribution to the Log Loss will be very large indeed. Naturally this is going to have a significant impact on the overall Log Loss for the classifier. The bottom line is that it’s better to be somewhat wrong than emphatically wrong. Of course it’s always better to be completely right, but that is seldom achievable in practice!

*SK-Learn's unified scoring API always maximizes the score, so scores which need to be minimized are negated in order for the unified scoring API to work correctly. The score that is returned is therefore negated when it is a score that should be minimized and left positive if it is a score that should be maximized.*