Task 4: Statistical Testing

Step 1: Magnitude classifier

The lexicon classifier you created in Task 1 does not use all the information available in the lexicon. Modify the lexicon reader and simple classifier to account for the strength/magnitude of the sentiment expressed by each word. The tester assumes the use of weight 4 for strong sentiment and weight 1 for weak sentiment.

Step 2: Sign test

To investigate whether the two systems are significantly different, we will use the **sign test**. In this case, the sign test is based on the binomial distribution.

Count all cases when system 1 is better than system 2, when system 2 is better than system 1, and when they are the same. Call these numbers *plus*, *minus*, and *null*.

We will use the sign test to return the probability of seeing a result at least as extreme as our observed counts under the assumption that the **null hypothesis** is true. This probability is called the **p-value** and it can be calculated for the two-sided sign test using the following formula:

$$2\sum_{i=0}^{k}\binom{n}{i}q^i(1-q)^{n-i}$$

where $n = 2\lceil\frac{null}{2}\rceil + plus + minus$ is the total number of cases and $k = \lceil\frac{null}{2}\rceil + \min\{plus, minus\}$ is the number of cases with the less common sign. In this experiment, $q = 0.5$.

Here, we divide the null cases evenly between the two signs, rounding up if necessary. Why is this better than rounding down? (*Hint*: we are looking for statistical significance) We then use the formula as above. The formula is multiplied by two because this is a two-sided sign test and tests for the significance of differences in either direction, so a difference in one of the directions is half as significant as it would be otherwise.

The numbers you calculate are high enough to cause overflow errors in languages like Java or C. In Python, integer values are not restricted by the number of bits, and instead expand dynamically to the limit of the available memory. However, this does not apply to floats, so you can still run into overflow errors. To circumvent this, make sure you use integer division when appropriate.

Step 3: Test your classifiers

Is the difference in the scores obtained by your simple classifier and the magnitude classifier statistically significant? What about the magnitude and Naive Bayes classifiers? Make sure you compare the results on the same examples and that you don't test on the training set. As usual, please see `tick4.py` as a guide for what is expected.

How does the p-value vary if you reduce the number of samples?

Before being ticked, experiment with different weightings for the magnitude classifier. Does the weighting affect performance? Make notes to show to your ticker.

Starred tick (optional)

If you vary the smoothing parameter from Task 2 so that you add a constant smaller than 1 (e.g. add 0.5 rather than add 1), do you obtain better performance? You can carry out this experiment with various values for the smoothing parameter to try and optimize it. Why would it be invalid (strictly speaking) to check for significant improvements by using the sign test on your best result?