1. To calculate the label for the sentence "I always like foreign films" we use the following equation:

$$P(Label \mid Sentence) = P(Label) \prod_{word \in Sentence} P(word|Label)$$

This equation gives proportional results as we forgo dividing by P(sentence) to save on computation as it scales everything to the same degree. We can use the equation to directly compute the probability for each class, but it is computationally more efficient to convert this to the log space for the same equation as so:

$$\lg\big(P(Label \mid Sentence)\big) = \lg\big(P(Label)\big) + \sum_{word \in Sentence} \lg\,(P(word|Label))$$

Where lg is log base 2, then after computing this for each Label we take:

$$arg \max_{Label \in LABELS}[\lg\big(P(Label|sentence)\big)]$$

With our table of values, and prior values this yields the following:

S = "I always like foreign films"

$P(pos \mid S) =$

$P(pos)P(I \mid pos)P(always \mid pos)P(like \mid pos)P(foreign \mid pos)P(films \mid pos)$

$= (0.4)(0.09)(0.07)(0.29)(0.04)(0.08)$

Then,

$\lg\,(P(pos \mid S)) =$

$\lg\,(0.4) + \lg\,(0.09) + \lg\,(0.07) + \lg\,(0.29) + \lg\,(0.04) + \lg\,(0.08)$

$\approx\ -18.71$

$P(neg \mid S) =$

$P(neg)P(I \mid neg)P(always \mid neg)P(like \mid neg)P(foreign \mid neg)P(films \mid neg)$

$= (0.6)(0.16)(0.06)(0.06)(0.15)(0.11)$

Then,

$\lg\,(P(neg| S)) =$

$\lg\,(0.6) + \lg\,(0.16) + \lg\,(0.06) + \lg\,(0.06) + \lg\,(0.15) + \lg\,(0.11)$

$\approx\ -17.42$

So, we would predict that this review is negative.

2.

a. Please see my code for implementation

b. movie-review-small.NB:

Here + stands for Comedy, and – stands for Action

Label	Token	Log(P(Token | Label))
+		-1.3219280948873622
-		-0.7369655941662062
+	fun	-2.0
+	couple	-2.415037499278844
+	love	-2.415037499278844
+	fly	-3.0
+	fast	-3.0
+	furious	-4.0
+	shoot	-4.0
-	fast	-2.584962500721156
-	furious	-2.584962500721156
-	shoot	-1.84799690655495
-	fun	-3.1699250014423126
-	fly	-3.1699250014423126
-	love	-3.1699250014423126
-	couple	-4.169925001442312

c. The above weights, when used for calculating the classification of:
S = “fast, couple, shoot, fly”
yields:
log[P(S | comedy)] = -13.737 and log[P(S | action)] = -12.510
So this model will classify S as Action

d. For output and code please see the attached documents.
My first strategy for this model was to separate all characters which are not alphanumeric from the tokens before them in preprocessing, take counts in reference to the feature vocabulary, and then compute the log weighted probabilities. This method on the main data sets yielded an accuracy of 81.34%. My first idea on improving the quality of my model’s classification was to remove very common words as I thought they may not be giving class specific information. I removed the words “the” , “to” , “be” and “of” which lowered the accuracy of the model marginally to 81.24%. Upon inspection of the vocabulary, I noticed that there are contractions and hyphenated words, and that there was very little symbols. As a result, I added “&” and “%” to the vocabulary and left all apostrophe and hyphen symbols attached to their tokens instead of separating them in preprocessing. This caused an increase in accuracy from 81.34% to 91.728%. I also noticed that the vocabulary had certain emoji patterns, but I did not have time to test how I could augment preprocessing to allow for them. One trend I noticed among the incorrect predictions was that the scores for each

incorrect prediction (pos/neg) were very close in value. I believe that a larger data set or an additional discounting scheme may be able to steer the close predictions farther onto the right path. With more time I would try both.