**Summer Semester 2020 SNLP Assignment 8**

**Name: Awantee Deshpande  
Id: 2581348  
Email:** [**s8awdesh@stud.uni-saarland.de**](mailto:s8awdesh@stud.uni-saarland.de)

**Name: Lakshmi Rajendra Bashyam  
Id: 2581455  
Email:** [**s8laraje@stud.uni-saarland.de**](mailto:s8awdesh@stud.uni-saarland.de)

**1. Feature Selection**

1.1 Pointwise Mutual Information  
The code for this part is in the file PMI.py. We compute the PMI for unigram count > 100 and bigram count > 50 for both, the negative and positive classes and get the following results:  
  
Top 20 Negative Class Unigrams PMI  
{'boll': 1.0136146221461328, 'uwe': 1.0104865845923172, 'seagal': 0.9926614718202373, 'mstk': 0.9824066112489906, 'unwatchable': 0.9698754183514585, 'incoherent': 0.9486062917423018, 'unfunny': 0.9306594385126794, 'waste': 0.9260872097438764, 'blah': 0.9173134132649419, 'horrid': 0.906197381070768, 'pointless': 0.9037901910651704, 'drivel': 0.9014594620641049, 'atrocious': 0.8986241613659672, 'redeeming': 0.8948650904543358, 'prom': 0.891163442337418, 'lousy': 0.8900450995306762, 'worst': 0.8851986499924681, 'laughable': 0.8789910111059321, 'awful': 0.8774059698954507, 'poorly': 0.8772524749956983}

Top 20 Positive Class Unigrams PMI  
{'edie': 0.9755786458324426, 'paulie': 0.9612233528553726, 'felix': 0.9200835332407387, 'matthau': 0.8987630487816116, 'victoria': 0.8834604438440112, 'flawless': 0.8785541914375781, 'mildred': 0.8779678492060202, 'gandhi': 0.8589337269801721, 'astaire': 0.8524304150595775, 'superbly': 0.8404190625508069, 'perfection': 0.8317428732587909, 'voight': 0.8074558870241155, 'wonderfully': 0.8073539333799695, 'brosnan': 0.8073071632426437, 'captures': 0.8056536443901301, 'peters': 0.7911540746950151, 'powell': 0.7840471882007553, 'bourne': 0.7805626634272997, 'refreshing': 0.7799143235912528, 'mustsee': 0.7796584358571857}

Top 20 Negative Class Bigrams PMI  
{('even', 'worth'): 1.031448617714144, ('prom', 'night'): 1.0314486177141435, ('terrible', 'movie'): 1.0314486177141435, ('worst', 'films'): 1.0183924648886973, ('uwe', 'boll'): 1.0137466159806845, ('total', 'waste'): 1.0067865634798745, ('worst', 'movies'): 1.0038363865062838, ('movie', 'sucks'): 1.003434241544547, ('badly', 'written'): 1.002879465517373, ('terrible', 'film'): 1.002879465517373, ('bad', 'bad'): 0.9975012857908059, ('worst', 'film'): 0.995469320608577, ('awful', 'movie'): 0.9877272402848256, ('worst', 'movie'): 0.9830855961527445, ('film', 'school'): 0.9737331198578564, ('poorly', 'made'): 0.9737331198578564, ('money', 'back'): 0.9733456631505754, ('dont', 'waste'): 0.9732675466256144, ('complete', 'waste'): 0.9700480730500003, ('power', 'rangers'): 0.9610592898227457}

Top 20 Positive Class Bigrams PMI  
{('red', 'sox'): 0.9692223162621568, ('gunga', 'din'): 0.9692223162621567, ('rob', 'roy'): 0.951944324830321, ('midnight', 'cowboy'): 0.9457633434381677, ('nancy', 'drew'): 0.9453755743077888, ('worful', 'movie'): 0.9432271077292121, ('police'nde, 'story'): 0.9412079400925601, ('excellent', 'movie'): 0.9248281969037032, ('perfectly', 'cast'): 0.9157830573006961, ('well', 'worth'): 0.9131277541912305, ('highly', 'recommended'): 0.9126387878957893, ('favorite', 'movies'): 0.9041272880402718, ('definitely', 'worth'): 0.8964659738268425, ('beautiful', 'film'): 0.8834924422362728, ('excellent', 'job'): 0.8804130493043005, ('michael', 'jackson'): 0.8606978594839876, ('wonderful', 'film'): 0.8606978594839876, ('walter', 'matthau'): 0.8581910038734126, ('rock', 'hudson'): 0.848928082544445, ('first', 'rate'): 0.8458399007568748}

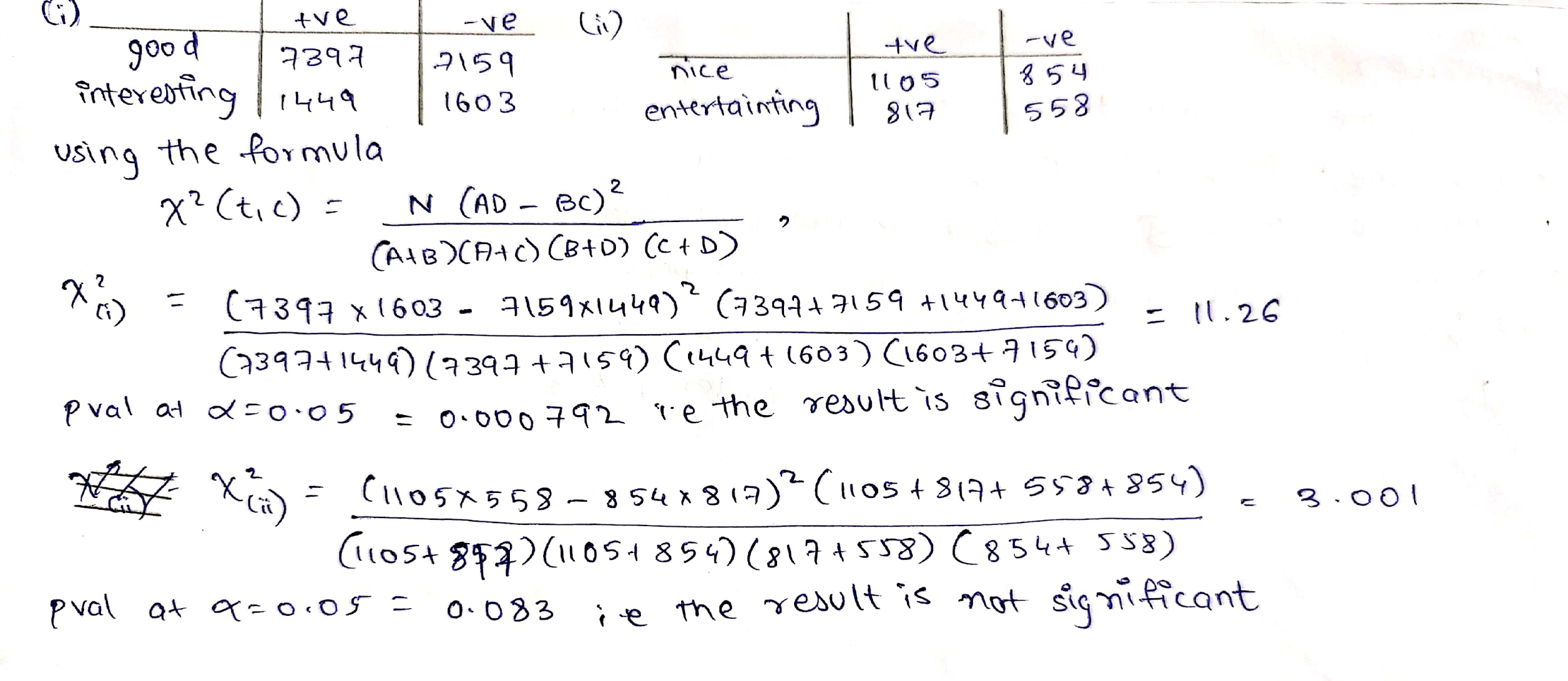
The PMI for words *good* and *bad* in the two classes are also computed.

|  |  |  |
| --- | --- | --- |
| PMI | Negative | Positive |
| *good* | 0.000413141374 | -0.00039938456 |
| *bad* | 0.696211725311 | -1.31984829203 |

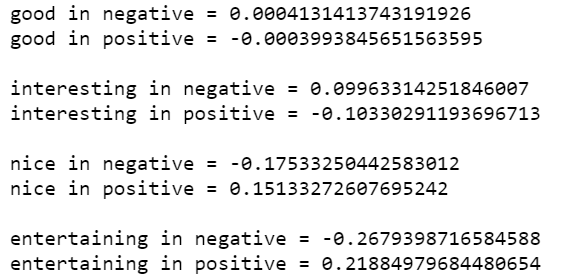
Analysis: PMI(x,y) = 0 means that the particular values of x and y are statistically independent. A positive PMI value means they co-occur more frequently than would be expected under an independence assumption, and negative PMI means they co-occur less frequently than would be expected.

In the above example, the top 20 words (both unigrams and bigrams) for the negative class contain words like ‘pointless’, ‘worst’, ‘terrible’ etc. Those for the positive class contain words like ‘excellent’, ‘favourite’, ‘refreshing’ etc. Adjectives are themselves indicators of negative and positive sentiments and it can be inferred from the words alone what class they probably come from (this is even easier for bigrams because of the context e.g. ‘beautiful film’). For noun terms, this is difficult unless we have prior knowledge of what entities are generally popular and which ones are not. PMI is an unreliable metric for less frequently occurring words, and if the term occurs only once, the PMI is the best over all the set of singletons. This is why we filter out the low frequency words from the calculation.  
For the words *good* and *bad*, it can be observed that the PMI of good is higher in the positive class and lower in the negative class as expected. It is surprising that the PMI of good alone is lower in the positive class than the negative one. But the count of *good* is higher in the negative class and we can state that contexts like ‘not good’, ‘i thought it was good, but…’ etc. will not be captured in the unigram alone, and hence the results.

1.2 Chi Square Test



We can see from the p-values that we can reject the null hypothesis for the words {good, interesting}. The distributions for these words are not dependent on the type of movie reviews. For {nice, interesting} we cannot reject the hypothesis. This can also be observed from the PMI values we compute in the previous example.

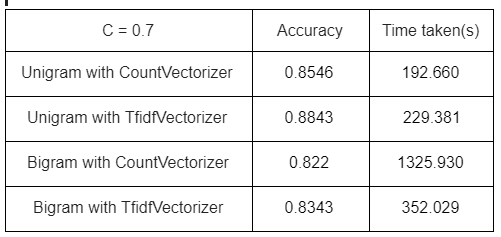
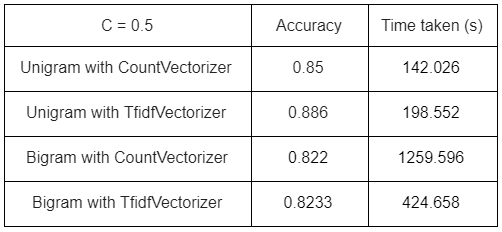


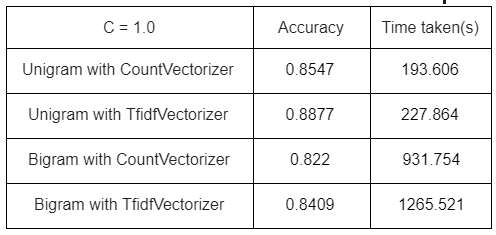
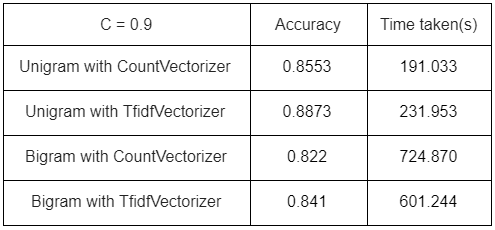
**2 Classification**

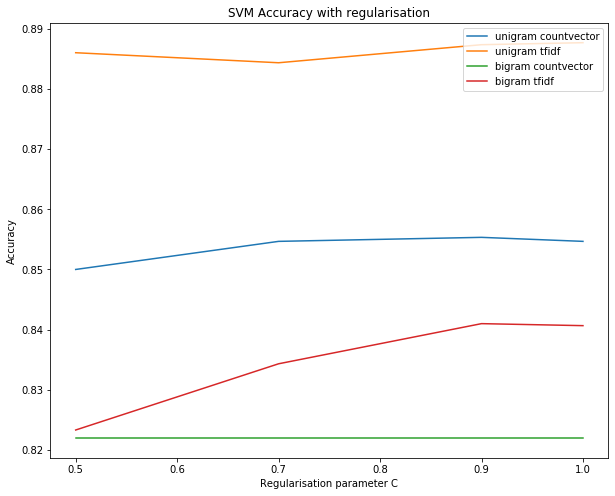
The code for the classifiers SVM and Multinomial Bayes are in the file text\_classification.py.

2.1 Support Vector Machines  
NOTE: Due to computation load, we run the code on 3/5th data only.

After splitting the data into 80% training size and 20% test size and performing SVM based classification with a linear kernel over the regularisation parameter range C = [0.5, 0.7, 0.9, 1.0], we get the following results.



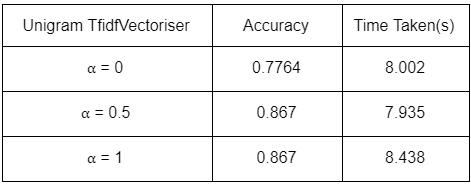
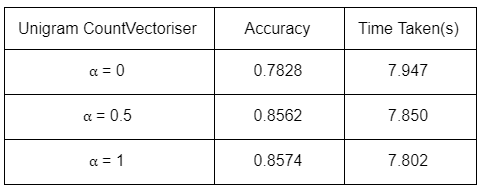


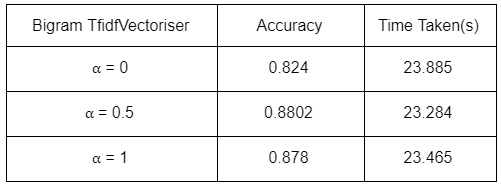
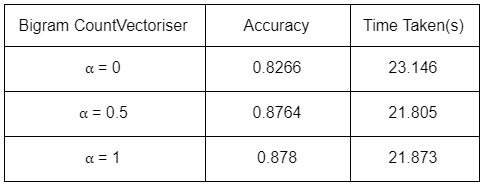


Analysis: We can observe that the accuracy of all the models in pretty much in the same range, with that of the Unigram-Tfidf being the highest. There is hardly any change in the model with an increase in the regularisation parameter C. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.  
Because there is not much deviation in accuracy with change in C, we can infer that the class separation for the positive and negative class vectors is distinct enough that a change in C does not change the test accuracy much.   
The tf-idf model works better in all cases over the simple count models, since tfidf places a weightage on the word occurrence in a document as well as in all the other documents, leading to better vectors. The best accuracy is obtained for unigrams with TfidfVectorizer and C = 1 (88.77 %).

2.2 Different Classifiers

For this task, we choose the Multinomial Bayes Classifier to train the classification model. We obtain the following results.





Analysis: The Naive Bayes family of classifiers is based on applying Bayes theorem with the assumption that every feature is independent of the other. It predicts the conditional probability of the class given the probability distribution of the word tokens. The Multinomial NB classifier is useful when the distribution has integer feature counts (i.e. word frequencies here), though in practice, even tfidf may work.  
The parameters for this algorithm are:  
1) ⍺ - Laplace smoothing parameter for calculating probabilities (0 for no smoothing)

2) fit\_prior - A boolean to specify whether class prior probabilities (here P(neg) and P(pos)) should be learned. If false, a uniform prior is assumed.

3) class\_prior - If prior probabilities are explicitly known beforehand, they can be passed via this parameter.

Here, the highest accuracy is 87.8% for the Bigram TfidfVectoriser combination, though the score is similar to SVMs. Naturally, the accuracy increases as we increase our smoothing parameter.

Comparison with SVMs:   
Since both, SVM with linear kernel and Multinomial NB can be interpreted as linear models, they show similar results. One major difference is that SVMs do not scale well with large data. Au contraire, Multinomial NB is quick. This can be observed in the time taken for training each model, and note that Multinomial NB has been trained on the *entire* dataset (the maximum accuracy is lower (85.9%) for 3/5th of the data). While Naive Bayes treats the features as independent, SVMs try to determine the interactions between the features to some degree, giving SVMs a slight edge over Multinomial NB. We can see that SVM has better accuracy than Multinomial NB on lesser data.