Report

1. Dataset splits

① Among the 3000 sentences, I choose 60% (1800) as training, 20% (600) as validation and 20% (600) as testing.

② Before splitting, I first shuttle all the sentences in the dataset.

2. Feature Selection

① Feature Vector 1 (FV1) ----Length: 817

FV1 is the original feature vector taken from the HW2 with Lemmatization and deleting some noisy words whose length is less than 2 or frequency is less than 5 times. The content $F_{i,j}$ in the feature means the count of word j in sentence i.

② Feature Vector 2 (FV2) ----Length: 256

FV2 keeps the top-256 most frequent words in the entire dataset.

③ Feature Vector 3 (FV3) ----Length: 256

FV3 keeps words with top-256 TF-IDF scores in the entire dataset.

3. Classification Algorithm

① I used three classification algorithms:

1) K-NN (Implemented by me with Euclidean distance);
2) Decision trees (NLTK package); 3) Naïve Bayes (NLTK package)

4. Performance Evaluation and Analysis (Python)

① Choosing hyper parameter with Accuracy on Validation Set

K-NN: choose the number of neighbors k that maximizes the classification accuracy.

Decision tree: choose the different depth of the tree to maximize the accuracy.

Naïve Bayes: no hyper parameter needs to be chosen.

Metrics	K	K-NN			Depth _ cutoff	Decision trees		
		FV-1	FV-2	FV-3		FV-1	FV-2	FV-3
Accuracy (TP+TN) /All	3	0.63333	0.63333	0.63833	50	0.775	0.687	0.662
	5	**0.65000**	**0.64667**	**0.64167**	60	0.770	**0.690**	0.660
	7	**0.65000**	0.64167	0.63167	70	**0.778**	**0.690**	0.663
	9	0.63667	0.62500	0.61667	80	0.775	**0.690**	0.677
	11	0.64333	0.62667	0.61500	90	0.770	**0.690**	**0.685**

As a result, I choose k = 5 and Depth _ cutoff = 90

② Overall Comparison

Metrics	Dataset	K-NN*			Decision trees			Naïve Bayes		
		FV-1	FV-2	FV-3	FV-1	FV-2	FV-3	FV-1	FV-2	FV-3
Time(s)	Training	0	0	0	162.8	**36.06**	51.63	1.109	0.336	**0.329**
	Test for a new tuple	0.282211	0.093459	**0.090507**	0.000036	**0.000022**	0.000033	0.002681	0.000841	**0.000832**
Accuracy (TP+TN) / All	Training	0.777	**0.782**	0.741	**0.913**	0.898	0.801	**0.893**	0.811	0.784
	validation	**0.642**	0.633	0.568	**0.773**	0.698	0.687	**0.820**	0.757	0.737
	Testing	**0.662**	0.645	0.658	**0.778**	0.715	0.685	**0.782**	0.752	0.705

Precision	Training	**0.736**	0.742	0.719	**0.862**	0.849	0.732	**0.894**	0.812	0.819
TP /	Validation	**0.641**	0.633	0.582	**0.732**	0.668	0.632	**0.821**	0.755	0.749
(TP + FP)	Testing	0.647	0.635	**0.655**	**0.741**	0.692	0.629	**0.781**	0.765	0.700
Recall	Training	0.841	**0.844**	0.765	**0.988**	0.974	0.961	**0.899**	0.818	0.741
TP /	Validation	**0.712**	0.706	0.610	**0.829**	0.735	0.826	**0.798**	0.728	0.676
(TP+ FN)	Testing	**0.773**	0.751	0.728	0.840	0.751	**0.867**	**0.768**	0.710	0.693
This table's data is not exactly the same with the table above, this is because the dataset after shuffle isn't always the same. However, in each table, datasets keep same. I should use the same shuffled dataset to run these two experiments in the future.										

③ How do the prediction results differ before and after feature selection?

1) Time: After feature selection, both offline and online efficiency costs reduce because of the shorter length of feature vector after pruning.
2) Accuracy, Precision, Recall: In most cases, pruned features are not as good as the original features, since the feature selection methods are not very good and pruned features have lost a lot of useful information. Among two feature selection methods, the top K frequency method is better than the top TF-IDF method. This is because TF-IDF method reduces scores of some words with higher frequency in general. This is one defect of the IF-IDF method in this task, since some top frequency words in general are also important, such as "not" 306 times, "good" 281 times, "very" 243 times, "great" 212 times.

④ Which classifier in this case is better?

1) Naïve Bayes is the best in this case, since the dependence among features are not serious.
2) Decision Tree is the second best.
3) K-NN is the worst. In this case, K-NN implemented by me has some limitations.
   a) If we have two feature vectors from two samples, there could be two cases.
      Case 1: Feature: "biscuits: 0, like: 1, …" and "biscuits: 0, like: 1, …".
      Case 2: Feature: "biscuits: 0, like: 0, …" and "biscuits: 0, like: 0, …".
      The distances in these two cases are the same, but the importance of both or neither existing "like" are different. Both "like" leads to the high probability of belonging to the same class. However, the calculation of K-NN hasn't shown this.
   b) After pruning features, the length of features is short. As a result, there may exists sentences whose feature vectors are all 0. Then the distance between them is 0, which means the closest. However, this closest distance is meaningless and will influence the results.

5. Potential Improvement

① Use more advanced features, such as CNN features, instead of the count of word in the sentence.
② Implement better feature selection methods, such as picking out top-K most informative features according to the Random Forests.
③ Use better classification algorithms, such as Random Forests, XGBoost, SVM and CNN, etc.
④ Optimize the K-NN Algorithm implemented by me to improve accuracy and speed up.