Semantic Pattern Classification

Duong Thanh Long
Machine Learning 2011-2012
Charles University
Final Report

Table of Contents

TASK A

1. Problem Abstract

Our target is classifying semantic pattern of 6 verbs (Ally, Arrive, Cry, Halt, Plough, Submit). Each verb has limited number of sense

Word	Number of Sense
Ally	6
Arrive	6
Cry	19
Halt	3
Plough	19
Submit	6

We are given raw data and are supposed to construct feature vectors. After that, these vectors will be used for machine learning (ML) algorithms. The assignment is divided in to 2 parts. The first part we are asked to implement a single learning algorithm. The second part we applied several ML algorithms and choose the best one. More notably, in this part we also have to develop our own feature set and test again it.

2. Data Description

For each verb, we have 250 instances. All annotated instances have been (randomly) selected from the BNC corpus. Each instance consists of 6 items.

Item	Description
sentence ID	ID of the sentence
pattern tag	the manually annotated class label
tokenized sentence	tokens are separated by spaces
morphologically analysed sentence	tokens are separated by tabs; each token includes - original word form - its lemma - its morphological tag
syntactic dependencies	obtained automatically using the Stanford dependency parser
named entity recognizer	It's an optional items, in fact we don't really need it

From instances, we create feature vectors. There are total 284 features for an instance (1 - pattern label, 83-morphology features, 200 semantic features). However, we shouldn't use all the features since some feature only have a small number of instances and thus, the result provide by machine learning algorithms base on this feature is unreliable. For that purpose, I define a variable namely, level of instance confidence (LIC). We choose features if and only if

Number of Instance of Feature >= LIC

So, the first steps we set **LIC = 10** and we eliminate a lot of unreliable features (nearly 100 features), so the remaining features is183 and will be reduced more in the last part of this report. The reasons for LCT = 10 is not so clearly, but with 5 < LIC < 15 the number of eliminated feature does not vary much.

3. Experiment

3.1. Preliminary

In order to execute source code successfully these following are required.

- First you need to change the working directory in R to the corresponding directory eg: "*D:/Short Report/*".
- File **ConvertToFeature.R** implement code to read all text file in *./Short Report/data/development.instances/* and output features to text files in *./Short Report/featureVectors/* respectively.
- File **code.R** construct machine learning algorithms.
- The following package is needed
  - **Hash**: I use this package to support the semantic features classification process. Since the semantic classes file "semantic-classes.wn" is very huge, using hash is appropriate approach
  - **Rpart** : this package provide method for constructing decision tree
  - **E1071** : we use SVM (support vector machine) from this package
  - **Adabag** : use for bagging and boosting technique.
  - **RWeka :** use for KNN algorithms.
  - **FSelector :** use for feature selection

3.2. Accuracy

For each verb, the accuracy of classifier is measure using this formula

$$\boldsymbol{Classifier\ Accuracy} = \frac{Number\ of\ Correct\ Classify\ Case}{Total\ Number\ of\ Test\ Case}$$

So, for each ML algorithms, we tune the different parameters and acquire 6 classifiers for a single verb (these classifier can be the same or different). The accuracy of an ML algorithm is as follow.

$$\boldsymbol{ML\ Algorithm\ Accuracy} = \sum_v \frac{p(v) * A(v)}{p(v)}$$

With A(v) is classifier accuracy and p(v) is the verbs' relative frequencies

Verb	**Relative Frequency**
Ally	0.0083%
Arrive	0.1307%
Cry	0.0257%
Halt	0.0183%
Plough	0.0076%
Submit	0.0483%
$\sum_v p(v)$	0.2389%

3.3. Base line

Base line of each verb is the classifier accuracy if we always classify verb base on the highest frequency. For example, the following table is frequency of label of Ally

1	**2**	**3**	**4**	**5**	**6**	**Ux**
19	50	119	44	7	3	8

So, all instances will be classified as label 3 (highest frequency). And the accuracy will be 119 / 250 = 47.6 %. Here the baseline of all verbs.

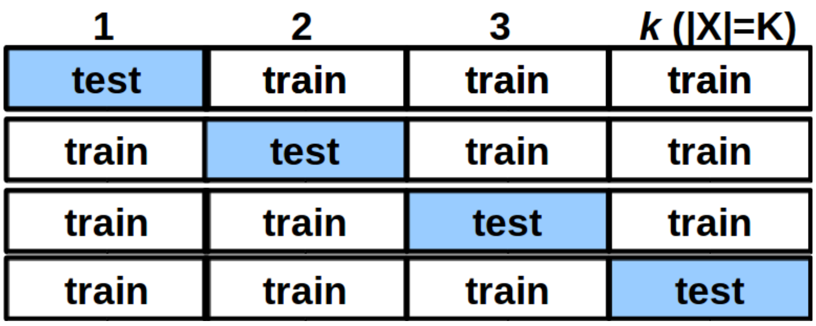
Verb	**Baseline**
Ally	48%
Arrive	68%
Cry	52%
Halt	84%
Plough	32%
Submit	71%

So, the algorithms base line base on weighted of each word will be:

$$\sum_v \frac{p(v) * A(v)}{p(v)} = \mathbf{66.24}\ \%$$

3.4. Cross-Validation

Cross validation is a technique to acquire better estimation of a model. Instead of just divide data to training data and testing data once, it repeats the process several times and the final accuracy will be the average. There are many techniques for cross-validation, but the most common is leave-one-out validation. Data is divided into K-folds, each iteration we use k-1 fold for training data, and 1 fold for testing data.



1	2	3	*k* (|X|=K)
test	train	train	train
train	test	train	train
train	train	test	train
train	train	train	test

With our data of 250 records, we will use **6-fold cross validation for every machine learning algorithms below.** The reasons for 6 fold cross validation is simple, by using 6 folds, the ratio of training data and test data is as the same as with unseen test data. (42/208 ~ 50/250) .

Moreover, the average accuracy solely is not enough to say a learning model is good a not. For example, 6 fold cross validation may be

Fold	Accuracy
Fold-1	40%
Fold-2	50%
Fold-3	60%
Fold-4	70%
Fold-5	80%
Fold-6	90%
Average	65%

As you can see, the average accuracy 65% is not meaningful when the range of value is so big that the model must be unstable. Therefore, we adopt “confident interval” to tackle the problem. And from now on, to take into consideration of choosing the right parameter for learning algorithm model, it must be **“Accuracy average” accompany with “Confidence Interval”**

With the assumption that accuracy is normal distribution, we can calculate the error using equation

error = qnorm(1- significant level)* standard deviation /sqrt(size of sample)

And the confident interval will be from (mean-error => mean + error). So with the above example, the 95% confident interval is 65 +/- 12.56

From the next part of this report, I will use **significant level of 0.05% for all my calculation.**

3.5. Develop the best model

3.5.1. Decision Tree

Decision Tree is constructed base on "Rpart" package.

rpart (formula, data=, method=, control=)

Formula	is in the format: outcome ~ predictor1+predictor2+predictor3+ect.
Data	specifies the dataframe
Method	"class" for a classification tree "anova" for a regression tree
Control	optional parameters for controlling tree growth.

We interested in parameter of **control part** from the command.

Minsplit	the minimum number of observations that must exist in a node, in order for a split to be attempted. Default value of Minsplit is 20
cp	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. Default value of cp is 0.01
Split	Gini : for gini index splitting criteria (Default value) Information : for information gain splitting criteria

Since information gain is normally slightly better than gini index, we always use information gain for splitting criteria. Here the result of decision tree with different parameters.

Cp	**Minsplit**	ACCURACY					
		Ally	Arrive	Cry	Halt	Plough	Submit
0.01	10	0.553-/+0.05	0.585-/+0.01	0.508-/+0.04	0.805-/+0.05	0.423-/+0.07	0.841-/+0.04
0.01	20	0.549-/+0.05	0.642-/+0.04	0.537-/+0.05	0.813-/+0.03	0.455-/+0.09	0.833-/+0.04
0.01	30	0.541-/+0.06	0.654-/+0.03	0.52-/+0.09	0.817-/+0.04	0.459-/+0.09	0.85-/+0.04
0.1	20	0.476-/+0.1	0.675-/+0.03	0.5-/+0.06	0.833-/+0.07	0.297-/+0.09	0.817-/+0.04

0.05	20	0.557-/+0.11	0.659-/+0.02	0.488-/+0.05	0.821-/+0.04	0.386-/+0.11	0.841-/+0.04
0.05	30	0.557-/+0.12	0.659-/+0.02	0.488-/+0.05	0.829-/+0.05	0.386-/+0.11	0.846-/+0.05

However, aside from average accuracy, the confident interval is important to say that

For each verb, we choose the best parameters and the overall accuracy of Decision Tree model is

$$\sum_v \frac{p(v) * A(v)}{p(v)} = \mathbf{69.6}\ \%$$

Decision tree is considered an **unstable learning algorithm** since small changes in training data cause large difference in generated models. So, we are going to apply **bagging and boosting** technique to restrain the un-stability. The key idea of bagging technique is creating different training sets. In overall, it includes 4 steps

- Take repeated bootstrap samples to create a sequence of training sets
- Generate new samples by drawing instances from the original sample with replacement
- Train classifiers using the training sets
- Classification by majority voting

Implementation of bagging in R uses **adabag** package.

bagging(formula, data, mfinal = 100, control)

Formula	Same as decision tree
Data	Same as decision tree
Mfinal	An integer, the number of iterations for which boosting is run or the number of trees to use.
Control	Same as decision tree

Since tuning parameter won't change much the result for decision tree and running time of bagging algorithm is extremely slow, we apply with the best parameters of cp (0.01), minsplit(30) with 20 iterations (Mfinal) which archive from above calculation.

Mfinal	**Cp**	**Minsplit**	ACCURACY					
			Ally	Arrive	Cry	Halt	Plough	Submit
20	0.01	30	0.621951	0.6747967	0.581301	0.8373984	0.46748	0.841463

Compare with the original algorithms, it's clearly that applying bagging get better result. However, *it's not good enough with respect to time we have to pay*.

$$\sum_v \frac{p(v) * A(v)}{p(v)} = \mathbf{70.26}\ \%$$

3.5.2. KNN – K Nearest Neighbor

Knn learning algorithm is include inside "RWeka" package. It's true that knn is provided in several package eg "class". However, RWeka is more powerful since it lets us customized a lot of thing.

IBk(formula, data, control = Weka_control())

Formula	is in the format: outcome ~ predictor1+predictor2+predictor3+ect.
Data	specifies the dataframe
control	Class of weka_controls - **K** : number of k-nearest neighbor - **X** : if X = true -> it will automatically search for k' < K that return the best value (using cross validation) - **I** : Neighbors will be weighted by the inverse of their distance when voting. (default equal weighting) - **F** : Weight neighbours by 1 - their distance Eg. control = Weka_control(K = 20, X = TRUE, I = TRUE)

Here the result of KNN using different parameters.

K	**I**	ACCURACY					
		Ally	Arrive	Cry	Halt	Plough	Submit
30	TRUE	0.545-/+0.14	0.675-/+0.03	0.524-/+0.06	0.833-/+0.07	0.374-/+0.11	0.715-/+0.11
20	TRUE	0.565-/+0.11	0.675-/+0.03	0.537-/+0.05	0.833-/+0.07	0.37-/+0.09	0.707-/+0.09
10	TRUE	0.561-/+0.1	0.675-/+0.02	0.533-/+0.06	0.829-/+0.05	0.382-/+0.07	0.748-/+0.05
5	TRUE	0.577-/+0.09	0.675-/+0.02	0.537-/+0.07	0.797-/+0.03	0.39-/+0.05	0.744-/+0.04
30	FALSE	0.537-/+0.14	0.675-/+0.03	0.524-/+0.06	0.833-/+0.07	0.378-/+0.11	0.711-/+0.1
20	FALSE	0.557-	0.675-	0.528-	0.833-	0.382-	0.711-

		/+0.11	/+0.03	/+0.06	/+0.07	/+0.1	/+0.09
10	FALSE	0.553-/+0.1	0.671-/+0.03	0.524-/+0.07	0.829-/+0.05	0.378-/+0.1	0.736-/+0.07
5	FALSE	0.565-/+0.11	0.683-/+0.03	0.533-/+0.05	0.825-/+0.04	0.39-/+0.05	0.748-/+0.05

So, the overall accuracy of KNN learning algorithms is

$$\sum_v \frac{p(v) * A(v)}{p(v)} = \mathbf{67.45}\ \%$$

3.5.3. SVM – Support Vector Machine

SVM is constructed using **e1071** packages.

svm(formular, data, scale = TRUE, type = NULL, kernel = "radial", degree = 3, gamma, cost = 1, class.weights = NULL, cross =)

Formula	is in the format: outcome ~ predictor1+predictor2+predictor3+ect.
Data	specifies the dataframe
Scale	Normally, SVM work better when data is scaled. So, SVM scales it by default
Type	SVM work well with both regression and classification. If class labels is categorized, SVM will automatically use classification
Kernel	Set the kernel function, default if "radial". Can be chosen from 4 types **linear**: u'*v **polynomial**: (gamma*u'*v + coef0)^degree **radial basis**: exp(-gamma*|u-v|^2) **sigmoid**: tanh(gamma*u'*v + coef0)
Gamma	parameter needed for all kernels except linear (default: 1/(data dimension))
Cost	Cost of constraints violation (default: 1). If cost is high, SVM go strict with slack (outlier).
Class.weights	In case of asymmetric class sizes, we may want to avoid possibly over proportional influence of bigger classes. So set class weight for each class.
Cross	Build in cross validation

Here the result of SVM using kernel function: **radial basis** (default)

Cost	Gamma	ACCURACY					
		Ally	Arrive	Cry	Halt	Plough	Submit
0.001	Default	47.551-/+5.08	68.031-/+4.61	52.381-/+3.86	83.624-/+6.64	32.414-/+3.07	70.867-/+6.24
0.1	Default	47.551-/+5.08	68.031-/+4.61	52.381-/+3.86	83.624-/+6.64	32.414-/+3.07	70.867-/+6.24
1	Default	59.108-/+8.1	68.428-/+4.37	60.782-/+2.92	83.624-/+6.64	47.213-/+3.69	80.449-/+5.23
10	Default	61.943-/+6.42	69.232-/+4.47	60.801-/+3.15	86.024-/+6.49	49.613-/+3.7	81.262-/+6.76
100	Default	61.943-/+6.42	69.232-/+4.47	61.605-/+3.36	85.627-/+7.13	50.019-/+3.56	81.262-/+6.76
1000	Default	61.943-/+6.42	69.232-/+4.47	61.605-/+3.36	85.627-/+7.13	50.019-/+3.56	81.262-/+6.76
100	0.00001	55.943-/+5.72	68.031-/+4.61	52.787-/+4.12	83.624-/+6.64	39.208-/+4.96	71.264-/+6.29
100	0.00005	60.743-/+7.08	70.422-/+4.3	61.595-/+3.08	84.417-/+6.9	48.422-/+2.61	82.056-/+5.74
100	0.0001	62.35-/+6.15	67.228-/+3	60.017-/+4.87	85.211-/+6.34	50.842-/+4.86	80.865-/+5.75
100	0.0005	58.759-/+5.38	65.215-/+4.61	55.643-/+5.29	82.791-/+6.67	50.387-/+4.29	80.449-/+5.03
100	0.001	58.798-/+3.71	61.992-/+3.01	55.236-/+4.21	81.988-/+6.55	48.384-/+4.55	80.846-/+4.78
100	0.1	49.139-/+6.23	68.031-/+4.61	52.381-/+3.86	83.624-/+6.64	33.208-/+2.61	70.867-/+6.24
100	1	49.139-/+6.23	68.031-/+4.61	52.381-/+3.86	83.624-/+6.64	32.414-/+3.07	70.867-/+6.24
100	10	49.139-/+6.23	68.031-/+4.61	52.381-/+3.86	83.624-/+6.64	32.414-/+3.07	70.867-/+6.24

So the overall of SVM using radias basis

$$\sum_v \frac{p(v) * A(v)}{p(v)} = \mathbf{72.06}\ \%$$

Using kernel function **polynomial** require one more parameter (degree). Using tune.svm we can search through a wide range of parameters. For cost value (2^{-10} -> 2^{10}), degree (2->10), gamma (10^{-5} -> 10^{5})

Word	Best parameter			Accuracy
	Cost	Degree	Gamma	
Ally	16	3	0.1	0.584
Arrive	2	2	0.01	0.708

Cry	128	3	0.01	0.564
Halt	0.0009765625	2	0.00001	0.836
Plough	4	2	0.1	0.46
Submit	8	2	0.01	0.808

So, the overall of SVM using polynomial kernel function

$$\sum_v \frac{p(v) * A(v)}{p(v)} = \mathbf{71.03}\ \%$$

3.5.4. Best Model Conclusion

The following table gives the conclusion of best developed model.

Model	**Accuracy**
Decision Tree	69.6
Decision Tree – Bagging	70.26
KNN	67.45
SVM – Gaussian	72.06
SVM – Polynomial	71.03

As expected, SVM out-perform over other learning methods (Decision Tree, Booting and Bagging for Decision Tree, and KNN). From the experiment we re-confirm that Gaussian kernel function is very robust and should be tried first since it's effective and quite easy to use (only need to tune 2 parameters). Thus, **we will use SVM using Gaussian kernel function for the part B** of this report.

However, compare with baseline, all these methods only improve a litter bit.

Model	**Baseline**	**Accuracy**	**Improved (%)**
Decision Tree	66.24	69.6	5%
Decision Tree – Bagging	66.24	70.26	6%
KNN	66.24	67.45	2%
SVM – Gaussian	66.24	72.06	9%
SVM – Polynomial	66.24	71.03	7%

It may be due to the fact that, *data we have is not enough to construct a reliable classifier*. We are provided with 250 records but some words we need to classify into nearly 20 classes.

TASK B

4. Dimension Reduction

Dimension Reduction is the process of reducing the number of feature under consideration. When there are too many feature vectors. It may become a mess and classifier doesn't really know what is important to base on. So, the main purpose of dimension reduction will be

- Finding meaningful features.
- Reduce noise in the feature set
- Support the data analysis (classification, clustering …)

There are many algorithms for dimension reduction, yet there are too main approaches

- Feature Ranking
- Subset Selection

In the first approach, features are ranked by some criteria and then features above a defined threshold are selected. We will use this method for the purpose of this assignment. The most important things about feature ranking is the function **which determine the weighted of features.** There are several functions to do that which is supported in R. (library **Fselector** and **mlbench** are required)

- Chi-Square Filter

```
chi.squared(formula, data)
```

- Correlation Filter
  - Pearson's correlation

```
linear.correlation(formula, data)
```

  - Spearman's correlation

```
rank.correlation(formula, data)
```

- Entropy-Based
  - Information Gain

```
information.gain(formula, data)
```

  - Gain Ratio

```
gain.ratio(formula, data)
```

- Symmetrical Uncertainty

```
symmetrical.uncertainty(formula, data)
```

- OneR

```
oneR(formula, data)
```

- Random Forest Filter

```
random.forest.importance(formula, data, importance.type = 1)
```

5. Experiment on new features.

In this part, we are asked to choose 3 words and apply our feature selector accompany with the best model which is developed in the early part (SVM – Gaussian). I choose 3 most hard-to-classify words (Ally, Cry, Plough). The current best accuracy for this using SVM is

Word	No of Meaning	Baseline	Current Best	Parameters	
				Cost	Gamma
Ally	6	48%	62.35-/+6.15	100	0.0001
Cry	19	52%	61.595-/+3.08	100	0.00005
Plough	19	32%	50.019-/+3.56	1000	Default

Here is the table acquired after applying feature selection and SVM – Gaussian learning algorithms with corresponding best parameters

Word	Selection Method	Number of Features	Accuracy
Ally	Entropy – Information Gain	50	60.734-/+8.37
		60	61.547-/+7.79
		70	61.566-/+5.25
		80	63.976-/+7.41
		90	62.35-/+6.58
		100	62.35-/+7.08
		110	62.35-/+6.15
		120	62.35-/+6.15
	Chi-Square	50	60.734-/+8.37
		60	61.547-/+7.79
		70	61.566-/+5.25
		80	63.976-/+7.41

		90	62.35-/+6.58
		100	62.35-/+7.08
		110	62.35-/+6.15
		120	62.35-/+6.15
	Random – Forest	50	65.931-/+8.72
		60	67.141-/+9.05
		70	65.544-/+6.43
		80	65.563-/+6.52
		90	64.344-/+7.09
		100	65.137-/+6.99
		110	62.35-/+6.15
		120	62.35-/+6.15
Cry	Random - Forest	50	60.376-/+1.95
		60	61.585-/+1.61
		70	60.782-/+2.54
		80	62.379-/+2.36
		90	62.389-/+3.98
		100	62.408-/+3.11
		110	61.595-/+3.08
		120	61.595-/+3.08
	Entropy – Information Gain	40	57.191-/+4.37
		50	57.975-/+4.27
		60	59.988-/+3.52
		70	61.198-/+2.13
		80	59.988-/+3.24
		90	60.801-/+3.15
		100	59.998-/+3.35
		110	61.595-/+3.08
		120	61.595-/+3.08
	OneR	40	57.985-/+4.33
		50	58.382-/+4.09
		60	58.788-/+3.19
		70	59.582-/+3.07
		80	57.985-/+3.58
		90	58.788-/+3.49
		100	59.185-/+4.03
		110	61.595-/+3.08
		120	61.595-/+3.08
Plough	Random Forest	40	53.175-/+3.19
		50	53.997-/+3.73
		60	58.788-/+4.04
		70	56.03-/+4.59

		80	54.017-/+3.91
		90	57.569-/+3.98
		100	53.997-/+2.6
		110	54.007-/+3.39
		120	52.41-/+3.35
	Entropy – Information Gain	40	51.587-/+3.2
		50	52.758-/+6.39
		60	51.181-/+4.38
		70	52.4-/+2.74
		80	53.63-/+4.39
		90	52.836-/+3.82
		100	51.607-/+4.69
		110	49.613-/+4.48
		120	51.22-/+3.47

6. Conclusion

In summary, we have best selection method and parameters.

Word	Selection Method	Number of Feature	SVM- Gaussian		Final Accuracy
			Cost	Gamma	
Ally	Random Forest	80	100	0.0001	65.563-/+6.52
Cry	Random Forest	80	100	0.00005	62.379-/+2.36
Plough	Random Forest	60	1000	Default	58.788-/+4.04

Compare with original accuracy, we improve a litter bit using feature selection method

Word	Original Accuracy	Final Accuracy	Improved (%) – Base on average
Ally	62.35-/+6.15	65.563-/+6.52	5%
Cry	61.595-/+3.08	62.379-/+2.36	1%
Plough	50.019-/+3.56	58.788-/+4.04	18%

It's clearly that, in the context of applying feature ranking algorithms to reduce feature vectors, **random forest weighted algorithms** is really good. It makes the best accuracy for all words we chose. However, I expected feature reduction to do more for word Ally and Cry.

TEST WITH HIDDEN DATA.

TASK A

Run the SVM Gaussian machine learning algorithms for the hidden test data with the best estimated parameters from part A estimation.

Order	Verb	Training Data Accuracy	Hidden Test Accuracy
1.	ally	62.35/+-6.15	70
2.	arrive	70.422/+-4.3	64
3.	cry	61.595/+-3.08	34
4.	halt	86.024/+-6.49	80
5.	Plough	50.019/+-3.56	28
6.	Submit	82.056/+-5.74	88

The other verbs are inside the interval except for “cry” and “plough”. **It’s really strange since the result in hidden test data is completely different from training data**. If the problem stem from the model, let’s try another one.

Order	Verb	Training Data Accuracy	Hidden Test Accuracy
1.	ally	0.553-/+0.05	0.54
2.	arrive	0.675-/+0.03	0.62
3.	cry	0.537-/+0.05	0.3
4.	halt	0.829-/+0.05	0.82
5.	Plough	0.459-/+0.09	0.18
6.	Submit	0.85-/+0.04	0.84

DECISION TREE - WITH HIDDEN TEST DATA

Order	Verb	Training Data Accuracy	Hidden Test Accuracy
1.	ally	0.577-/+0.09	0.6
2.	arrive	0.675-/+0.02	0.64
3.	cry	0.537-/+0.05	0.36
4.	halt	0.833-/+0.07	0.84
5.	Plough	0.39-/+0.05	0.24
6.	Submit	0.748-/+0.05	0.82

KNN - WITH HIDDEN TEST DATA

Once again the problem comes from “cry” and “plough” solely for both decision tree and KNN machine learning algorithms. So, the problem is not likely come from the model. As far as I concerned, it stems from data itself. Also, please notice that, both "cry" and "plough" are consider a hard classify word due to high number of word senses (19 senses). So, let's have a look at the hidden data.

Word	Category	Training Data	Hidden Test Data
Cry	1	131	19
	2	2	1
	3	0	1
	4	59	20
	5	0	1
	6	7	
	7	13	3
	8	4	
	10	1	
	12	1	
	13	1	
	16	4	
	17	5	2
	18	2	
	19	1	
	Ux	19	3
Plough	1	81	15
	3	13	2
	4	5	2
	5	17	5
	7	17	5
	8	18	4
	9	3	
	10	1	
	11	7	
	12	10	1
	13	2	1
	14	2	
	15	52	9
	16	3	
	17	2	1
	18	11	2
	19	2	
	ux	4	3

TASK B

Firstly, we would like to run our best model (SVM Gaussian after doing feature selection) to evaluate again hidden test data.

Order	Verb	Training Data Accuracy	Hidden Test Accuracy
1.	Ally	65.563-/+6.52	70
2.	Cry	62.379-/+2.36	34
3.	Plough	58.788-/+4.04	26

Again, “cry” and “plough” accuracy surprisingly low on hidden test data. **The underlying reasons must be data sparseness**. So, methods like cross validation or data bagging should help us.

Cross validation

We combine both training data and hidden test data into a single data, which will have 300 records. Then, we **run SVN Gaussian** cross validation (6 folds) on newly created data. Here’s the result.

Order	Verb	Training Data Accuracy	Newly Created Data
1.	Ally	65.563-/+6.52	64.667-/+3.25
2.	Cry	62.379-/+2.36	61.333-/+3.47
3.	Plough	58.788-/+4.04	52.667-/+6.22

Adding the hidden data to training data cause confidence interval to broaden, denoting that the hidden data is a kind of **“unusual”**.

Data Bagging

We again run data bagging on the hidden test data. And here’s the result.

Order	Verb	Training Data Accuracy	Hidden Test Data
1.	Ally	0.621951	0.56
2.	Cry	0.581301	0.54
3.	Plough	0.46748	0.5

My failure in this hidden test data is that, I should foresee the unstable of the system. Instead of choosing SVN Gaussian machine learning algorithms, **I should choose Data Bagging instead**.

REFERENCES

http://cran.r-project.org/web/packages/FSelector

http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R

http://en.wikipedia.org/wiki/Dimension_reduction