Verb-Sense Disambiguation

Septina Dian Larasati
NPFL054 2009/2010
Introduction to Machine Learning (in Computer Linguistic)
Final Project

Table of Contents

1. Project Description

This report delivers comparison of three machine learning method applied on one of Natural Language Processing tasks. The task is to select proper sense of the given verb in a given context or known as **Verb-sense disambiguation task**. For example English word '*read*' as a verb will have more than one sense, two among many others are (1) interpreting something that is written or printed and (2) to hear and understand. In this case the given context will disambiguate between the senses like in the sentence "*I read the book*" will apply to the (1) sense and the sentence "*I read you loud and clear*!" will apply to the (2) sense. Verb-sense disambiguation is a subtask of Word-sense disambiguation, where the subject to be disambiguated is not any words in general instead restricted to only verbs.

The automation on choosing the correct verbs proper sense in Czech has been done by (Semecký, 2008). That work also performed several different Machine learning methods on the experiment. The verbs that will be disambiguated in this experiment are two verbs in Czech. Those verbs are ***'přihlížet'*** and ***'odpovídat'***, which have two and three senses respectively. The three machine learning methods that applied to solve the task are **Decision Tree Learning (DT)**, **Support Vector Machines Learning (SVM)**, and **Instance-Based Learning** methods, which is **K-Nearest Neighbour (KNN)**.

The information of the context is given by sets of features of the verb. The more detailed description of the features will be found on Data Description section. The parameters that are adjusted during the tuning will be discussed later in Experiment section also containing brief results of the three the machine learning methods applied. The results will be delivered in accuracy and then compared to their baseline scores. Detail of the experiment result can be found in **Error! Reference source not found.** section. To wrap all the experiments, the overall work done and future improvement will be discussed in Conclusion section.

2. Data Description

The data that are used in this experiment is in the same form as the data used in (Semecký, 2008). The data are represented as vectors of atomic values hereinafter called features. The values of the features can be divided into two types of data; those are numerical and categorical values. The numerical values can be naturally ordered in spite of whether they are continuous or discrete, e.g. number of animate substantives in the sentence. While the categorical values are without any natural order, e.g. part of speech of the verb, although they can be also expressed by number, e.g. Czech word cases in number 1, 2, 3, etc. Boolean or binary value is special categorical value where they have basically two values, `true` and `false`.

As in (Semecký, 2008), the data is divided into five different types of features. Those types of features are:

- **Morphological** – morphological information about the lemmas of the current verb and two preceding and following words.
  For each word, there are 15 features taken from 15 positions of the morphological tags that are based of Czech positional morphology. Among those 15 features, 3 features are ignored, which makes it 60 features in total.
- **Syntax-Based** – information taken from automatic syntactic parser.
  This type of features groups as in the following:
  - Reflexive *se* (1 feature)
  - Reflexive *si* (1 feature)
  - Subordinate verb (1 feature)
  - Superordinated verb (1 feature)
  - Subordinating conjunctions (37 features)
  - Substantive cases (7 features)
  - Adjective cases (7 features)
  - Prepositional cases (7 features + 69 lexicalized prepositional phrase features)
- **Idiomatic** – information of idiomatic expression that occurs in the sentence. (181 features)
- **Animacy** – information about the animacy that occurs in the sentence. (18 features with 4 features ignored)
- **WordNet** – information taken from WordNet top-ontology classes. (128 features)

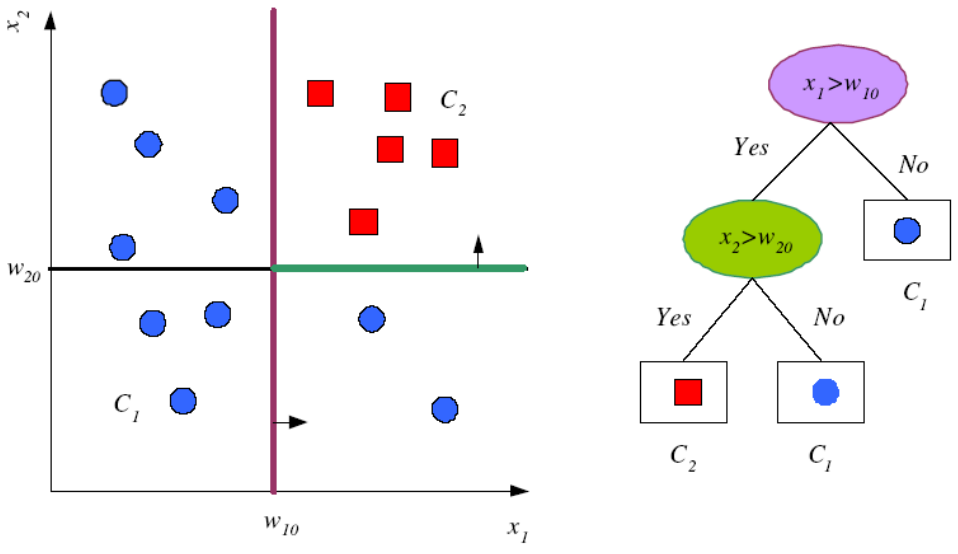
All the features are in categorical values, or more specifically in boolean values, except some the Anymacy type of features. The booleanized version of the Morphological features are also provided, which it has the number of feature blown up into 705 features.

3. Machine Learning Methods

The classification of the possible senses of the verbs is made by applying three machine learning methods. Those methods are Decision Tree Learning, Support Vector Machines Learning, and Instance-Based Learning methods. The theoretical background for each method will be explained in the next subsections.

3.1 Decision Tree Learning

Decision Tree Learning is one of machine learning methods that use Decision Tree as the learned discrete-valued function representation to predict its value. Decision Tree is a tree that composed of internal decision nodes (including a root node) and terminal nodes. Each internal decision node t is a test function $f_t(x)$ of an attribute x with discrete outcomes labeling the edges. The functions divide the input space into small regions.



(Credits: Ethem Alpaydin)

Figure 1. Decision Tree

Each path in the Decision Tree corresponds to conjunctions of attributes tests where the whole tree will corresponds to disjunction of conjunctions of the attribute tests.

This Decision Tree Learning method suitable for problems with the following characteristics:

- Instances in attribute-value pairs
- Discrete target function values
- Require disjunctive descriptions
- The training data that may contain errors
- The training data that may contain missing attribute values

One of the basic Decision Tree Learning algorithms is the ID3 algorithm. This algorithm constructs the tree in top-down manner where the attributes that best classifies will be sorted top-down from the root node. The attributes are selected statistically according to the training data. Here are two splitting criteria to measures the attributes, the Information Gain and Gini Index. Here are the formulas for the Information Gain criteria.

$$Gain(D, A) = H(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} H(D_v)$$

D	=	Set of training example
H	=	Entropy
A	=	Attribute/Feature
D_v	=	Subset of D of which it has v as a value of the given attribute/feature (A)

Figure 2. Information Gain

$$H(D) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

D	=	Set of training example
H	=	Entropy

p_i = Probability of the given class in D

Calculation of $0 \log_2 0$ will be 0.

Figure 3. Entropy

Here is the formula for Gini Index

$$Gini(A) = 1 - \sum_{i=1}^{k} \left(p(c_i|t)\right)^2 - \sum_{i=1}^{n} p(t_i) \sum_{j=1}^{k} p\left(c_j|t_i\right)\left(1 - p\left(c_j|t_i\right)\right)$$

A	=	Attribute/Feature
$p(c_i\|t)$	=	Probability of assigning an example to class c_i in node t
$p(t_i)$	=	Probability node t_i
$p(c_i\|t_i)$	=	Probability of assigning an example to class c_i in node t_i

Figure 4. Gini Index

3.2 Support Vector Machines Learning

The idea of Support Vector Machines Learning is to separate the data with a hyperplane and extend it to non-linear decision boundaries. The general equation of the hyperplane, where x is a point vector and w is the corresponding weight vector, is

$$f(x) = \langle w \cdot x \rangle + b$$

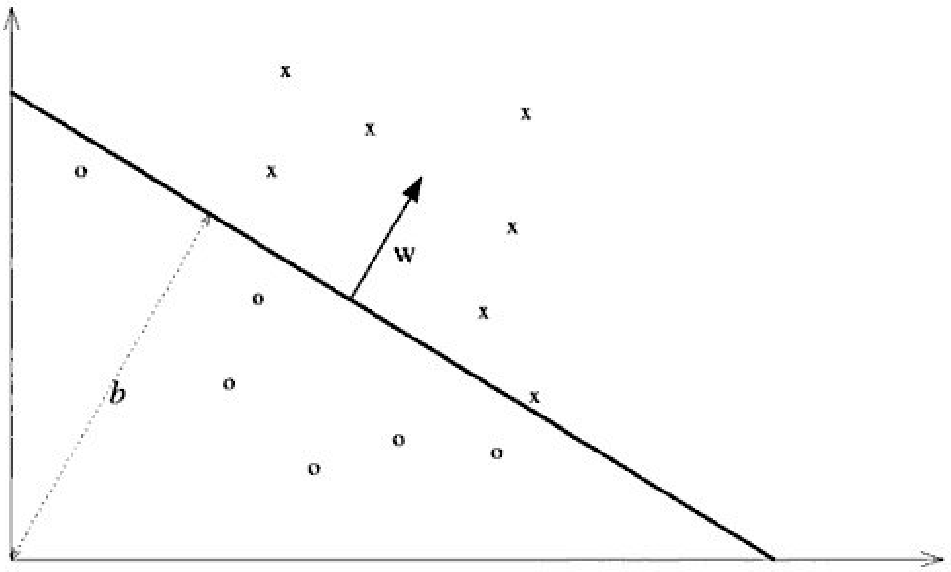


Figure 5. Linear Classifier

If the data is separable, the linear classifier is the linear line that separates the points. The closest points beside the separating line are called the support vectors. To measure the margin, two parallel lines to the separating lines are drawn passing through the support vectors. The margin is the distance between those two parallel lines. The best linear classifier is the one with the maximum margin (Maximum margin linear classifier). With *w* and b defined, the separating hyperplane is defined.

A slack variable are introduced when the data are linearly non separable and with this the outliers are captured. When the data are separable by a nonlinear region, SVM use a kernel function that maps the data to a higher dimensional space of features, rather than fitting the line to non linear curve.

As the line separating the space into two different regions, this method performs well in binary classification. But, it is also possible to apply it to multi-class classification.

3.3 Instance-Based Learning

Instance-Based Learning methods is an approach to approximating a real-valued or discrete-valued target function by storing the training data and retrieve similar related instances from the memory every new query instances encountered to be classified. By this, different target functions can be constructed for each distinct query. The disadvantages are the cost of classifying new instances can be high because the computation takes place at the classification and typically considering all the attributes of instances to retrieve similar training examples.

The Instance-Based Learning that is being used in the experiment in this report is K-Nearest Neighbour Learning. The distance metric in this method is Euclidean distance and it will consider K-Nearest Neighbour with no weighting function. Below shown how this method fitting the local points.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

Figure 6. Euclidean Distance

For discrete function

$$\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

For continuous-valued function

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=i}^{k} f(x_i)}{k}$$

Where $\delta(a, b) = 1$ if a = b, and 0 otherwise.

$\hat{f}$ = target function approximation

f = target function

$x_i, \ldots, x_j$ = nearest examples according to distance metric to x_q.

Figure 7. Approximating Target Function

The classification is done by voting on the nearest neighbouring class of the instance to be classified.

4. Experiment

This chapter will give the detailed descriptions about the experiments on disambiguating the word senses of the words ***'přihlížet'*** and ***'odpovídat'***. The feature selection of the data will be explained first, followed by the architecture of the system implementation, and then moving further to the descriptions of baseline set up and several settings of the tuning runs.

4.1 Data Preparation

The context information is provided as sets of features to disambiguate the verbs given. Hereinafter *'přihlížet'* and *'odpovídat'* will be referred as Verb 1 and Verb 2 respectively. These sets of features are divided by their type. The given Table 1. Set of Features are the descriptions of the set of features and the target feature.

The features that have more than one value for the whole training data will be selected for the tuning. Since the morphological feature has two different form, *m* and *n*, the training set will be divided into three training sets, those sets are sets with

- all selected features
- all selected feature excluding *m*
- all selected feature excluding *n*

Table 1. Set of Features

Type	Number of Features			Description
	Initial	Selected for Verb 1	Selected for Verb 2	
m	75	51	55	Morphological features
n	705	576	334	Morphological features in Boolean values
s	131	17	18	Syntax-Based features
i	118	3	1	Idiomatic features
a	18	11	13	Animacy features
w	128	0	0	WordNet features
Total	**1175**	**658**	**421**	
x	1	1	1	Target feature: Verb frame feature

The values of the features are represented in `.data` files and the descriptions of the features are represented in `.names` files. The data are divided into training and test data. Here are the statistics of the training and test data for both of the verbs.

Table 2. Training and Test Data Statistics

	Number of instances		Number of Possible Senses
	Training set	Test set	
Verb1	66	33	2
Verb2	65	32	3

	Verb 1		Verb 2		
Classes	**1**	**2**	**1**	**2**	**3**
Number of instances in the training data	36	30	21	4	40
Number of instances in the test data	18	15	10	2	20

In SVM and KNN the categorical valued features, which also includes boolean valued features, are sorted and then mapped into a numerical sequence.

4.2 System Architecture

The implementation is done in R system software running on Windows operating system. The Decision Tree Learning experiments are using the **'`rpart`' package** while the Support Vector Machines Learning and K-Nearest Neighbour Learning experiments are using **'`e1071`' package**. The experiments are done by first drawing the baseline score, running the default setting, and then changing the parameters to find the best scores that can be achieved.

The evaluations of the experiments are measured in a simple accuracy metric as below,

$$accuracy = \frac{number\ of\ correctly\ classified\ instances}{number\ of\ instances}$$

Figure 8. Accuracy Equation

Not all the features that were initially provided are used in the tuning. The features are selected in a manner that mentioned previously in Data Preparation section. The following tuning runs descriptions are using the data with selected features.

4.3 Baseline Setting

In the beginning of the experiment the Baseline scores are set for both Verbs. The baseline is computed by doing a random guess based on the probability of target values in the training data. The random guesses are generated based on the training data distribution and repeated 1000 times. Here is the result along with 95% confidence level of two-tailed confidence interval.

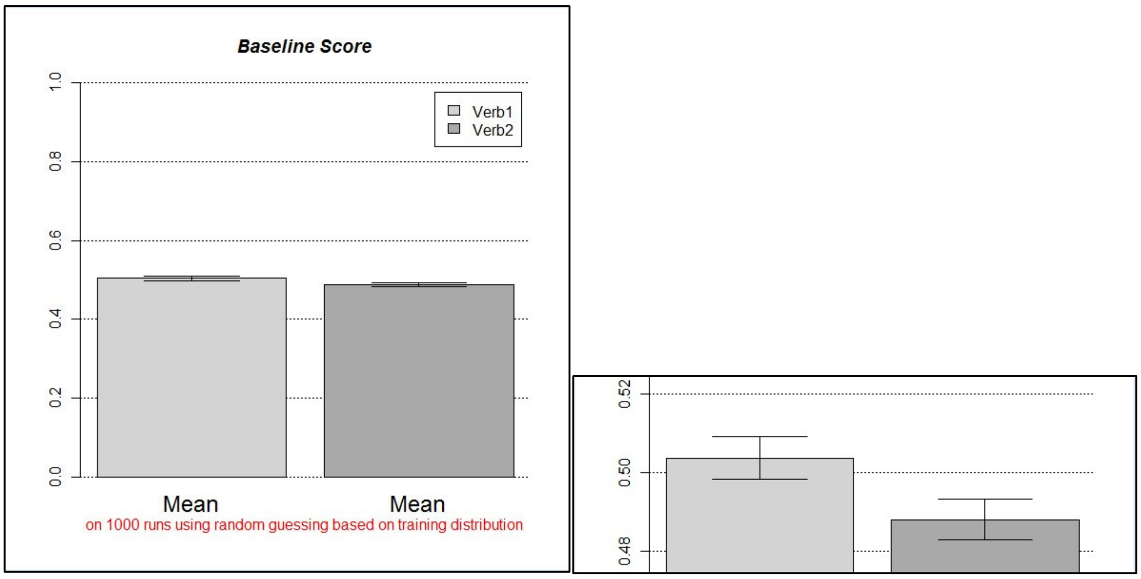


Figure 9. Baseline Score:
Generated by random guessing based on training data distribution

4.4 Default Setting

The default scores are made by defining the data on the method's functions and running it with default parameters setting. Given further is the description for the parameters for each method, that are tuned on the experiments, along with theirs default values.

4.4.1 Decision Tree Learning

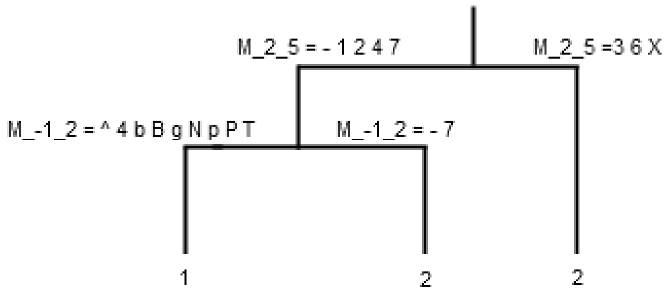
Given below are the default parameters and their default value if available.

```
rpart(formula, data, weights, subset, na.action = na.rpart, method, model = FALSE, x =
     FALSE, y = TRUE, parms, control, cost, ...)
rpart.control(minsplit=20, minbucket=round(minsplit/3), cp=0.01, maxcompete=4,
     maxsurrogate=5, usesurrogate=2, xval=10, surrogatestyle=0, maxdepth=30, ...)
```

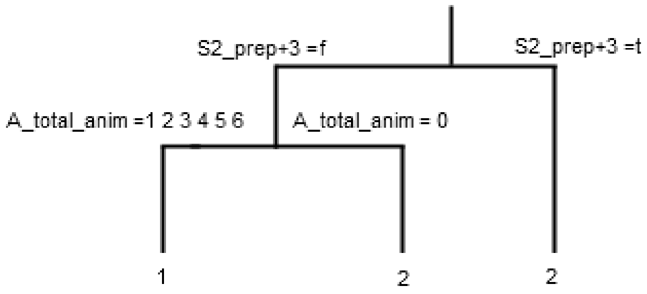
Given below the simplified parameters descriptions as in `rpart` package help for those features that are tuned during the experiment

- **`minsplit`** – The minimum number of observations that must exist in a node, in order for a split to be attempted. The default value is 20.
- **`cp`** – Complexity Parameter. The default value is 0.01.
- **`split`** – The splitting index that can have `information` or `gini` as its value. The default value is `gini`.

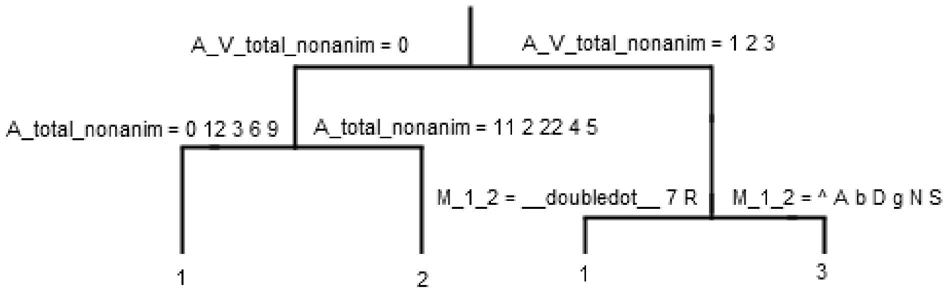
Then the `rpart` function is run in default setting with different set of features as mentioned in Data Preparation section. By looking at the tree results, we can see that only few of the features are more important than the other features available. As seen on the tree, most of those few features are coming from the morphological (m) features set, which show that this set stands out among any other features sets. The resulting trees in three different set of features for both verbs are given.



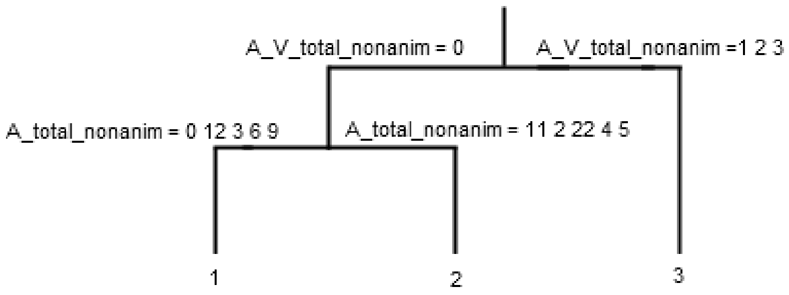
(1)



(2)


A_V_total_nonanim = 0
A_V_total_nonanim = 1 2 3
A_total_nonanim = 0 12 3 6 9
A_total_nonanim = 11 2 22 4 5
M_1_2 = __doubledot__ 7 R
M_1_2 = ^ A b D g N S
1
2
1
3

(3)



(4)

Figure 10. Verb 1 Baseline Tree, (1) with all selected features and all selected features excluding *n*; (2) with all selected features excluding *m*; Verb 2 Baseline Tree, (3) with all selected features and all selected features excluding *n*; (4) with all selected features excluding *m*

4.4.2 Support Vector Machine Learning

Given below are the default parameters and their default value if available.

```
svm(formula, data = NULL, ..., subset, na.action = na.omit, scale = TRUE)
svm(x, y = NULL, scale = TRUE, type = NULL, kernel = "radial", degree = 3, gamma = if
      (is.vector(x)) 1 else 1 / ncol(x), coef0 = 0, cost = 1, nu = 0.5, class.weights
      = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1, shrinking = TRUE,
      cross = 0, probability = FALSE, fitted = TRUE, ..., subset, na.action =
      na.omit)
```

Given below the parameters simplified descriptions as in `e1071` package help for those features that are tuned during the experiment

- **`kernel`** – The kernel used in training and predicting. The default value is `radial`.
- **`gamma`** – Parameter needed for all kernels except `linear`. The default value is 1/(data dimension).

All the features' values are sorted and mapped into a numerical sequence. Then the `svm` function is run in default setting with different set of features as mentioned in Data Preparation section. Here are the settings for the baseline:

- all selected features

```
Verb 1 Parameters:
        SVM-Type:  eps-regression
      SVM-Kernel:  radial
            cost:  1
           gamma:  0.001519757
         epsilon:  0.1
     Number of Support Vectors:  63
```

```
Verb 2 Parameters:
        SVM-Type:  eps-regression
      SVM-Kernel:  radial
            cost:  1
           gamma:  0.002375297
         epsilon:  0.1
     Number of Support Vectors:  56
```

- all selected feature excluding *m*

```
Verb 1 Parameters:
        SVM-Type:  eps-regression
      SVM-Kernel:  radial
            cost:  1
           gamma:  0.001647446
         epsilon:  0.1
     Number of Support Vectors:  64
```

```
Verb 2 Parameters:
        SVM-Type:  eps-regression
      SVM-Kernel:  radial
            cost:  1
           gamma:  0.002732240
         epsilon:  0.1
     Number of Support Vectors:  54
```

- all selected feature excluding *n*

```
Verb 1 Parameters:
        SVM-Type:  eps-regression
      SVM-Kernel:  radial
            cost:  1
           gamma:  0.01219512
         epsilon:  0.1
     Number of Support Vectors:  55
```

```
Verb 2 Parameters:
        SVM-Type:  eps-regression
      SVM-Kernel:  radial
            cost:  1
           gamma:  0.01149425
         epsilon:  0.1
     Number of Support Vectors:  56
```

4.4.3 K-Nearest Neighbour Learning

Given below are the default parameters and their default value if available.

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

Given below the parameters simplified descriptions as in `e1071` package help for those features that are tuned during the experiment

- **`k`** – Number of neighbours considered. The default value is 1.
- **`prob`** – If this is true, the proportion of the votes for the winning class are returned as attribute `prob`. The default value is `FALSE`.
- **`use.all`** – Controls handling of ties. If true, all distances equal to the `k`th largest are included. If false, a random selection of distances equal to the `k`th is chosen to use exactly `k` neighbours. The default value is `TRUE`.

4.5 Tuning

To get the best accuracy, all the parameters are adjusted during the tuning. After building some models, only few parameters are affecting the accuracy. Provided below are the parameters for each method applied that are adjusted during the tuning that have some effect on the accuracy and each default value of the parameters

Table 3. DT Accuracy Table:
with different values on the *minsplit* and *cp* in different feature sets[1]

Verb 1					
minsplit	**cp**	**split**	**accuracy**		
			all features	**all features without *n***	**all features without *m***
default	default	gini	0.91	0.91	0.88
default	0.037	gini	0.91	0.91	0.88
default	0.037	information	0.91	0.91	0.88
16	default	information	0.91	0.91	0.88
16	0.037	information	**0.91**	**0.91**	**0.88**
Verb 2					
default	default	gini	0.5	0.5	0.66
default	0.02	gini	0.5	0.5	0.66
default	0.21	information	0.781	0.78125	0.875
16	default	information	0.875	0.875	0.875
16	0.035	information	**0.875**	**0.875**	**0.875**

The resulting trees are as below, where the important decision features for the classification are shown,

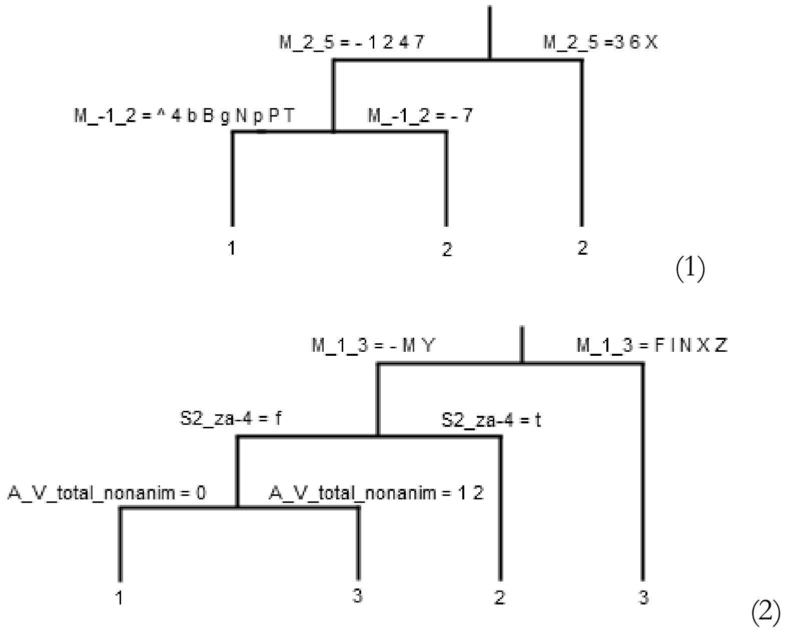


Figure 11. The best resulting decision tree after doing the tuning for (1) Verb 1 and (2) Verb 2

[1]The numbers in bold are some of the best scores achieved

Table 4. SVM Accuracy Table: with different values on the *kernel* and *gamma* in different feature sets[1]

Verb 1				
kernel	**i in gamma = 1/i*dim**	**accuracy**		
		all features	**all features without *n***	**all features without *m***
linear	default	0.273	0.636	0.091
polynomial	default	**0.636**	0.727	**0.636**
sigmoid	1	0.333	0.848	0.182
sigmoid	2	0.364	0.939	0.182
sigmoid	3	0.394	**0.97**	0.242
sigmoid	4	0.455	**0.97**	0.303
sigmoid	5	0.515	**0.97**	0.424
polynomial	1	**0.636**	0.727	**0.636**
polynomial	2	0.606	0.545	0.606
polynomial	3	0.606	0.545	0.606
polynomial	4	0.545	0.545	0.606
polynomial	5	0.545	0.545	0.545
Verb 2				
linear	default	0.25	0.59375	0.40625
polynomial	default	0.625	0.625	**0.69**
sigmoid	1	**0.66**	0.65625	0.625
sigmoid	2	**0.66**	0.65625	0.65625
sigmoid	3	0.625	**0.72**	0.625
sigmoid	4	0.625	0.6875	0.625
sigmoid	5	0.625	0.6875	0.625
polynomial	1	0.625	0.625	**0.69**
polynomial	2	0.625	0.625	**0.69**
polynomial	3	0.625	0.625	0.625
polynomial	4	0.625	0.625	0.625
polynomial	5	0.625	0.625	0.625

[1]The numbers in bold are some of the best scores achieved

Table 5. KNN Accuracy Table: with different values on the *k, prob,* and *use.all* in different feature sets[1]

Verb 1					
k	**prob**	**use.all**	**accuracy**		
			all features	**all features without *n***	**all features without *m***
1	default	default	0.85	0.79	0.79
2	default	default	0.79	0.79	**0.85**
3	default	default	**0.88**	**0.85**	**0.85**
4	default	default	0.67	0.76	0.79
5	default	default	0.76	0.79	0.7
6	default	default	0.76	0.76	0.67
7	default	default	0.76	0.79	0.7
3	FALSE	FALSE	**0.88**	**0.85**	0.79
3	TRUE	FALSE	**0.88**	**0.85**	0.79
3	FALSE	TRUE	**0.88**	**0.85**	0.79
3	TRUE	TRUE	**0.88**	**0.85**	0.76
Verb 2					
1	default	default	0.625	0.6	**0.875**
2	default	default	0.72	**0.75**	0.84
3	default	default	0.72	0.72	0.84
4	default	default	0.75	0.69	0.81
5	default	default	0.75	0.69	0.81
6	default	default	0.66	0.69	0.84
4	FALSE	FALSE	0.66	0.625	
4	TRUE	FALSE	**0.78**	0.656	
4	FALSE	TRUE	**0.78**	0.72	
4	TRUE	TRUE	0.72	0.66	
2	FALSE	FALSE		0.625	
2	TRUE	FALSE		0.66	
2	FALSE	TRUE		**0.75**	
2	TRUE	TRUE		0.66	
1	FALSE	FALSE			**0.875**
1	TRUE	FALSE			**0.875**
1	FALSE	TRUE			**0.875**
1	TRUE	TRUE			0.84

The parameters descriptions are explained in 4.4.2 Support Vector Machine Learning. On the gamma parameter, which the default value is 1/(data dimension), its denominator is tuned by a number that multiplies the data dimension and further will be known as *i*. The whole process is implemented in R and the models with the adjusted parameters are calculated and accuracies are reported.

5. Evaluation

The best accuracies with the corresponding adjusted parameters are highlighted in bold in the previous chapter tables. More detailed comparisons between the correctly and incorrectly classified instances are given below.

Figure 12. Verb 1 Detailed Comparison

DT		
result / data	**1**	**2**
1	**17**	1
2	2	**13**

SVM		
result / data	**1**	**2**
1	**18**	0
2	1	**14**

KNN		
result / data	**1**	**2**
1	**15**	3
2	1	**14**

Figure 13. Verb 2 Detailed Comparison

DT			
result / data	**1**	**2**	**3**
1	**8**	0	2
2	1	**1**	0
3	1	0	**19**

SVM			
result / data	**1**	**2**	**3**
1	**1**	7	2
2	0	**2**	0
3	0	1	**19**

KNN			
result / data	**1**	**2**	**3**
1	**9**	0	1
2	1	**0**	1
3	1	0	**19**

As seen in the overall accuracy graph, most of the parameters adjustments for all classifier are successfully improve the accuracy scores compare to the baseline score. The percentages of the improvements are relatives to the number instance in the test set. For example the 21.9% improvement on Verb 2 evaluation tuned on DT methods means that the tuned version has 7 instances more correctly classified that can be different instances.

In Verb 1 case, the SVM classifier almost classifies all the test data perfectly except one misclassification. This may happen because SVM works perfectly in binary classification, which the case of Verb 1. While in Verb 2 case, the DT and KNN classifier performs better. The KNN may fail classifies the test instances because the proportion of the instance's classes in the training data are uneven and mostly it votes more for one of the classes ('3' class). This classifier fails to classify '2' class where the training instances with that class only occurs four times while the k parameters for the voting are set to three.

5.1 Statistical Evaluation

Hypothesis testing is conducted to see the equality of the performance in terms of accuracy between two methods. The hypothesis testing is conducted in pair wise fashion for every pair in these three methods:

Decision Tree: The parameters setting: split = information, minsplit = 16, minbucket = 4, cp = 0.037.

SVM: The parameters setting: kernel = sigmoid, gamma = 0.0003

KNN: The parameters setting: k = 3, prob = FALSE, use.all = TRUE

Test Sets – the test sets for the hypothesis testing are taken randomly from the original test set provided for the task. In this hypothesis testing there are 30 different test sets, where each set contains 20 instances. The instances are taken randomly with replacement from the original test set.

Evaluation Metric – The metric use for measuring the performance is accuracy as mentioned in previous chapters.

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{var_1}{n_1} + \frac{var_2}{n_2}}}$$

Figure 14. t-value formula

$$d.o.f = N - 2 = n_1 + n_2 - 2$$

Figure 15. Degree of freedom

Alpha – The alpha is 0.05

Degree of freedom – The degree of freedom is 58

t-alpha – $t_{0.05} = 2.00$

Hypotheses – the testing are conducted in pair wise fashion for every method pair. Every pair has their own corresponding null and alternate hypothesis. For each method pair here is the hypotheses in general term:

- **H_0** : The two methods under question performances are equal based on the results given
- **H_1** : There are differences among the performance of the two methods under question based on the results given

Table 6. Methods performances on the test set
Every row shows the performance of the methods on a single test set based on accuracy. There are 30 different test sets comprise of 20 instances each.

Test Set	**Verb 1**			**Verb 2**		
	DT	**SVM**	**KNN**	**DT**	**SVM**	**KNN**
1	0.70	1.00	0.70	0.90	0.80	0.90
2	0.55	0.90	0.85	1.00	0.75	0.90
3	0.75	1.00	0.80	0.70	0.55	0.70
4	0.85	1.00	0.85	1.00	0.75	1.00
5	0.45	1.00	0.90	0.90	0.45	0.85
6	0.85	1.00	0.75	0.80	0.55	0.85
7	0.30	0.95	0.80	0.95	0.80	0.80
8	0.45	1.00	0.80	0.85	0.80	0.80
9	1.00	1.00	0.95	1.00	0.65	1.00
10	0.70	0.95	0.85	0.80	0.65	0.70
11	1.00	1.00	0.90	0.95	0.65	0.90
12	0.65	0.95	0.55	0.90	0.80	0.85
13	0.90	1.00	0.75	0.95	0.90	0.95
14	0.95	1.00	0.80	0.75	0.75	0.80
15	0.80	0.95	0.75	0.95	0.65	0.95
16	0.70	0.90	0.80	0.85	0.75	0.90
17	1.00	1.00	0.95	0.80	0.60	0.85
18	0.45	1.00	0.75	0.90	0.60	0.90
19	0.95	1.00	0.85	1.00	0.75	0.85
20	0.65	1.00	0.85	0.90	0.65	0.90
21	0.95	0.95	0.80	0.90	0.80	0.95
22	0.80	0.90	0.75	0.90	0.65	0.85
23	0.95	1.00	0.90	0.85	0.70	0.85
24	0.55	1.00	0.80	0.90	0.75	0.75

25	0.95	1.00	0.85	0.85	0.55	0.80
26	0.95	1.00	0.90	0.95	0.85	0.80
27	0.60	0.95	0.75	0.80	0.75	0.75
28	0.75	1.00	0.80	0.95	0.80	0.80
29	0.85	1.00	0.80	1.00	0.80	0.95
30	0.40	1.00	0.75	0.85	0.85	0.90
$\bar{X}$	0.75	0.98	0.81	0.89	0.71	0.86
var	0.42	0.001	0.006	0.006	0.01	0.006
SD	0.20	0.034	0.08	0.078	0.11	0.08
n	30	30	30	30	30	30

Table 7. t-value for every pair of methods

	Verb 1					**Verb 2**			
d.o.f = 58	*t*	**DT**	**SVM**	**KNN**		*t*	**DT**	**SVM**	**KNN**
alpha = 0.05	**DT**	-	6.17	1.58		**DT**	-	7.49	1.72
$t_{0.05}$ = 2.00	**SVM**		-	10.69		**SVM**		-	6.04
	KNN			-		**KNN**			-

As seen on the t-values, only pair of DT and KNN that is not rejecting the null hypothesis, while the other pairs reject it. Based on the results given by hypothesis testing, DT's and KNN's performances are equal.

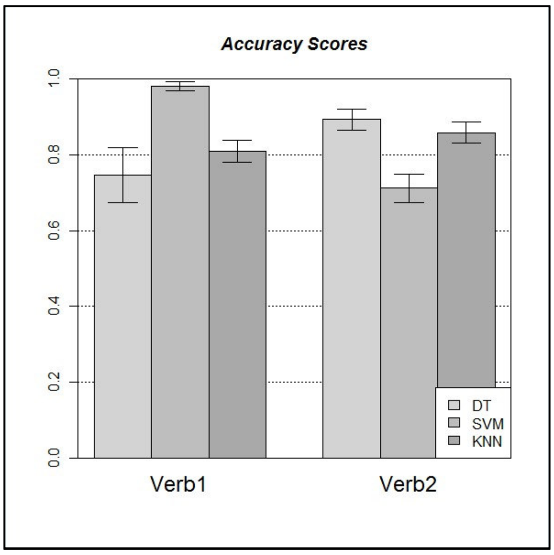


Figure 16. Hypothesis Testing - Comparison between the three methods in terms of Accuracy.

6. Conclusion

The three machine learning methods that have been applied gives inconsistent performance based on the results between the tasks of verbs. Statistically DT and KNN perform equally on the tasks given as to compare to SVM. SVM performs better and almost perfect in verb 1 task with average accuracy of 0.98, while SVM performs the opposite in the verb 2 task with average accuracy of 0.71. Based on the result, SVM is preferred for Verb 1 task, while in Verb 2 task DT and KNN, which are perform equally, are preferred.

Among the big numbers of the features that are given to solve the task, there are few crucial features that can be used in DT which gives the same performance as if using bigger number of features. These features are *S2_prep+3* and *A_total_anim* for Verb 1 task and *A_V_total_nonanim*, *A_total_nonanim*, and *M_1_2* for Verb 2 task.

Based on the experiments conducted, the default setting in Decision Tree already gives a very good performance, and if the cp is tuned it will give a slightly better performance.

For the future work it might be interesting is to try is to apply the same method with the same set of features with tasks in different languages. It may show how the methods work language independently or the translation from other languages may or may not help the improvement cross language.

Bibliography

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines: and Other Kernel-based Learning Methods*. Cambridge: Cambridge University Press.

Mitchell, T. M. (1997). *Machine Learning*. New York: WCB McGraw-Hill.

Semecký, J. (2008). *Verb Valency Frames Disambiguation*. Prague: Institue of Formal and Applied Linguistics, Charles University in Prague.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory*. Springer.

Venables, W., & Ripley, B. D. (2002). *Modern Applied Statistics with S*. Birkhäuser.

Venables, W., & Ripley, B. (2000). *S Programming*. Springer.

http://ufal.mff.cuni.cz/~hladka/ML/LECTURES/jsmath/test/support-vector-machines.html

http://cran.at.r-project.org/web/packages/rpart/rpart.pdf

http://roadrunner.cancer.med.umich.edu/comp/docs/R/rpart.pdf

http://sirad.pd.infn.it/glast/ground_sw/dc2/code/rpart.pdf

Table of Figures

***