PFL054 Term project
2011/2012

Semantic Pattern Classification

Ema Krejčová

1 Introduction

The aim of the project is to construct classifiers which should assign semantic patterns to six given verbs, as accurately as possible. Each verb has a set of usual usage patterns (based on the Pattern Dictionary of English Verbs), defined by its semantic and syntactic characteristics. It is assumed that if we are able to automatically assign these patterns, it will help us in the general problem of word sense disambiguation which is vital in many fields of computational linguistics.

1.1 The data

The data obtained for this task consist of six text files, one for each verb. Each file contains 250 sentences with the verb, extracted from BNC, with manually annotated pattern tag. In each sentence, the verb in question is highlighted by angle brackets. For each sentence, also a part-of-speech tag list and a dependency structure (both automatically assigned) are included in the file. There is also a named entity list, provided by a Named Entity Recognizer, but this list was not used in this task.

The verbs in question are *ally* (with 6 semantic patterns), *arrive* (6 patterns), *cry* (18 patterns), *halt* (3 patterns), *plough* (17 patterns), and *submit* (5 patterns).

These data represent the training data for the construction of the classifiers. There are 300 more sentences (50 for each verb) for testing the classifiers, but these were not available for the development process, they will be only used for testing and evaluation afterwards.

1.2 Data preprocessing

First what had to be done was to convert the given sentences into feature vectors which could be then processed by the classifiers. For this I created a Perl script which took as its input the text file `{verb}.txt` and outputted a list of feature vectors.

The default set of 275 features[1] was given by the assignment (see Appendix A). Apart from morphosyntactic features that were extracted directly from the POS tagging and dependency structures, there were also semantic features, based on 50 semantic classes extracted from WordNet, which were also given by the assignment.

Most of the features were binary, represented in the output feature list as values "n" and "y", and a few were categorical, represented by character strings. If needed, the feature script can convert these values to numeric.

The task was then divided in two steps:

[1]In fact, the assignment contained 283 features. But since two of the given semantic classes were empty and each semantic class was used to define four features, eight features would necessarily consist of constant zeroes, so I did not implement these eight features.

A. use the default feature set (or its subset), experiment with at least three classification methods, choose the one which seems best and tune it for each of the given verbs.

B. choose three of the six verbs and create new feature lists for them, possibly better than the default features used in step A. Then again, develop a model which will assign the patterns as best as possible.

Apart from the feature extraction from the data, the whole experiment was carried out in the R environment.

2 Baseline for the experiment

As a baseline for the experiment, with which the accuracy of the developed models with be compared, we will use the accuracy of a simple classifier, which assigns to each verb its most frequent pattern tag in the training data. The distribution of patterns differs for different verbs – sometimes there is one pattern which is much more frequent than the others, so even this simple classifier is quite successful.

To get a single number for all six verbs, we will use a weighted average, with weights of verbs defined by their frequency in the BNC. The formula is the following:

$$\frac{\sum_v p_v * A_v}{\sum_v p_v}$$

where p_v represents the relative frequency of the verb v, and A_v the accuracy of the corresponding classifier. The frequencies are shown in Table 1.

p_{ally}	0.0083%
p_{arrive}	0.1307%
p_{cry}	0.0257%
p_{halt}	0.0183%
p_{plough}	0.0076%
p_{submit}	0.0483%
$\sum_v p_v$	0.2389%

Table 1: Relative frequencies of the verbs

The baseline accuracy computed on all 250 training instances is shown in Table 2.

ally	47.6%
arrive	68%
cry	52.4%
halt	83.6%
plough	32.4%
submit	70.8%
weighted average	**66.2%**

Table 2: Baseline accuracy

3 Task A

For this task, I divided the development data for each verb into two parts – 30 sentences were randomly selected as development test data, the rest was used as development train data.

I experimented with three classification methods – Naive Bayes, Decision Trees and Support Vector Machines. For each method, first a specific baseline was computed, which is the accuracy of the classifier with default parameters (i.e. default in the R system) and the full feature set. Next, the tuning process for each method and each verb was the following:

1. from the default set, remove features which have just a single value or have two values, but one of them appears less than four times
2. from the remaining features, select those which improve the accuracy most
3. tune the method's parameters
4. evaluate the accuracy – by 5-fold cross-validation on the development train data, with a 95% confidence interval. Just for comparison, also the accuracy on development test data was computed.

After the first step, the feature set was reduced significantly:

	ally	arrive	cry	halt	plough	submit
# of features	98	104	111	122	129	107

I decided to select the features first and then tune the parameters, because it seemed plausible that the most useful features will be most useful regardless of the parameters of the method (that parameter tuning might improve absolute performance, but the relative contribution of the features should not change much).

In the following, the *accuracy on train set* refers to the average accuracy achieved by 5-fold cross-validation on the development train data (220 instances). The *accuracy on test set* refers to the accuracy on the development test data (the remaining 30 instances).

3.1 Naive Bayes

The baseline accuracy for NB (with the default feature set and default parameters implemented in the R system) is in Table 3:

The selection of a suitable subset of features was done by the greedy algorithm, i.e. starting with an empty set one feature at a time was tested (using 5-fold cross-validation) and the one which improved the performance most was added to the set. In the case of Naive Bayes, two features were taken at the beginning (i.e. all possible pairs were tested at the beginning), because the algorithm implemented in the R system requires at least two features. The selection process was stopped when the overall accuracy stopped increasing significantly, which was usually somewhere around 35 features or earlier (for a graphical representation see Figure 1).

All 220 instances from the development train set were used for training the model in each round.

verb	accuracy on train set	confidence interval	accuracy on test set
ally	58.2%	± 12.9%	60%
arrive	62.3%	± 12.4%	66.7%
cry	63.2%	± 3.7%	63.3%
halt	74.5%	± 3.7%	83.3%
plough	63.6%	± 6.9%	60%
submit	77.7%	± 6.1%	80%
weighted average	**66.3%**		69.8%

Table 3: Baseline accuracy for the Naive Bayes classifier

When the set of features was established, the `laplace` parameter of the method was tuned – values of 0,1,2, and 3 were tested, using 5-fold cross-validation. However, for all the verbs, the value of 0 came best.

Table 4 shows the optimized results:

verb	# of features	laplace	accuracy on train set	confidence interval	accuracy on test set
ally	32	0	68.2%	± 10.7%	70%
arrive	19	0	77.3%	± 7.2%	63.3%
cry	29	0	76.4%	± 6.5%	73.3%
halt	29	0	88.2%	± 5.4%	86.7%
plough	33	0	75%	± 8.5%	46.7%
submit	17	0	90%	± 6.5%	96.7%
weighted average			**80.2%**		72.6%

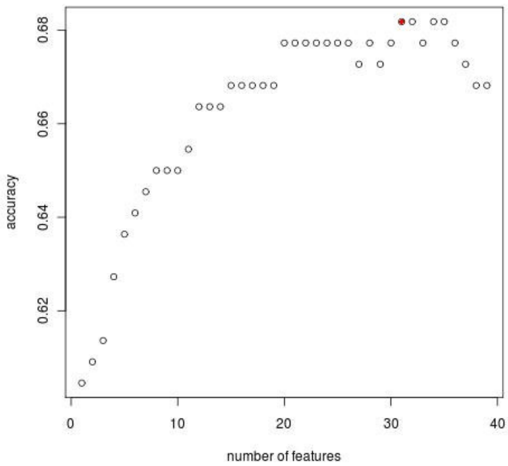
Table 4: Naive Bayes classifier – optimized results

For comparison, I tried also to tune the `laplace` parameter with all the features, with no prior selection. The results are in Table 5.

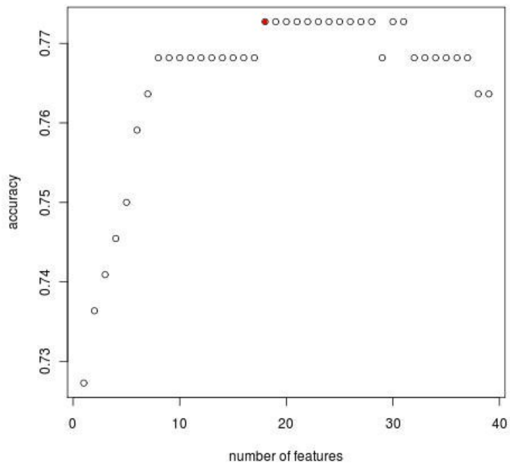
verb	# of features	laplace	accuracy on train set	confidence interval	accuracy on test set
ally	98	2	60%	± 10.1%	60%
arrive	104	3	68.2%	± 4.9%	76.7%
cry	111	0	63.2%	± 3.7%	73.3%
halt	122	3	81.8%	± 9.6%	83.3%
plough	129	0	63.6%	± 6.9%	56.7%
submit	107	2	81.4%	± 5.4%	90%
weighted average			**70.9%**		78.3%

Table 5: Naive Bayes classifier – tuned results, but no feature selection

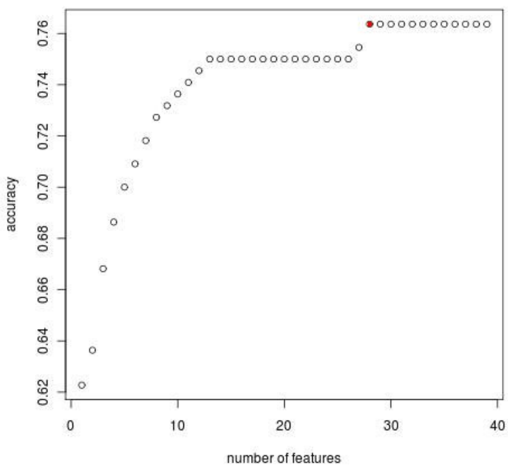
The results of the classifier were considerably better, when only some features were selected.



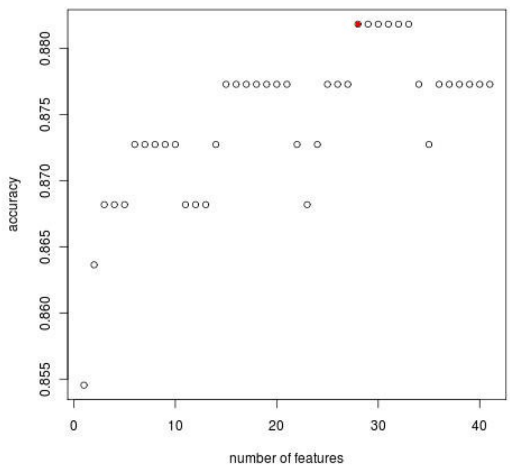
(a) The verb *ally*



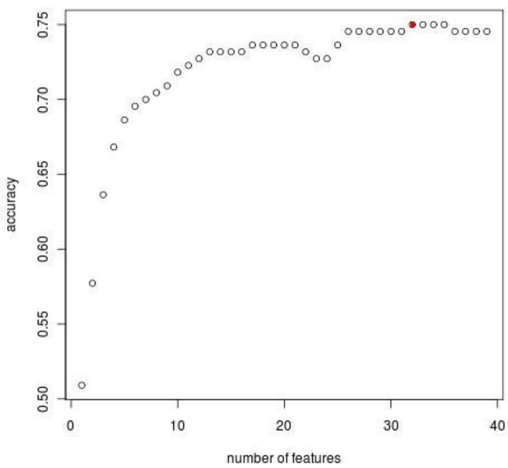
(b) The verb *arrive*



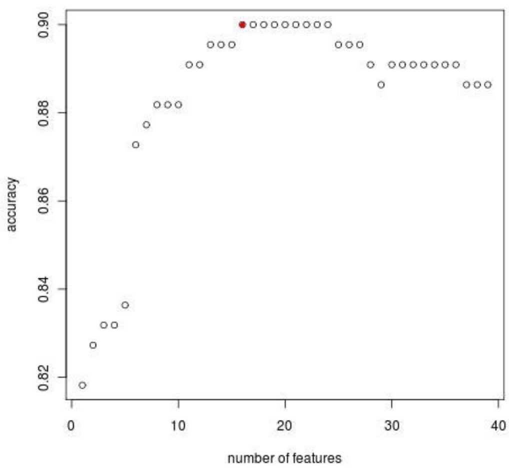
(c) The verb *cry*



(d) The verb *halt*



(e) The verb *plough*



(f) The verb *submit*

Figure 1: Feature selection. The dark dot shows where the selection was stopped.

3.2 Decision Trees

Baseline ("non-tuned") accuracy for the Decision Tree classifier is shown in Table 6:

verb	accuracy on train set	confidence interval	accuracy on test set
ally	55%	± 6.7%	63.3%
arrive	66.8%	± 9.1%	73.3%
cry	61.8%	± 7%	53.3%
halt	83.6%	± 9.4%	83.3%
plough	58.2%	± 10.9%	50%
submit	83.6%	± 5%	90%
weighted average	**70.3%**		74.2%

Table 6: Baseline accuracy for the Decision Tree classifier

With Decision Trees the greedy algorithm did not prove useful for feature selection. It chose a subset of features, but the performance with this subset was not better than the performance achieved when the choice was left to the `rpart` function implemented in R (i.e. when the function `rpart` was called with all the features and chose the features for splits by itself).

For the parameter tuning I used the R function `tune.rpart` and tuned the parameters `cp` (with values 0.1, 0.05, 0.01, 0.005, and 0.001) and `minsplit` (with values 2, 4, and 8). The final results are shown in Table 7, the resulting trees are in Figure 2.

verb	# of features	cp	minsplit	accuracy on train set	confidence interval	accuracy on test set
ally	14	0.01	4	57.7%	± 12.2%	56.7%
arrive	4	0.05	2	69.5%	± 6.5%	66.7%
cry	12	0.01	4	65%	± 9.1%	50%
halt	2	0.05	2	81.8%	± 7.2%	83.3%
plough	12	0.005	4	60.5%	± 4.3%	63.3%
submit	7	0.005	8	82.7%	± 8.1%	83.3%
weighted average				**71.9%**		69.1%

Table 7: Tuned parameters for the Decision Tree classifier

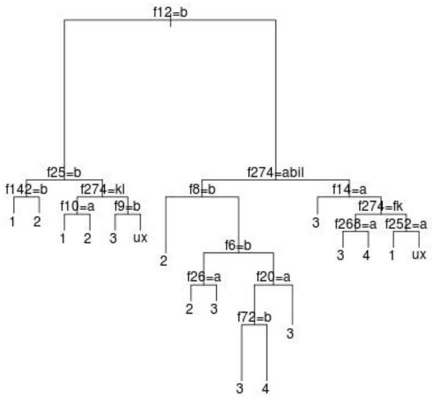
3.3 Support Vector Machines

Similarly to the previous classifiers, first the baseline ("non-tuned") accuracy was computed for the SVM classifier. It is shown in Table 8.

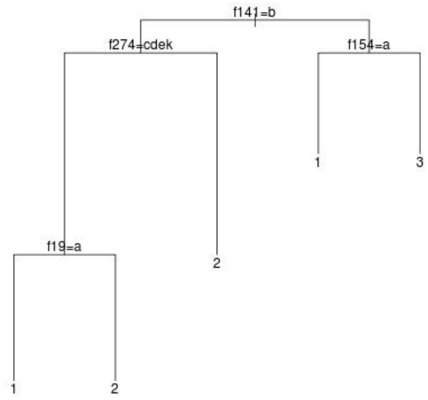
The greedy algorithm for feature selection did not prove useful for the SVM classifier either. Therefore, I used all the features from the default set (apart from the constant or almost constant ones) for parameter tuning.

Parameter tuning was done by the R function `tune.svm`, and parameters `gamma` and `cost` were tuned. Values for `gamma` were 0.0625, 0.125, 0.25, 0.5, values for `cost` were 1, 2, 5, 10, 50, 100.

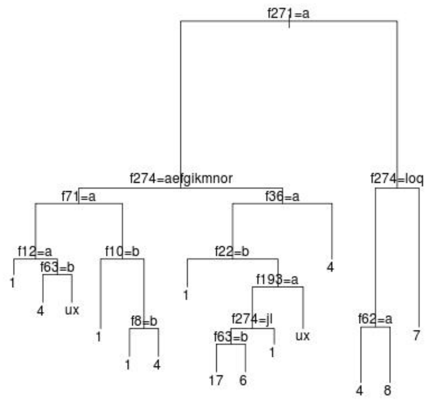
The final results are shown in Table 9.



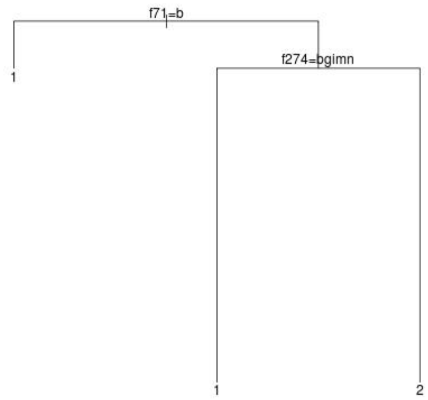
(a) The verb *ally*



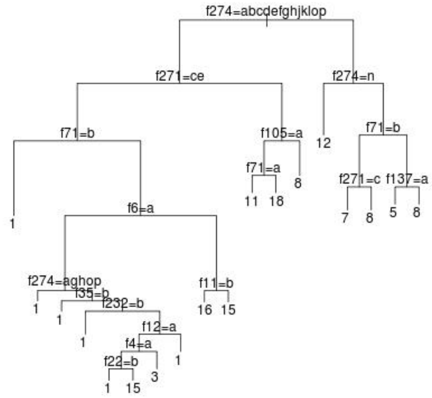
(b) The verb *arrive*



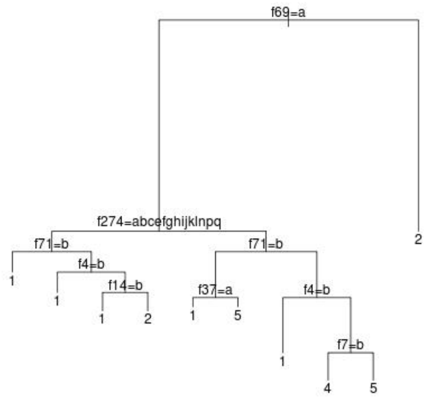
(c) The verb *cry*



(d) The verb *halt*



(e) The verb *plough*



(f) The verb *submit*

Figure 2: Decision trees

verb	accuracy on train set	confidence interval	accuracy on test set
ally	55.9%	± 16.8%	53.3%
arrive	68.6%	± 7.6%	63.3%
cry	53.2%	± 9.3%	46.7%
halt	83.6%	± 6.7%	83.3%
plough	38.2%	± 11.4%	40%
submit	70.9%	± 9.7%	70%
weighted average	**66.9%**		63.3%

Table 8: Baseline accuracy for the SVM classifier

verb	gamma	cost	accuracy on train set	confidence interval	accuracy on test set
ally	0.0625	10	64.1%	± 15.8%	53.3%
arrive	0.0625	2	71.4%	± 5.9%	66.7%
cry	0.0625	5	67.3%	± 5.1%	73.3%
halt	0.0625	1	83.6%	± 6.7%	83.3%
plough	0.0625	5	66.8%	± 9.1%	66.7%
submit	0.0625	2	85%	± 4.3%	93.3%
weighted average			**74.2%**		73.6%

Table 9: Tuned parameters for the SVM classifier

3.4 Results for the Task A

In conclusion to the Task A it can be said that all three methods achieved better performance than baseline. Interestingly, the specific baselines, computed as the performances of the classifiers without tuning any parameters, did not exceed much the primitive baseline, which corresponds to assigning the most frequent pattern to every verb. However, tuning the parameters brought some improvement, and surprisingly the Naive Bayes classifier comes as best. The results are summarized in Table 10.

method	baseline	tuned performance
Naive Bayes	66.3%	**80.2%**
Decision Trees	70.3%	71.9%
SVM	66.9%	74.2%

Table 10: Overall results of Task A

The classifier, which I chose in the part A of the experiment, is therefore the Naive Bayes classifier, with the laplace parameter set to 0 and with the reduced feature set for each verb (see section 3.1 for details).

Apart from selecting the best possible classifier, we can have a look whether we can learn something interesting about the features from the selections performed with the Naive Bayes classifier and the Decision Tree. Table 11 shows which features were chosen by either of these classifiers and which appeared in both.

verb	features NB	features DT	both
ally	f2, f6, f8, f9, f12, f15, f16, f19, f20, f21, f26, f29, f35, f36, f38, f39, f40, f47, f51, f55, f57, f62, f70, f73, f104, f128, f138, f193, f210, f232, f266, f272	f6, f8, f9, f10, f12, f14, f20, f25, f26, f72, f142, f252, f268, f274	f6, f8, f9, f12, f20, f26
arrive	f4, f5, f6, f7, f26, f35, f36, f39, f41, f45, f46, f57, f73, f154, f168, f196, f217, f244, f272	f19, f141, f154, f274	f154
cry	f2, f3, f4, f12, f13, f15, f16, f22, f24, f26, f27, f29, f32, f37, f41, f46, f48, f55, f57, f58, f62, f69, f81, f124, f140, f143, f202, f271, f274	f8, f10, f12, f22, f36, f62, f63, f71, f193, f271, f274	f12, f22, f62, f271, f274
halt	f0, f1, f2, f3, f6, f16, f18, f21, f22, f24, f27, f28, f33, f52, f54, f57, f58, f59, f63, f70, f71, f73, f106, f143, f154, f170, f206, f222, f274	f71, f274	f71, f274
plough	f1, f6, f7, f9, f14, f16, f21, f23, f28, f45, f52, f56, f57, f71, f96, f98, f105, f111, f126, f148, f150, f161, f181, f184, f205, f209, f211, f232, f234, f252, f253, f271, f274	f4, f6, f11, f12, f22, f35, f71, f105, f137, f232, f271, f274	f6, f71, f105, f232, f271, f274
submit	f0, f1, f3, f4, f5, f7, f8, f12, f15, f16, f26, f29, f36, f69, f71, f75, f124	f4, f7, f14, f37, f69, f71, f274	f4, f7, f69, f71

Table 11: Features selected as most useful. For description of all features see Appendix A.

If we were to find some general tendency, we see the feature most often considered useful is f274 – the presence of a specific prepositional modifier. It appears for all the six verbs in the DT classifier and for three of the verbs in the NB classifier. Among others that appear often are f271 – the presence of a phrasal verb particle, f12 – nominal following the target verb, and f6 – the VBG tag on the target verb (which means the verb is in gerund or present participle form).

4 Comparison with A. Tamchyna's results

I had the opportunity to compare my results with those of my colleague Aleš Tamchyna [2]. Despite the fact we worked with identical data in the experiment, we obtained different results. To some extent, this was to be expected, because the methods we used are probabilistic in principle and some randomness is always present. But of course I cannot rule out the possibility of an error in my computation.

Our conclusions from the Task A are different: The classifier which comes as best in A. Tamchyna's experiment, is SVM. In my experiment, it is Naive Bayes.

And what are the visible differences in our approaches:

- the number of features removed from the default set – A. Tamchyna removed features that have only one value or their second value appears only once, he ended with cca 140-150 features for each verb. I removed also features, which had their second value less than four times, which yielded cca 100-120 features for each verb. This difference probably does not influence the results very much.

- the ranges of the classifiers' parameters for tuning – here we differ quite a lot, especially for the DT and SVM classifiers we ended with very different parameters. However, the tuned performance (weighted average) of our classifiers on the development data is quite close:

	DT classifier	SVM classifier
AT	72.9 %	74.6%
EK	71.9 %	74.2%

- the Naive Bayes classifier – there is the biggest difference in the result:

	NB classifier
AT	71.6 %
EK	80.2 %

There is also probably the biggest difference in approach – I greatly reduced the feature set for this classifier using the greedy algorithm (see section 3.1), for each verb there remained only up to 33 features. This considerably improved the performance. It seems reasonable, because the training data is relatively small, and having too many features is rather a disadvantage, their combinations are not present enough times and classifier is not able to learn them. On the other hand, the greedy algorithm which I used for selecting the most useful features might have the same problem, and it is also possible that the selected features are suitable for our development data, but unsuitable for different data.

5 Comparison with V. Kríž's results

Vincent Kríž's diploma thesis is also focused on finding classifiers for automatic pattern assignment, but of course deals with the problem much more thoroughly. He works with more verbs and experiments in much more detail with selecting the best features and tuning the best parameters for each verb and each classification method. For this comparison, I am going to look only on his results for the six verbs I used here and only with morpho-syntactic features (V. Kríž then also experimented with different types of semantic features).

V. Kríž did not use the Naive Bayes classifier, but instead he worked with the k-nearest neighbours classifier and AdaBoost, which I did not. Concerning feature selection, he achieved the best results with a set he calls Best58 – a list of 58 features chosen by a greedy algorithm (for details see [1], pp. 74–75).

The comparison of our results is the following:

	DT classifier	SVM classifier	NB classifier	kNN classifier	AdaBoost
VK	74.3%	74.8%	x	73.1%	**76%**
EK	71.9%	74.2%	**80.2%**	x	x

V. Kríž reports the SVM classifier as best. It is because he computes a weighted average over all the 30 verbs he worked with. When computed only over our 6 verbs, it is AdaBoost which gives the best result. However, it is still below the result of Naive Bayes which came out in my experiment.

6 Task B

In the task B, I experimented with adding "dictionary" features, i.e. features of the type "is a word in the neigbourhood of the target verb equal to X?". I chose the verbs *plough*, *cry*, and *arrive*.

As a starting point, I took the results from the part A. I used the Naive Bayes classifier with a reduced set of features. As the initial set of features for all the verbs I used the features that appeared at least three times in the selections in the part A (see Table 11). These were: f1, f2, f3, f4, f6, f7, f12, f15, f16, f21, f26, f29, f36, f57, f71, f73, and f274. To this set, I added new features for each verb.

The new features were created based on the patterns, which contain a lexical set, e. g. pattern 1 for the verb *plough*:

[[Human = Farmer] ˆ [Animal |{horse|ox|bull}]] plough (Location | {field|acre|furrow|land|area|moorland|field|fallow|pasture|stetch|round|soil|...} = Field)

In some cases, I also added words which were not explicitly mentioned in the pattern, but can logically occur there, e.g. in the pattern 7 for the verb *plough*:

[Human | Institution] plough [Money] [{into} Business Enterprise | Activity]

Here I replaced the given type [Money] with a set {money|cash|profit|capital|million|thousand}. For descriptions of all the added features see Appendix B.

6.1 Results of the Task B

The results of Task B are summarized in Table 12. Even with the initial 17 features only, the classifier performed better than baseline (see Table 2) for the verbs *plough* and *cry*. Adding new features improved the performance for all the three verbs, but not above the level achieved in Task A (shown here in the last column, for details see Table 4).

verb	NB with initial features			NB with added features				Task A
	acc. on train set	conf. int.	acc. on test set	laplace	acc. on train set	conf. int.	acc. on test set	acc. on train set
plough	43.2%	8.2%	46.7%	1	**64.5%**	3.2%	60%	**75%**
cry	58.2%	8.4%	56.7%	1	**65.9%**	8.2%	66.7%	**76.4%**
arrive	64.5%	5.5%	56.7%	3	**74.1%**	6.5%	70%	**77.3%**

Table 12: Overall results of Task B

The results of Task B show that "dictionary" features might be of some help. However, it is necessary to select properly the initial set of features, which I did not do. Therefore, the classifiers are not tuned to their best possible performance, I just wanted to experiment with new features to see whether they might be of any use. Also, to find the most suitable dictionary features, we would probably need more development data. It is necessary to have enough examples for each pattern and look carefully into them to extract the words which are most significant for each pattern.

Appendix A

List of Default Features

Syntactic properties of the target verb (TV):

f0	TV in passive voice
f1	modality 1 – TV with *would* or *should*
f2	modality 2 – TV with *can*,*could*, *may*, *must*, *ought*, *might*
f3	negation – TV negated

Morphological tags on TV:

f4	VBN tag on TV
f5	VBD tag on TV
f6	VBG tag on TV
f7	VBP tag on TV
f8	VB tag on TV

Properties of the words following and preceding the TV (+- 3 positions):

f9	nominal-like, TV - 3	f27	modal, TV - 3	f45	wh-adverb, TV - 3
f10	nominal-like, TV - 2	f28	modal, TV - 2	f46	wh-adverb, TV - 2
f11	nominal-like, TV - 1	f29	modal, TV - 1	f47	wh-adverb, TV - 1
f12	nominal-like, TV + 1	f30	modal, TV + 1	f48	wh-adverb, TV + 1
f13	nominal-like, TV + 2	f31	modal, TV + 2	f49	wh-adverb, TV + 2
f14	nominal-like, TV + 3	f32	modal, TV + 3	f50	wh-adverb, TV + 3
f15	adjective, TV - 3	f33	preposition "to", TV - 3	f51	verb, TV - 3
f16	adjective, TV - 2	f34	preposition "to", TV - 2	f52	verb, TV - 2
f17	adjective, TV - 1	f35	preposition "to", TV - 1	f53	verb, TV - 1
f18	adjective, TV + 1	f36	preposition "to", TV + 1	f54	verb, TV + 1
f19	adjective, TV + 2	f37	preposition "to", TV + 2	f55	verb, TV + 2
f20	adjective, TV + 3	f38	preposition "to", TV + 3	f56	verb, TV + 3
f21	adverb, TV - 3	f39	wh-pronoun, TV - 3	f57	lemma "to be", TV - 3
f22	adverb, TV - 2	f40	wh-pronoun, TV - 2	f58	lemma "to be", TV - 2
f23	adverb, TV - 1	f41	wh-pronoun, TV - 1	f59	lemma "to be", TV - 1
f24	adverb, TV + 1	f42	wh-pronoun, TV + 1	f60	lemma "to be", TV + 1
f25	adverb, TV + 2	f43	wh-pronoun, TV + 2	f61	lemma "to be", TV + 2
f26	adverb, TV + 3	f44	wh-pronoun, TV + 3	f62	lemma "to be", TV + 3

Presence of elements syntactically dependent on TV:

f63	nominal subject
f64	clausal subject
f65	direct object
f66	indirect object
f67	passive nominal subject
f68	passive clausal subject
f69	clausal complement
f70	complementizer ("that", "whether", etc.)
f71	any object
f72	adverbial modifier
f73	adverbial clause modifier
f74	purpose clause modifier
f75	temporal modifier

Semantic classes of the words immediately preceding or following the TV and its subjects and objects:

f76	TV -1	class: origin	f84	TV -1	class: living	f92	TV -1	class: human
f77	TV +1	class: origin	f85	TV +1	class: living	f93	TV +1	class: human
f78	subject	class: origin	f86	subject	class: living	f94	subject	class: human
f79	object	class: origin	f87	object	class: living	f95	object	class: human
f80	TV -1	class: natural	f88	TV -1	class: plant	f96	TV -1	class: creature
f81	TV +1	class: natural	f89	TV +1	class: plant	f97	TV +1	class: creature
f82	subject	class: natural	f90	subject	class: plant	f98	subject	class: creature
f83	object	class: natural	f91	object	class: plant	f99	object	class: creature

f100	TV -1	class: animal
f101	TV +1	class: animal
f102	subject	class: animal
f103	object	class: animal
f104	TV -1	class: artifact
f105	TV +1	class: artifact
f106	subject	class: artifact
f107	object	class: artifact
f108	TV -1	class: form
f109	TV +1	class: form
f110	subject	class: form
f111	object	class: form
f112	TV -1	class: substance
f113	TV +1	class: substance
f114	subject	class: substance
f115	object	class: substance
f116	TV -1	class: solid
f117	TV +1	class: solid
f118	subject	class: solid
f119	object	class: solid
f120	TV -1	class: liquid
f121	TV +1	class: liquid
f122	subject	class: liquid
f123	object	class: liquid
f124	TV -1	class: gas
f125	TV +1	class: gas
f126	subject	class: gas
f127	object	class: gas
f128	TV -1	class: object
f129	TV +1	class: object
f130	subject	class: object
f131	object	class: object
f132	TV -1	class: composition
f133	TV +1	class: composition
f134	subject	class: composition
f135	object	class: composition
f136	TV -1	class: part
f137	TV +1	class: part
f138	subject	class: part
f139	object	class: part
f140	TV -1	class: group
f141	TV +1	class: group
f142	subject	class: group
f143	object	class: group
f144	TV -1	class: function
f145	TV +1	class: function
f146	subject	class: function
f147	object	class: function
f148	TV -1	class: vehicle
f149	TV +1	class: vehicle
f150	subject	class: vehicle
f151	object	class: vehicle
f152	TV -1	class: representation
f153	TV +1	class: representation
f154	subject	class: representation
f155	object	class: representation
f156	TV -1	class: software
f157	TV +1	class: software
f158	subject	class: software
f159	object	class: software
f160	TV -1	class: place
f161	TV +1	class: place
f162	subject	class: place
f163	object	class: place
f164	TV -1	class: occupation
f165	TV +1	class: occupation
f166	subject	class: occupation
f167	object	class: occupation
f168	TV -1	class: instrument
f169	TV +1	class: instrument
f170	subject	class: instrument
f171	object	class: instrument
f172	TV -1	class: garment
f173	TV +1	class: garment
f174	subject	class: garment
f175	object	class: garment
f176	TV -1	class: furniture
f177	TV +1	class: furniture
f178	subject	class: furniture
f179	object	class: furniture
f180	TV -1	class: covering
f181	TV +1	class: covering
f182	subject	class: covering
f183	object	class: covering
f184	TV -1	class: container
f185	TV +1	class: container
f186	subject	class: container
f187	object	class: container
f188	TV -1	class: building
f189	TV +1	class: building
f190	subject	class: building
f191	object	class: building
f192	TV -1	class: 3rdorderentity
f193	TV +1	class: 3rdorderentity
f194	subject	class: 3rdorderentity
f195	object	class: 3rdorderentity
f196	TV -1	class: dynamic
f197	TV +1	class: dynamic
f198	subject	class: dynamic
f199	object	class: dynamic
f200	TV -1	class: property
f201	TV +1	class: property
f202	subject	class: property
f203	object	class: property
f204	TV -1	class: relation
f205	TV +1	class: relation
f206	subject	class: relation
f207	object	class: relation
f208	TV -1	class: cause
f209	TV +1	class: cause
f210	subject	class: cause
f211	object	class: cause
f212	TV -1	class: phenomenal
f213	TV +1	class: phenomenal
f214	subject	class: phenomenal
f215	object	class: phenomenal
f216	TV -1	class: communication
f217	TV +1	class: communication
f218	subject	class: communication
f219	object	class: communication
f220	TV -1	class: condition
f221	TV +1	class: condition
f222	subject	class: condition
f223	object	class: condition
f224	TV -1	class: existence
f225	TV +1	class: existence
f226	subject	class: existence
f227	object	class: existence
f228	TV -1	class: experience
f229	TV +1	class: experience
f230	subject	class: experience
f231	object	class: experience
f232	TV -1	class: location
f233	TV +1	class: location
f234	subject	class: location
f235	object	class: location
f236	TV -1	class: manner
f237	TV +1	class: manner
f238	subject	class: manner
f239	object	class: manner
f240	TV -1	class: modal
f241	TV +1	class: modal
f242	subject	class: modal
f243	object	class: modal
f244	TV -1	class: possession
f245	TV +1	class: possession
f246	subject	class: possession
f247	object	class: possession
f248	TV -1	class: purpose
f249	TV +1	class: purpose
f250	subject	class: purpose
f251	object	class: purpose
f252	TV -1	class: quantity
f253	TV +1	class: quantity
f254	subject	class: quantity
f255	object	class: quantity
f256	TV -1	class: social
f257	TV +1	class: social
f258	subject	class: social
f259	object	class: social
f260	TV -1	class: time
f261	TV +1	class: time
f262	subject	class: time
f263	object	class: time
f264	TV -1	class: usage
f265	TV +1	class: usage
f266	subject	class: usage
f267	object	class: usage

Miscellaneous

f268	plural subject
f269	plural object
f270	TV used in an infinite phrase (outside subject)
f271	presence of a phrasal verb particle with TV
f272	presence of a prepositional modifier
f273	presence of a marker (subordination conjunction, apart from “that” and “whether”)
f274	presence of a prepositional clausal modifier

All features f0–f270 are binary (yes/no), features f271–f274 are categorical, with the preposition/marker/particle in question as their value.

Appendix B

List of Added Features

plough

- in a window ± 5 words from TV, there is one of the words *money, cash, profit, capital, million, thousand* or a word containing digits
- among 5 words following TV, there is the word *back*
- among 6 words following TV, there is the word *into*
- in a window ± 5 words from TV, there is one of the words *field, acre, land, area, pasture, soil, ground*
- among 5 words following TV, there is the word *sea* or *ocean*
- among 5 words following TV, there is the word *path* or *furrow*
- among 3 words following TV, there is the word *on, onwards* or *ahead*
- among 3 words following TV, there is the word *through*

cry

- among 4 words following TV, there is the word *eyes, heart* or *head*
- among 6 words following TV, there is the word *shoulder*
- the word following TV is *over*
- the word following TV is *out*
- the word following TV is *off*
- the word following TV is *for*
- the word following TV is *with*
- the word following TV contains the string *self* or *selves*

arrive

- in a window ± 6 words from TV, there is one of the words *document, collection, exhibition, aid, package, container, letter, cheque, information, mail, goods*
- in a window ± 6 words from TV, there is one of the words *solution, decision, figure, result, consensus, agreement, answer, interpretation, conclusion*
- the word following TV is *at*
- the word following TV is *to* or *from*
- the word following TV is a punctuation mark

References

[1] Kríž, Vincent (2012): Klasifikátor pro sémantické vzory užívání anglických sloves. Diploma thesis, MFF UK, Praha.

[2] Tamchyna, Aleš (2012): Semantic Pattern Classification, Final Report. Unpublished manuscript.