Empirically Evaluating Co-Training

Student:
W.E. van Beusekom, 9928863

Supervisors:
Dr. I.G. Sprinkhuizen-Kuyper
Dr. L.G. Vuurpijl

May 27, 2009
Radboud University Nijmegen

Foreword

"We hebben allemaal wat, we zijn allemaal raar
Toch houden we van elkaar."

Many thanks go out to Ida Sprinkhuizen-Kuyper and Louis Vuurpijl for keeping pointing me into the right direction, and to Egon van den Broek for giving me pointers about setting up the data set and feature extraction.

I would further like to thank, in alphabetical order: Freek Batstra for accompanying me through MSN during lonesome hours, Helen Oosterling for her pep-talks, Leonie Wiering for her unflinching constructive attitude, Leon van Gulik for taking the time to be a friend, Lisa Hauke for being a wonderful roommate, Petra Blom for being a loyal friend, Stijn Voortman for being Stijn Voortman, and, Theo van Beusekom for making hard efforts to share his wisdom with me. Finally, I would like to thank myself for not being too hard on me.

Abstract

Co-training is a classification scheme needing only a small set of training instances for correct classification. The main question assessed in this thesis was how co-training performance varies with varying representativeness of the training data. 1280 co-training runs have been made, to test the generalization accuracy of co-training classification when using different selections of the training data. The results indicate that the availability of training data that are typical for their class or a distribution in the training data matching the a priori distribution of the corpus as a whole is a good condition for the generalization accuracy of co-training.

1 Introduction

A classifier is a program which implements a method for distinguishing data into classes, based on the given training examples. In a supervised learning task, a classifier is trained on a set of manually labeled example data. The amount of example data that is needed can be huge. If, for example, the data consists of pictures of good and bad quality tomatoes, one may need thousands of example pictures to train the classifier with, so that it can accurately classify future pictures. And there should be at least one human being who painstakingly labels each of these pictures, among which are rotten tomatoes. Once labeled, this set of training data is then plugged into the supervised learning classifier to allow it to form a hypothesis with which it should accurately predict the classes of yet unlabeled pictures.

Co-training is a classification scheme with embedded classifiers that are trained using the same training set. What co-training does, according to Blum and Mitchell, the designers of co-training [2], is to let two or more pre-trained classifiers each classify fresh, unlabeled data - for which a high confidence is achieved - for the new automatically labeled data and put these in the shared training set. The classifiers are then trained with the updated training set, newly labeled data are picked, and so on. This way, what starts as a pair of classifiers with low accuracy on the whole set, can become a pair of classifiers with high accuracy. An algorithmic description of co-training is given in Section 3.

Manually labeling hundreds and thousands of samples in a classification training set can be enormously time-consuming, and might not be a very inspiring chore. If this classification task were done using a co-training classification scheme, there would be no need to invest days, weeks, or even months into the manual labeling work. This is because co-training allows correct classification starting with only a small set of labeled instances by using multiple feature views on the same large set of unlabeled instances. The major advantage of the co-training strategy is that usually a small set of instances is enough for sufficient classification.

1.1 Related Work

Blum and Mitchell [2] succesfully applied co-training with a Naive Bayes classifier to universities' web pages, classifying them as either 'home pages' or 'academic courses'. Two kinds of features were used: the web page content text and the hyperlinks pointing to the web page. After 30 co-training iterations, the classification error dropped by a factor of more than two.

Nigam and Ghani [7, 8] adapted the approach of Blum and Mitchell, namely co-training with a Naive Bayes text classifier using different views, but they artificially created two fully independent and redundant feature views in the data set. The result of this experiment was that algorithms with the artificial feature split indeed yield better performance than algorithms without.

Zhang and Lee [10] classified full-color images from the internet using co-training with embedded SVM classifiers, the independent views being 'color histogram' and 'text associated with the image'. They found that co-training indeed performs better than other implementations of the SVM algorithm.

Kiritchenko and Matwin [5] compared the performance of two classifiers, Naive Bayes and SVM, when classifying emails as 'interesting' and 'uninteresting', each being co-trained with the views 'bag of words from subject lines' and 'bag of words from body'. Surprisingly Naive Bayes performed very poorly, performance decreasing with iterations. On the other hand SVM performed well and had the best perfomance when the initial training set contained five times more negative examples than positive.

In the original co-training design by Blum and Mitchell, each classifier should be given a view on the data that is different from the views given to the other classifiers. These feature views should not be completely correlated given the class and they should be redundantly sufficient - i.e., each view on any instance should, by itself, contain sufficient information to accurately classify it. Goldman and Zhou [4] adapted co-training to enable co-training classification with different classifiers, instead of different views. They co-trained two different learning algorithms, ID3 and HOODG, on the same data set. In a series of experiments ranging over eight different data sets, after at most five iterations, a decrease in the error rate ranging from 0% to 100% was reported compared to that of the ID3 and HOODG classifiers individually.

1.2 Research Question

This work is a proof of concept investigating how the co-training performance depends on the *representativeness* of the initial training data. This representativeness can be defined in two ways. First, the training examples themselves can be categorized as typical and untypical for their class. Instances that are typical for their class have more instances of the same class in close proximity than untypical instances. Second, the training set as a whole can be distributed just like the distribution in the entire corpus, or not. Two kinds of feature sets were extracted from raw pictures of textures. There were four classes of pictures, and two-class co-training was applied alternately with each class being the positive class while the other three classes were regarded as the same negative class. Furthermore, co-training with different feature sets was compared to co-training with the same feature set. The generalization accuracy of the different co-training runs was finally compared.

It is expected that co-training performance is better when the training instances are typical for their class than when they are untypical for their class, and that the performance also is better when the training set has a distribution matching the distribution of the corpus than when it is not.

Section 2 discusses how the pipeline processing the raw data to co-training-usable instances was implemented. How the bitmaps were preprocessed into features is elaborated on in Section 2.1. How the features were made usable for the experiments in terms of typicalness and distribution is explicited in Sections 2.2 and 2.3. The co-training algorithm is the subject of Section 3, with greater detail about its embedded classifiers in Section 3.1 and about the confidence measures in Section 3.2. The method of this thesis' experiment is discussed in Section 4, the results of the experiment in Section 5 and Section 6 rounds up with concluding remarks and suggestions for future research.

2 Processing pipeline

See Figure 1 for the complete pipeline from data preprocessing through the co-training experiments. The entire pipeline was written in Matlab code (see Appendix A). Prior to the co-training runs, 61 texture bitmaps from the VisTex database were processed through the following stages: preprocessing, typicalness estimation, and instance distribution. The preprocessing stage is discussed in Section 2.1. During this stage, the bitmaps are quantized, split into smaller bitmaps and finally features are extracted from it. Typicalness estimation and instance distribution are both methods to manipulate the representativeness of the initial training data for the co-training algorithm. Both methods are discussed in Section 2.2 and in Section 2.3, respectively.

2.1 Bitmap preprocessing

For the corpus, 61 pictures were downloaded from the on-line VisTex database. They were truecolor, 24-bits 512x512-pixels, representing different kinds of textures: 13 pictures of barks, 20 of man-made fabrics, 17 of leaves, and 11 of terrains. All the pictures were quantized to a 32-color color map and split into 64 smaller pictures of size 64x64 pixels. This number of colors was found to work quite well in supervised learning classification [3]. This resulted in 3904 bitmaps to work with. See Figure 2 for an example of a resulting bitmap for an example picture for each of the four textures after such preprocessing. A detail of the pipeline's preprocessing stage is shown in Figure 3.

Two bags of features were extracted from the resulting bitmaps: one bag of features $\mathbf{F}_1$ based on the color correlogram, and one on the (color) histogram feature, $\mathbf{F}_2$. The feature set $\mathbf{F}_1$ was based on the color correlograms of the tex-

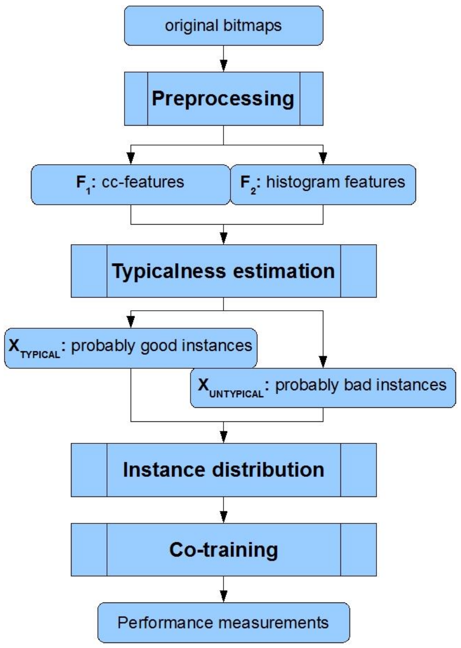

Figure 1: The processing pipeline.

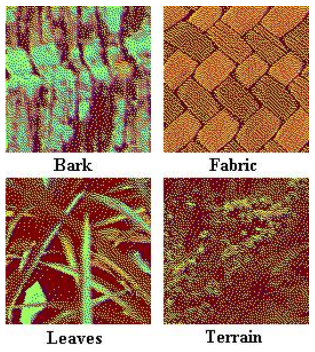

Figure 2: Four example pictures from the preprocessing output; one of each class.

ture bitmaps and consists of vectors with eight elements. The color correlogram is a matrix with elements $\mathbf{C}_{\bar{d}}(i, j)$ that contains the occurrence of two colors i and j over a distance $\bar{d}$, thus coding the structure in the picture. Each element in a $\mathbf{F}_1$ vector represents a different kind of information about the bitmap's structure, as shown in Table 1. The author's work on color corellograms is based on [3]. The histogram feature set $\mathbf{F}_2$ consists of vectors with 32 elements, each element representing the frequency of a specific color in a bitmap.

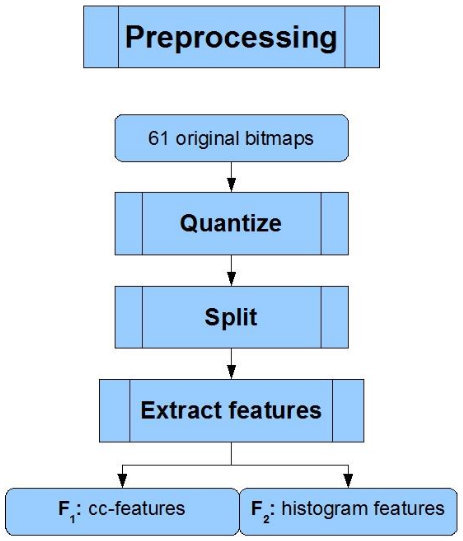

Figure 3: Detail of the pipeline's preprocessing stage.

2.2 Typicalness estimation

Let $\mathbf{L}_0$ be the set of initial training instances, i.e., before it is used with co-training. One way to look at $\mathbf{L}_0$'s representativeness is to see how typical each instance is for its own class. In the typicalness estimation stage of the pipeline, the data were divided into a set of instances that were called typical for their class, and instances that were untypical for their class.

An instance's class typicalness was defined as follows. For every instance i in a feature set $\mathbf{F}$, the number $\texttt{nNeighbors}_{i,\mathbf{F}}$ of neighbors of the same class as that instance, within a specific constant distance, is counted. This constant was fixed at the median of Euclidean distances between instances of the same class.

An instance's class typicalness $\texttt{typic}_{i,\mathbf{F}}$ given a feature set is $\texttt{nNeighbors}_{i,\mathbf{F}}$ normalized to the range $[0,1]$ by:

$$\texttt{typic}_{i,\mathbf{F}} = \frac{\texttt{nNeighbors}_{i,\mathbf{F}} - min_j(\texttt{nNeighbors}_{j,\mathbf{F}})}{max_j(\texttt{nNeighbors}_{j,\mathbf{F}}) - min_j(\texttt{nNeighbors}_{j,\mathbf{F}})},$$

where $min_j(\texttt{nNeighbors}_{j,\mathbf{F}})$ is the smallest $\texttt{nNeighbors}_{j,\mathbf{F}}$ given feature set $\mathbf{F}$ and $max_j(\texttt{nNeighbors}_{j,\mathbf{F}})$ is the largest $\texttt{nNeighbors}_{j,\mathbf{F}}$ given feature set $\mathbf{F}$. The result is an estimation measure that increases with typicalness; $\texttt{typic}_{i,\mathbf{F}} = 1$ holds for an instance that is the most typical for its class given the feature set, $\texttt{typic}_{i,\mathbf{F}} = 0$ holds for an instance that is the most untypical for its class given the feature set.

The intuition of this definition is that the typicalness of an instance decreases as it lies more at the perimeter of its class cluster and/or more away from a high-density pocket of instances of a class. This implies an inverse re-

Feature	Descriptions
Energy	$\sum_{i,j} \mathbf{C}_{\bar{d}}(i,j)^2$ Energy is a measure of the homogeneity of the image's pixel values. It increases as the homogeneity increases.
Entropy	$-\sum_{i,j} \mathbf{C}_{\bar{d}}(i,j) log \mathbf{C}_{\bar{d}}(i,j)$ Entropy is a measure of the randomness of the image's pixel values. It increases as the randomness increases, and decreases as the homogeneity increases.
Inverse Difference Moment	$\sum_{i,j} \frac{1}{1+(i-j)^2} \mathbf{C}_{\bar{d}}(i,j)$ IDM increases as the high values of the matrix are near its main diagonal. Intuitively, high IDM indicates an image where pixels having approximately the same values are clustered together.
Inertia	$\sum_{i,j}(i-j)^2 \mathbf{C}_{\bar{d}}(i,j)$ Inertia increases as the low values of the matrix are more near its main diagonal. It measures the opposite of IDM. Intuitively, low inertia indicates an image where pixels having approximately the same values are clustered together.
Cluster Shade	$\sum_{i,j}((i-\mu_x)+(j-\mu_y))^3 \mathbf{C}_{\bar{d}}(i,j)$ Cluster Shade and Cluster Prominence increase as the asymmetry in the matrix increases.
Cluster Prominence	$\sum_{i,j}((i-\mu_x)+(j-\mu_y))^4 \mathbf{C}_{\bar{d}}(i,j)$ *see Cluster Shade*
Correlation	$\sum_{i,j} \frac{(i-\mu_x)(j-\mu_y)\mathbf{C}_{\bar{d}}(i,j)}{\sigma_x \sigma_y}$ Correlation is a measure of image complexity, because higher correlation corresponds to higher correlation between a matrix's entries.
Haralick's Correlation	$\frac{\sum_{i,j}(xy)\mathbf{C}_{\bar{d}}(i,j) - \mu_R \mu_C}{\sigma_x \sigma_y}$ Compared to correlation, Haralick's correlation is an even more sensitive measure of an image's complexity.

Table 1: The color-correlogram-based features. $\mathbf{C}_{\bar{d}}(i,j)$ is an element of the so-called color-corellogram matrix. These are all the elements in an $\mathbf{F}_1$ feature vector.

lationship between typicalness and misclassification chance: as an instance lies more at the perimeter of the class cluster, its class typicalness decreases and its misclassification chance increases.

Since the typicalnesses given the feature set are normalized, the typicalnesses

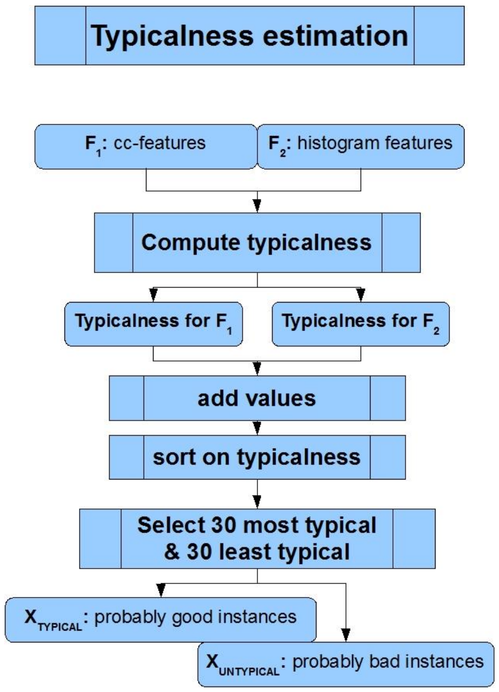

Figure 4: Detail of the processing pipeline's typicalness estimation stage.

of an instance given two feature sets can be summed so that we get an overall typicalness measure that lies within range $[0, 2]$. The intuition of this method has to do with the fact that $\mathbf{F}_1$ and $\mathbf{F}_2$ are different ways to represent the raw information contained in the bitmaps. For instance, the former represents the pixel structure of bitmaps, and the latter explicitly does not. Figure 5 shows that there are very little low-typicalness instances, especially when compared to the number of high-typicalness instances. In the majority of cases, this method yielded more typical instances than untypical instances (barks: 5.17% typical instances against 3.00% untypical instances; fabrics: 7.03% against 8.67%; leaves: 8.46% against 4.96%; terrains: 24.01% against 13.78%).

The final step of the typicalness estimation stage of the pipeline is to acquire a set of typical instances and a set of untypical instances. The bag $\mathbf{X}_{\text{TYPIC}}$ contains all the instances with typicalness of at least 1.8 (2 is the maximum) and $\mathbf{X}_{\text{UNTYPIC}}$ contains the instances with class typicalness of at most 0.6 (0 is the minimum). These bags can be used to join instances for the initial training set of a co-training run.

2.3 Instance distribution

Another way to look at the representativeness of the initial training set $\mathbf{L}_0$ is to see how the class distribution of that set corresponds to the a priori class

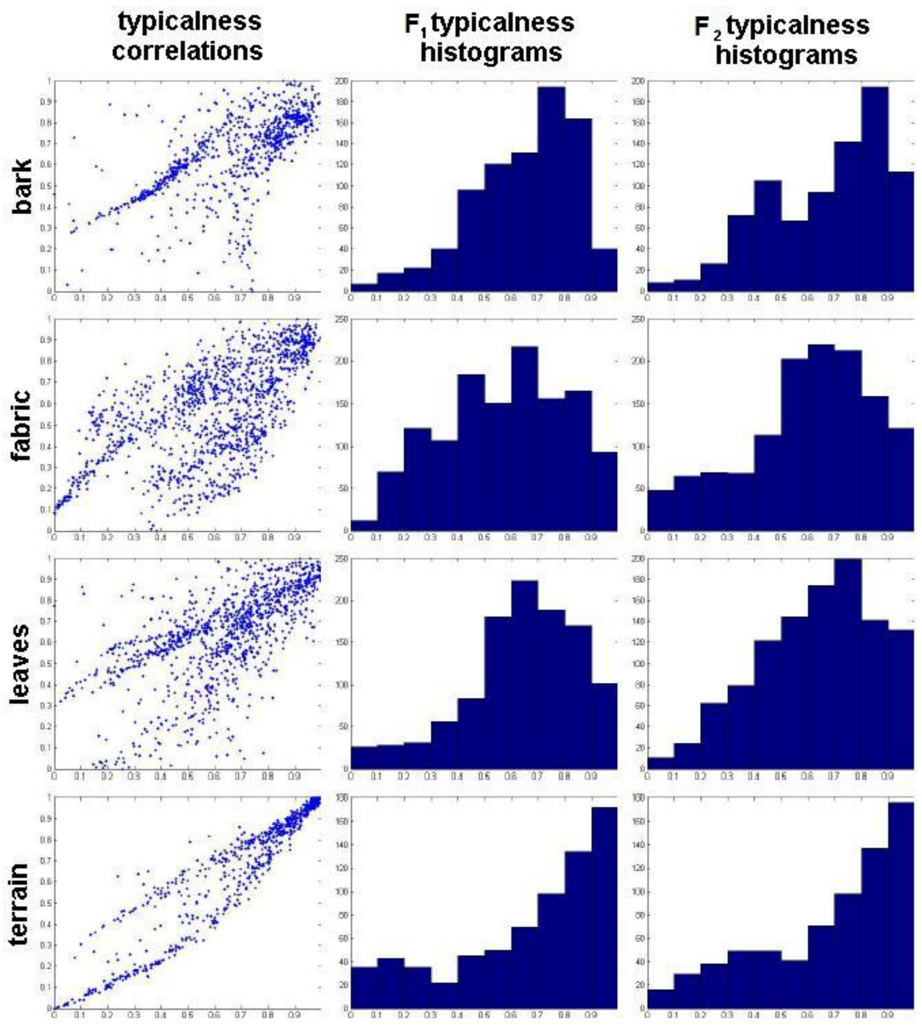

Figure 5: Intraclass typicalness correlations, typicalness histograms in the color corellogram view, typicalness histograms in the histogram view. In the intraclass typicalness correlation graphs, the horizontal axis corresponds to typicalness in the color corellogram feature set, while the vertical axis corresponds to typicalness in the histogram feature set. Typicalness in the color correlogram view and typicalness in the histogram view correlated moderately to high (bark: .6424, fabric: .6195, leaves: .6405, terrain: .9439), indicating that the shape of the instance clusters are roughly the same in both views, given the class. For each class, the typicalness distributions are nearly the same in both views, with very little low-typicalness instances and many high-typicalness instances.

distribution of the corpus as a whole. Given the size of the initial training set and the bag from which instances are picked ($\mathbf{X}_{\text{TYPIC}}$ or $\mathbf{X}_{\text{UNTYPIC}}$), the instance distribution stage randomly grabs instances from the bag such that the class distribution in $\mathbf{L}_0$ is one of the following: `matching` or `nonmatching`.

In `matching`, only the percentage of the positive class was made to match its a priori chance. The `nonmatching` option ensures that the percentage of the positive class in $\mathbf{L}_0$ is closest to 100% minus its a priori chance.

For example, if $\mathbf{L}_0$ has twenty instances in total, if the option `matching` was selected, if $\mathbf{L}_0$ should contain only untypical instances, and if the chosen positive class is Bark, then four instances in that set should be Bark (the other sixteen anything but Bark) from the bag $\mathbf{X}_{\text{UNTYPIC}}$. This way, the set has a distribution that matches the a priori distribution of Bark (which is $832/3904 = 21.3\%$) as closely as possible. If the option `nonmatching` was selected, sixteen instances in the set should be Bark.

3 Co-Training

Pictures, texts, sounds, and other objects are easily classified by human beings as belonging to a certain class. For a computer the task is much harder. The instance space $\mathbf{X}$ is defined as the representation of all the objects that human beings are able to unambiguously classify. A classifier is a program that maps from the instance space $\mathbf{X}$ to a discrete set of class labels $\mathbf{Y}$. If a classifier can do a perfect job at classifying anything from the instance space we hand to it, then the classifier is said to have found the target function $f(\mathbf{x})$ that optimally separates the instance space into its classes. The classification problem is the problem of finding that target function.

Classical solutions (i.e., supervised learning solutions) to this problem need a large number of examples for comparison with the unknown instances, but co-training makes it possible to accurately classify using only a small number of labeled instances and a lot of unlabeled instances. The co-training algorithm implemented for this thesis is given in Figure 6. See for further details on the co-training algorithm programmed by the author in Appendix B.

Co-training starts with several weak classifiers trained with the same small pool of labeled training data. Iteratively, each classifier selects confidently labeled new training instances from a large set of unlabeled instances with which to enrich their shared set of training data. In the original algorithm, each classifier should have a distinct view on the data [2] but there is also at least one succesful case of co-training use where it was sufficient that each classifier was distinct while being coupled on the same view on the data [4]. Because the data is either handled or viewed differently, heterogenous information on which to base classification hypotheses is added every iteration. During the algorithm, instances flow from $\mathbf{U}$ to $\mathbf{B}$ to $\mathbf{L}$, as shown in Figure 7.

In this thesis both different views and different classifiers are considered. What makes each view in this thesis distinct was adressed in Section 2.1. The classifiers embedded in this thesis's co-training algorithms are elaborated on

Given:

- labeled training set $\mathbf{L}$
- unlabeled buffer set $\mathbf{B}$
- unlabeled validation set $\mathbf{U}$

Loop for k iterations:

- Train classifiers h_1 and h_2 using the $\mathbf{F}_1 and \mathbf{F}_2 views of \mathbf{L}, respectively. Let \mathrm{h}_1$ classify the instances in $\mathbf{B}$.
- Compute confidences of the classified instances in $\mathbf{B}$ with h_1's confidence measure.
- Allow h_1 to move n most confidently labeled negative instances and p most confidently labeled positive instances from $\mathbf{B}$ to $\mathbf{L}$.
- Let h_2 classify the instances in $\mathbf{B}$.
- Compute confidences of the classified instances in $\mathbf{B}$ with h_2's confidence measure.
- Allow h_2 to move n most confidently labeled negative instances and p most confidently labeled positive instances from $\mathbf{B}$ to $\mathbf{L}$.
- Choose $2p + 2n$ instances from $\mathbf{U}$ to replenish $\mathbf{B}$.

Figure 6: Semi-formal description of the co-training algorithm. Adapted from [2].

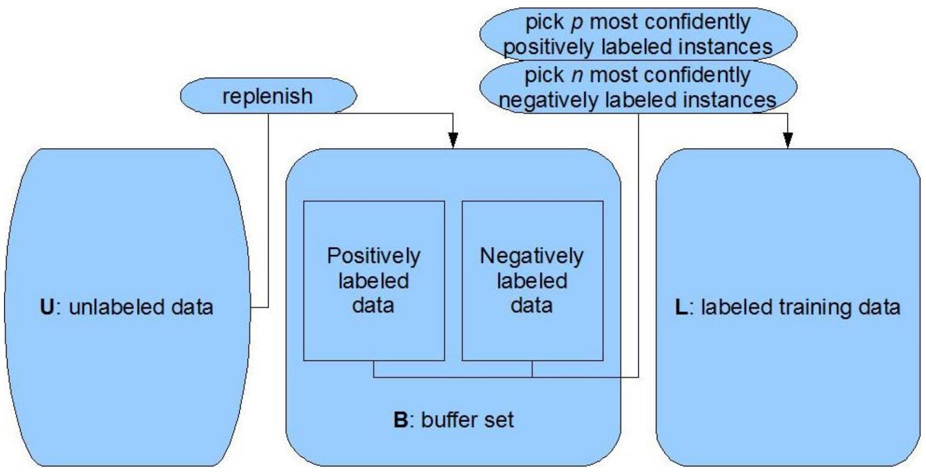
pick *p* most confidently positively labeled instances
pick *n* most confidently negatively labeled instances
replenish
U: unlabeled data
Positively labeled data
Negatively labeled data
B: buffer set
L: labeled training data

Figure 7: Outline of how the instances flow during co-training from **U** to **B** to **L**.

in Section 3.1. How the confidences of the classifications of new instances are measured is discussed in Section 3.2.

3.1 Classifiers

In this thesis, two classifiers are used: Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN). Our particular implementations of k-NN and SVM were part of the object-oriented Matlab library called Spider [11]. The purpose of SVM during its training phase is to find the hyperplane that separates the data space into distinct class-specific clusters of data as optimally as possible, so that new and unknown data points lie on the correct side of the hyperplane as much as possible [1]. The algorithm's name comes from the so-called support vectors, the points that define the hyperplane and the margin around around the hyperplane.

In realistic situations, the clusters of training points overlap and/or there is no way to put a linear hyperplane between the class clusters. In order to be able to train and classify with SVM, two solutions are available. The first solution is the kernel function, which projects the entire data space to a higher-dimension data space where it is possible to put a linear hyperplane between the class clusters (see Figure 8). If, however, there are still points that lie on the wrong side of the hyperplane, these errors can be handled by using the second solution, viz. allowing some amount of errors during the training phase [1].

The second classifier of this thesis is the K-nearest-neighbour classifier (short: k-NN). Training in k-NN is simply storing the training data and the corresponding labels. Classification then can occur as follows. Every new unknown instance is classified by the same label as the class of the majority of its k nearest training instances.

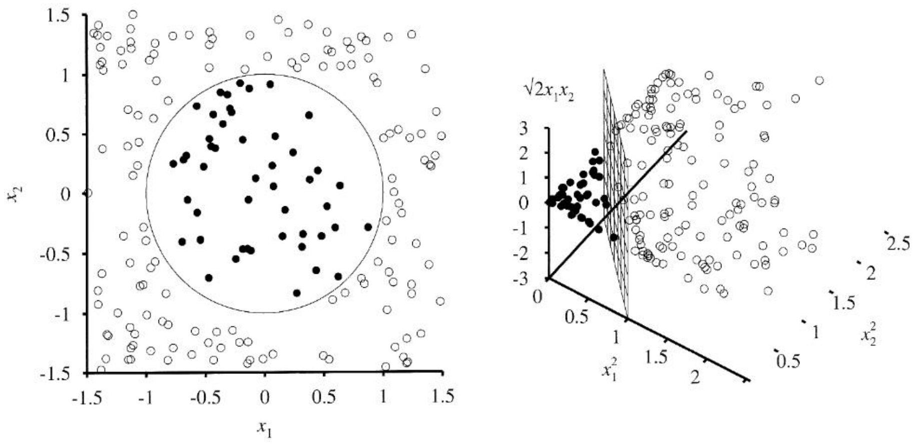

Figure 8: An example of kernel projection (from Russel and Norvig [9]), where a two-dimensional space (left) is mapped onto a three-dimensional space (right) such that data points can be separated into two sets of classes using a linear hyperplane.

3.2 Confidence measures

As explained, the most confidently classified instances are selected to enrich the training data with, but the question how this confidence is measured, remains. For this thesis's SVM classiers, a function SVM_CONF has been implemented to measure how reliable an instance classification is. For the k-NN classifiers, a function KNN_CONF was implemented.

A classified instance's confidence according to SVM_CONF depends on the training data and the hyperplane according to those training data. In the instance space, regions with high and low SVM_CONF can be constructed, such that new instances in regions around the training data and away from the hyperplane are given high SVM_CONF, whereas instances near the hyperplane are given low SVM_CONF (see Figure 9).

The confidence measure for a k-NN classified instance KNN_CONF is inversely proportional to the average distance to its seven nearest same-class neighbours. Preliminary co-training runs indicated that seven nearest same-class neighbours yielded iteratively increasing accuracy. Longer average distance corresponds to lower KNN_CONF.

4 Method

As a proof of concept, this thesis' main interest lies in how well co-training performs by varying specific aspects of the representativeness of the co-training algorithm's initial training set. These aspects were the typicalness of the instances in the initial training set and whether or not the training set has a distribution that matches the a priori distribution. The corpus consisted of

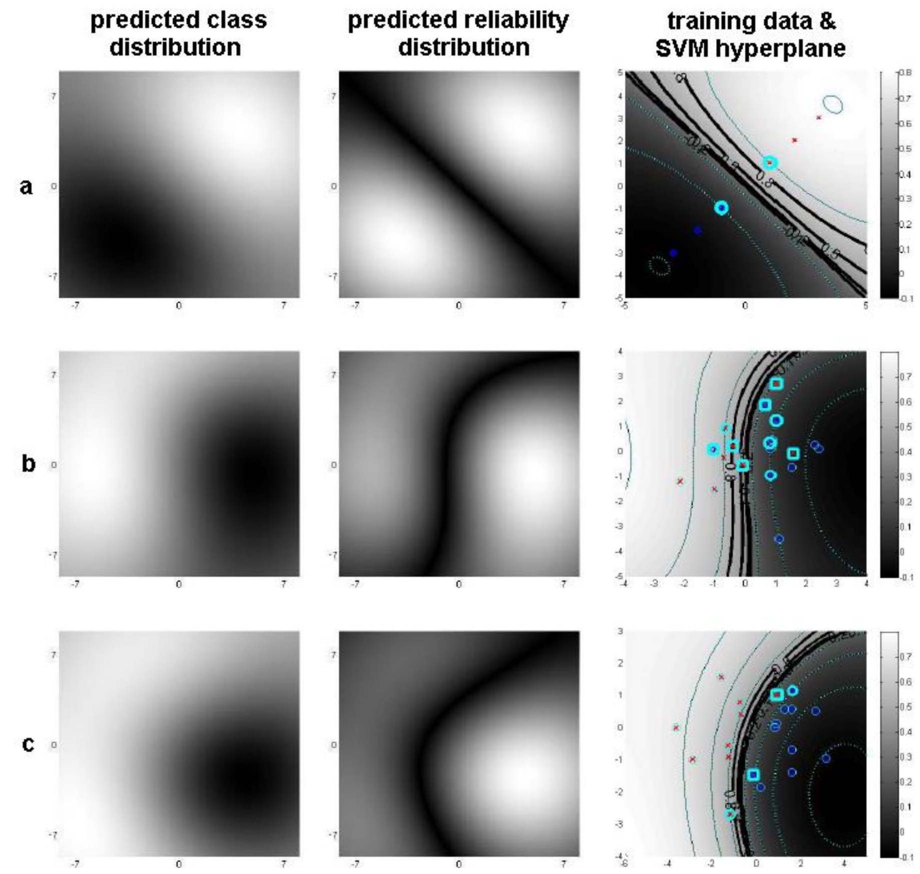

Figure 9: Visualisations of the confidence measure SVM_CONF. The horizontal axes are the axes of the first feature, the vertical axes are the axes of the second feature. The rows a through c contain pictures about three different randomly generated distributions. Instances in the darker regions in pictures in the first column are more likely to have the negative class, while instances in lighter regions in the pictures of this column are more likely have the positive class. Thus, instances in gray regions of these pictures have high uncertainty about their class. In the pictures of the second column, the confidence of instance classifications increases with the brightness of the region the classified instances are at. Note that the regions in the first two columns roughly correspond to the positions of the instances in the third column and to the positions of the hyperplanes. In the last column, marked instances are the support vectors and the isometric lines correspond to fixed values of the predicted class probability.

3904 instances of texture pictures: 832 bark pictures, 1280 man-made fabric pictures, 1088 leaf pictures, and 704 terrain pictures. Half the corpus was used for co-training runs. The other half was kept apart from the co-training runs to measure the embedded classifiers' generalization accuracy before and after co-training. Every picture could be seen in two views, as seen in the independent feature sets: color correlogram-based feature set $\mathbf{F}_1$ and color histogram feature set $\mathbf{F}_2$.

The data were used in a number of different co-training classification runs, differing in the (i) estimated typicalness of every instance in the initial training set $\mathbf{L}_0$, (ii) the class distribution of the initial training set $\mathbf{L}_0$, (iii) the texture class to be considered as positive class while instances of the other three classes were considered as the same negative class, and (iv) the classifier-view pair used for training and classification.

How the typicalnesses were estimated, was described earlier in section 2.2. The way instances could be distributed was described in section 2.3. 'Classifier-view pair' refers to which type (or types) of classifier was (or were) used during co-training, as well as which feature set was assigned to each of the two classifiers. See Table 2 for details on the pairs. Note that two of the pairs each have different classifiers and the same view, whereas the other two each have the same classifier and different views.

Different view pairs	View(s)	Learner type(s)
Pair 1	both	SVM-gaussian
Pair 2	both	k-NN
Different learner pairs	View(s)	Learner type(s)
Pair 3	histogram	both
Pair 4	cc-features	both

Table 2: The four classifier-view pairs utilized for training and classification in the co-training runs. Pairs 1 and 2 each have the same classifier but different views, whereas pairs 3 and 4 each have different classifiers but the same view.

		bark	fabric	leaves	terrain
SVM	cc-features	.7883	.6737	.7233	.8217
SVM	histogram	.7883	.6737	.7233	.8214
k-NN	cc-features	.6742	.6146	.6530	.8098
k-NN	histogram	.6931	.7107	.6172	.8225

Table 3: Ceiling estimations. For every cell, the estimation was made by selecting the best performing training set out of 100 randomly drawn training sets of forty instances.

Since there are two bags $\mathbf{X}_{\texttt{typic}}$ and $\mathbf{X}_{\texttt{untypic}}$, two kinds of $\mathbf{L}_0$'s class distribution, four classes, and four pairs, there are in total 64 different kinds of co-training runs. For every kind, 20 runs were done, each with random allocations of 20 instances in the initial $\mathbf{L}_0$, 20 instances in $\mathbf{B}$ and 1912 instances initially in $\mathbf{U}$. These numbers were chosen such that $\mathbf{L}$ was small, and, as will be seen, this number seemed sufficient for correct classification in many cases.

Furthermore, the ceilings of the data set have been established by selecting the best performing training set from 100 randomly drawn training sets. The randomly drawn training sets all had forty instances, making them the same size as the training sets after the fifth and final co-training iteration in our co-training runs. The ceilings in Table 3 are used as a reference for the resulting generalization accuracies of the co-training runs.

5 Results

All the resulting generalization accuracies after 5 iterations of co-training are shown in Figure 10. Figure 11 shows the generalization accuracy improvements.

More often than not, typical instances in the training set leads to better final performance than untypical instances. Indeed, the only pair-specific and positive class-specific significant differences found between performances of typical and untypical runs had this trend (see Table 4). Furthermore, there was evidence for the hypothesis that distribution in the initial training set matching the a priori distribution is a proper condition for good co-training performance (see Table 5). All these significant differences were found in runs with differing views (SVM-only pair and kNN-only pair) as well as in runs with the same views (histogram-only pair and color correlogram feature-only pair).

Pair	Positive class	Distribution
kNN	bark	non-matching
kNN	fabric	non-matching
histo	fabric	non-matching
cc	fabric	non-matching
kNN	leaves	non-matching
kNN	terrain	matching
kNN	terrain	non-matching
histo	terrain	matching
histo	terrain	non-matching
cc	terrain	non-matching

Table 4: All runs yielding significant differences in generalization accuracy while typicalness differed. All differences consisted of runs with typical initial training set leading to better generalization accuracy than runs with untypical instances. It appears that the difference is most often found with non-matching initial training set distribution.

Most of the SVM-only runs (SVM-only pair) started with a generalization accuracy that was nearly that of the ceiling and remained that way. A few runs with terrain positive class started with generalization accuracy under that level, and achieved ceiling-level generalization accuracy. Surprisingly, non-matching runs with fabric positive class generally had negative improvement, with the majority of these runs performing below 50%.

Unfortunately, there were cases where the final generalization accuracy was below chance level. Table 5 shows the runs that structurally performed below chance had non-matching distribution in the training set.

Co-training using different views with k-NN classifiers (kNN-only pair) led

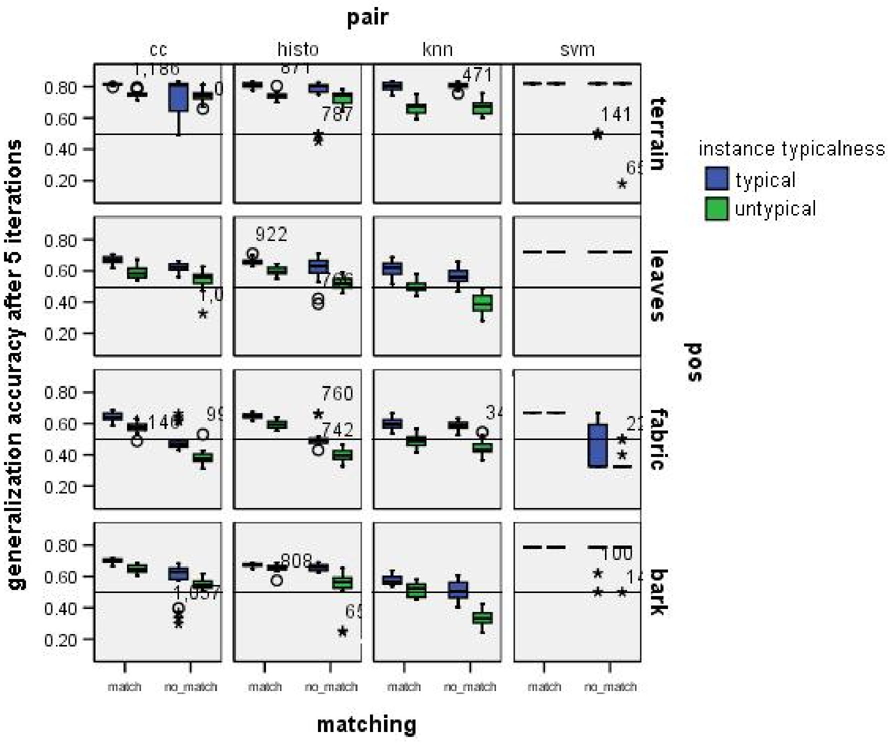

Figure 10: Generalization accuracies after 5 co-training iterations. The 1280 co-training runs are split into 2*2*4*4 box plots, twenty runs per box plot. Significant differences between two box plots are those where the box plots do not overlap. Circles are outliers, stars are extreme outliers. Note how runs with untypical instances (light green box plots) tend to end with lower accuracies than runs with typical instances (dark blue box plots). Note also how runs with non-matching distribution tend to end with lower accuracies than runs with matching distribution. Further details on these results are elaborated on in the text.

to lower rather than higher generalization accuracies than using the same views but different classifiers (histogram-only pair and color correlogram feature-only pair). This negative difference compared to same-view runs was especially pronounced in the kNN-only pair runs shown in Table 6, i.e., the ones with generalization accuracy below chance level.

Finally, only runs with non-matching training set distribution had extreme outliers in the final generalization accuracy.

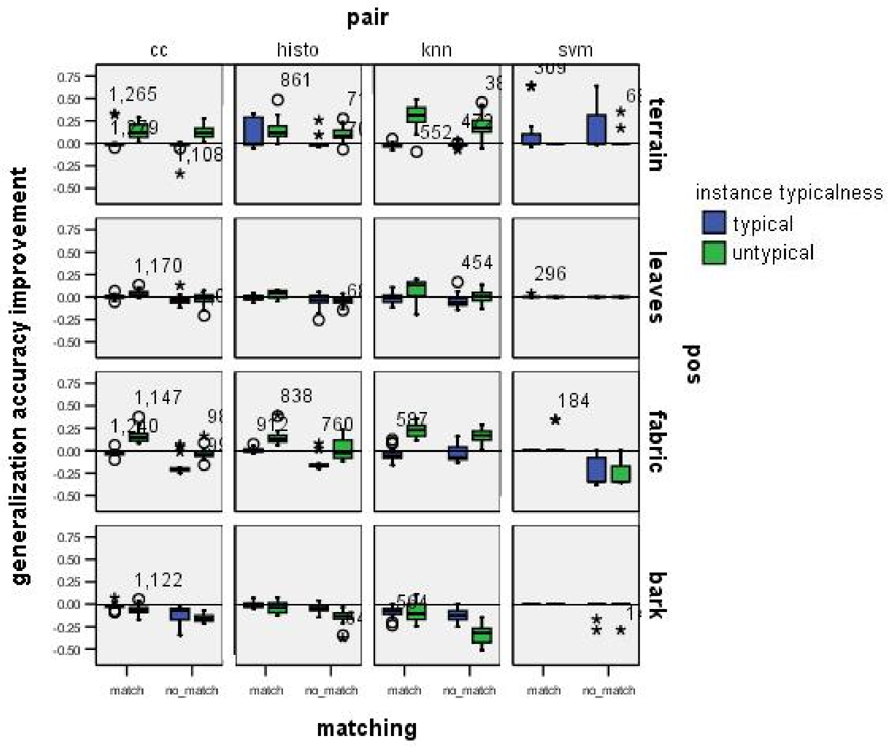

Figure 11: Generalization accuracy improvements over 5 co-training iterations. The 1280 co-training runs are split into 2*2*4*4 box plots, twenty runs per box plot. Significant differences between two box plots are those where the box plots do not overlap. Circles are outliers, stars are extreme outliers. Note how runs with untypical instances (light green box plots) tend to have larger change in performance than runs with typical instances (dark blue box plots). Further details on these results are elaborated on in the text.

6 Discussion

The results indicate that training data containing instances that are typical for their class as well as a distribution in the training data that matches the a priori distribution have a positive effect on the generalization accuracy of co-training runs. When the distribution doesn't match, the results can be disastrous: a distribution in the training set that doesn't match the a priori distribution is a predictor for below chance level generalization accuracy.

Under nearly all circumstances, co-training didn't seem to be necessary to achieve high generalization accuracy on this data set when using SVM classifiers and different views. Even before co-training, the accuracies of SVM classifiers trained with $\mathbf{L}_0$ are quite high.

The distribution of the training set matters a great deal for co-training. Not

Pair	Positive class	Typicalness
knn	bark	untypical
cc	bark	typical
cc	bark	untypical
histo	fabric	typical
histo	fabric	untypical
cc	fabric	typical
cc	fabric	untypical
knn	leaves	untypical

Table 5: All runs yielding significant differences in generalization accuracy while the distribution of the training set differed. All differences consisted of runs with matching initial training set distribution leading to better generalization accuracy than runs with non-matching initial training set distribution.

Pair	Positive class	Distribution	Typicalness
knn	bark	non-matching	untypical
svm	fabric	non-matching	untypical
histo	fabric	non-matching	untypical
cc	fabric	non-matching	untypical
knn	leaves	non-matching	untypical

Table 6: The kinds of runs that structurally performed below chance level. All of them were runs with untypical instances in an initial training set having a distribution that didn't match the a priori distribution. They occurred in runs with differing views as well as runs with the same views.

only can a non-matching distribution lead to performance below chance level, it can make the generalization accuracy more variable. The only extreme outliers were found in these runs and the deviation from the mean tends to be greater in these runs than in the other ones. Since the a priori distribution of the data is unknown in real situations, this may mean it is problematic to effectively apply co-training without a reliable estimate of the distribution in the population. Another way to interpret the negative effect of non-matching distribution, is that these runs had too few negative examples to effectively discriminate the positive class from the rest. It may be that after more iterations and more negative examples into the training set, generalization accuracy becomes better.

The current experiment measured the generalization accuracy only before and after co-training. It was assumed that any differences in the generalization accuracy before and after co-training could be interpolated and extrapolated over iterations. For example, when an improvement was found after five iterations, it was assumed that this improvement would at least not become negative over the next iteration. What this experiment did, was let co-training undergo an empirical analysis about its performance, without allowing detailed analysis about the inner mechanical workings of co-training.

In the current experiment, performance was measured on a specific set which would be the population of the data in realistic settings. On the one hand, there are positive results about the possibility of using only a small training set for classification. Co-training can be used to classify a huge unlabeled set, and under most circumstances, most of them would be labeled accurately. On the other hand, co-training users should be aware that performance may be worse than chance if the initial training set is highly unrepresentative.

If one would take co-training out of this laboratory setting, the population and its distribution are not known so there is no direct way to measure the generalization accuracy of the classifiers trained during co-training. In real situations, the accuracy of co-training on some unlabeled set could be estimated by manually evaluating the training set in pilot runs with a limited number of iterations, trying out a limited number of initial training sets. If users are satisfied with the preliminary results of these iterations, they may opt to continue co-training the rest of the unlabeled data with the initial training set yielding the highest accuracy during the pilot runs.

The current initial training set size is twenty instances, and high performances were achieved with this amount. It would be interesting to see how co-training performs after experimenting with even smaller training sets.

No work has been done on co-training with three or more views, although suggestions have been made in this direction [6]. The current work can be extended by, for example, splitting the histogram feature set into two separate feature sets.

In the current work, SVM classifiers pick new instances before k-NN classifiers do, and it is not known if this order biased the results towards SVM classification. The alternative would be to permit an overlap in the picked instances, i.e., that the same instances can be picked by both classifiers. But that would leave us with the possibility that an instance is picked as the positive class by one classifier, and as the negative class by the other classifier, which is the reason why the current work employed an order in the picking. Additional comparison co-training runs where k-NN classifiers can pick first and SVM classifiers can pick next should illuminate whether the order in which classifiers pick instances matters.

The representativeness of the training set, in terms of instance typicalness and matching set distribution, has a large impact on co-training performance, which raises the question how high representativeness in the training set can be guaranteed. As for the instance typicalness, a human expert may increase instance typicalness in the training set by selecting the instances that have a high frequency of similar instances. This should help increase co-training performance, because instances that have a lot of similar 'neighbors' in the population of possible instances have high typicalness in our operational definition of that term, which enhances performance. It would be illuminating to see to what degree human typicalness estimation has the same effect on performance as the statistical typicalness estimation as done in the current work. Letting the distribution of the training instance match the a priori distribution would be harder, as the a priori distribution can only be estimated and never known for certain.

The current work should be considered as a proof of concept. A data set of 3904 instances is rather small, and test results with a larger data set are more reliable. More classes could be added to the data set as well, with which more negative examples can be used during co-training runs.

Overall, we can conclude that co-training is a promising classification method. With only a small training set, the maximum performance can be achieved. What the human expert using co-training can do, is try to pick the most representative instances from the data set as training data.

References

[1] Asano, A. (2004). Support vector machine and kernel method. In: *Pattern Information Processing (2004 Autumn Semester) Session 12.*

[2] Blum, A., and Mitchell, T. (1998). Combining Labeled and Unlabeled Data with Co-Training. In: *Proceedings of the 1998 Conference on Computational Learning Theory,* p. 92-100.

[3] Van den Broek, E.L. (2005). *Human-Centered Content-Based Image Retrieval.* PhD Thesis, Nijmegen: NICI, Radboud University Nijmegen, Netherlands.

[4] Goldman, S., and Zhou, Y. (2000). Enhancing Supervised Learning with Unlabeled Data. In: *Proceedings of the 17th International Conference on Machine Learning,* p. 327-334.

[5] Kiritchenko, S., and Matwin, S. (2000). *Email Classification with Co-Training.* Proceedings of CASCON 2001, p. 192-201, Toronto, Canada, 2001.

[6] Mitchell, T. (1999). *The Role of Unlabeled Data in Supervised Learning.* Proceedings of the Sixth International Colloquium on Cognitive Science.

[7] Nigam, K., and Ghani, R. (2000). *Understanding the Behavior of Co-Training.* In: Proceedings of KDD-2000 Workshop on Text Mining.

[8] Nigam, K., and Ghani, R. (2000). *Analyzing the Effectiveness and Applicability of Co-Training.* In: Proceedings of Ninth International Conference on Information and Knowledge (CIKM-2000), p. 86-93.

[9] Russell, S., and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Pearson Education, Inc..

[10] Zhang, D., and Lee, W.S. (2005). *Validating Co-Training Models for Web Image Classification.* In Proceedings of the SMA Annual Symposium 2005, NUS, Singapore, Jan 2005.

[11] http://www.kyb.tuebingen.mpg.de/bs/people/spider/, last visited on March 2nd, 2009.