Active Learning with Support Vector Machines

Andreas Vlachos

Master of Science
School of Informatics
University of Edinburgh
2004

Abstract

This thesis examines the use of support vector machines for active learning using linear, polynomial and radial basis function kernels. In our experiments we used named entity recognition which was treated as a binary task and as a multiclass task and we also tackled shallow parsing. We report savings in annotation costs ranging from 80% to 95% depending on the task. We observed that the distribution of labels in the selected instances during active learning could provide us with a stopping criterion in cases where one class can be considered to be the majority class of the dataset. Finally, using the confidence estimation of the SVM classifier, we define a stopping criterion that appears to be efficient in all our active learning experiments.

Acknowledgements

I would like to thank my supervisor, Miles Osborne, who guided me throughout this task. I am obliged to Chih-Jen Lin for his help in tuning LIBSVM. Many thanks to Andrew and Christophoros for their help and insight in maths, as well as proofreading my thesis. Many thanks to Beatrice, Ben, Marcus and Shipra for their help in the parallel experiments we ran. Finally, to my friends and colleagues who kept me sane the long hours of work.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Andreas Vlachos*)

To my friends who have made my life fun to live all this time. To my family who have supported me through the years.

Table of Contents

List of Figures

Chapter 1

Introduction

In recent years, Statistical Natural Language Processing has become a very important field of research. The amount of textual information available in electronic formats has increased significantly because of the wide use of computers and the improved storage facilities. Moreover, the Internet and the world wide web have become a facility available to a lot of people. As a consequence, there is increasing interest in machine learning methods to perform various natural language processing tasks in order to make efficient use of these large amounts of information. These tasks include but are not limited to text classification, part-of-speech tagging and named entity recognition.

Research has concentrated on developping and improving machine learning methods for such tasks. Some of the most widely used are maximum entropy, C4.5 and support vector machines. When attempting to tackle a natural language task, researchers experiment with various machine learning methods and a certain annotated dataset that the methods are trained upon. In order to determine which is the most suitable for the task at question, they also use an independent test set for evaluation.

Even though more suitable machine learning methods for a task can yield better results, it is generally accepted that the almost certain way of improving performance is to increase the size of the training dataset. As Banko and Brill (2001) suggest, even with huge datasets, the performance can always be increased by adding more training data. The reason is that the sparsity and the variety of language makes impossible the construction of a training dataset that covers all the possible cases. However, annotations usually come at a cost which is prohibitive to annotate very large amounts of data. Interestingly, the sparsity of the language suggests something more. When using randomly selected data for annotation, we are probably annotating a lot of data that might not be that useful for training a machine learning method because identical or similar instances have already been annotated.

Active learning is a technique that attempts to deal with this problem. It suggests that the data for annotation should be selected in such way that only the most informative examples are included, still achieving performance competitive with using randomly selected data. Thus, the cost of annotation is reduced or, alternatively, better performance can be obtained for the same annotation cost. Typically, the selections for active learning are made by a machine learning method which determines the informativity of the examples.

In this thesis, the machine learning method used for active learning is support vector machines (SVMs). This method has significant theoretical advantages and it has shown impressive performance in many tasks such as text categorization and handwritten digits recognition. Since it is considered as the state-of-the-art in machine learning, it is an attractive candidate to be used for the purpose of active learning.

The results of active learning using support vector machines presented in the thesis are interesting. The methods described involved the use of linear, polynomial and radial basis function kernels, adapted to binary and multiclass classification. In our experiments, they were applied to named entity recognition and shallow parsing, reducing the amount of training data needed by 80% to 95%, depending on the task. Moreover, findings concerning the confidence of the SVM classifier and the distribution of class labels in the selected instances provided ways of defining a stopping criterion for active learning.

The structure of the thesis is the following. Chapter 2 provides the necessary background in active learning and support vector machines. Chapter 3 discusses how SVMs were used for active learning. Chapter 4 describes the tasks and the software used in order to perform experiments. Chapter 5 presents the results of these experiments, along with an evaluation of the methods presented in chapter 3. Chapter 6 discusses the issue of a stopping criterion for active learning. Finally, chapter 7 summarises our results and suggests future work.

Chapter 2

Literature Review

In this chapter a literature review is made in order to provide us with the necessary background to explore the subject of this thesis. The review starts with active learning which is the main axis of the thesis. It then continues with a presentation of support vector machines, which is the machine learning method to be used in active learning. Finally, the multiclass extensions of support vector machines are discussed, which are also going to be used for the purpose of active learning.

2.1 Active Learning

Active learning is a term referring to machine learning frameworks in which the learning algorithm selects the instances to be labeled and included to its training material. Thus, it is expected that the amount of training data needed to train a supervised learning method can be reduced significantly. The motivation behind it is that the cost of manual annotation for producing training material is high, because human annotators are normally involved in the process. In addition, given the Zipf-like distribution of natural language, training material is never enough to cover all the possible cases. Banko and Brill (2001) showed that increasing the amount of training data results in improving the generalization performance, indifferently to the size of the training set. In their experiments they used a task for which training data were available in abundance, which is not the case with most of the tasks. Therefore, instead of annotating random instances to produce training data, active learning suggests to annotate those instances that are expected to maximally benefit the learning method.

The typical active learning setting consists of the following components, as described in Tong and Koller (2000). The data are divided into (typically few) labeled instances X and pool of unlabeled instances U. There is also a learner l which is trained on the labeled data[1] and a query module q. The module q decides which instances of U will be selected to be labeled

[1]Unless the learning method uses transduction, like transductive SVMs in Tong and Koller (2000).

and added in X, which in turn will be used to train l. In a passive learning setting, q selects instances randomly, as opposed to active learning where the most informative instances are chosen.

The efficiency of active learning methods is measured in two ways. The more popular one is the reduction in the training data needed in order to achieve a certain level of performance. The second is the increase in performance for a certain amount of training data. A common baseline for active learning is random selection of data for annotation and incorporation in the training data.

Active learning is very promising in reducing the amount of training data needed and has been applied to various tasks. Baldridge and Osborne (2003) applied it to parse selection and report savings in annotation costs up to 73%. Tong and Koller (2000) and Schohn and Cohn (2000) presented impressive results in text classification using active learning. In another publication, Thompson et al. (1999) apply it to natural language parsing in order to perform information extraction. Active learning has also been applied to spoken language understanding in Tur et al. (2003). Sassano (2002) used it to reduce the training material needed for Japanese word segmentation, reporting that active learning achieved equal performance with random selection using only 17.4%. In the following sections, the two most widely used active learning methods are described, namely uncertainty based sampling and query by committee.

2.1.1 Uncertainty Based Sampling

Uncertainty based sampling (Lewis and Gale (1994), Cohn et al. (1995)) is based on measuring the confidence of the classifier on unseen instances. It is expected that the classifier would benefit more from being trained on instances on which it is more uncertain when attempting to classify them. Uncertainty sampling requires a probabilistic classifier that assigns to unlabeled instances each possible label with a certain probability. Then it computes the entropy for the distribution of each instance and selects the instance or the instances with the highest entropy to be manually annotated and incorporated in the training data. High entropy for an instance suggests that the learner is highly uncertain for the classification it makes. Therefore, the learner would benefit from having such instances annotated as training examples.

2.1.2 Query by Committee

Query by committee (Seung et al. (1992)) is a method which is based on measuring the agreement among a committee of classifiers. The committee of classifiers is trained on the labeled material available and then it is presented with the unlabeled instances. The instances presenting the higher disagreement among the classifiers are manually annotated and incorporated in the training data. One of way of measuring the disagreement is the vote entropy metric (Argamon-Engelson and Dagan (1999)). The intuition behind it is that if a committee of classifiers cannot agree on the label of an instance it can be attributed to the hypothesis that their training set does not include enough or any similar instances, thus giving rise to conflicting decisions by the classifiers.

It must be said that query by committee is benefitted by classifiers that work in a different manner so that their decisions are uncorrelated. However, the purpose of this thesis is to study the use of a specific classifier for active learning, support vector machines. Therefore, query by committee is not considered of interest. Support vector machines are described in the following section.

2.2 Support Vector Machines

Support Vector Machines were introduced as a machine learning method by Cortes and Vapnik (1995). The main intuition behind them is that given a two-class training set they project its datapoints in a higher dimensional space and attempt to specify a maximum-margin separating hyperplane between the datapoints of two classes. This hyperplane is optimal in the sense that it generalizes well to unseen data. A more detailed description of SVMs follows, based on Burges (1998) and Vapnik (1998).

The training input of SVMs consists of datapoints that are vectors of real-valued numbers. The dataset is then projected to higher dimensional feature space, using a function that satisfies Mercer's condition, the kernel function. Explicitly, following the notation from Burges (1998):

Let the datapoints of the dataset be vectors $\{x_1, \ldots, x_n\}$ that belong in the feature space $\mathbb{X} \subseteq \mathbb{R}^d$, associated with their labels $y_i \in \{-1, 1\}$. Let Φ be the function that maps the datapoint to the higher dimensional (possibly infinite dimensional) feature space $\mathbb{H}$:

$$\Phi : \mathbb{X} \mapsto \mathbb{H}$$

There exists a kernel function K or simply kernel defined by:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{2.1}$$

The kernel function is of great importance. As mentioned in Vapnik (1998), in order to train SVMs one does not need the to consider the feature space in its explicit form. This is due to the fact that only the inner products between support vectors and the vectors of the feature space are required. Therefore, the problem that arises from the high dimensional feature space is alleviated, because it allows the computations to take place in the original feature space of the problem. The use of the kernel functions is usually referred to as the ”kernel trick” and it was introduced by Aizerman et al. (1964).

After projecting the datapoints to the higher dimension space, SVMs try to identify the optimal hyperplane that separates the two classes. As mentioned earlier, optimality refers to the generalization ability of the hyperplane. As expected, there can be many more than one separating hyperplane for a specific projection of a dataset (Figure 2.1(a))[2]. The optimal one is the one that separates the data with the maximal margin (Figure 2.1 (b)). SVMs identify the datapoints near the optimal separating hyperplane which are called support vectors. The distance of the support from the separating hyperplane is called the margin of the SVM classifier.

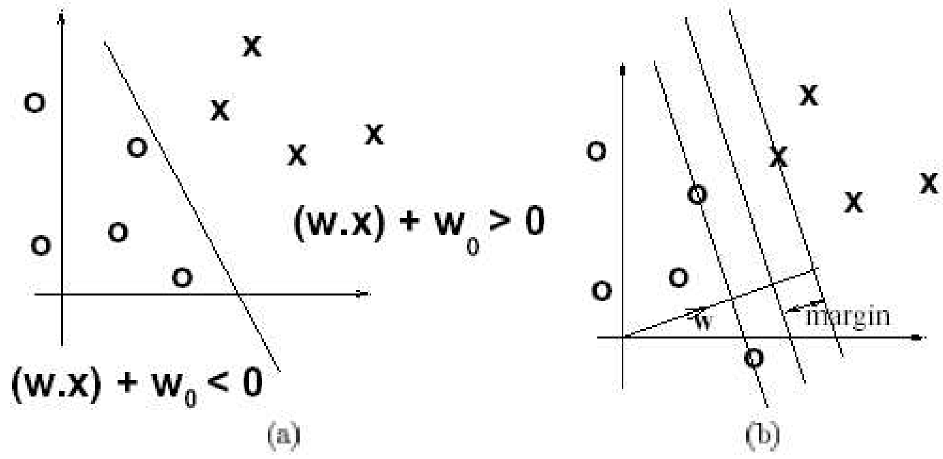


Figure 2.1: (a) A separating hyperplane. (b) The maximum margin separating hyperplane.

During testing, the distance of the unseen datapoints from the separating hyperplane is calculated. Depending on the sign of the value of this distance, the datapoint is classified as belonging to the positive or the negative class. Its calculation requires only the support vectors identified during training. More explicitly, given a set of support vectors $x_1, \dots, x_l$ with their respective labels $y_1, \dots, y_l$, the decision function for a novel datapoint x will be:

$$f(x, \alpha) = sign\left(\sum_i y_i \alpha_i^0 K(x, x_i) + b\right) \tag{2.2}$$

where α_i^0 is a set of parameters to be defined using an optimization process described later.

It should be noted that the non-linear decision function (Eq. 2.2) in the input space $\mathbb{X}$ is equivalent to the following linear decision function (Eq. 2.3) in the higher dimensional space $\mathbb{H}$, because of Mercer's condition (Eq. 2.1):

$$f(x, \alpha) = sign\left(\sum_i y_i \alpha_i^0 < \Phi(x_i)\Phi(x) > +b\right) \tag{2.3}$$

The result of the equation 2.2 is the decision value of the classifier for the datapoint x and

[2]The images were taken from Chris Williams' lecture notes on SVMs, course material for the course Data Mining and Exploration (http://www.inf.ed.ac.uk/teaching/modules/dme/)

represents the distance of the datapoint to the separating hyperplane. In order to specify the α_i parameters, one needs to maximize the functional:

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{2.4}$$

subject to constraints:

$$\begin{aligned} &\sum_{i=1}^{l} \alpha_i y_i = 0, \\ \alpha_i \geq 0, \qquad & i = 1, 2, \ldots, l. \end{aligned} \tag{2.5}$$

It should be noted that the training data might not be separable even when they are projected in the higher dimensional space. In that case a cost parameter C is introduced, which represents the penalty for allowing datapoints to exist on the wrong side of the separating hyperplane. The value of C can vary significantly depending on the dataset, therefore in order to define its optimal value one approach is to perform cross-validation on the training dataset. In this case of non-linearly separable data, the constraints for the α_i parameters becomes:

$$\begin{aligned} &\sum_{i=1}^{l} \alpha_i y_i = 0, \\ 0 \geq \alpha_i \geq C, \qquad & i = 1, 2, \ldots, l. \end{aligned} \tag{2.6}$$

For more details on SVMs, the interested reader should refer to Burges (1998) and Vapnik (1998).

Kernel functions that are commonly used are the polynomial kernel ($K(x_i, x_j) = (x_i \cdot x_j + 1)^p$) and the radial basis function (rbf) kernel ($K(x_i, x_j) = e^{\|x_i - x_j\|^2 / 2\sigma^2}$). Another kernel widely used is the linear kernel, which is in effect a polynomial kernel of degree 1. It is worth mentioning that SVMs using this simpler kernel can be optimized in order to run in significantly smaller times than those using the more general polynomial kernel. The parameters p and σ are to be defined by the user, probably with cross-validation.

In spite of the use of the kernel trick, discovering the optimal hyperplane poses a quadratic programming problem, which is superlinear in the size of the dataset ($\Omega(n^2)$, according to Vavasis (1991)). Platt (1999) presented a way of speeding up the process of training SVMs, but the time required still remains an problem.

Apart from improving the training time, one more important consequence of the kernel function is that it defines the shape of the separating hyperplane to be discovered by SVMs. This arises because the hyperplane belongs to the high dimensional feature space $\mathbb{H}$, whose geometry depends on the kernel function and, for nonlinear choices of kernel, is no longer 'flat'. As it would be expected from theory and is verified empirically (Joachims (1998b)), the choice of the kernel and its parameters affects the generalization performance significantly. Usually, the more powerful the kernel is, the better the results are. In Figure 2.2, it can be seen that while SVMs with a linear kernel cannot find a separating hyperplane in the dataset, using a polynomial kernel it becomes feasible, because the later kernel can identify more complicated boundaries[3]. In Figure 2.3, using more complicated datasets, we can observe that the radial basis function kernel succeeds in separating the dataset, while the polynomial kernel fails. Of interest is the fact that very different separating boundaries are discovered by the two kernels. These observations stress the importance of the choice of kernel for a certain task. However, discovery of more complex hypotheses usually comes at the expense of increased training time, because more complex kernels result in more computations. This could be troublesome in applications that require multiple trainings, which is the case with active learning.

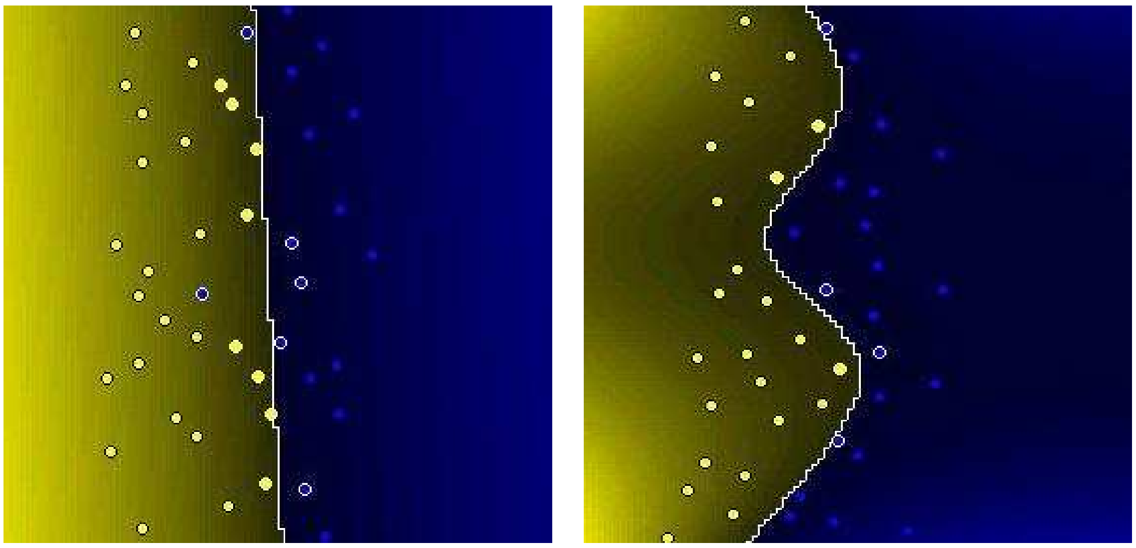


Figure 2.2: Linear SVMs on the left, polynomial SVMs on the right. Note how the polynomial kernel discovers a hyperplane that separates the data more efficiently.

It should also be clarified that the decision values produced using the decision function 2.2 during testing should not be related to probabilities. They can take values in $\mathbb{R}$, unlike the probabilities that lie inside $[-1,1]$. The absence of probabilistic output from SVMs could be an obstacle in using them in certain applications. One of the most common active learning methods, uncertainty based sampling, requires probabilistic classifiers.

The decision value though is a measure of the confidence an SVM classifier has in its decision for a datapoint. The larger the absolute value, the more confident the classifier is in the decision, because the decision value represents the distance of the datapoint from the optimal hyperplane. However, it must be said that the decision values produced by SVMs with different kernels and/or datasets are not comparable. Let's say that we have the decision functions f_1 and

[3]The images were created using the SVM applet available at the Computer Learning Research Center at the Royal Holloway University of London (http://www.clrc.rhul.ac.uk/resources/appletsoverview.htm)

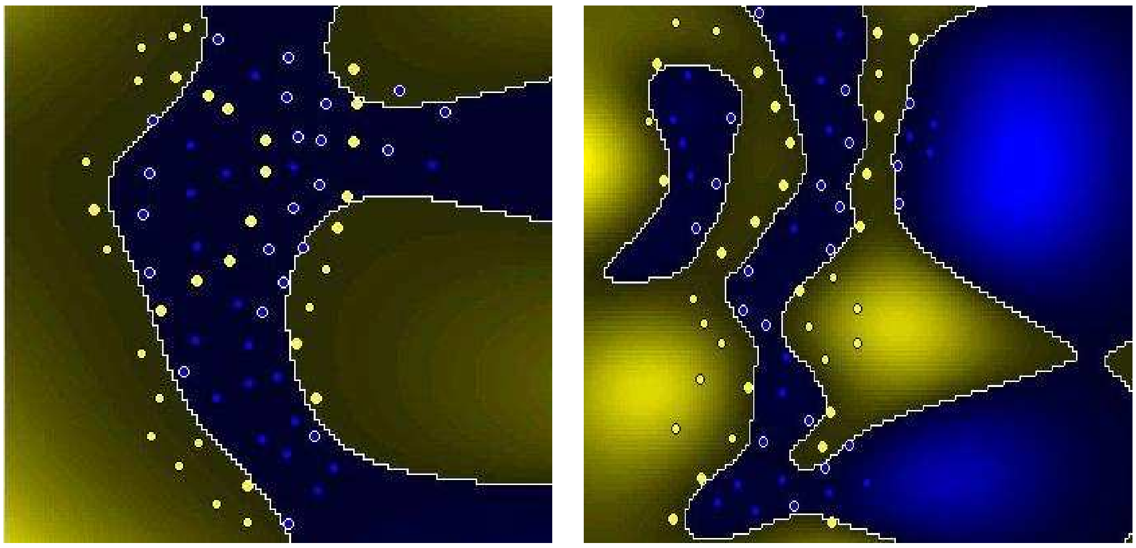


Figure 2.3: Polynomial SVMs on the left, radial basis function SVMs on the right. Note how much different is the hypothesis discovered by the radial basis function SVMs from the hypothesis discovered by the polynomial SVMs.

f_2 that are obtained training SVMs with different parameters and/or different training set. For a certain datapoint x, $f_1(x) = f_2(x)$ does not necessarily mean that both decisions are equally confident. Some research has looked at mapping the decision values produced by SVM to probabilities, such as that of Platt (2000), Sollich (1999) and Vapnik (1998). These methods though need further training. Platt (2000), which is the most widely-used, requires fitting a sigmoid function to perform the mapping, which increases the training time. Moreover, the accuracy of the probability estimates they yield depends on the efficiency of their estimation process which, as expected, cannot be perfect.

Support Vector Machines have significant theoretical advantages over other machine learning methods. They don't require any independence assumptions, unlike Naive Bayes. SVMs are capable of discovering non-linear separating boundaries between classes, while Maximum Entropy can discover only linear ones. Finally, even though SVMs accept only numerical features, categorical features can be used too, by mapping them to numerical features using one-out-of-m encoding. In practice, this means that one can use both numerical and categorical features to describe the instances of the problem, which cannot be done easily with Maximum Entropy or Naive Bayes.

Support Vector Machines have been applied to various problems, demonstrating top or near-top performance. Joachims (1998b) and Dumais et al. (1998) tackled the text categorization task of the Reuters database comparing various machine learning methods. In both papers, SVMs had the best performance. Moreover, Liu et al. (2002) and LeCun et al. (1995) have obtained the best results in handwritten digits recognition using them. Their empirical success makes them attractive candidates to be used in active learning is spite of the non-probabilistic output and the slow training. As it is mentioned in Tong and Koller (2000), the success of active learning is dependent on the performance of the learning algorithm used.

2.3 Multiclass SVMs

Support Vector Machines are binary classifiers in their basic form. Their theoretical advantages and their practical success motivated researchers to investigate extensions to multiclass problems. It should be noted here that with the term multiclass we refer to problems in which any instance is assigned exactly one class label. Such problems are otherwise called mutually exclusive multiclass problems. An example is named entity recognition, as defined in Tjong Kim Sang and De Meulder (2003). This is unlike the Reuters text categorization task, in which each document can be assigned zero, one or more labels. Tasks like this are tackled by building a binary classifier for each possible label, thus making the application of SVMs to the task by Joachims (1998b) and Dumais et al. (1998) straightforward.

Several ways of extending Support Vector Machines to deal with multiclass problems have been suggested in the literature. They can be divided in two categories, following Hsu and Lin (2001). Those that consider the whole dataset with all the classes at once and solve the multiclass problem directly and those that decompose the problem into constructing several binary classifiers and combining their output.

Examples of direct approaches are those presented by Vapnik (1998)[4] and Crammer and Singer (2000). Intuitively, they attempt to find separating boundaries for all the classes in one step. For each class they define a decision rule similar to that of the binary SVMs (Eq. 2.2) and during testing the test datapoint is assigned the label of the decision rule that yielded the highest (positive) margin. Such methods however present numerical difficulties due to the large number of variables that need to be optimized and are rather difficult to implement.

For these reasons, so far, the approaches that decompose the problem into binary classification have been more popular and widely used. The standard method is the one-against-all method (Vapnik (1998)). For each class a binary SVM classifier is constructed, discriminating the datapoints of that class against the rest. Testing is very similar to that of the direct approaches described earlier. Each classifier yields a decision value for the test datapoint and the classifier with the highest positive decision value assigns its label to the datapoint. The comparison between the decision values produced by different SVMs is valid because the training parameters and the dataset remain the same, changing only the labels.

Another indirect way of extending SVMs to solve multiclass problems is the one-against-one method. In this case, for N classes $N(N+1)/2$ classifiers are built, one for each pair of

[4]This method was presented also independently by Weston and Watkins (1999)

classes. In order to obtain a classification from such a group of classifiers two schemes have been suggested and tested. The Max Wins algorithm by Kreßel (1999) performs voting among the classifiers and the selected label is the one with the most votes among the classifiers. Platt et al. (2000) introduced an algorithm, DAGSVM, which places the one-against-one classifiers constructed on a Directed Acyclic Graph and derives the decision from it. It is worth noting that in the one-against-one method the decision values of the classifiers are not comparable because they use different portions of the training set.

In two papers that contain comparisons among these methods (Platt et al. (2000) and Hsu and Lin (2001)), none proves itself significantly better in classification accuracy. From the indirect methods, Max Wins and DAGSVM are proven faster in training than one-against-all. This can be attributed to the fact that while they build more classifiers, each of the classifiers is constructed faster because it takes into consideration a portion of the data. DAGSVM is also a little faster in testing than Max Wins because it does not apply all the decision rules produced during training to each testing datapoint. Comparing the indirect methods to the direct methods, Hsu and Lin (2001) suggest that the latter are far slower to train. Finally, it should be mentioned that like the basic binary SVMs, all the reported multiclass SVMs do not yield probability estimates. This is an important obstacle to overcome in order to use them for active learning.

2.4 Chapter summary

In this chapter, we described the techniques and the methods on which this thesis is going to build. Active learning is a technique used to reduce the amount of training material needed to train a method by selecting the most informative instances. Support vector machines is a top-performing, well-founded machine learning technique. While it was developed for binary classification tasks, there are several extensions of it to multiclass classification. The main disadvantages of SVMs are slow training and non-probabilistic output. In the following chapter we are describing how we overcome these disadvantages in order to use them in active learning for both binary and multiclass classification tasks.

Chapter 3

Active Learning with Support Vector Machines

The theoretical advantages and the empirical success of Support Vector Machines makes them an attractive choice as a learning method to use with active learning. As it has been observed empirically (Tong and Koller (2000)), the efficiency of an active learning method is highly dependent on the performance of the learning algorithm used for querying the unlabeled instances. Apart from reaching higher performance in the classification task, it also achieves greater savings in terms of the labeled data needed. Moreover, as mentioned in Schohn and Cohn (2000) and Campbell et al. (2000), SVMs could also provide us with a stopping criterion for active learning, a subject discussed in chapter 6. In this chapter we present our approach of using SVMs with active learning. In the first part we describe their application to binary classification problems and in the second we extend it to multiclass classification problems. Two are the main issues that should be overcome, the lack of probabilistic output and the slow training phase.

3.1 Binary classification

In order to employ SVMs in the active learning setting, we considered uncertainty based sampling (Lewis and Gale (1994)). The other method described, query by committee, needs more than one different classifiers, so it is not considered. Therefore, uncertainty based learning becomes a natural choice, since we wanted to explore the use of SVMs in active learning. However, as mentioned earlier, this method requires a probabilistic classifier that assigns probability estimates for each class that the datapoint could belong. Unfortunately, SVMs are not probabilistic classifiers. Their output is a decision value whose sign determines which of the two possible labels will be assigned to the datapoint. The absolute value of this output is the distance from the separating hyperplane that was discovered by SVMs during training. It is therefore straightforward to consider the absolute decision value as a measure of confidence of the classifier for its prediction. This intuition is further confirmed by the way the output of SVMs is mapped to probabilities in Platt (2000). In that paper, the fitting of the parameters of the following function is proposed:

$$P(y=1|f(x)) = \frac{1}{1+exp(Af(x)+B)} \tag{3.1}$$

In the above function, y is the label, $f(x)$ is the decision value of the classifier for the instance x and A,B are the parameters that need to be defined. It can be easily proved mathematically that the probability the datapoint in question belongs to the class of positive examples increases monotonically with the decision value f, as long as $A < 0$.

It becomes obvious that, in order to perform uncertainty based sampling using SVMs, we do not need to obtain probabilistic output but we can use the decision values directly. This observation enables us to avoid the training process that is involved in defining the parameters of the function 3.1. Returning to the active learning formulation used in the previous section, the query module becomes as follows. The trained learner, which in our case is SVMs, assigns decision values to the unlabeled data and then it selects those with the least absolute decision value.

Another intuition that we considered to explore was to measure the effect that each candidate datapoint would have in the confidence of the predictions of the classifier. This idea however was not developed further because it would require training SVMs twice for each candidate datapoint. Even so, it was proven useful in defining a stopping criterion in a later chapter.

It must be mentioned that this active learning method has been proposed earlier in the literature, but with different justifications. In Tong and Koller (2000), the argument of the version space (Mitchell (1982)) is used in combination with SVMs using linear kernel. The version space is the set of consistent hypotheses with the training data, which in the case of SVMs is the area in which only the separating hyperplanes belong. It is argued then that by choosing datapoints that lie close to the optimal separating hyperplane is a good heuristic to maximally reduce the version space. This is the method called *Simple Margin* in their paper, which is then further refined using more complex heuristics named *MaxMin Margin* and *Ratio Margin*. The intuition behind both is to select to add the datapoint which maximally reduces the version space of the SVM classifier. In order to achieve this, for each unlabeled datapoint two SVM classifiers have to be trained, one for each possible label the datapoint can take and then the datapoint that results in the biggest reduction is added to the training data. This is not feasible in our experiments because we start with more than 200000 unlabeled datapoints and also because we considered non-linear kernels which are slower to train.

Similar analysis is made in Schohn and Cohn (2000), where they give a more geometrical interpretation of the effect of choosing datapoints that lie close to the separating hyperplane. They also make some interesting observations on the possible performance gains by training on a small but refined subset of randomly selected dataset over training on the randomly selected dataset itself. Furthermore, they discuss the savings in time that result from active learning and introduce a stopping criterion, which we are going to discuss in a later chapter. Both these studies restrict themselves to linear SVMs because of their justification. Campbell et al. (2000), following a different analysis, extended it to SVMs using any kind of kernel. They also defined the same stopping criterion as Schohn and Cohn (2000) and in addition they perform formal analysis of the case where the dataset is not linearly separable.

Our analysis, while it is not as formal as those of the previously mentioned efforts, is very simple and intuitive. It is not restricted to linear kernels and most importantly, it can be extended to the case of multiclass mutually exclusive classification problems.

3.2 Extending to multiclass classification

The successful use of SVMs in active learning for binary classification problems motivated us to explore their application to multiclass problems. Again, the starting point of our approach was to apply uncertainty sampling. As mentioned earlier, none of the variants of multiclass SVMs is probabilistic. Attempts to obtain probabilistic output from multiclass SVMs have been made by Hastie and Tibshirani (1998) and Wu et al. (2004). These methods are applicable to the one-against-one scheme. In both cases, an iterative optimization procedure is required in order to define the parameters needed to obtain probabilistic output. As it can be expected, such procedures could be very time consuming, so we chose to avoid them.

Following our previous analysis, we try to estimate the confidence of an SVMs classifier over a decision without using probabilities. For this purpose, we selected the one-against-all method described earlier. Using this method, the decision values produced by the SVMs classifier built for each class are comparable with each other, so that they can be combined without scaling. Since the decision values were the confidence indicator used for binary classification active learning, we attempted to use them in the case of multiclass classification problems.

More formally, given an n-class classification problem, the one-against-all formulation trains n SVMs classifiers with decision rules $f_i, i = 1, \ldots, n$, like the decision rule of Equation 2.2. Then the classifiers are applied to each instance x of the pool of unlabeled instances, producing n decision values, $f_i(x), \ldots, n$. We present the following ways to combine these decision values in order to obtain a confidence function $c(x)$ that will serve as selection criterion in active learning:

sum A rather naive idea is to sum over all the absolute values of the decision values, which is equivalent to estimating the total confidence on the instance. More formally:

$$c(x) = \sum_{i=1}^{n} \mid f_i(x) \mid \tag{3.2}$$

product Combining the decision values using the **sum** metric however has the following problem. Very large decision values can affect it dramatically, while values close to zero become insignificant. Thus, the decision value of a confident binary classifier is more important to the confidence estimation using **sum**, while the not confident classifiers do not affect it that much. In order to resolve these issues, we used the absolute product of the decision values:

$$c(x) = \prod_{i=1}^{n} \mid f_i(x) \mid \tag{3.3}$$

It should be mentioned that in Tong (2001) exists a similar argument for combining SVMs to perform active learning for non-mutually exclusive multiclass problems.

difference Since we are performing uncertainty based sampling, we tried to imitate the entropy metric which is used with probabilistic classifiers. Recall, the entropy for the label distribution of a probabilistic classifier is defined as:

$$E(x) = -\sum_{i=1}^{n} p(c_i|x) log p(c_i|x) \tag{3.4}$$

It has the property of approximating 0 as the classifier becomes certain of its decision. Obviously, it cannot be used with non-probabilistic classifiers. In our case, the larger the absolute decision value, the greater the confidence of the classifier. For mutually exclusive multiclass problems using one-against-all SVMs, ideally we would like to have one classifier yielding a large positive decision value and all the other classifiers yielding very low negative ones. We would also like to penalize the cases where more than one classifiers yield positive values the same datapoint, which means that they both predict that the datapoint belongs to their positive class. To express these requirements, we subtract from the largest positive decision value all the rest. More formally:

$$i_{max} = argmax_i f_i(x)$$
$$c(x) = f_{i_{max}}(x) - \sum_{i \neq i_{max}} f_i(x) \tag{3.5}$$

difference-2 In Schapire et al. (1997) appears a definition of margin or confidence for classifiers whose prediction is the result of a vote over a set of base classifiers. They define the confidence of a prediction as the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label. In the active learning setting however, the correct label is not known during selecting new instances to include in the training data. In order to adapt it to active learning, we considered as the weight of the correct label the decision value of the label assigned by the multiclass SVMs and we subtract from it the second largest decision value. More formally:

$$\begin{aligned} i_{max} &= argmax_i f_i(x) \\ j &= argmax_{j \neq i_{max}} f_j(x) \\ c(x) &= f_{i_{max}}(x) - f_j(x) \end{aligned} \tag{3.6}$$

Estimating the confidence with Function 3.6 still penalizes the incident that two of the SVM classifiers consider that datapoint belongs to their class, as with Function 3.5.

min Another, rather naive way of defining the confidence of the predictions of multiclass SVMs is to consider the confidence of the whole multiclass classifier equal to the confidence of the least certain classifier. Since the confidence of each classifier is measured with the absolute decision value, then the confidence c over the prediction of the datapoint x becomes:

$$c(x) = min \mid f_i(x) \mid \tag{3.7}$$

In spite of its simplicity, this method performs rather well in our experiments. The reason for this is probably that it allows the binary classifiers to select datapoints that are beneficial to each one of them independently.

At this point it must be said that the methods presented and their analysis is concentrated on using the confidence estimation as an active learning selection criterion. This purpose is very different than actually trying to estimate the confidence of the classifier, for which we are going to perform an analysis of these methods in a later chapter.

3.3 Chapter summary

In this chapter we described in theory how we used Support Vector Machines in the context of active learning. Firstly, we provided a different justification for an existing method of using SVMs for active learning, which is based on using the non-probabilistic output as an indication of confidence. Then we extended this justification in order to use it in multiclass classification tasks. The methods discussed in this section are going to be compared and evaluated in later chapter.

Chapter 4

Experimental setup

In order to evaluate the methods discussed in the previous chapter, we ran various experiments. In this chapter we present the datasets and the software used. The datasets came from two very popular tasks in natural language processing, named entity recognition and shallow parsing. Named entity recognition was treated both as a binary and as a multiclass classification problem. Shallow parsing is a very different task in the sense that it has many classes with widely varying number of instances each. The diversity of the datasets was considered important to the quality of the evaluation. The software was evaluated in terms of its suitability to the objectives of the thesis.

4.1 Data

In order to evaluate the performance in shallow parsing (or chunking) and named entity recognition, we used data from two CoNLL tasks((Sang and Buchholz (2000)) and (Tjong Kim Sang and De Meulder (2003)) respectively). As mentioned earlier, Support Vector Machines take as input only numerical features. Since these tasks come from the Natural Language Processing domain, most of the features are categorical. In order to use SVMs, all the categorical features were mapped to binary numerical features, using one-out-of-m encoding. At this point it should be mentioned that both tasks can be treated as a sequencing tasks. This treatment however would have implications with the active learning process, because we would have to select sequences instead of individual tokens to add to the dataset. Morever, the methods described in the previous chapter are not developed for selecting sequences. In the following sections, the datasets used for each task are described.

4.1.1 Named Entity Recognition

Named entity recognition (NER) is a problem receiving significant attention by researchers. Originally, the task was concerned with identifying sequences of words in texts that are names of people or locations, or organizations. Recently, though it has been extended to other domains, such as biomedical texts. The purpose there is to identify sequences of words that are of special interest, such as names of proteins and cells[1].

In our experiments, we used the English corpus (training and development set) from the CoNLL 2003 shared task[2]. The size of the dataset is 203621 training instances and 51362 testing instances. Named entity recognition was treated in two ways, first as a binary classification task and then as a multiclass classification task.

4.1.1.1 Binary NER

The binary task in Named Entity Recognition is to discriminate entities from non-entities, which is also called segmentation. The class distribution in the dataset is very skewed, only 15% of the words are entities. The features used are extracted from a five-word window with the word in question in the middle of it. Namely, these features are:

- Current word
- Current part-of-speech
- If the first letter is capital
- If all the letters of the word are capitalized
- If it contains digits
- If it contains punctuation marks
- Words and POS tags of the 2 preceeding and the 2 following words
- 2-letter and 3-letter suffixes

After mapping the categorical features to numerical, the total number of (binary) features rises to 114524. Since only a few of these numerical features are active (17) in each instance, the resulting dataset is very sparse. We did not use any features that would depend on previous or next decisions of the classifier, such as whether the previous word is a named entity or not. The reason for this is that when performing active learning, in order to use such features, we would have to annotate sequences of words, while our selection criteria defined in the previous chapter are considering each word independently.

4.1.1.2 Multiclass NER

For the multiclass version of the task, we considered all the entity types defined by Tjong Kim Sang and De Meulder (2003). These are names of persons, locations, organizations and a miscellaneous class containing all the entities that could not be included in any of the previously

[1]For further information on this, one should check the BioNLP/NLPBA shared task (http://research.nii.ac.jp/ collier/workshops/JNLPBA04st.htm).

[2]The corpus is freely available from http://cnts.uia.ac.be/conll2003/ner/.

mentioned categories. The tagging scheme followed in the preparation of the data is the IOB (Ramshaw and Marcus (1995)). For each type of entity there are an I-XXX tag and a B-XXX tag and for non-entities the tag O is used. Words are tagged as O when they do not belong to an entity and as I-XXX when they do. Whenever two entities of the same type are consecutive, then the first word of the second entity is tagged as B-XXX. This is the only case where the B-XXX tags appear. As a consequence of this, the dataset is split into 9 classes, 4 of which have very few instances.

In our case, following the approach of Mayfield et al. (2003) who used SVMs on the same dataset, we considered the B-XXX tags as I-XXX tags. Apart from reducing the sparsity of the data, in our case it reduced significantly the time needed for the experiments. This is because since we are using the one-against-all version of multiclass SVMs, by turning the B-XXX tags to I-XXX tags we reduce the number of SVMs needed to train and test in each active learning round. Finally, the feature set used for the multiclass NER is the same with the one of the segmentation task.

4.1.2 Shallow parsing

The purpose of shallow parsing, as defined in Sang and Buchholz (2000), is to divide text into syntactically related non-overlapping groups of words (chunks). In our experiments, we used the data from the shared task of CoNLL 2000[3]. The sizes of the training set and the testing set are 211727 and 49389 instances respectively. Each word belongs to a syntactic class, such as belonging to a verb phrase (VP) or a noun phrase (NP). As with the NER dataset, the IOB tagging scheme was used, with the difference that the first word of each chunk is always tagged with a B-XXX tag. There are 11 syntactic categories and therefore 23 different classes (2 classes for each tag plus the O class). The number of instances in each class varies significantly, from 1 for the I-LST class to approximately 63000 examples for the I-NP class.

The features for each word used were extracted from a 5-word window, in which the word in question is in the middle. Following Kudoh and Matsumoto (2000) who used SVMs on the same dataset, the words themselves and their respective part-of-speech tags are used as features. After transforming them to numerical, we obtained 92789 features. As with NER, the resulting dataset for shallow parsing is very sparse. We did not consider B-XXX and I-XXX as one class as we did for NER because the slightly different tagging scheme results in substantial numbers of B-XXX tags.

[3]The corpus is freely available from: http://cnts.uia.ac.be/conll2000/chunking/

4.2 Software

There are various software packages implementing Support Vector Machines and most of them are freely available for academic use. Before starting experiments with active learning, we tested some of them in order to evaluate their capabilities and speed. This was considered necessary, because active learning involves training an SVM classifier in each round of selections and given the superlinear time needed to find an optimal hyperplane, any savings in time could be important in order to obtain results in reasonable time. The following packages were evaluated:

SVMTorch SVMTorch is described in Collobert and Bengio (2001). It contains implementations of linear, polynomial, radial basis functions and sigmoidal kernels, as well as support for user defined ones. It is also capable of multiclass classification, using the one-against-all scheme. However, in the multiclass case it does not output the individual classifiers decision values which is necessary for our active learning method. Furthermore, in our experiments it was slower than the other software packages.

LIBSVM LIBSVM is described in Chang and Lin (2001) and was used to perform the comparison among multiclass SVM methods in Hsu and Lin (2001). Apart from multiclass classification it also includes implementation of mapping the output of SVMs to probabilities, in both binary and multiclass classification (Wu et al. (2004)). It supports linear, polynomial, radial basis functions and sigmoidal kernels. The authors provide the users with a script that helps tuning the radial basis functions kernel with cross-validation on the dataset. It doesn't yield decision values, so we could not use it for our active learning experiments. Even so, we used it to tune the parameters for the radial basis functions kernel.

SVMlight SVMlight (Joachims (1998a)) is widely used in the scientific literature. It has been employed in applications of SVMs to various tasks, including NER and shallow parsing (Mayfield et al. (2003) and Kudoh and Matsumoto (2000) respectively). Until the time this evaluation was performed there was no option for multiclass classification[4]. It has support for linear, polynomial and radial basis function kernels, as well as for user-defined ones. Its great advantage is that the linear kernel implementation is much faster compared to the other SVM software packages. This was very helpful for obtaining results is reasonable time. Linear kernel speed, along with outputting decision values were the reasons for using it throughout our experiments. In order to make up for the lack of capability for multiclass classification, we implemented on top of SVMlight the simple one-against-all multiclass SVMs scheme described earlier.

[4]The newest version (6.00) performs multiclass classification but it was not available when the experiments were run.

The above evaluation was performed for the purposes of this thesis and it is not meant to a complete comparison. The performance of a method or software package cannot be determined or compared using only one dataset. The observations and conclusions of this evaluation could be useful to other experiments, however one should keep in mind that software packages change and improve in time, both in performance and capabilities.

Chapter 5

Active Learning Experiments

In this section we present results from active learning experiments described in Chapter 3. In all the experiments that follow, a randomly chosen 1% of the available training data was used as seed data in order to initiate the active learning experiments. In each active learning round, 1% of the available training data is added to the training set used to train the classifier, unless otherwise stated. The test set used to evaluate the performance of the classifiers is the same across all the experiments for each of the tasks considered. The SVM parameters used in the experiments were the default selections of SVMlight for linear and polynomial kernels. For the radial basis function kernel, we used LIBSVM along with a script provided with it to optimize the parameters on a randomly selected 1% of the training data which was used to initiate the active learning experiments.

5.1 Binary classification

As mentioned earlier, in order to evaluate SVMs active learning for binary classification tasks, we used the task of discriminating named entities from non-named entities in text. As it was mentioned earlier, the distribution of the class labels in the dataset of the task is very skewed. The tokens that are entities are the minority class (15% of the dataset). In order to evaluate the performance of the classifier trained in each active learning step we used two evaluation metrics, both based on the F-score (Van Rijsbergen (1979)):

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.1}$$

Precision is the fraction of the correctly predicted minority class units (True Positive) over the total number of minority class units predicted by the classifier (True Positive plus False Positive). Recall is the fraction of the correctly predicted minority class units (True Positive) over the total number of the minority class units existing in the testing set (True Positive plus False Negative). More formally:

$$Precision = \frac{TP}{TP+FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP+FN} \tag{5.3}$$

The first evaluation method used was the F-score (Eq. 5.1), calculating precision and recall considering each token of the corpus individually. The second evaluation method used was calculating the F-score again but considering sequences of tokens of the same type as the units for estimating precision and recall. In order to avoid confusion, we will refer to them as F-simple and F-sequence respectively. In order to illustrate the difference more clearly, we give an example from named entity recognition. Assume that the lines in the box of Figure 5.1 are taken from the output of a classifier. The first column contains the token, the second the true label and the third the predicted label, while the ”O” label represents the majority class and ”ENT” the minority.

The	O	O
European	ENT	ENT
Union	ENT	O
rejects	O	O
the	O	O
American	ENT	ENT
proposal	O	O

Figure 5.1: Example from named entity recognition used to compare F-score and F-sequence.

Precision calculated for the F-simple measure would be 1, since all the tokens predicted to belong to the minority class belonged there indeed. Recall would be 2/3, since the minority token ”Union” was classified as belonging to the majority class. Therefore, F-simple becomes 0.8. For the F-sequence, precision and recall are calculated on sequences of tokens. In order to count a correct prediction for precision or recall the classifier needs to identify a sequence of minority class tokens correctly. Consequently, precision in this case is 0.5 since only one of the two sequences of ”ENT” tags corresponded to an actual sequence of ”ENT” tokens, which in our example is the word ”American”. The sequence ”European Union” which was partly recognized as ”ENT” does not count. Calculated in a similar manner, recall is 0.5 and therefore F-sequence becomes 0.5. It is clear that F-sequence is a harder metric.

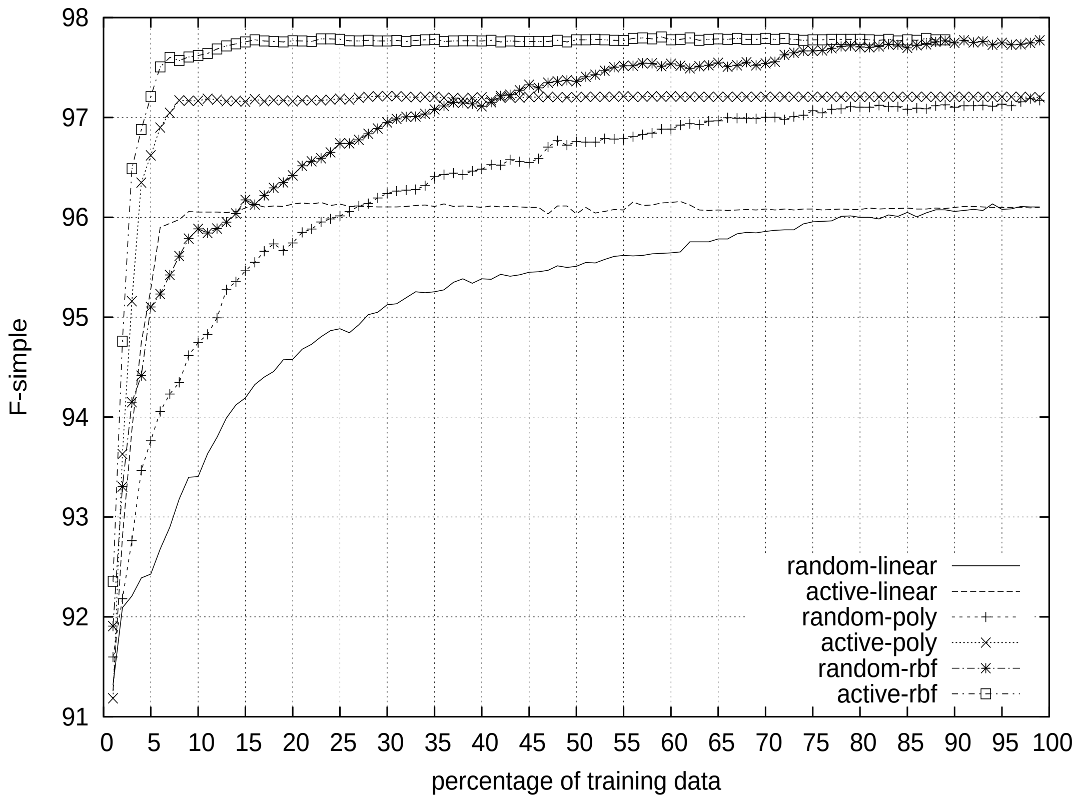
While the F-simple should be considered more appropriate to evaluate the performance, since both the active learning method and the classifier produced consider each token individually without making use of any sequencing information, there are good reasons for reporting the results with the F-sequence. Firstly, our results can be compared with those of other researchers using the same corpus, since the F-sequence is computed by the official evaluation script provided by CoNLL. Secondly and more interestingly, we have an opportunity to see how the lack of sequencing information affects the performance of the trained classifiers when judged as a sequencing task. As Baldridge and Osborne (2004) suggest, a desirable property of an active learning method is that the annotations made using a certain combination of machine learning algorithm and features can be reused effectively with a different combination of features and/or estimation procedure. This is important because the machine learning methods and the feature sets might evolve over time. In our case, such evolution could mean adding sequencing features to the feature set, so if our active learning method performs well even when judged as sequencing task without using sequencing information, then we have a strong indication that selections made would be useful when such information is used.

5.1.1 Active learning with a variety of kernels

In this section we present results using the method described in section 3.1 for binary classification active learning. In Figure 5.2, we evaluate the performance of SVMs with three different kernels using the two evaluation metrics described earlier. The graph on the top is produced using F-simple and the graph in the bottom using F-sequence. In each graph, two curves for each of the three kernels (linear, polynomial, radial basis function) appear, one for random selection and one for active learning. Using both ways of evaluating the performance of our classifiers, we achieve savings of 90-95%, depending on the kernel.

A first observation is that the curves obtained using the two metrics are very similar. The main difference is that the performances estimated with F-sequence are a little lower than those estimated with F-simple, which is expected since F-sequence is a harder metric. Also, the performance peaks later in the active learning curves when measured with F-sequence. This is more noticeable in the case of the linear kernel, in which when measured using F-sequence the performance initially rises steeply but close to the peak the curve flattens. However, these differences do not alter the shape of the curves significantly. Therefore we can assume that the active learning method used could select training instances for a learning method that uses sequencing information, which is a desirable property as it was mentioned earlier. Even so, further testing is required in order to confirm this hypothesis. Since the curved obtained using the two metrics are similar, further results will be reported using F-sequence in order to be compatible with the results reported by other researchers.

The performance of the active learning compared to random selection for each kernel is very impressive. For all the kernels the top performance is reached using less than 10% of the available training data. Also, the performance rises as we move from linear to polynomial and to radial basis function kernel. This agrees with the comment in Tong and Koller (2000) that



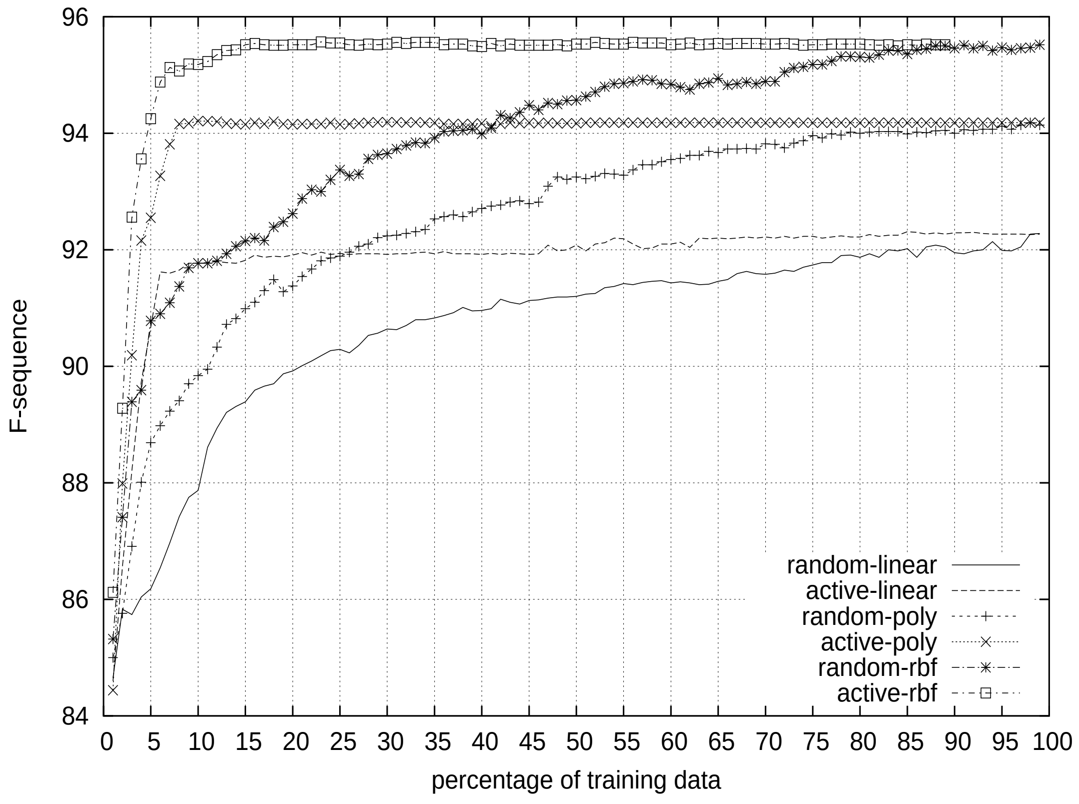


Figure 5.2: Learning curves with SVMs using various kernels. At the top the evaluation is performed with F-simple and at the bottom with F-sequence.

better learning method results in more efficient active learning. For a given training set size, SVMs with radial basis function kernel perform better than SVMs with a polynomial kernel and in turn SVMs with a polynomial kernel perform better than SVMs with a linear kernel.

The results we present in this thesis cannot be directly compared to earlier work (Tong and Koller (2000), Schohn and Cohn (2000), Campbell et al. (2000)) because different dataset was used. However, since the active learning method used is the same in spite of the different justification, the excellent performance of it can be confirmed by our experiments. Interestingly, in line with Schohn and Cohn (2000), we observed that the performance was often a little better using less but selected data than using the full dataset. According to Campbell et al. (2000), the efficiency of active learning in the entity segmentation task shows that the dataset is rather sparse. Even so, assuming that similar datasets are very common in natural language processing tasks, active learning appears as a very promising technique.

We also explored the effect of granularity in our active learning further. Therefore, we reduced the amount of training data added in each round of active learning from 1% to 0.1%. The graphs in figure 5.3 show learning curves for each kernel. In each graph, for a certain kernel, curves with random selection, active learning adding 1% and active learning adding 0.1%. It should be noted that we kept the percentage of the initial training data the same in all the experiments, 1% of the available training data randomly selected. In order to demonstrate the effect of granularity better, we concentrated in the initial part of the learning curves. In all three graphs, we confirm the observation of Schohn and Cohn (2000), that increasing the granularity results in steeper active learning curves. This is expected because in each active learning round the classifiers trained are improved and their estimation of uncertainty over a certain training instance changes. Therefore, by updating the trained classifiers more frequently we can achieve equal performance with fewer data.

At this point though, it should be noted that since smaller granularity requires more frequent re-training of the classifier, it increases the total running time needed for the active learning. In the case of SVMs where the training time is $\Omega(n^2)$ in the size of the dataset, the running time could be a problem, depending on the size of the dataset. In addition, in a realistic active learning setting increasing the granularity might not be always possible or beneficial. In the experiments presented in this thesis, as well as the experiments in most active learning publications, all the training data is annotated before the active learning experiments start. When using active learning employing human annotators, asking for very few annotations in each round might be inefficient and/or expensive.

5.1.2 Active learning using different kernels for selecting and testing

As it is suggested in Baldridge and Osborne (2004), we ran some experiments trying to evaluate how useful would be the selections of active learning using a certain learning method when used

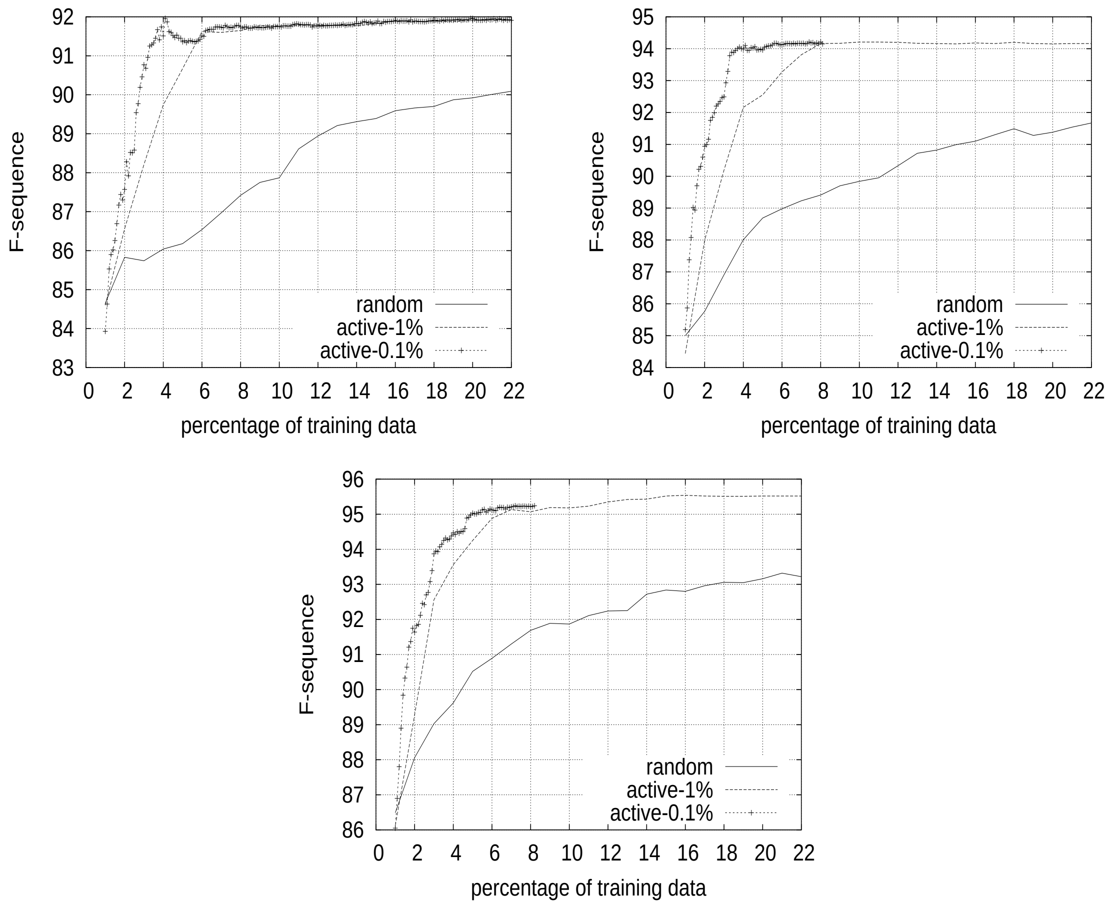


Figure 5.3: Learning curves with SVMs using various kernels demonstrating the effect of granularity. Top-left linear, top-right polynomial, bottom centre radial basis function.

to train a different learning method. This scenario is of great importance in order to evaluate the efficiency of active learning methods, because ideally we would like to reuse the data selected and annotated in later experiments with (hopefully) improved learning methods.

In order to test this hypothesis, we considered SVMs with two of the kernels used in the experiments of the previous section, the linear and the radial basis function. They were chosen because they represent the worst and the best learning methods respectively. We ran two experiments, one using the linear kernel to select instances for the radial basis function kernel and one with the roles reversed.

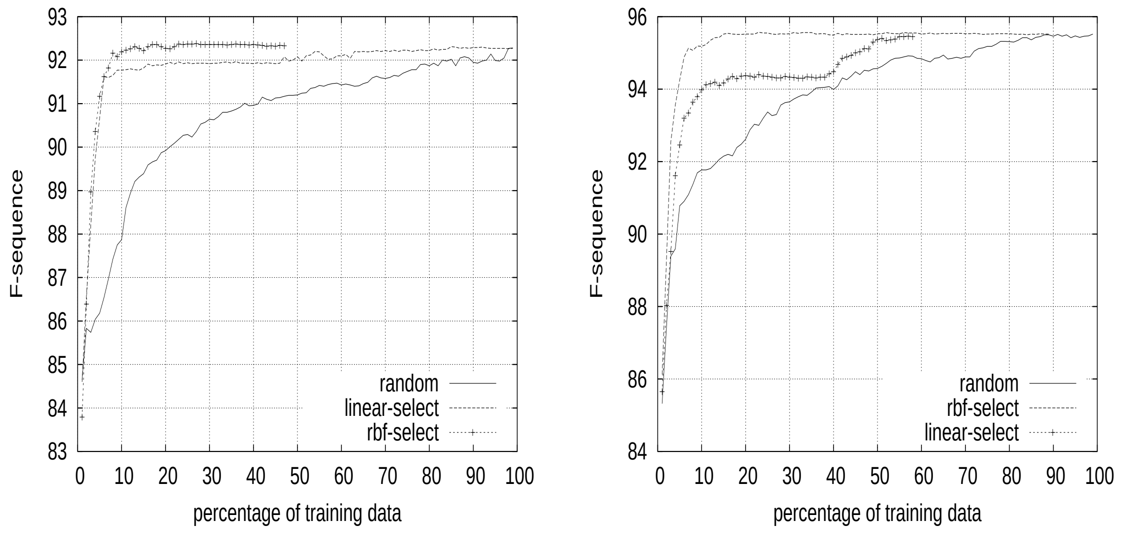


Figure 5.4: Learning curves using selections of SVMs with a certain kernel to train SVMs with a different kernel. On the left, training linear SVMs with radial basis function SVMs selecting instances. On the right, training radial basis function SVMs with linear SVMs selecting instances.

The results of these experiments are shown in Figure 5.4. Each of the graphs shows the performance of SVMs with certain kernel trained with selections made by itself, with selections made by SVMs using a different kernel and with random selections. On the left graph we observe that when SVMs with linear kernel are trained with selections made by SVMs using radial basis function the performance is even better than when SVMs with linear kernel select for themselves. This could be expected, since SVMs with radial basis function kernel perform better. These results further confirm the argument that superior learning method results in more efficient active learning.

When the roles are reversed (right graph), SVMs with linear kernel select instances for SVMs with radial basis function kernel less efficiently than when the latter select for themselves. It appears that the selections made by SVMs with linear kernel are very inefficient, especially near the peak of the performance. This could be attributed to the fact that the hypothesis discovered by linear SVMs is very different and much simpler than that of the rbf-SVMs.

This inhibits them from finding the necessary datapoints in order to refine the more complex hypothesis of rbf-SVMs. Even so, the performance of rbf-SVMs trained on the selections of linear SVMs is much better than using random selections.

Baldridge and Osborne (2004), who ran similar experiments for parse selection, report that in some cases the performance of a learning method trained on data selected by a different learning method is worse than when trained on randomly selected data. Compared to their experiments, the methods we used are not totally different, since we used the same feature sets and SVMs changing only the kernels. Even so, as it was mentioned in an earlier chapter, the choice of kernel affects the hypotheses discovered significantly as it was shown in Figures 2.2 and 2.3. Therefore, the observation that the selections made by the linear kernel perform significantly better than random selection when training the SVMs with radial basis function kernel is very promising.

5.2 Multiclass classification

In order to evaluate the methods presented for multiclass SVM active learning, we used the tasks of named entity recognition and shallow parsing. The main difference between them is the number of classes in each task, which is 5 and 23 respectively. In order to evaluate the performance of the learned classifiers we used the evaluation script provided by CoNLL with the corpuses, which calculates the F-sequence metric described earlier.

In Figure 5.5, we present the learning curves for named entity recognition. Each curve corresponds to one of the selection criteria of estimating the confidence of the classifier over the unlabeled datapoints that were presented in chapter 3. In addition, the learning curve for random selection is included. For clarity, only the initial part of the curves is shown.

As it can be observed, SVM active learning, using any of the criteria presented for estimating the confidence, performs significantly better than random selection. The active learning curves do not vary significantly. Even so, several comments can be made. The confidence estimation using *sum* (Eq. 3.2) performs worse than *prod* (Eq. 3.3), a finding that we had predicted in the analysis of the methods. Among the rest, *diff* (Eq. 3.5) and *diff-2* (Eq. 3.6) demonstrate the steepest learning curves. However, they reach the peak of the performance later than the *min* (Eq. 3.7) criterion. This is shown more clearly in the left graph of Figure 5.6.

This observation can be justified by the way *diff* is defined, compared to *min*. The *diff* method estimates the uncertainty of the unlabeled datapoints, by combining the decision values of all the binary classifiers that are used for multiclass SVMs. Therefore, it selects instances that are maximally informative for all the binary classifiers at the same time. This can be very beneficial, especially in the early stages of active learning. However, when a certain level of performance has been reached, *diff* is not effective in identifying the datapoints that would

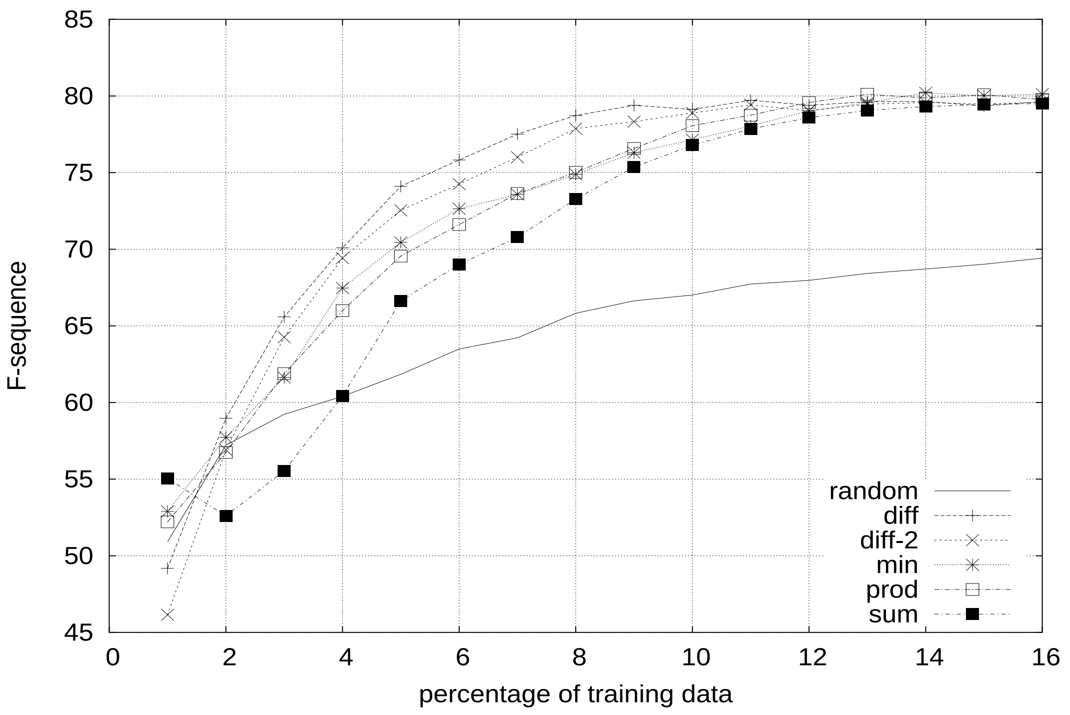


Figure 5.5: Learning curves for the NER task. Each curve corresponds to a different method of estimating the confidence.

contribute in reaching the performance peak. The same arguments apply to a lesser extent to *diff-2*, which takes into account only the two largest decision values and behaves very similarly to *diff*. On the other hand, *min* selects the instances that are maximally beneficial for one of the binary classifiers, irrespectively of the significance of the instances to the other classifiers. This behavior is useful when attempting to reach the peak of the performance for a certain training dataset, because the selections are tailored to the individual needs of each classifier. The drawback in this case is that the learning curve in the beginning is not very steep.

Recalling the purpose of active learning which is to reduce the annotation cost, *diff* and the similarly behaving *diff-2* seem as the best choices. The cost of increasing the performance from a relatively high level to its peak is usually too much and the performance gain too little. One should probably consider replenishing the pool of unlabeled instances instead of annotating more instances from the same dataset. Therefore, a steeper learning curve is always desirable. However, there are cases where reaching the peak of the performance is more desirable than maintaining a high performance per cost rate. This could be realistic in situations where replenishing the pool of unlabeled instances is not possible, because data, even unlabeled, are hard to be gathered. Moreover, as Schohn and Cohn (2000) suggest, in cases where all the data are labelled, active learning could reach the peak of performance in less time than naively training on the whole dataset.

In order to combine the advantages demonstrated by *diff* and *min*, we designed a hybrid method, *diff+min*. The latter consists of using the two simple criteria in turns. The results in the right graph of Figure 5.6 demonstrate that *diff+min* is a rather effective compromise between *diff* and *min*. A similar hybrid method is also mentioned in Tong and Koller (2000), in order to combine advantages of the *Ratio* method in the early rounds and of the *Simple* method in later rounds. However, a definition of the point during the active learning process should the *Simple* method start selecting instances is required. Obviously, this point is dependent on the dataset and the granularity used, therefore, in our opinion this method cannot be easily applied.

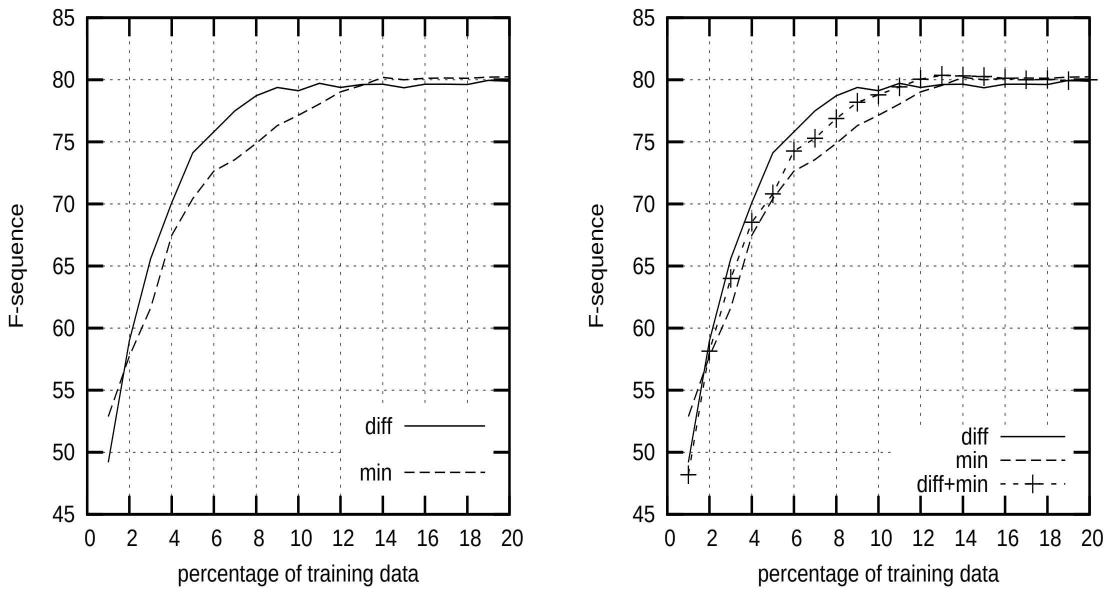


Figure 5.6: On the left, learning curves for *diff* and *min*. On the right, learning curves for *diff*, *min* and the hybrid method *diff+min*.

We also applied the multiclass SVMs active learning method to the task of shallow parsing. The results are presented in Figure 5.7. Again, each curve corresponds to one of the methods of estimating the confidence of the classifier over the unlabeled datapoints that were presented in chapter 3 and the initial part of the curves is shown for clarity.

A first observation is that shallow parsing appears to be a much harder task than named entity recognition, in terms of the amount of training material needed to reach the peak of the performance curve during active learning. While all the criteria perform better than random sampling, not all of them reached the peak of the performance in a few active learning rounds as they did with NER. Most notably, *diff* and *sum* reach it after selecting 50% of the data. The increased difficulty of the dataset could be attributed to the large number of classes (23).

The bad performance of *sum* was expected from the analysis in chapter 3. Adding naively the absolute decision values is certainly not the most efficient way of estimating the confidence

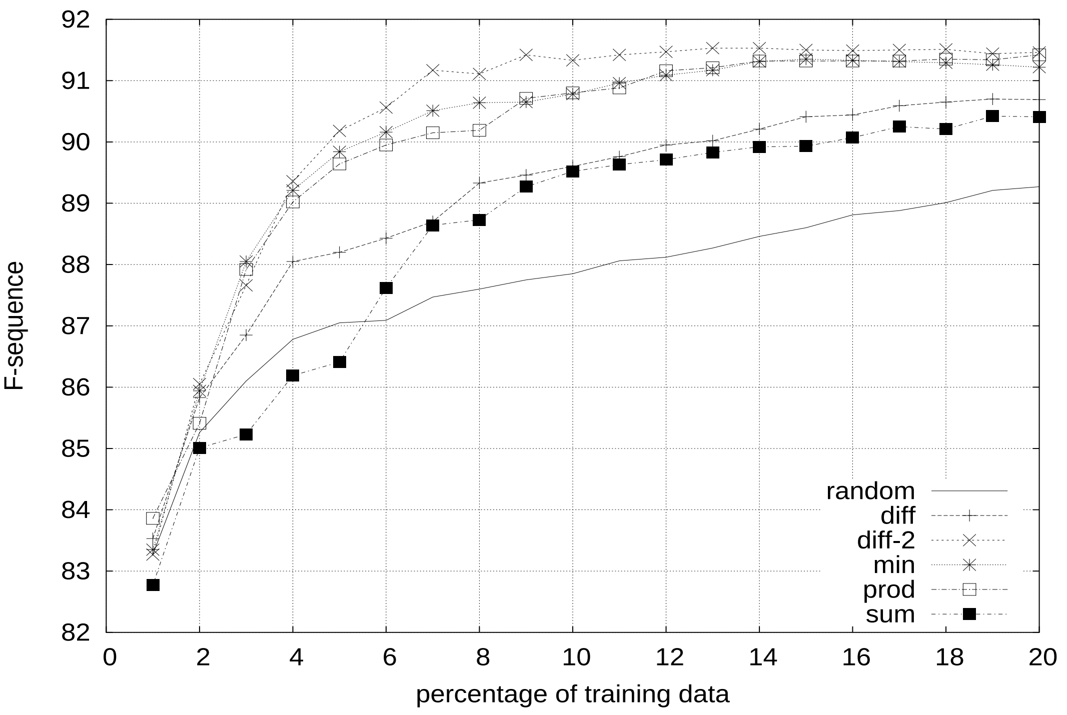


Figure 5.7: Learning curves for the shallow parsing task. Each curve corresponds to a different method of estimating the confidence.

over a datapoint because it can be dominated by the largest absolute values which might not be of importance to estimation. The increase in the number of classes of the dataset exposed this weakness more clearly. *Diff*, while it had the steepest learning curve in NER it does not perform well in shallow parsing. The reason for this is again the large number of classes which results in many decision values that are included in the estimation. Recalling the way the confidence is estimated with *diff* (Eq. 3.5), we subtract from the largest decision value all the rest. In the case of shallow parsing however, we subtract 22 decision values. Obviously, some of them should be quite negative, since some of the classifiers are expected to confidently classify the datapoint in question as not belonging to their positive class. Therefore, the final estimation is affected severely by these decision values, not allowing less confident classifiers to affect it.

The large number of classes appears to be handled more efficiently by the other three criteria, *diff-2*, *min* and *prod*. The first, which is the best performing of all, preserves its efficiency in spite of the large number of classes because it takes into account only the two largest decision values in order to estimate the confidence of the datapoint. In our experiments it achieved the performance peak using only 10% of the data. In a similar fashion, *min* takes into account only the smallest absolute decision value, so it is not affected by the number of classes of the dataset. It should be noted here, that unlike NER, *min* did not reach the peak of the performance earlier than *diff-2*, so we did not develop a hybrid method. This could be also an effect of the large number of classes in the shallow parsing task but more experiments should be conducted in order to explain this behavior. Finally, *prod* behaves similarly with *min* and much better than *sum*.

Concluding from the experiments with multiclass SVMs active learning, it appears that *diff-2* is the most efficient criterion among those presented in chapter 3. It demonstrated impressive performance in both tasks we used for evaluation and it was not affected by the number of classes involved. *Min* also demonstrated efficiency in both tasks. Moreover, it was the fastest to reach the performance peak in NER, which could be of use in cases where the objective is to obtain the absolute maximum of the performance of a dataset with as little data as possible. Finally, *prod* in spite of being rather simple, showed adequate performance. In our opinion, more thorough testing is required in order to evaluate the potential of the methods proposed.

5.3 Chapter summary

In this chapter, we evaluated the methods described in chapter 3 on using support vector machines for active learning. We conducted experiments for binary classification using the segmentation task with various kernels and explored the effect of granularity. The results in all the cases were impressive. Furthermore, we conducted experiments having SVMs with different kernels to select and train. This allowed us to observe that it is possible to reuse the data from active learning sessions with different kernels. Finally, we conducted multiclass experiments with two tasks, named entity recognition and shallow parsing, and observed the effect of the number of classes on the methods evaluated.

Chapter 6

Stopping Criteria for Active Learning

In the experiments presented in the previous chapter, in accordance with the practice followed in the literature, we estimated the savings achieved as the percentage of the training data needed to reach the performance achieved when using all the training data available. Therefore, we are allowed to say that we could have avoided the annotation cost of the rest of the dataset.

While this approach is valid when simulating active learning experiments, in practice it would not be feasible. The reason is that we need to know the performance achieved using all of the training data available, which requires annotating the whole dataset before we start the active learning process. Obviously, this contradicts the purpose of active learning, which is to avoid annotating all the training material available. In the active learning curves presented earlier, the performance rises very steeply during the first active learning rounds and then, as it reaches its peak, the rate the performance increases diminishes until the curve flattens out. Ideally, we would like to take advantage only of the initial rounds, where the improvement in performance in each active learning round is high.

Therefore, a very interesting question in active learning arises. How could someone decide when to stop annotating more data, without knowing in advance the performance using all the available training dataset? Looking at the graphs presented earlier is not very helpful. The active learning curves are steep in the beginning and they flatten out after some point. However, we cannot confidently suggest at any point that the performance would not improve (significantly) by adding more data, unless we actually do so. It should be noted that this is an area with very limited amount of previous research.

In this chapter, we attempt to provide a stopping criterion for active learning based on the use of the methods described earlier. First, we discuss our ideas in the context of active learning for binary classification. Then, we extend them to cover the case of multiclass classification. Finally, we discuss our findings and compare them with earlier work.

6.1 Tracking selections

In order to investigate further the behavior of active learning using SVMs we tracked the selections made in each round. In Figure 6.1, we present graphs for the label distribution of the selected examples for each of the kernels used. Next to each of these graphs, we include the graph of the active learning performance curve and the random selection performance curve of the SVM classifier using that kernel, so that comparisons can be made.

In Figure 6.1, we observe that initially the selections made during active learning by SVMs with all the kernels are balanced. After some percentage of the training set is used, the active learner starts selecting datapoints that are entities less frequently. For every kernel, this happens after the performance has reached its peak and early enough in order to avoid a significant amount of annotation. A stopping criterion for active learning that could be induced from these observations is that when the label distribution becomes skewed, then one should stop annotating more data from this dataset.

We also tracked the selections made in our multiclass experiments. In Figure 6.2, the graph on the left shows the class distribution of the selections made by linear SVMs for the named entity recognition task, using the *diff* criterion, which was the best performing. The graph on the right shows the performance curve of the method compared to random selection.

Recalling from the description of the task, the ”O” class accounts for 85% of the instances and therefore it is considered the majority class. In a fashion similar to the binary version of the task, a ”knee” can be observed in the curve of the number of examples of the majority class that are selected during active learning. This happens after the performance curve has flattened out. Similar behavior is demonstrated by all the other methods as well. These observations provide further evidence to the argument that the distribution of the labels could provide a stopping criterion for active learning. Adapting it to the multiclass case, we could suggest that when the proportion of the majority class instances selected increases then one should stop annotating more instances from the dataset.

We tried to confirm this stopping criterion in our experiments with shallow parsing. However, this was not feasible because of the different distribution of class labels in the dataset. As mentioned in the description of the task, there are 23 classes with cardinality that ranges from 1 to approximately 63000. Most importantly there is no dominant class in the dataset, since the second class in size has around 55000 instances. Consequently, the stopping criterion that one should stop active learning when the proportion of the majority class instances selected increases is not applicable to shallow parsing, at least not in this form.

The distribution of class labels appears as an effective stopping criterion in cases where a dominant class exists in the dataset. However, further investigation is needed to determine if it is a property of the active learning method or an artefact related to the dataset. It could be just a deficiency of the dataset we are using. It might also be of interest to investigate the behavior

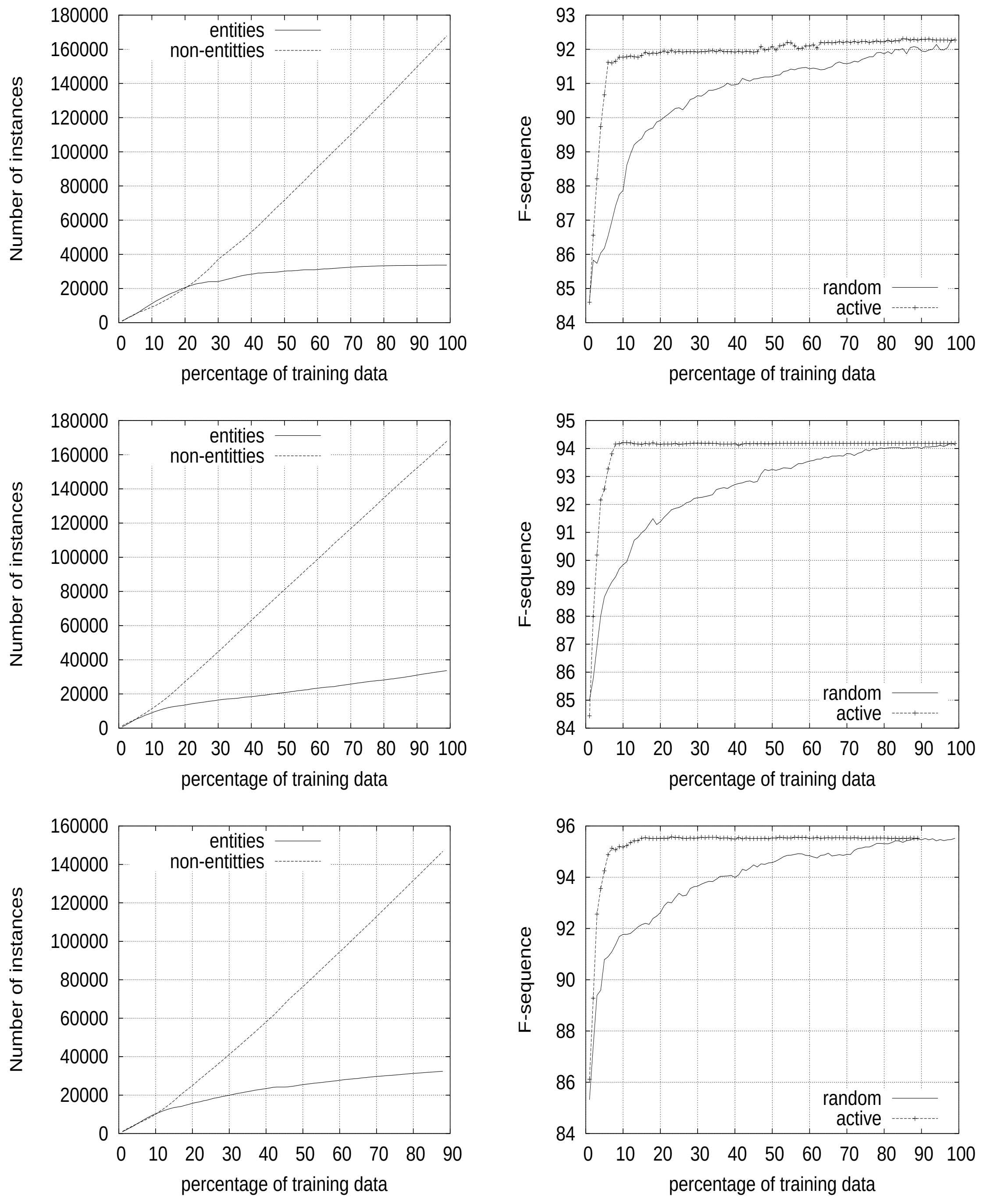


Figure 6.1: Kernels from top to bottom: linear, polynomial, radial basis function. For each kernel, the left graph shows the label distribution and the right graph the performance curve.

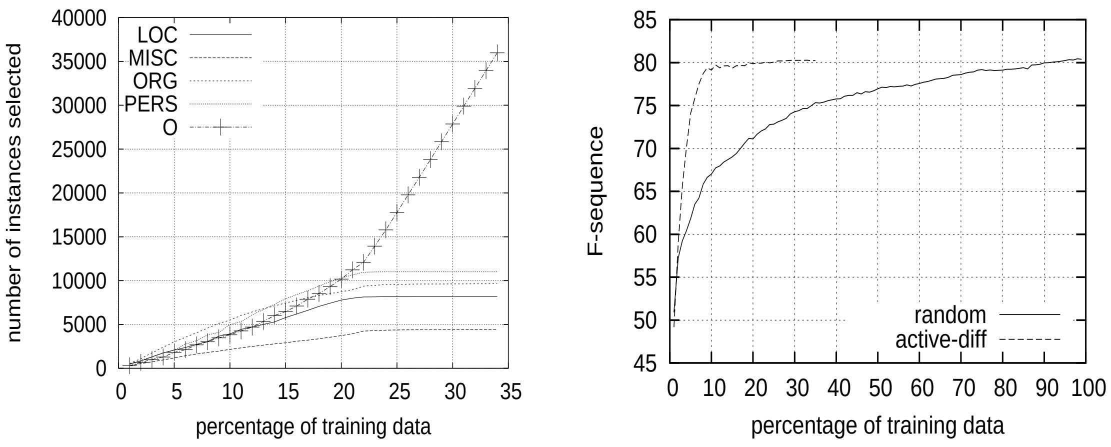


Figure 6.2: The left graph shows the label distribution for active learning with linear kernel for NER and the right graph shows the performance curve.

of active learning selections using methods based on different learning models like maximum entropy.

6.2 Tracking decision values

6.2.1 Binary classification

As mentioned earlier, the performance curves of active learning are not useful in defining a stopping criterion. Therefore we decided to track apart from the performance, the confidence of the classifier too. This idea was inspired by Schapire et al. (1997), where they perform a similar analysis for boosting. Justifying this intuition further, in Freund and Schapire (1999) it is suggested that boosting and support vector machines work in a very similar manner. According to them, both methods attempt, implicitly or explicitly, to maximize the minimal margin from any training datapoint.

In order to perform our analysis, we used as measure of confidence the decision values, which were used as confidence indicators during active learning in order to perform uncertainty sampling. The first way was to estimate the confidence of the classifier on the test set used to measure the performance. For this purpose, in each active learning round we calculated the sum of the absolute decision values yielded from the classifier for the datapoints of the test set.

The results are presented in the left column of graphs in figure 6.3. In each of these graphs, the upper curve is for the confidence estimation using the active learning selections and the lower curve is for the confidence estimation using random selections. The graphs in the right column show the performance of active learning and random selections so that assumptions can be made. As it can be observed, the confidence estimation when training using randomly selected data rises monotonically, in a manner similar to its performance curve. Surprisingly, the confidence estimation when performing active learning rises steeply in the beginning, it surpasses the maximum of the random selection curve, peaks and then decreases gradually to meet the random selection curve when using the full training dataset. Interestingly, the peak is always near the point where the performance curve flattens out. The flattening of the performance curve in active learning terms means that more annotated datapoints from the pool of training material we are currently selecting from would not result in significant gains in performance, or any gains at all if the peak has already been reached.

Experimenting further with the estimation of the confidence using the test set, we tried to make use of the labels, which were available anyway in the experiments in order to estimate the performance of the classifiers. To accomplish this, for each prediction on the test set, if it was correct we added its absolute decision value and if it was wrong we subtracted it. This change in the estimation of the confidence did not change the shape of the curves so the graphs are not presented. Furthermore, in order to estimate the prediction in this manner we need the labels of the test set, which might not be available as they were in our case, increasing the annotation cost. Therefore, estimating the confidence of the classifier summing over the absolute decision values should be preferred.

Tracking the sum of the absolute decision values on the test test appears to be a reliable stopping criterion. However, it requires an independent test set, which does not need to be labeled but it still needs to be excluded from the pool of unlabeled data and needs feature extraction performed on it. Therefore, we attempted to estimate the confidence of the classifier using only the training set.

The first attempt was to use the labeled training data as a confidence estimator. Since the labels for these datapoints are available, we estimated the confidence in the same manner as when we used the labels to estimate the confidence from the testing set, adding the absolute decision value of each correct prediction and subtracting it for each incorrect decision. However, the size of the training set increases as we perform active learning, so we averaged the previous sum over the size of the labeled training set. The results are shown in the left column of graphs in Figure 6.4. Again, in each of these graphs, the upper curve is for the confidence estimation using the active learning selections and the lower curve is for the confidence estimation using random selections. The graphs in the right column show the performance of active learning and random selections so that assumptions can be made.

As it can be observed, when performing random selection the confidence estimation rises steadily as expected, due to the increase of the labeled training set. This is not the case when performing active learning, in which the confidence decreases steeply, reaches a minimum and rises again, to a level higher than were it started.

The initial drop in the confidence estimation is justified by the way the labeled training set changes during active learning. In the early stages, the classifier is not very confident

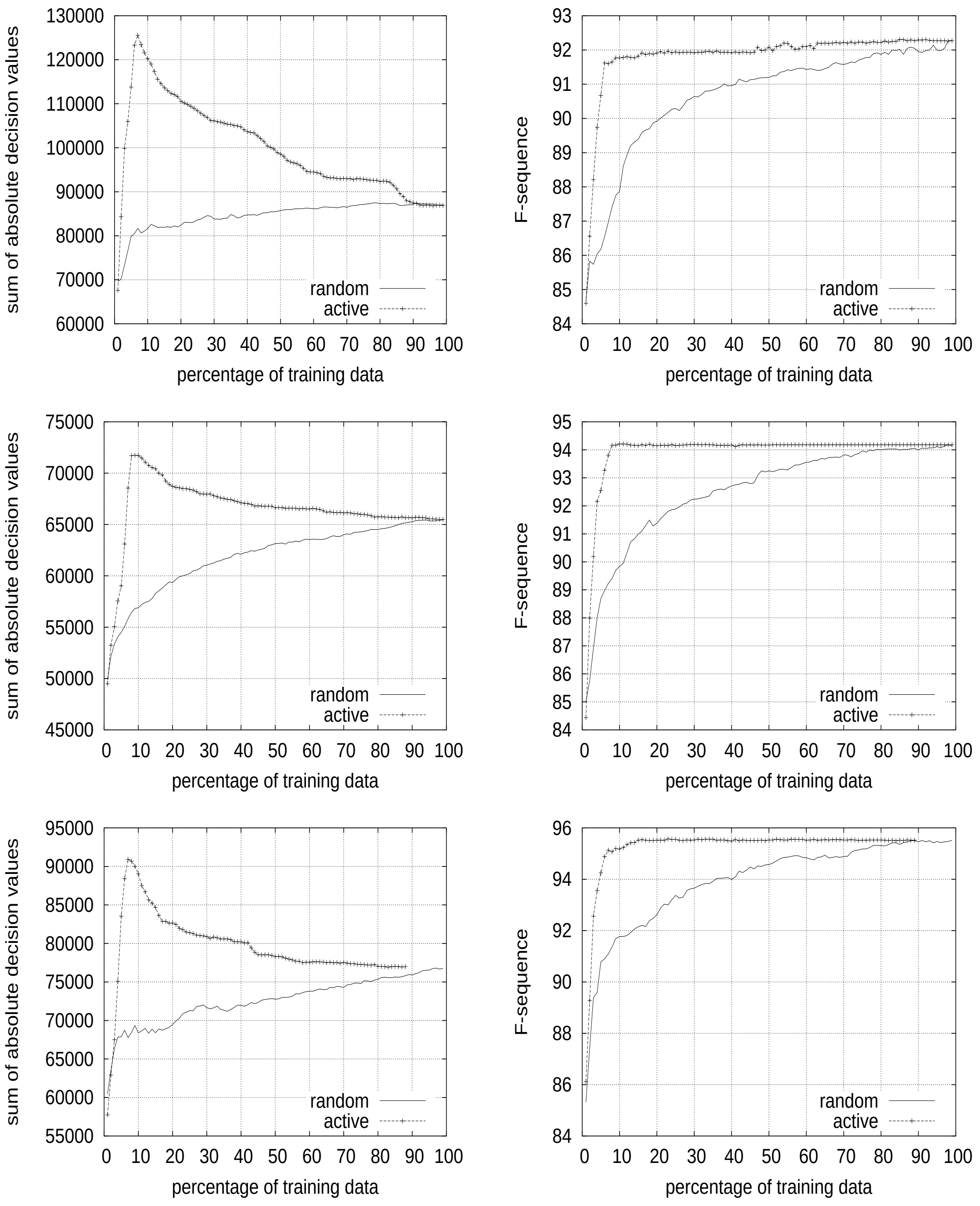


Figure 6.3: Kernels from top to bottom: linear, polynomial, radial basis function. For each kernel, the left graph shows the confidence estimation on the test set and the right graph the performance curve.

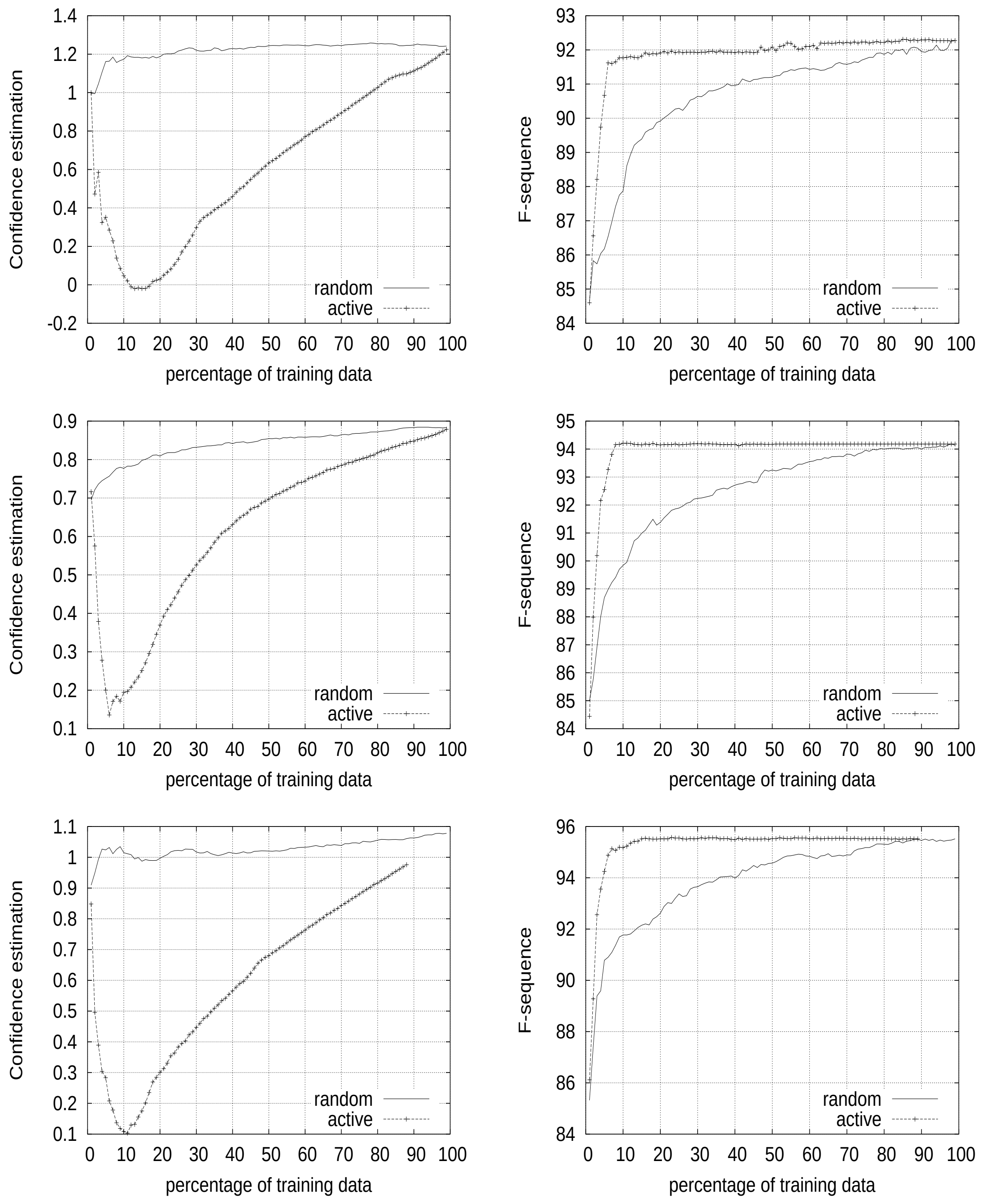


Figure 6.4: Kernels from top to bottom: linear, polynomial, radial basis function. For each kernel, the left graph shows the confidence estimation on the labeled training set and the right graph the performance curve.

and, as it is supposed to do, it adds the hardest instances of the unlabeled training data to the labeled training data. The confidence is probably increased, but so is the difficulty of the dataset on which the confidence estimation is obtained. Therefore, the estimation of the confidence drops, in spite of improving the classifier. In later stages however, since gradually easier examples are added to the labeled dataset, the estimation of confidence increases. On the contrary, the confidence estimation for random selection always increases because the difficulty of the labeled training set remains the same.

Apparently, this way of estimating the confidence of the classifier cannot be trusted because it is affected by the change in the difficulty of the labeled training data, which is inevitable during the active learning process. Since we cannot separate the effect of the changing dataset from the actual confidence estimation of the classifier, it can not be used in defining a stopping criterion. From a more abstract point of view, using only the labeled training data in defining a stopping criterion for active learning means that we are attempting to decide to stop annotating data after some point without taking into consideration what kind of data are available for the task. Obviously, this is very hard and there is no evidence that it is feasible.

Since the problem in the previous way of estimating the performance of the classifier was impaired by the changing dataset used for the purpose, we attempted to use the whole training dataset, labeled or not. Our prediction was that it would provide us with a reliable stopping criterion as the confidence estimation on the testing set, but without the need of an independent set. As the graphs in Figure 6.5 suggest, our prediction was right.

For all the kernels, the curves for estimating the confidence of the classifier by summing the absolute decision values on the datapoints of the training set peak when the performance curves flatten. Therefore, we can use it as a stopping criterion for active learning with SVMs, which suggests that once the confidence of the classifier estimated on all the training material available (labeled and unlabeled) drops, then annotating more of the unlabeled training material would not improve the performance of the classifier significantly. Similar results were obtained from experiments with different granularities, further confirming the usefulness of the criterion. Figure 6.6 shows such experiments with SVMs using linear kernel. Each graph of the right column shows the performance curve for certain granularity and the left graph in the same row show the confidence estimation. A difference that can be observed is that when reducing the amount of data added in each active learning round, apart from reaching the top performance sooner, the peak of the confidence of the classifier is usually higher. Similar graphs were obtained for the other kernels.

So far, we have not provided an explanation to why the confidence of the SVM classifier drops after a certain point in the active learning process. It appears to be rather a paradox, because SVMs are supposed to find the optimal hyperplane in each dataset, so that their confidence should not be decrease.

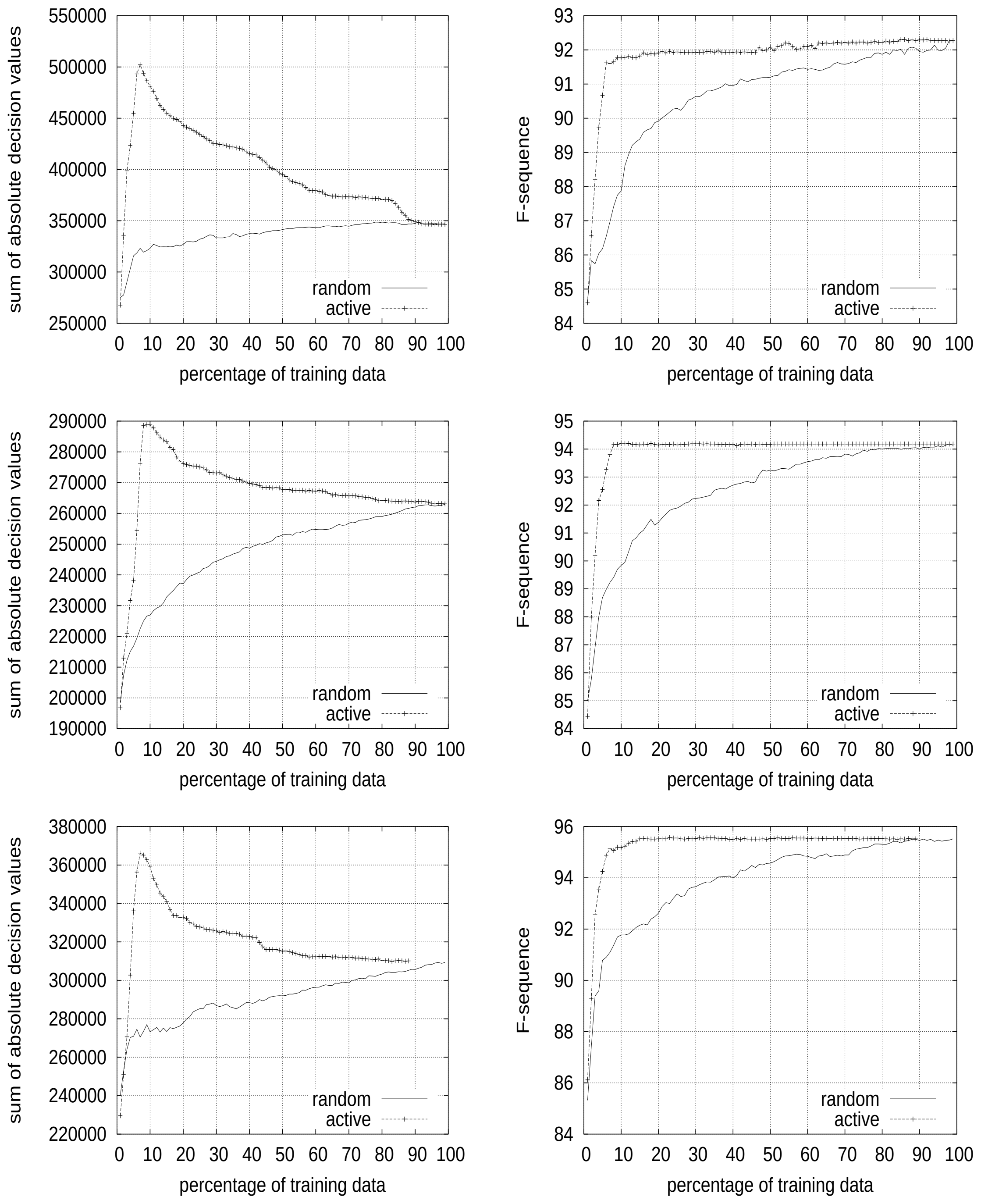


Figure 6.5: Kernels from top to bottom: linear, polynomial, radial basis function. For each kernel, the left graph shows the confidence estimation on the training set and the right graph the performance curve.

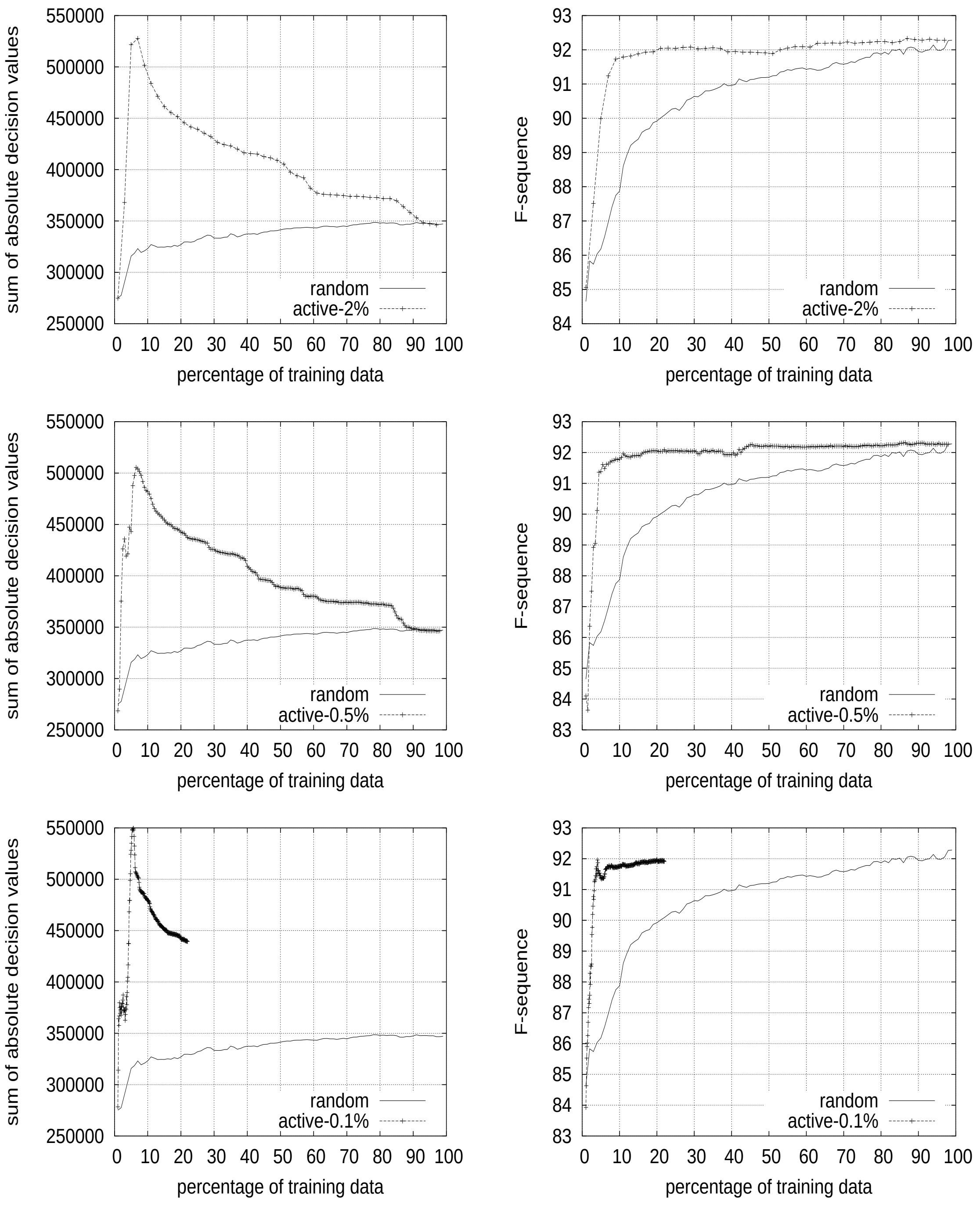


Figure 6.6: Amount of data added in each active learning round from top to bottom: 2%, 0.5%, 0.1%. For level of granularity, the left graph shows the confidence estimation on the training set and the right graph the performance curve.

Schohn and Cohn (2000) make similar observations about the performance of SVMs. In their experiments in text classification they notice that the performance of the SVM classifier is better when trained on fewer but selected examples rather than using all the training material available. In our experiments, this observation was confirmed in most of the cases, especially when using little granularity. This could be attributed to the level of granularity used. Even in the case of adding 0.1% of the data in each round, the batch of examples was 203 examples for the segmentation task, which is far greater than the batch of 8 used in their experiments. Commenting on this, Schohn and Cohn (2000) suggest that it could be the result of noise of the dataset added in later active learning rounds that reduces the performance.

This suggestion could be used to justify the drop in the confidence we observe. Recalling the way the decision value is obtained from an SVM classifier mentioned earlier is of use. We repeat it here for better comprehension:

$$f(x,\alpha) = sign\left(\sum_i y_i \alpha_i^0 K(x,x_i) + b\right) \tag{6.1}$$

In Equation 6.1, in order to obtain the decision value we sum over linear combination of the results of the kernel function applied to the datapoint in question x and each of the support vectors x_i. When noisy datapoints are introduced in the training dataset, some of them become support vectors and therefore they affect the decisions and the decision values. The decisions are less prone to change because they depend only on the sign of the decision values. However, the decision values themselves are more prone to alter. A support vector which is represents a noisy datapoint could reduce the decision value without changing the final decision explaining away the difference in the shape of the performance curve and the confidence estimation curve. It could also be the case that the instances that are selected in later active learning rounds are not very distinctive examples, therefore acting as noise the dataset constructed in the earlier rounds. It must be said at this point that more training material results always in more support vectors and consequently more values to add in Equation 6.1. Therefore, an increase in the confidence estimation is expected as more training data are added, which is the case with random selection. During active learning, the number of support vectors increases more steeply but never drops. This the reason we suggest that the rise-peak-drop pattern observed can be attributed to the informativity of the data added.

6.2.2 Extending to multiclass classification

The idea of using the confidence estimation of the classifier on the training set as a stopping criterion was proven effective in active learning for binary classification tasks. Therefore, we tried to extend it and apply it to multiclass classification tasks.

In chapter 3, we provided 5 ways of estimating the confidence of an one-against-all multiclass SVM classifier over a datapoint. The discussion that followed was concerned with the efficiency of the estimation as a selection criterion for active learning. However, many of the arguments mentioned there do not hold for the purpose of estimating the confidence of the classifier.

Min (Eq. 3.7) for example, which was quite successful as a selection criterion, is certainly a bad choice for prediction estimation, because it takes into account only the least certain of the binary classifiers. As shown in the experiments, it is beneficial for the multiclass classifier to include in its training set a datapoint on which one of the binary classifiers is most uncertain. However, the confidence of the least certain binary classifier is not representative of the overall confidence of the multiclass classifier. As when selecting instances for active learning, *sum* (Eq. 3.2), is expected not to perform well because a few very confident classifiers can overshadow inconfident classifiers. For this reason, *prod* (Eq. 3.3) should be a more accurate estimation. *Diff* (Eq. 3.5) could be a good option, but its efficiency is severely impaired in datasets with a lot of classes. The latter though, should not be a problem for *diff-2*.

In order to compare the efficiency of these methods, we tracked their estimations during the active learning experiments we conducted. At this point, it should be noted that the multiclass methods described in chapter 3 are used in two ways in the experiments that follow. One method is used as the selection criterion for active learning and one method (not necessarily the same) is used to estimate the confidence of the classifier on the training set.

In Figure 6.7, we present graphs for all the confidence estimation methods mentioned. In each of the graphs, one curve is for active learning and one for random selection. They were obtained from experiments for the task of named entity recognition task. The selection criterion for active learning is the *diff*, which showed the best performance in the task. The bottom left graph shows the performance curves for random selection and active learning with *diff*.

The main observation is that all the methods of estimating the confidence yield very similar curves. Moreover, they all follow in broad lines the same pattern with the confidence estimation curves for the experiments with binary classification. That is, they demonstrate a rise in the early stages of active learning and they reach a peak above the maximum confidence estimation reached by random selection. This peak is near the point where the maximum performance is obtained. Finally, they drop until they meet the confidence curve for random selections.

The justification of this behavior is a straightforward extension of the justification given for the binary task. The active learning method seems to be able to identify efficiently the datapoints that are more distinctive instances in order to learn from them. However, more deviations from the pattern described are observed, compared to the binary classification experiments. This is expected, since instead of having one classifier selecting datapoints for itself, all the binary classifiers of multiclass SVMs are combined in the selection process. This results in training each classifier on material that is not optimal for it, consequently reducing its

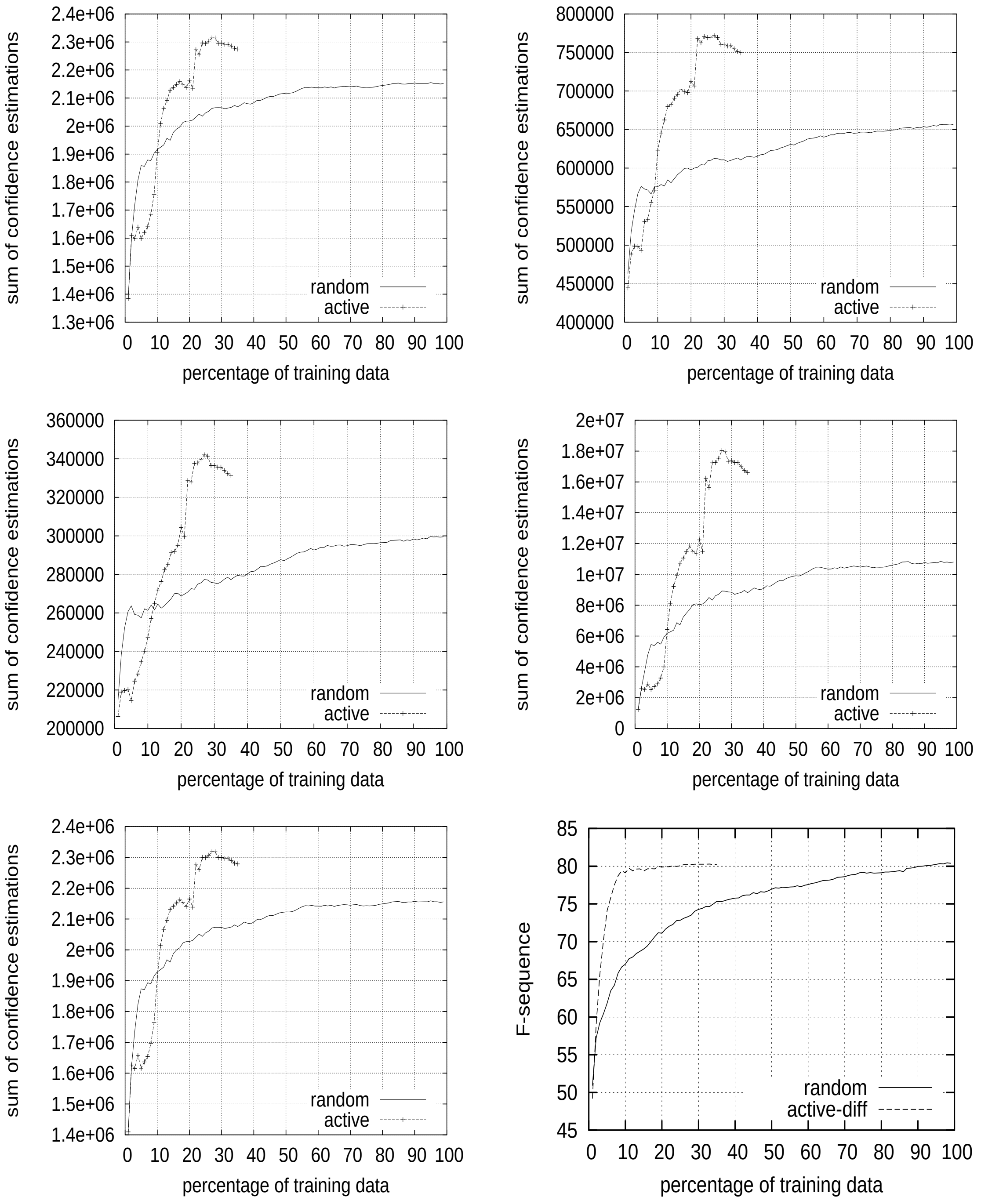


Figure 6.7: Experiments with different with different ways of estimating the confidence of multi-class SVMs classifier. Top to bottom, left to right: *diff*, *diff-2*, *min*, *prod*, *sum*. The bottom right graph shows the performance. The task used is NER and the active learning selection criterion is *diff*.

confidence.

Similar curves were obtained using the other active learning selection criteria. Interestingly though, while they all yielded curves following the same general pattern, some of them appeared to follow the pattern in a more distinctive way. To illustrate this, in Figure 6.8, we present the confidence estimation curves obtained using each of the active learning selection criteria for the NER task. The confidence estimation method used was *diff-2*.

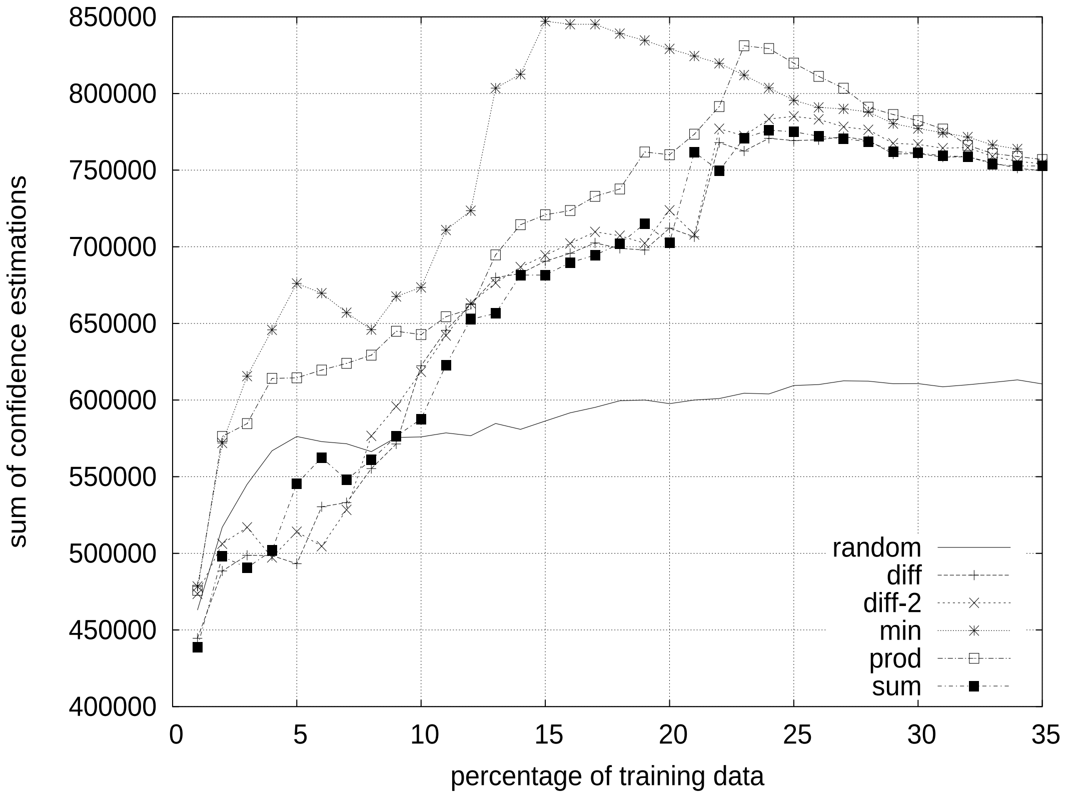


Figure 6.8: Learning curves for the NER task. Each curve corresponds to a different selection criterion for active learning. The method of estimating the confidence was *diff-2*.

The active learning selection criterion that follows more closely the pattern of increase, peak and decrease is *min*, followed by *prod*. *Diff* and *diff-2*, even if they demonstrated better performance in terms of learning curves, they might be troublesome when trying to apply the stopping criterion based on the confidence estimation curve. *Sum* behaves similarly to the previous two methods.

The justification of this phenomenon is similar to the explanation given to the ability of *min* to reach the peak of the performance earlier than *diff* and *diff-2* (Fig. 5.6). Since *min* allows the most uncertain binary classifier to select for itself irrespectfully of the others, the criterion selects instances that are optimal for each of the classifiers individually. This is not allowed by *diff* and *diff-2*. The argument explains also the behavior of *prod*, because in this criterion a very unconfident binary classifier can affect significantly the estimation.

To further evaluate the methods of estimating the prediction confidence, we ran similar experiments with shallow parsing. As before, in Figure 6.9, we present graphs for each of the confidence estimation methods mentioned. In each of the graphs, one curve is for active learning and one for random selection. Finally, the bottom left graph shows the performance curves for random selection and active learning. The selection criterion for active learning is the *diff-2*, which demonstrated the best performance in the task.

As it is shown by these graphs, the confidence estimation method *diff-2* is the one that demonstrated the pattern more distinctively, in spite of the many classes involved in the task. *Sum* and *diff* also behave adequately, even though the drop they present in their confidence estimation is not decisive. *Min*, as expected in the analysis in the beginning of this section, does not follow the pattern at all. Finally, *prod* behaves very differently than the other methods. For active learning, it yields a curve that increases dramatically in the early stages and after it reaches its peak it fluctuates around it. It should be noted that the point where the fluctuations start is close to the point where the performance reaches its peak. For the random selections, the curve produced increases constantly and the rate of the increase is becoming greater as more data are included in the training dataset.

Our explanation is that some of the binary classifiers are very confident even with very little data, while others are not. This is expected since classifiers with only few positive examples, which is the case in the shallow parsing dataset, are likely to be very confident from the early stages. On the other hand, the rather unconfident classifiers increase their confidence as more training data is selected. Since the estimation according to *prod* is the product of the decision values, the unconfident ones in the beginning keep the estimation at low levels. As more data is added to their training set they become more certain of their predictions and therefore the overall confidence estimation increases dramatically. This increase is happenning earlier in the active learning than random selection, because the instances that improve the inconfident classifiers are selected earlier during active learning.

The *diff-2* method of estimating the confidence of the multiclass SVM classifier on the training set appears to be more stable and helpful in order to apply our stopping criterion concerning the performance estimation of the classifier. Therefore we used it to evaluate which of the active learning selection criteria behaves better in terms of following the pattern of rise-peak-drop more closely.

The results are shown in Figure 6.10. As with named entity recognition, *min* demonstrates a distinctive peak which would be useful in applying our stopping criterion. *Diff-2* behaves almost identically, probably because like *min* it does not take into account all the decision values, therefore selecting instances that improve the confidence of each binary classifier significantly. The other three methods demonstrate a peak in their confidence estimation later and not that

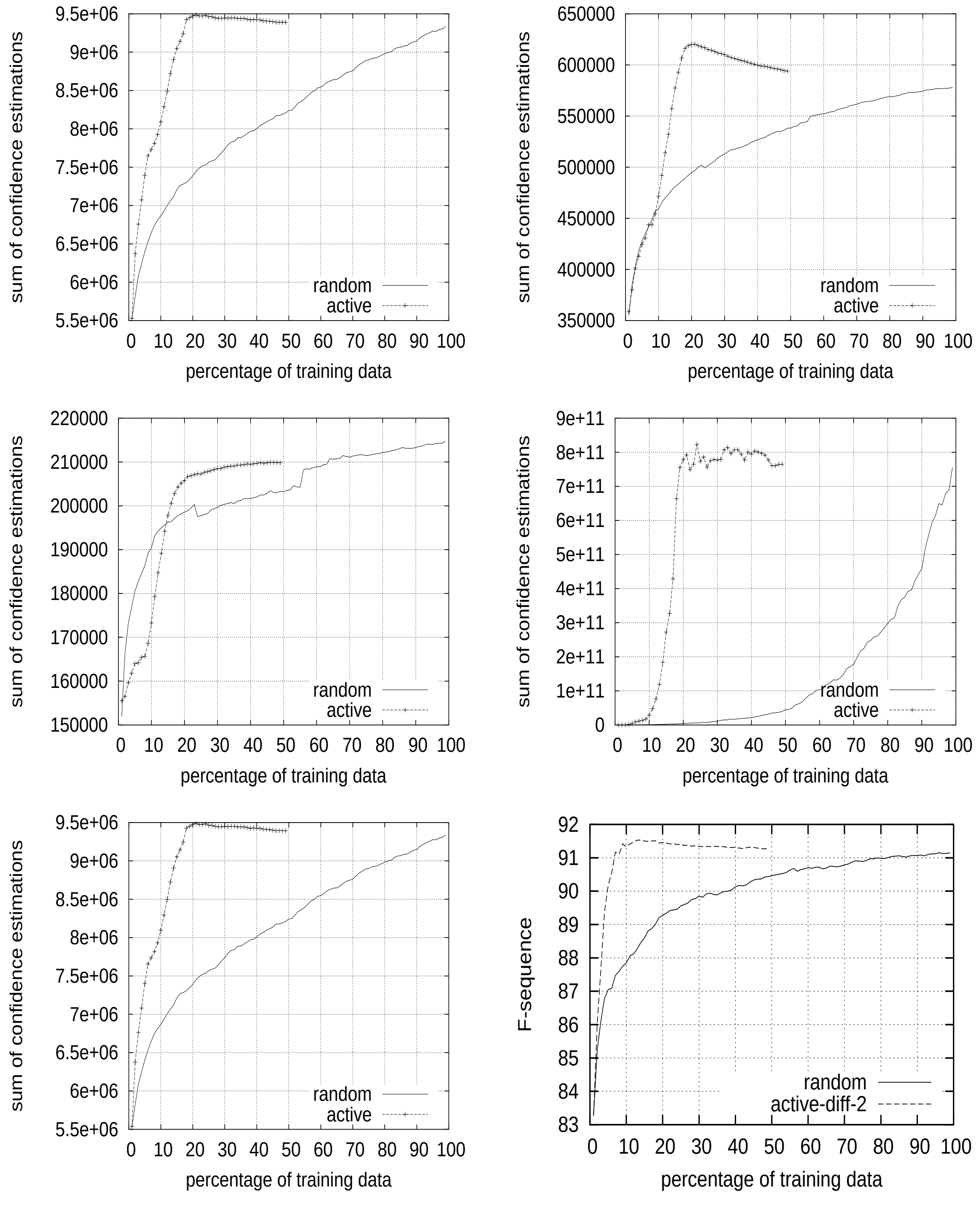


Figure 6.9: Experiments with different ways of estimating the confidence of multiclass SVMs classifier. Top to bottom, left to right: *diff*, *diff-2*, *min*, *prod*, *sum*. The bottom right graph shows the performance. The task used is shallow parsing and the active learning selection criterion is *diff-2*.

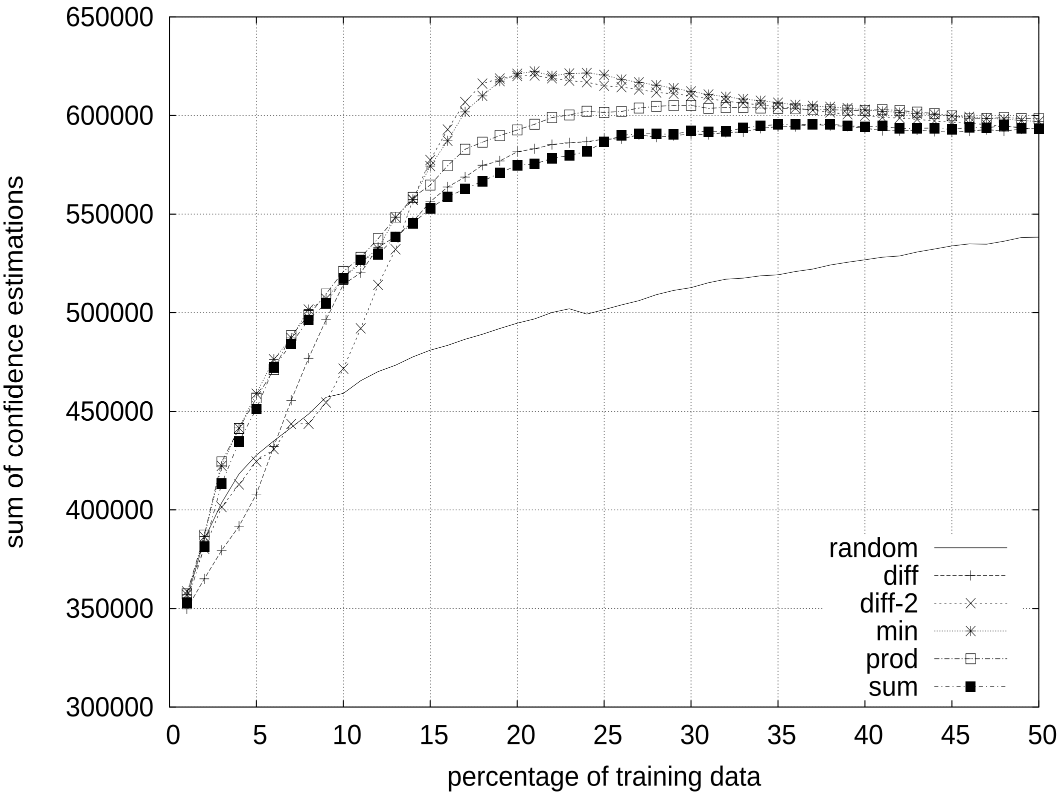


Figure 6.10: Learning curves for the shallow parsing task. Each curve corresponds to a different selection criterion for active learning. The method of estimating the confidence was the *diff-2*.

distinctively.

From the above experiments, we conclude that the best combination of methods is *min* for active learning selections and *diff-2* for confidence estimation. *Min*, while it may not produce the steepest active learning curves, it still performs competitively, especially in tasks with many classes like shallow parsing. Moreover, its confidence estimation curve demonstrates the most distinctive peak among the methods in both tasks, thus making it the best choice in a realistic active learning setting in which the top performance is not known beforehand. *Diff-2,* as a confidence estimation metric behaved well in both tasks. All the confidence curves produced by it followed the pattern of rise-peak-drop more closely than with any of the other methods. For these reasons, this combination of methods is going to be used in the discussion of the following section.

6.3 Previous work - Discussion

The results of the previous section are of interest. The confidence of the classifier appears to be of use as a stopping criterion. We ran experiments with different granularities for the segmentation task and with two different datasets for its multiclass extension. In all the experiments the stopping criterion, which is to stop annotating when a steady drop in the confidence of the resulting classifier is observed, was confirmed.

One note must be made here, about when the confidence estimation criterion we suggest is considered satisfied so that we stop annotating more data. This is important because so far we have not presented any quantitative formulation of it. In many of the experiments performed, the curve of the confidence estimations rises and drops without any local maximums. However, in cases with increased granularity, some fluctuations are observed, as in the bottom left graph of Figure 6.6. Similar fluctuations are observed also in the confidence curves multiclass experiments (Fig. 6.8). While the confidence curve maintains the very distinctive peak near the peak of the performance curve, the early fluctuations inhibit us from stating that at any point the confidence estimation drops one should stop annotating more material, which is correct in most cases. The fluctuation could be explained away by the very limited training data the classifier is trained on at the initial stages of active learning. Further investigation is required in this direction. For the purpose of this thesis, the stopping criterion for active learning is going to be a steady drop in the confidence estimation of the classifier. This can be easily detected by tracking the confidence estimation curve during active learning.

In the literature, one other stopping criterion for SVM active learning is suggested in Schohn and Cohn (2000) and Campbell et al. (2000). Their criterion is based on the idea of the margin of an SVM classifier. As mentioned in chapter 2, the margin of an SVM classifier is the distance of the support vectors from the separating hyperplane. Schohn and Cohn

(2000), based on the geometric justification of the use of the decision values for active learning with linear SVMs, they suggest that one should stop annotating when all the unlabeled examples are outside the margin of the classifier. They assume that such datapoints would not alter the separating hyperplane if they were added to the dataset. In order to determine when a datapoint lies inside the margin, they use the property that these datapoints have absolute decision values less than one. This can be confirmed easily during active learning when estimating the decision values of the unlabeled datapoints. The criterion then becomes that one should stop active learning when none of the candidate datapoints has absolute decision value less than one. Campbell et al. (2000) extended this stopping criterion to non-linear kernels and they added a validation step at which a bound on the remaining training error can be stated with some confidence. This step is supposed to be evaluated by the user and if the latter is satisfied then the active learning stops. Since having a user judging the performance is beyond our consideration, in our discussion we are going to use the simpler form of the criterion presented in Schohn and Cohn (2000).

We compared the two criteria in our experiments. We performed experiments with the three kernels (linear, polynomial, radial basis function) and various granularities. Both criteria appear to behave well in the sense that they are satisfied soon after the performance curve has started to flatten out. Recalling the introduction of the chapter, ideally a stopping criterion should terminate active learning just when the gains in performance from adding more data are considered insignificant or too expensive in terms of annotation costs.

In Figure 6.11, experiments with the linear kernel are made with different granularities. In all the cases, by the active learning round all the unlabeled datapoints are outside the margin of the classifier, a drop in the confidence estimation has been observed earlier. The lag between our stopping criterion to the that of Schohn and Cohn (2000) is increased when selecting more data in each active learning round. Moreover, one should consider the consequences of increased granularity upon running time and the implications that could arise in a realistic active learning setting, which were discussed in the previous chapter. Therefore, the efficiency demonstrated by our stopping criterion when selecting more data in each active learning round is a significant advantage.

Then we investigated the effect of the kernel choice. The graphs in Figure 6.12 compare the behavior of the two criteria using linear, polynomial and radial basis function kernels, selecting 1% of the training data in each active learning round. The results show that the efficiency of the criterion of Schohn and Cohn (2000) decreases when using polynomial and radial basis function kernels.

This observation can be justified using some elements from the description of SVMs in chapter 2. As mentioned, polynomial and radial basis function kernels can identify more complex hypotheses in the training set. As a consequence, each time new datapoints are added to

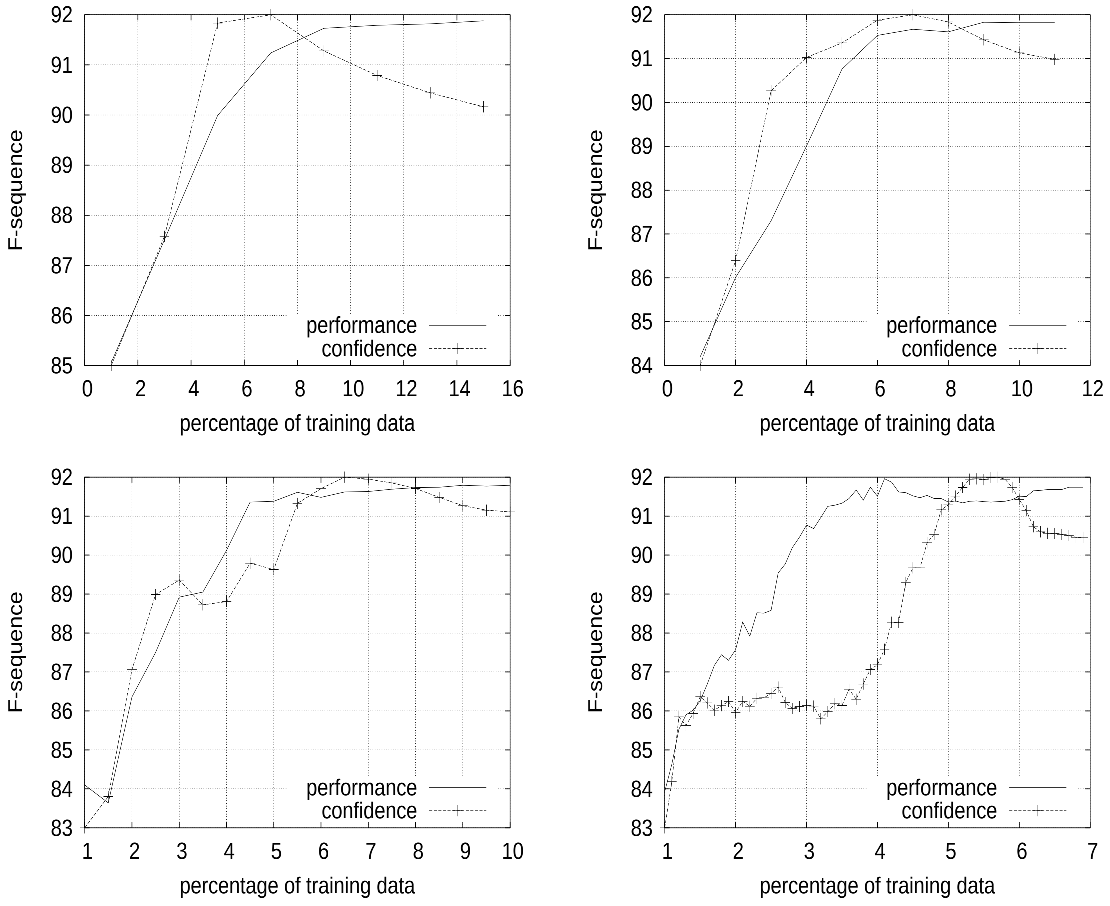


Figure 6.11: Experiments with different amount of data added in each active learning round clockwise from top left: 2%, 1%, 0.1%, 0.5%. For each level of granularity, the left graph shows the confidence estimation on the training set and the performance until the criterion of Schohn and Cohn (2000) is satisfied.

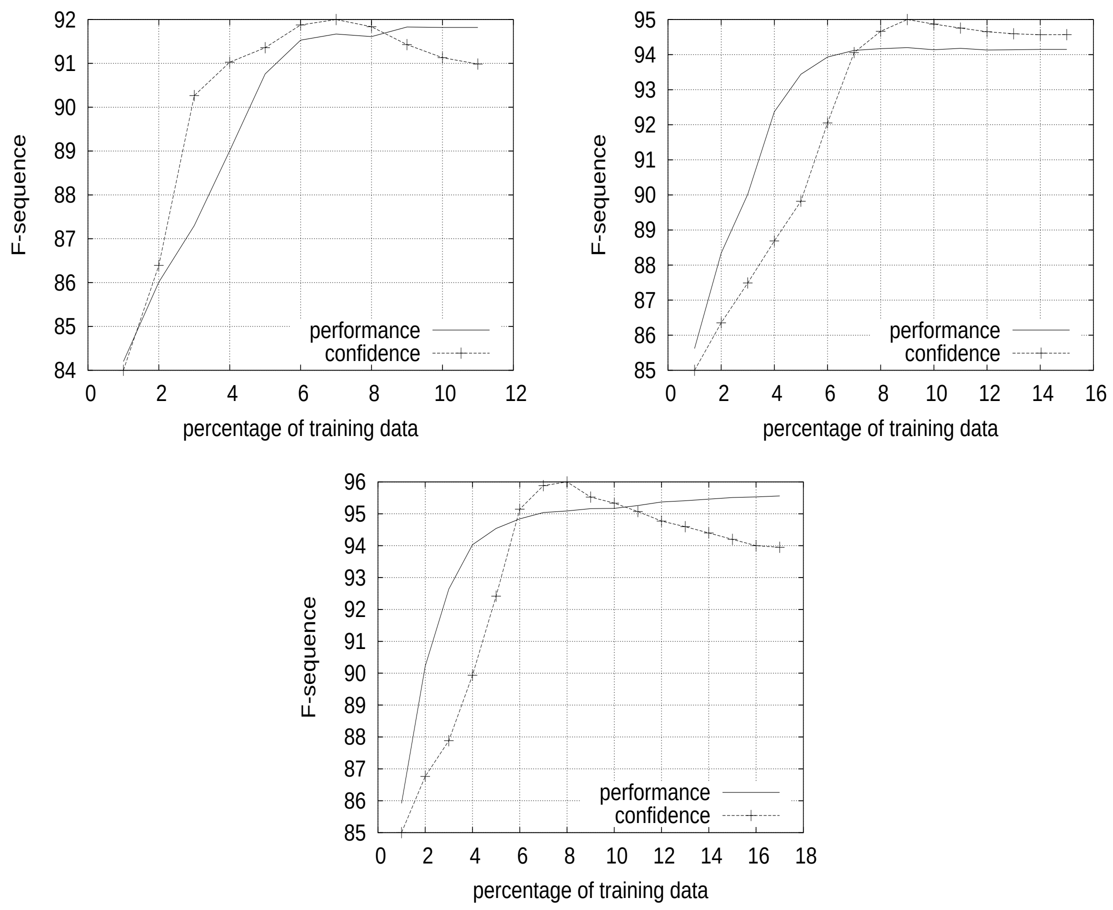


Figure 6.12: Kernels: top-left linear, top-right polynomial, bottom centre radial basis function. For each kernel, the graph shows the confidence estimation on the training set and the performance curve until the criterion of Schohn and Cohn (2000) is satisfied.

the training set, the hypothesis changes significantly. Therefore, more active learning rounds are needed to identify a hyperplane that has no unlabeled datapoints closer than the support vectors.

In order to support the observations made on the graphs, we present several indicative results, based on the graphs in Figures 6.11 and 6.12, comparing the two stopping criteria. There is no absolute way of determining when our stopping criterion is fulfilled, since a ”steady drop in the confidence estimation of the classifier” is not a quantitative expression. To overcome this difficulty, we used the graphs in the aforementioned figures to determine a steady drop in the confidence curves, so that the reader can confirm our results. In the Table 6.1, ”SC00” refers to the stopping criterion by Schohn and Cohn (2000) and ”conf” is the criterion based on the confidence of the classifier introduced in this thesis. The granularity is measured in the percentage of the initial training data added in each active learning round and the performance using F-sequence.

AL settings		data selected		performance reached	
kernel	granularity	SC00	conf	SC00	conf
linear	0.1%	6.9%	6.2%	91.74%	91.65%
linear	0.5%	10%	8.5%	91.79%	91.74%
linear	1%	11%	9%	91.83%	91.82%
linear	2%	15%	11%	91.88%	91.79%
polynomial	1%	15%	12%	94.15%	94.13%
rbf	1%	17%	10%	95.56%	95.17%

Table 6.1: Comparative results for active learning stopping criteria. SC00 is the Schohn and Cohn (2000) criterion, conf is the criterion introduced in the thesis.

It should be noted here that Schohn and Cohn (2000) are more interested in obtaining the maximum performance of a certain dataset using SVMs, rather than reducing the annotation cost using active learning. Even so, our stopping criterion reaches an adequate level of performance while stopping the active learning process earlier.

One more comment must be made about the stopping criterion of Schohn and Cohn (2000). While it suggests that the active learning should stop when all the unlabeled datapoints are given absolute decision values greater than one, we observed that if someone continues labeling then in many cases the minimum absolute decision value of the unlabeled datapoints drops again below one. Continuing the active learning process, we observed that it fluctuates to rise above one again in later rounds. This behavior raises questions about the efficiency of the heuristic it is based on, that labeling datapoints that are not in the margin of the SVM classifier does not alter the hypothesis.

To our knowledge, there is no previous work suggesting an active learning stopping criterion for multiclass SVMs. However, using the graphs of Figure 6.13 we will give quantitative results to illustrate the efficiency of the stopping criterion introduced. The fulfillment of the stopping criterion is determined again by inspecting the confidence curves. The curves are shown until the point when we consider the stopping criterion fulfilled, in order to simulate the experiments more realistically. Using the stopping criterion, active learning for named entity recognition stops when 22% of the data is used, reaching 80.21% F-sequence score. For shallow parsing, active learning is stopped at 30% reaching 91.26% F-sequence score. It should be noted that in both cases there is room for improvement since the stopping criterion is fulfilled significantly later after the performance curve has flattened out.

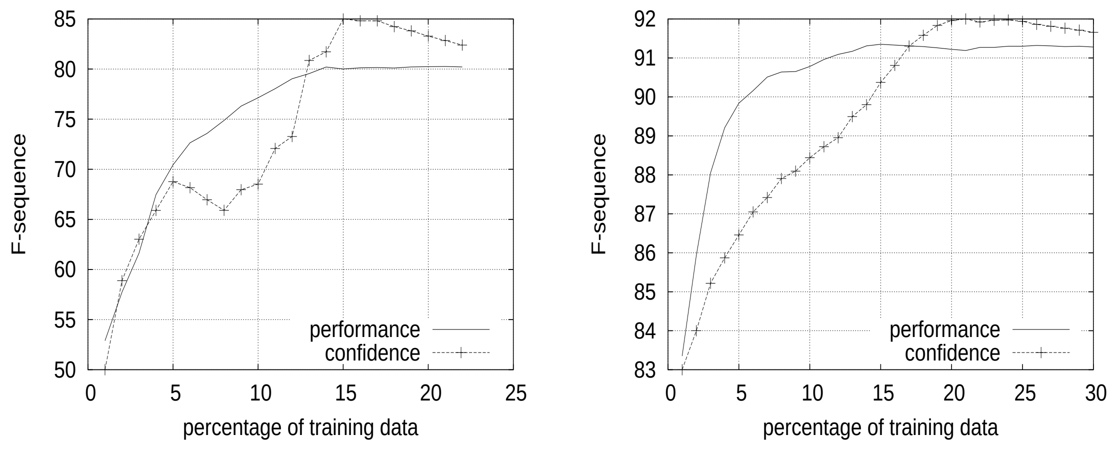


Figure 6.13: On the left, performance and confidence curves for NER. On the right, performance and confidence curves for shallow parsing.

Further investigation is required towards identifying an efficient stopping criterion for active learning with SVMs. The behavior of the confidence estimation should be explained in a more formal way that would actually predict such behavior instead of justifying it afterwards. Moreover, the stopping criterion should be quantified in such way that human supervision is not required. Even so, its efficiency compared to the criterion of Schohn and Cohn (2000), along with its successful extension and application to multiclass problems, makes it an attractive direction for further research.

6.4 Chapter summary

In this chapter we attempted to define a stopping criterion for active learning using SVMs. We considered two approaches. The first was to track the class label distribution of the selections made by the active learning. This approached could not be extended easily to datasets with no majority class. The second was to track the decision values of the SVM classifier and attempt to estimate its confidence. In order to achieve this, we measured the confidence of the predictions of the classifier on the training dataset, labeled and unlabeled. We used it efficiently in the case of a binary problem, named entity segmentation. Then, we extended it to multiclass problems, namely named entity recognition and shallow parsing. There, we explored several methods of defining the confidence of the multiclass SVM classifier. We evaluated the methods in combination with the selection criteria described in chapter 3. Finally, we gave quantitative results of the use of the stopping criterion and we compared it with previous work in the field.

Chapter 7

Conclusions - Future Work

This thesis dealt with the use of support vector machines for active learning. We confirmed the impressive performance of SVM active learning in the tasks of named entity recognition and shallow parsing. Moreover, using our alternative justification for the method used, we extended it efficiently for the first time to multiclass classification tasks. Finally, we introduced a new stopping criterion based on the confidence of the SVM classifier. Apart from being more efficient in our experiments than the criterion presented in the literature, it was successfully extended to multiclass classification with good results.

The results of this thesis show interesting directions for further research. The multiclass SVM method used, one-against-all, has the advantage of simplicity but it is not the most efficient in terms of training time. Therefore, it would be of interest to use other methods of multiclass SVMs for active learning such the one-against-one scheme, or those presented by Vapnik (1998) and Crammer and Singer (2001) which attempt to solve the multiclass SVM problem in one step. The one-against-one scheme could be combined with query by committee active learning. Moreover, in order to reduce the training time of the experiments it would be of interest to explore the incremental versions of SVMs presented by Cauwenberghs and Poggio (2000) and Rüping (2002).

In our experiments we observed that the choice of kernel is important for both active learning and performance. Therefore, it would be worthwhile to explore the performance of kernels that are specially designed for natural language processing tasks, such as the convolution kernels proposed in Collins and Duffy (2001). Another direction is to attempt to develop active learning that takes into account sequencing information, which would be more efficient for tasks that could benefit from it.

Concerning the stopping criterion based on the confidence of the classifier we presented, a more thorough and strict mathematical analysis should be provided that would predict the experimental results obtained. Such an analysis could provide a quantitative expression of the criterion, so that its fulfillment can be determined without human supervision. It would also be of to explore whether it is applicable to problems with real-valued numerical features, such as text classification with TF-IDF weighting used. It should also be investigated whether other machine learning methods demonstrate similar behavior in their confidence during active learning.

Another direction that should be followed is the effect of pool size and how the stopping criterion is affected. In our experiments, all the unlabeled data were considered in each active learning round. Sassano (2002), using the same binary classification SVM active learning applied to Japanese word segmentation, reports that using a smaller pool of unlabeled data that is replenished results in improved savings. It would be of interest to investigate the interaction of the multiclass SVM active learning and the stopping criterion presented with this approach.

Finally, in this thesis many times we referred to real-life considerations and how they could affect the results of active learning. Taking into account such considerations could alter the evaluation of the methods dramatically, as mentioned in Baldridge and Osborne (2004). Since the experiments presented in the thesis are all simulations and a lot of comments and arguments refer to realistic settings with human annotators, we consider it important to evaluate our methods and test our hypotheses in real-life experiments,

Bibliography

Aizerman, M., E.M.Breverman, and Rozoner, L. (1964). Theoretical foundations of the potential function method in patter recognition learning. *Automation and Remote Control*, 4:821–837.

Argamon-Engelson, S. and Dagan, I. (1999). Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360.

Baldridge, J. and Osborne, M. (2003). Active learning for hpsg parse selection. In *Proceedings of the 7th Coference on Natural Language Learning*, Edmonton, Canada.

Baldridge, J. and Osborne, M. (2004). Active learning and the total cost of annotation. In *Proceedings of EMNLP04*.

Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Meeting of the Association for Computational Linguistics*, pages 26–33.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Campbell, C., Cristianini, N., and Smola, A. (2000). Query learning with large margin classifiers. *Proceedings of ICML2000 (Stanford, CA, 2000).*, page 8.

Cauwenberghs, G. and Poggio, T. (2000). Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1995). Active learning with statistical methods. In Tesauro, G., d. Touretzky, and Leen, T., editors, *Advances in Neural Information Processing*, volume 7, pages 705–712. The MIT Press.

Collins, M. and Duffy, N. (2001). Convolution kernels for natural language. *Advances in Neural Information Processing Systems*.

Collobert, R. and Bengio, S. (2001). SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Crammer, K. and Singer, Y. (2000). On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based methods. *Journal of Machine Learning Research*, 2:265–292.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Machine Learning*. ACM Press.

Freund, Y. and Schapire, R. (1999). A short introduction to boosting.

Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press.

Hsu, C. and Lin, C. (2001). A comparison on methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

Joachims, T. (1998a). Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, A. S., editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

Joachims, T. (1998b). Text categorization with support vector machines: learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Kreßel, U. (1999). Pairwise classification and support vector machines. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods — Support Vector Learning*, pages 255–268, Cambridge, MA. MIT Press.

Kudoh, T. and Matsumoto, Y. (2000). Use of support vector learning for chunk identification. In Cardie, C., Daelemans, W., Nedellec, C., and Tjong Kim Sang, E., editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 142–144. Lisbon, Portugal.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. In Fogelman, F. and Gallinari, P., editors, *International Conference on Artificial Neural Networks*, pages 53–60, Paris. EC2 & Cie.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In Croft, W. B. and van Rijsbergen, C. J., editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE. Springer Verlag, Heidelberg, DE.

Liu, C., Nakashima, K., Sako, H., and Fujisawa, H. (2002). Handwritten digit recognition using state-of-the-art techniques. In *FHR02*, pages 320–325.

Mayfield, J., McNamee, P., and Piatko, C. (2003). Named entity recognition using hundreds of thousands of features. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 184–187. Edmonton, Canada.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18:203–226.

Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. pages 61–74.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. pages 185–208.

Platt, J. C., Cristianini, N., and Shawe-Taylor, J. (2000). Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems*, (12):547–552.

Ramshaw, L. and Marcus, M. (1995). Text chunking using transformation-based learning. In Yarovsky, D. and Church, K., editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

Rüping, S. (2002). Incremental learning with support vector machines.

Sang, E. T. K. and Buchholz, S. (2000). Introduction to the CoNLL-200 shared task: Chunking. In Cardie, C., Daelemans, W., Nédellec, C., and Sang, E. T. K., editors, *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop, Lisbon, 2000*, pages 127–132. Association for Computational Linguistics, Somerset, New Jersey.

Sassano, M. (2002). An empirical study of active learning with support vector machines for japanese word segmentation.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann.

Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Computational learning theory*, pages 287–294.

Sollich, P. (1999). Probabilistic interpretation and bayesian methods for support vector machines.

Thompson, C. A., Califf, M. E., and Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. In *Proc. 16th International Conf. on Machine Learning*, pages 406–414. Morgan Kaufmann, San Francisco, CA.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Tong, S. (2001). Active learning: Theory and applications.

Tong, S. and Koller, D. (2000). Support vector machine active learning with applications to text classification. In Langley, P., editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 996–1006, Stanford, US. Morgan Kaufmann Publishers, San Francisco, US.

Tur, G., Schapire, R. E., and Hakkani-Tr, D. (2003). Active learning for spoken language understanding.

Van Rijsbergen, C. J. (1979). *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

Vavasis, S. A. (1991). *Nonlinear optimization: complexity issues*. Oxford University Press, Inc.

Weston, J. and Watkins, C. (1999). Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*.

Wu, T.-F., Lin, C.-J., and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.