THE NATIONAL UNIVERSITY
of SINGAPORE

School *of* Computing
Lower Kent Ridge Road, Singapore 119260

TRB6/03

Continuous Naive Bayesian Classification

Vinsensius Berlian Vega S. N. and Stephen Bressan

June 2003

Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

Continuous Naive Bayesian Classifications

Vinsensius Berlian Vega S. N. VINSENSI@COMP.NUS.EDU.SG
Stéphane Bressan STEPH@NUS.EDU.SG
Department of Computer Science, National University of Singapore, 3 Science Drive, Singapore 117543

Abstract

The most common model of machine learning algorithms involves two life-stages, namely the learning stage and the application stage. The cost of human expertise makes difficult the labeling of large sets of data for the training of machine learning algorithms. In this paper, we propose to challenge this strict dichotomy in the life cycle while addressing the issue of labeling of data. We discuss a learning paradigm called Continuous Learning. After an initial training based on human-labeled data, a Continuously Learning algorithm iteratively trains itself with the result of its own previous application stage and without the privilege of any external feedback. The intuitive motivation and idea of this paradigm are elucidated, followed by an explanation on how it differs from other learning models is laid out. Finally, empirical evaluation of Continuous Learning applied to the Naive Bayesian Classifier for the classification of newsgroup articles of a well-known benchmark is presented.

1. Introduction

Many components and processes of modern Information Management Systems rely on machine learning algorithms and techniques. Classification, for example, is one of the key themes in Information Retrieval and Digital Libraries. Other areas include, to name a few, document analysis, document understanding (Esposito, *et. al*. 1998), language identification, summarization (Kupiec, Pederson, & Chen, 1995), intelligent agents, text routing, document clustering, and relevance feedback (Bartell, Cottrell, & Blew, 1994), etc.

Our own research was motivated by the design and development of information tools for the Indonesian Language. Motivated by the lack of effective information retrieval tools for Indonesian Language speakers, we set ourselves the twofold goal of identifying and investigating research issue pertinent to Information Retrieval for the Indonesian Language (Vega & Bressan, 2001a) and of building a full-fledge effective Indonesian web search engine. In order to do so, one important step that must be accomplished is to build a web crawler that will find and index Indonesian web pages from the Internet. An implementation of a tool to sieve out non-Indonesian pages is therefore essential. In (Vega & Bressan, 2001b), we propose a learning algorithm to tackle the problem. The algorithm is a positive only learning algorithm based on weighted trigrams. With this method, we were able to attain an initial performance of over 94% recall and 88% precision.

The already high performance coupled with the requirement that the algorithm needs to be adaptive towards changes in language trends, had led us into asking whether the algorithm could create its own new experience (e.g. data for training) and effectively learn from it. The idea of algorithm that creates its own experience is timely. With the explosion of machine-readable information, manually constructing experience (e.g. manually collecting and labeling data) could prove to be cumbersome and costly. Coupled with the high dynamism of information medium, content, and user requirements, algorithms that need minimum human intervention are desirable. We formulated the first notion of Continuous Learning according to which, new data labeled by the classifier (or by other kind of machine learning algorithm) is used to extend the training set indefinitely. The paradigm targets two important goals. First and foremost, it is aimed to reduce the amount of manual experience[1] needed. The desired effect of which is clear, that is to lessen the amount of human labor. Second, it is intended to make the underlying algorithm adaptive towards the change in environment. For example, in the past "Bill Clinton" would be a major keyword under the topic of U.S. politics. However as time passes, the major keyword has changed to "George Bush". Note that we use the term underlying algorithm here. The reason is because Continuous Learning is a learning paradigm, which needs to be applied to some learning algorithm, i.e. the underlying algorithm. In (Vega & Bressan, 2001b), we report the initial application of continuous learning idea to language identification and distinction. The result was

[1] By manual experience we mean experience or data that are not automatically gathered by the algorithm itself and/or data that need to be manually labeled

encouraging. Continuous learning improved the initial 94% recall and 88% precision to 100% recall and about 90% precision.

This paper elaborates the generic idea of Continuous Learning. Section 3 explicates the intuition behind Continuous Learning. An application of continuous learning to a classification algorithm is presented to exemplify. Following that, some experimental results are described in section 4. In the next section, previous related researches are discussed and compared. Finally, in the conclusion we summarize our contribution and outline our plan for future work.

2. Continuous Learning and its application

The most common model of Machine Learning involves two life-stages. The first stage is the training stage. In this stage, a learning algorithm gains experience from some form of teacher or oracle. The experience is commonly given in form of a set of training examples, be it perfectly labeled, a mixture of labeled and unlabeled (Blum & Mitchell, 1998; Nigam, *et. al*. 2000), probabilistically labeled (Smyth, 1995), or even erroneously labeled (Kearns & Li, 1993). Given the training set, the learning algorithm then uses it to construct or update its internal parameters. This process could be a single-pass over the training data (e.g. Naive Bayesian Classifier and Candidate-Elimination in (Mitchell, 1997)) or it could be iterative/multi-pass method (e.g. ID3 and EM algorithm (Dempster, Laird, & Rubin, 1977)). By iterative, we mean that the algorithm runs through or analyze a fixed set of training data multiple times until some condition is reached or satisfied.

In the next stage, which we call the application stage, the trained algorithm is then used as is to solve problems that it was originally intended for without any further significant modification towards the internal parameters of the algorithm. In other words, the learning process only occurs during the training stage.

2.1 Continuous Learning

Continuous Learning extrapolates this two life-stages model. Here the learning process is interwoven in the application stage. After the initial training, the algorithm is presented with some data that it needs to process. The most common task of a learning algorithm would be to perform some form of labeling (e.g. relevancy, categories, probability assignment, etc). Normally the information output by the algorithm is of the same kind as the information associated with the training data set. Let us say that the process transforms the given data from unlabeled data to *self-labeled* data. In the continuous learning algorithm, these self-labeled data are then used to retrain (or incrementally train) the algorithm. The interweaving of learning and labeling can continue ad lib. Here is the key of Continuous Learning, *trusting one's own judgment to enlarge one's knowledge base*. Although it may seem peculiar for an algorithm to do so, such a strategy is quite commonly used by humans, we believe. In some instances, it seems that the foundation of our knowledge is augmented with experience funded on this sole and self-knowledge without external feedback. Similarly, a Continuous Learning algorithm would indefinitely learn from its own judgment without any external feedback or guidance. External feedback includes predefined fitness function (such as those used in genetic algorithm), reward function (like in reinforcement learning), as well as human intervention. Simply put, a continuously learning algorithm is entirely on its own in the effort to improve its performance.

Given that the algorithm is going to determine its own fate, the arising question is to determine under which circumstances a performance increase actually occurs. One would expect that, if the performance of the algorithm after the initial training is relatively high, Continuous Learning have chances to improve its performance. However, should the initial performance be insufficient, the algorithm could worsen over time. This is not unlike the ability of better students to study on their own. The existence of such a threshold has been verified in the application of Continuous Learning to language distinction (Vega & Bressan, 2001b). We now illustrate the algorithm and the existence of a threshold for the performance improvement on a Naïve Bayesian Classifier.

2.2 Continuous Naive Bayesian Classifier

As an illustrating example, we propose to apply Continuous Learning to a Naive Bayesian Classifier. The Naïve Bayesian Classifier is a simple and effective as well as commonly used Machine Learning algorithm

The Continuous version of Naive Bayesian Classifier goes as follows. First, a generic Naive Bayesian Classifier is trained using a set of manually labeled initial training examples. Then, in the application stage, stream of documents is given to it to be classified. For each document, the classifier determines the most appropriate class of the document based on its current probability terms table. The assigned class is then assumed as the true class of the document. The classifier then incorporates the self-labeled documents as a new example of the assigned class, which is further used to update its internal probability terms table. The pseudo code for Continuous Naive Bayesian Classifier is given below. Note that in step 1, the document *d* is not obtained from any predefined set, rather it is obtained from the environment, e.g. the web, sensors' input, etc.

```
[Initial Training]
For each class j
   Build the probability terms P(W|v_j)
    based on S_j, set of sample documents
    of class j
```

```
[Continuous Learning]
Loop forever
   1. obtain an arbitrary new document d
      to be classified

   [Data labeling]
   2a. k ← classify document d from
       based on current P(W|V)
   2b. output k as label of d

   [Training from Self-labeled Data]
   3a. Let S_k = S_k ∪ {d}
   3b. update the probability terms
       P(W|v_k) using the updated S_k

End Loop
```

3. Experimental Results

In this section, the results gathered from the experiments done with the Continuous Naive Bayesian Classifier are presented. The base Naive Bayesian Classifier is implemented according to the algorithm given in (Mitchell, 1997). In each experiment, the Continuous Naïve Bayesian Classifier was trained with a set of manually labeled documents. Then, one by one, new documents from a predefined set were given to the Continuous Naïve Bayesian Classifier to be classified and learned from. At some interval, the current state of the classifier, i.e. the values of the probability terms P(W|V), is saved. Along with that, the performance of the Continuous Naïve Bayesian Classifier at the current state is measured against a test set.

From the results, we observed that in general, performance increase is attainable. However, the increase depends on the inherent difficulty of the problem and the initial performance. Initial performance here refers to the performance of the classifier after it has been trained with a set of manually (and correctly) labeled documents. While inherent difficulty of the problem refers to how hard the task in terms of how much experience is needed before a level of performance could be attained.

3.1 Experimental Setup

The data set used for the experiment is the 20 Usenet Newsgroups data set used in (Nigam, *et. al.,* 2000) (obtained from http://www.cs.cmu.edu/~textlearning). The data set contains 19997 postings that are spread among the 20 newsgroups, namely alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, and talk.religion.misc. Each of the 20 newsgroups, except soc.religion.christian, contains exactly 1000 postings. To balance up the numbers, we decided to create three postings into the group that contains only 997 documents by duplicating three different entries from the group. This duplication is justifiable, as a great portion of newsgroup postings are replies of previous postings that sometimes carry a significant portion of the previous postings' content. With the duplication, the total size of the data set is now 20000 and is distributed evenly to 20 newsgroups.

Before the experiment was carried out, the data set was preprocessed. First, the Usenet headers were removed. Then it was tokenized. Token is defined as contiguous alphabetic characters. No stemming was done, but we did stop-word removal. These settings are similar to that done in (Nigam, *et. al.*, 2000). With one little difference, in that for the stop word list, we used the stop word list provided in the SMART Information Retrieval[2].

3.1.1 Definitions and Data Setup

As described in section 2.2, Continuous Naive Bayesian Classifier will learn indefinitely from documents given to it to be classified. In the experiments, the process is simulated by asking a trained classifier (i.e. classifier that has been trained with labeled data) to judge some sets of documents (which will be used by the Continuous Learning algorithm to retrain itself). At some interval (i.e. after a certain number of documents are presented to the continuous classifier) the classifier's performance is measured. The set of documents used between two consecutive intervals is called running set and the period between two consecutive intervals is termed as iteration, just for the lack of better word. Note that the term iteration here means differently with the term iterative used earlier. In the earlier usage, iterative means the algorithm

Given the above settings, the data set needs to be divided into three subsets: initial training set, set of running sets, and test set. In our experiment, the initial training set was comprised of the first n-documents from each newsgroup. The next 800-minus-n-documents from each newsgroup composed the set of running sets. Documents from the running sets were handed to the classifier in chronological order. The test set was chosen to be the last 20% of the postings from each newsgroup, similar to what was done in (Nigam, *et. al.,* 2000). This simulate the situation where initial training set is collected from past documents, the current documents are used to continuously train the classifier, and the overall performance is measured in terms of its classification accuracy over future documents. Note also that none of the running sets overlapped.

3.1.2 Varying Initial Performance

The aim of the experiments is to discover whether the proposed framework could yield some increase of

[2] The SMART system (version 11.0) was developed at Cornell University and is available from ftp.cs.cornell.edu/pub/smart.

performance and under what condition could it do so. To obtain a good picture of what is happening, various parameters are introduced. The first parameter is the size of initial training set used. The reasoning is straightforward. Initial performance of the classifier vary with different number of documents used in initial training, by varying it we could observe how great the effect of Continuous Learning given different starting performance. In our experiments, we varied the initial training set to be 10, 20, 30, 40, or 50 documents per group.

Given that different initial training set sizes were used, the issue of how large the running set need to be resolved. A running set of 10 documents per group means differently for classifier X, trained initially with 10 documents per group, or classifier Y, trained initially with 50 documents per group. For classifier X, the self-labeled 10 documents per group have the equal weigh as the manually labeled 10 documents per group that it sees during the initial training. However for classifier Y, 10 per group self-labeled documents weigh much less than the 50 manually labeled documents per group that it was given in the initial training. Thus, suppose that the two classifiers wrongly label the entire running set, the effect would be greater for classifier X than to classifier Y. Therefore, to be fair, we should compare the performance of classifier X after seeing a running set of 10 documents with the performance of classifier Y after seeing a running set of 50 documents. In the light of this, we opted to set the saving and measurement interval to be equal to the size of the initial training set, e.g. for classifier X, the classifier state is saved and its performance is measured every 10 documents. While for classifier Y, it is done every 50 documents. In each of our experiments, the classifier was given 10 running sets, i.e. going through 10 iterations.

3.1.3 Varying Inherent Difficulty

Varying the size of initial training set show how the proposed idea performs under different starting position, i.e. different initial performance. However, altering the initial training set size may not be enough, as performance is also determined by how hard the problem is. Modifying the initial training set size challenges the idea to solve a same problem under different initial condition. Here is when the second parameter comes into play. Given a fixed size initial training set, we alter the difficulty of the problem by altering the number of groups that the classifier needs to classify. It may not be so obvious as why reducing the number of classes (or groups) means reducing the difficulty as well. As even for two groups, e.g. soc.religion.christian and talk.religion.misc, it might be very difficult to separate the two. However, as mentioned earlier, we would like to view difficulty as the amount of experience needed to achieve a certain level of performance. For example: assuming that the each document has an equal probability to be belong to a group, to achieve performance of 50% accuracy for the two-group problem, the classifier needs no experience, as it could be as easy as always assigning new documents into one of the classes. However, it is not the case for four-group problem. To exemplify the experiments, let's say for an initial training set of 10 documents per group, we change the problem from classifying new documents into one of 20 newsgroups to classifying into 7 newsgroups. Suppose also that the 7 newsgroups are selected to be the first 7 groups when the 20 groups are listed in alphabetical order. Thus, for this experiment, we take 7000 documents (1000 for each group). Out of this 7000, the last 20% or 1400 documents are set aside to be the test set. The initial training set and the running sets are formed as per normal. Hence, with this variation, we hope to discern the effectiveness of Continuous Learning for different inherent problem difficulty level. Here we use seven different numbers of classification problems: 7, 10, 12, 14, 16, 18, and 20 newsgroups.

3.1.4 Comparison with Upper Bound Performance

To further comprehend the effect of Continuous Learning, we compared the performance of each experiment setting of Continuous Learning with self-labeled data against the theoretical upper bound performance. The theoretical upper bound performance is achieved by feeding the algorithm with manually (and correctly) labeled data for it to learn from, instead of letting it learn from self-labeled data that may contain some error. Naturally, we expect the performance of the all-information setting to surpass that of the self-labeled. What we look into is whether there is some relation in performance between the two configurations.

3.2 Results

From our experiments we gathered a total of 12 pair of graphs, which depict the classification accuracy attained for 10 iterations. As stated earlier, in each iteration, the Continuous NBC is given a set of training set, whose size is equivalent to the size of the initial training set. Four pairs of graphs are shown in this section whereas the rest can be found in the Appendix. In each pair, the performance graph of the continuous learning with self-labeled data is contrasted with the all-information performance.

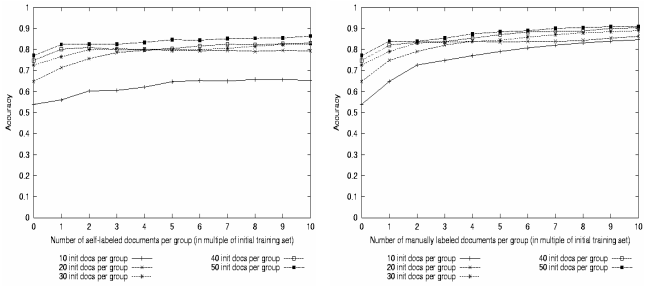

Fig. 1. Accuracy for classification on 7 newsgroups varying the initial training set

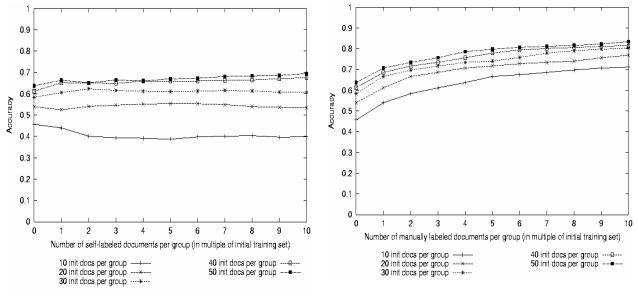

Fig. 2. Accuracy for classification on 16 newsgroups varying the initial training set

From Figure 1 and 2, we can see that Continuous Learning were able to improve performance given a good initial performance. Below some initial performance, however, Continuous Learning caused the performance to drop over time. Further, faced against a harder problem (i.e. more groups), not only that initial performance is generally lower, the improvement is attained at somewhat slower rate or even not at all. This claim is further validated with the two pairs of graphs below.

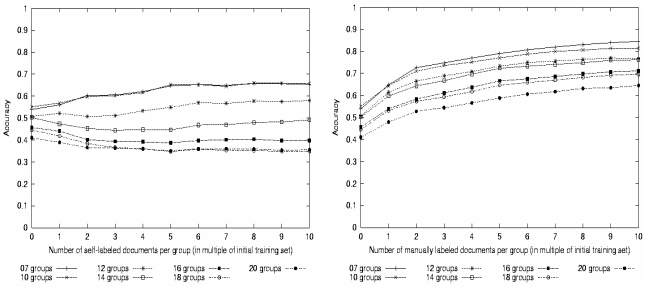

Fig. 3 Accuracy for initial training set of 10 documents per group and varying number of groups

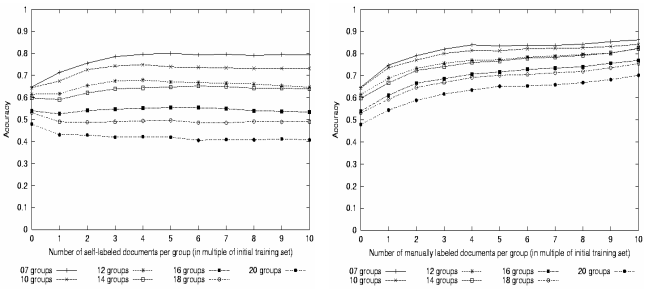

Fig. 4. Accuracy for initial training set of 20 documents per group and varying number of groups

The overall shape in the four pairs of graphs above clearly support the intuition behind the idea of Continuous Learning, in that an already good performing learning algorithm could improve itself by learning from self-labeled data. Vice-versa, not-so-good performing algorithm should be wary if it to use self-labeled data to improve itself, as it could end up with declining performance. It is also shown in the above that for a similar initial performance, the Continuous Learning concept works better for easier problem. Again this in line with the intuition that for easier subjects, less formal training is needed for a person to be able to self-study effectively. From this we can say that the minimum value for the initial performance needed to guarantee performance gains depends on the inherent difficulty of the problem.

In each pair of graph, we can observe that the performance of all-information (i.e. manually labeled) setting is almost always increasing. This is quite obvious as in all-information, the data are all correctly labeled. The important point to note is that, for the self-labeled plots that are increasing (e.g. figure 1), their shape mimics the shape of the all-information plots. It seems to imply that for self-labeled settings that experience performance increase, the performance increase in self-labeled setting is a fraction of that in all-information setting.

4. Related Work

To the best of our knowledge, this idea of continuous learning has never been explicitly and formally formulated before, although the idea is somewhat straightforward or even obvious. The most relevant works are those on using unlabeled data to boost performance (Blum & Mitchell, 1998; Nigam, *et. al.*, 2000). Although not directly equivalent, they have some common ground with the idea proposed in the paper. Both are aimed to reduce the need of large labeled sample data. Both tries to leverage the value of unlabeled data, using the information gathered from the small set of labeled data, to increase performance. Both avoid the need of external feedback.

The difference lies on how the unlabeled data are obtained and used. In (Blum & Mitchell, 1998; Nigam, *et. al.*, 2000), the training data set D consists some labeled data L and some unlabeled data U. The algorithms are first trained using L. After that, U is used to train the algorithms in the iterative fashion until some condition or state is satisfied. In the Continuous Learning paradigm, the training data set D consist only labeled data L, i.e. D = L. The algorithm is then trained with L. Whenever a document d is presented to the algorithm to be evaluated (or to be classified), it will give its judgement j (e.g. d belongs to class j) and retrain itself with d as if j is the actual label of d.

In fact, the concept of Continuous Learning can be applied to the algorithms for learning from unlabeled data, such that the algorithm will continue to collect unlabeled data, which upon reaching a certain size *n* will be used to retrain the algorithm. The resulting algorithm is shown below. Note that the initial training stage below is exactly the conventional learning from unlabeled data.

```
[Initial Training = Conventional Learning
 From Unlabeled Data]
Let D = L ∪ U
Let A be an algorithm capable of learning
    from unlabeled data
Train algorithm A using L
Train algorithm A using U

[Continuous Learning]
Let U = ∅
Loop forever
   1a. obtain a new document d to be
       classified
   1b. Let U = U ∪ {d}

   [Data labeling]
   2a. k ← classify document d using
       algorithm A
   2b. output k as label of d

   [Continuous Learning from Unlabeled
    Data]
   3. If sizeOf(U)=n
      then
           Train algorithm A using U
           Let U = ∅
      End of if

End Loop
```

Compared with other online algorithm (e.g. reinforcement learning), the key difference is the non-existence of external feedback (e.g. predefined fitness function, reward function, and human interference) in Continuous Learning.

5. Conclusion

In this paper, we proposed the idea of Continuous Learning. The idea is inspired by the human natural ability to use experience to learn in the absence of a teacher. It is conceived with two objectives in mind, namely (1) reducing the need of human labor in creating, setting up, and labeling data for training a learning algorithm and (2) forging an adaptive algorithm. These goals are critical in the ever-growing and ever-changing realm of modern information systems. Continuous Learning was contrasted to other learning models, such as the common two life-stages model (single pass or multi-pass/iterative) and other seemingly similar approach (e.g. reinforcement learning and EM). The key distinguishing point is that in Continuous Learning, the learning process is extended indefinitely relying only on self-labeled examples without external feedback.

We have illustrated how the framework could be applied to Naive Bayesian Classifier. The empirical results obtained from a comprehensive set of experiments corroborate the intuition behind the proposed paradigm: under the right circumstances it can learn. It is shown that the effectiveness of Continuous Learning is not only determined by the initial performance, but also by the inherent difficulty of the problem. Harder problems would require higher initial performance for the Continuous Learning to be successful. In addition, the comparison with all-information accuracy suggests that learning from self-labeled data improves the performance proportionally to that of learning from manually labeled data.

In other words, a learning algorithm that has reached a certain level of performance can be expected to improve its own performance autonomously.

Yet an important task remains, namely the formalization of the Continuous Learning paradigm. We are currently attempting to define a theoretical framework that could help us to identify and quantify the conditions under which the Continuous Learning algorithm can actually learn.

References

Bartell, B. T., Cottrell, G. W., and Belew, R. K. (1994). Learning the optimal parameters in a ranked retrieval system using multi-query relevance feedback. *In Proceedings Symposium on Document Analysis and Information Retrieval.*

Blum, A. and Tom Mitchell, T (1998). Combining Labeled and Unlabeled Data with Co-Training. *In Proceedings of the 11th Annual Conference on Computational Learning Theory.*

Dempster, A.P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B 39:1-38*.

Esposito, F., Malerba, D., Semeraro, G., Fanizzi, N., and Ferili, S. (1998). Adding Machine Learning and Knowledge Intensive Techniques to a Digital Library Service. *In International Journal on Digital Libraries,* 2(1), pp3-19, Springer-Verlag.

Kearns, M. and Li, M. (1993). Learning in the presence of malicious errors. *SIAM Journal on Computing, 22(4): 807-837.*

Kupiec, J., Pederson, J., and Chen, F. (1995) A trainable document summarizer. *In Proceedings of SIGIR '95, pp. 68-73, ACM Press.*

Mitchell, T. M. (1997) *Machine Learning.* McGraw-Hill.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (2000) Text Classification from Labeled and Unlabeled Documents using EM. *In Machine Learning*, 39(2/3). pp. 103-134.

Smyth, P. (1995) Learning with Probabilistic Supervision. In Petsche, T., Hanson, S., and Shavlik, J. (Eds.) *Computational Learning Theory and Natural Learning Systems 3.* Cambridge, MA: MIT Press, pp.163-182.

Vega, V. B. and Bressan, S. (2001) Indexing the Indonesian Web: Language Identification and Miscellaneous Issues. *In Poster Proceedings of 10th World Wide Web Conference.*

Vega, V. B. and Bressan, S. (2001) Continuous-Learning Weighted-Trigram Approach for Indonesian Language Distinction: A Preliminary Study. *In Proceedings of 19th International Conference on Computer Processing of Oriental Languages.*

Appendix: Performance Graphs

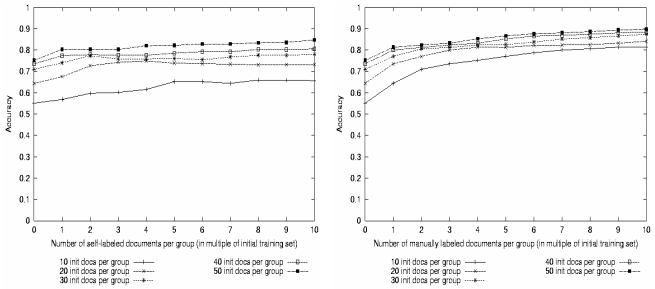

Fig. 5. Performance graph for experiment with classification on 10 newsgroups and variable size of initial training set

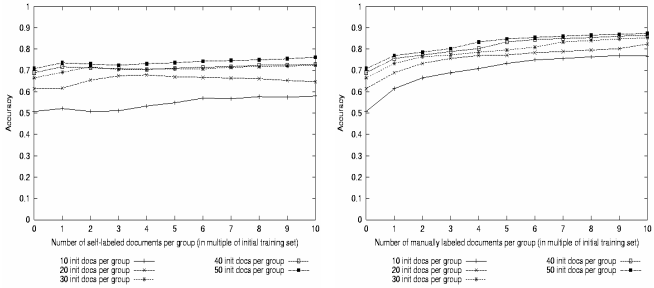

Fig. 6. Performance graph for experiment with classification on 12 newsgroups and variable size of initial training set

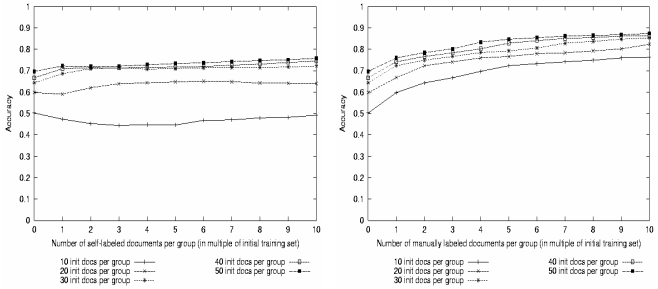

Fig. 7. Performance graph for experiment with classification on 14 newsgroups and variable size of initial training set

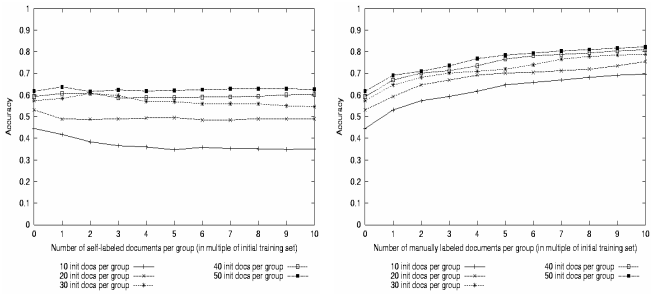

Fig. 8. Performance graph for experiment with classification on 18 newsgroups and variable size of initial training set

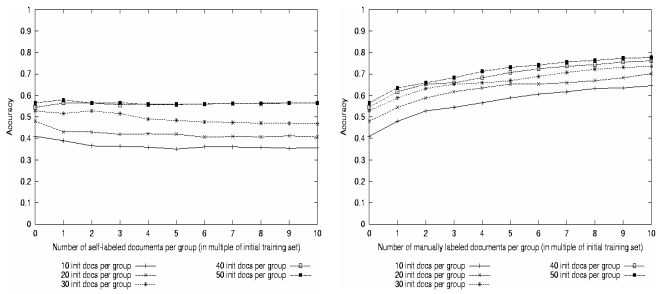

Fig. 9. Performance graph for experiment with classification on 20 newsgroups and variable size of initial training set

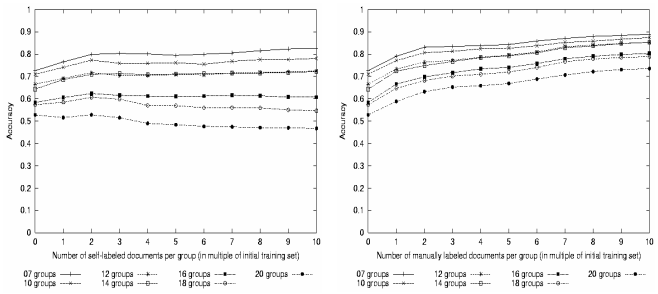

Fig. 10. Performance graph for experiment with 30 documents per group in the initial training set and varying number of newsgroups to be learned

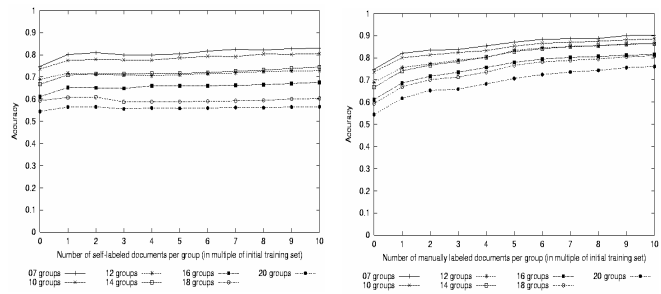

Fig. 11. Performance graph for experiment with 40 documents per group in the initial training set and varying number of newsgroups to be learned

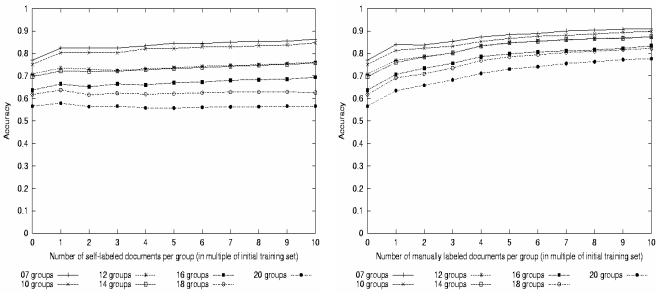

Fig. 12. Performance graph for experiment with 50 documents per group in the initial training set and varying number of newsgroups to be learned