Journal

of

Engineering, Science and Technology

Performance Analysis of Machine Learning Algorithms for Multi-class Document Using WEKA

Debby Erce Sondakh

Faculty of Computer Science, Universitas Klabat, Manado, Indonesia
debby.sondakh@unklab.ac.id

Abstract

This research aims to assess and compare the performance of six machine-learning algorithms for text classification namely decision rules, decision tree, k-nearest neighbor (k-NN), naïve Bayes, regression, and Support Vector Machine (SVM). These six algorithms are compared using multi-class text document. The comparison was done in terms of their effectiveness, the ability of classifiers to classify the document in the right category. Precision, recall, F-measure, and accuracy are the four effectiveness measurements that were applied. The result shows that decision rule's performance was the worst. SVM, decision tree, regression, and naïve Bayes have high effectiveness value. SVM can classify text quite well in average of 3.42 seconds to build each classifier model. Decision tree and regression can classify text with higher accuracy values rather than SVM, but slower in building the model. Among the six algorithms Naïve Bayes classifiers has the highest effectiveness value, while the model development time is the shortest as well. The average model building time is 0.03 second.

Keywords – text classification, multi-class document, machine-learning approach

I. INTRODUCTION

Information retrieval system aims to obtain relevant information from a collection of large number of information. As the number of digital text documents spread over the internet continues to grow every day, it triggers the need for a system that can organize the documents, and as well as make it easy for users to get the right and useful information. A number of algorithms and tools have been developed and implemented to retrieve information from large repositories.

Data mining provides solution to handle the rapid growth of data. Using data mining technique, the documents are grouping into classes in order to simplify the process of retrieving information from large set of data [1]. In data mining, there are two main approaches of grouping documents namely classification and clustering. The first method, classification, groups the documents into fixed categories based on documents' predefined labels. On the other hand, clustering method grouping the documents based on documents' similarity.

This research aims to examine and compare text documents classification algorithms, specifically the machine learning based classification algorithms.

II. DOCUMENT CLASSIFICATION

Document classification is defined as grouping documents into one or more categories based on predefined label. Document classification starting with the learning process to determine the category of the document, is called supervised learning. This research investigated the text documents. Reference [3] and [4] defined text classification as a relation between two sets, set of documents, $d = (d_1, d_2, \cdots, d_n)$ and set of categories $c = (c_1, c_2, \cdots, c_m)$. d_i is *i-th* document to be classified. c_j is *j-th* predefined category for a document. n is the number of documents to be classified, and m is the total of predefined category in c.

Text classification is the process of defining a Boolean value for each pair $(d_j, c_i) \in D \times C$, where D is the set of documents and C is a set of predefined categories. Classification is about to approximate the classifier function (also called rule, hypothesis, or model):

$$f: D \times C \rightarrow \{T, F\}$$

The value T (true) assigned to pair (d_j, c_i) indicates that document d_j includes in category c_i . Otherwise, the value F indicates that document d_j is not a member of category c_i.

Document is a sequence of words [5]. In information retrieval document is stored as set of words, also called vocabulary or feature set [6]. Vector Space Model is employed as document representation model. A document is an array of words, in the form of binary vector with value of 1 when a word present in the document or value of 0 for absences of a word. Each document is included in the vector space

$R^{|V|}$, $|V|$ is the size of *vocabularies* V. For a collection of documents, called dataset, documents are represented as $m \ x \ n$ matrix, where m is the number of documents and n is the words. Matrix element a_{ij} denotes the occurrence of word j in document i which is represented as binary value.

There are two main approaches that can be applied for classifying document, i.e. rulebased approach and machine learning approach. In rule-based approach, also called knowledge engineering, the rules that define the categories of documents are assigned manually by an expert. Then, the documents are grouped into categories that have been defined [3]. Using this method, rule-based classifier is able to produce an effective classification with good accuracy. However, its dependency on an expert to assign the rules manually becomes the main drawback. When the categories are about to change then the previous expert who defined the rules must be involved. Over all, this method requires high cost and takes time in classifying large number of documents [2].

III. MACHINE LEARNING BASED CLASSIFICATION

To overcome the weaknesses of rule-based classifier, machine learning based approach is applied to perform classification. This method is also called inductive process or learner, in which the document classification is running automatically using the text label that have been defined first (predefined class). Machine learning based classifiers learn the characteristics of the set of documents, which have been classified into category c_i . Using these characteristics the inductive process is done to obtain new characteristics that the new documents must have to be included in a category. So, inductive process is a way of building the classifiers automatically from set of documents that have been pre-classified. This method can overcome the problems of large document dataset, reducing labor cost, while the accuracy is comparable to the rules resulted from a supervisor.

A. Decision Rules

Decision rules using DNF rule to build a classifier for category c_i . DNF rule is a conditional rule consists of disjunctiveconjunctive clause. This rule describes the requirements for the document to be classified into categories defined; 'if and only if' the document meets on of the criteria in DNF clauses. The rules in DNF clauses represent categories' profile. Each single rule comprise of category's name and the 'dictionary' (list of words included in that category). A collection of rules is the union of some single rule using logic operator "OR". Decision rules will choose the rules whose scope is able to classify all the documents in training sets. Rules set can be simplified using heuristic without affecting the accuracy of resulting classifier.

Sebastiani in [3] explained, DNF rules are built in a bottom-up fashion, as follows:

1. Each training document d_j is $\eta_1, \ldots, \eta_n \rightarrow \gamma_i$ clause where $\eta_1, \ldots, \eta_n$ are the words

contain in document d_j , and γ_i is the category c_i when d_j satisfy the criteria of c_i, otherwise it is $\overline{c_i}$.

2. Rules generalization. Simplifying the rules by removing the premise from clauses, or merging clauses. Compactness of the rules is maximized while at the same time not affecting the 'scope' property of the classifier.

3. Pruning. The resulting DNF rules from step 1 may contain more than one DNF clauses, which able to classify documents in the same category (overfit). Pruning is done to 'cut' the unused clauses from the rule.

B. Decision Tree

Decision tree decomposes the data space into a hierarchical structure called tree. In textual data context, data space means the presence or absence of a word in the document. Decision tree classifier is a tree comprise of:

a. **Internal nodes**. Each internal node stores the attributes, i.e. collection of words, which will be compared with the words contained in a document.

b. **Edge**. Branches that come out of an internal node are the terms/conditions represent one attribute value.

c. **Leaf**. Leaf node is a category or class of documents.

Decision tree classifying document d_j by testing term weight of the internal nodes label contained in vector $\overline{d_j}$ recursively, until the document is classified at a leaf node. Label of the leaf node will be the document's class.

Decision tree classifiers are built in a top-down fashion [3]:

1. Starting from the root node, document d_j is tested whether it has the same label as the node's (category c_i or $\overline{c_i}$).
2. If the does not fit, select the k-th term (t_k), divide into classes of documents that have the same value as t_k. Create a separated sub-tree for those classes.

3. Repeat step 2 in each sub-tree until a leaf node is formed. Leaf node will contain the documents in category c_i.

The tree structure in decision tree algorithm is easy to understand and interpret, and the documents are classified based on their logical structure. On the contrary, this algorithm requires a long time to do the classification manually. When misclassification at the higher level occurs, it will affect the level below, and the possibility of overfit is high.

Sebastiani [3] explains, to reduce overfitting, several nodes can be trimmed (pruning), by withholding some of the attributes that are not used to build the tree. These attributes determine whether a leaf node will be pruned or not. The next step is comparing the class distribution in used attributes versus unused attributes. If the class distribution of the training documents used to construct the decision tree is different from the class distribution of the class distribution of the training documents retained for pruning, then the nodes are overfit to training documents and can be pruned.

C. k-Nearest Neighbor

In machine learning field k-nearest neighbor (k-NN) algorithm belongs to lazy learner group. Lazy learners, also called example-based classifier [3] or proximitybased classifier [7], do the classification task by utilizing the same existing category labels on the training documents with labels on the test documents.

k-NN starts by searching or determining the number of k nearest neighbor of the documents to be classified. Input parameter k indicates the number of document level to be considered in calculating document (d_j) classification function, $CSV_i(d_j)$. A document is compared with the neighbor classes, to calculate their similarity. Document d_j will become member of category c_i if there are k training documents that are similar to d_j in category c_i . k-NN classification function is defined as follows:

$$CSV_i(d_j) = \sum_{d_z \in Tr_k(d_j)} RSV(d_j, d_z) \cdot [\![\Phi(d_z, c_i)]\!]$$

- $RSV(d_j, d_z)$ is a measure of relationship between testing document d_j with training document d_z.

- $Tr_k(d_j)$ is the set of k testing document d_z to maximize the function $RSV(d_j, d_z)$.

D. Naïve Bayes

Naïve Bayes is a kind of probabilistic classifier that utilize mixture model, a model that combine terms probability with category, to predict document category probability [7]. This approach define classification as the probability of document d_j , which is represented as term vector d_j =

$\langle w_{1j}, \ldots, w_{|T|j} \rangle$, belongs to category c_i.

Document probability is calculated using the following equation:

$$P\left(c_i \middle| \vec{d_j}\right) = \frac{P(c_i)P\left(\vec{d_j} \middle| c_i\right)}{P\left(\vec{d_j}\right)}$$

where $P\left(\vec{d_j}\right)$ is the probability of document d_j (randomly chosen), $P(c_i)$ is the probability of a document to become classified in category c_i.

The size of document vector $\vec{d_j}$ may be large. Therefore, naïve Bayes applies word independence assumption. According to word independence assumption two different document vector coordinates are disjoint [3]. In other words, a term probability in a document does not depend on others. So, the presence of a word has no affect on others, so called 'naïve'. Probabilistic classifier naïve Bayes is expressed in the following equation:

$$P\left(\vec{d_j} \middle| c_i\right) = \prod_{k=1}^{|T|} P\left(w_{kj} \middle| c_i\right)$$

There are two commonly used naïve Bayes variants, namely Multivariate Bernoulli and Multinomial Model.

a. **Multivariate Bernoulli Model**. This model using the term occurrence in document as the document feature. Term occurrence is represent as binary value, 1 for term presences and 0 otherwise. Term occurrence frequency is not taken into account for document classification modeling.

b. **Multinomial Model**. As oppose to multivariate model, this model considers the term occurrence frequency. Document is defined as 'bag of words', along with term frequency of each word.

Classification modeling is conducted based on these occurrence frequencies in the document. Multinomial model has better performance compare with the other naïve Bayes variants [8, 9].

E. Regression-based

Regression is a statistic method for analyzing correlation between two real-value attributes. Regression is used for classifying numeric variables. The purpose of this method is to find matrix F which transforms the matrix A to A' such that $A' = A$.

There are two widely used regression methods for classifying text document, Linear Least Squares Fit (LLSF) and Logistic Regression Classifier. This paper reviews the LLSF method. Suppose that the predicted category label is $p_i = \bar{A} \cdot \bar{X} + b$ and y_i is the actual category label. LLSF calculates the value of vector $\bar{A}$ and scalar b such that error classification on the set of training documents can be minimized.

Reference [3] explains, every document d_j has two vectors, i.e. input vector $I(d_j)$ from the terms weight matrix $|T|$, and output vector $O(d_j)$ from the category matrix $|C|$. $|C|$ consists of m rows and c columns, where m is the number of documents in matrix $|T|$ and c is the number of categories. $|T|$ and $|C|$ have the same row number.

Document classification is defined as the process of determining the output vector $O(d_j)$ of document d_j, according to the value of input matrix $I(d_j)$. The classifiers are constructed by calculating $\bar{M}_{|C|\times|T|}$ matrix such that $\bar{M}I(d_j) = O(d_j)$. Matrix element $\bar{M}$, m_{ik} represent category c_i and term t_k.

F. Support Vector Machine

Similar to regression-based classification, SVM represents documents as vectors. This approach aims to find a boundary, called decision surface or decision hyperplane, which separates two groups of vectors/classes. The system was trained using positive and negative samples from each category, and then calculated boundary between those categories. Documents are classified by first calculating their vectors and partition the vector space to determine where the document vector is located. The best decision hyperplane is selected from a set of decision hyperplane $\sigma_1, \sigma_2, \ldots, \sigma_n$ in vector space $|T|$ dimension that separate the positive and negative training documents. The best decision hyperplane is the one with the widest margin [3, 7].

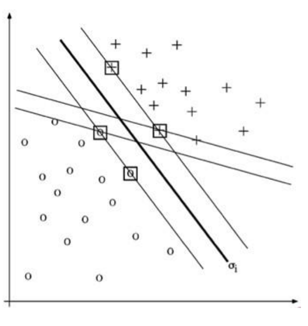

Figure 1. Contoh Support Vector Classifier [3]

Fig. 1 shows how SVM work. The cross (+)

and circle (**O**) symbols represent two training document categories. Cross symbols for the positive ones and circle symbols otherwise. The lines represent decision hyperplanes, there are five decision hyperplanes on the example in Fig. 1. Box symbols are the support vectors, i.e. the documents whose distance against decision hyperplanes will be computed to determine the best hyperplane. σ_i is the best one. Its normal distance against each training documents is the widest. Thus, $\sigma_i$become the maximum possible separation barrier.

IV. CLASSIFIER EVALUATION

Experimental approach was applied as document classifier evaluation method, to measure the effectiveness of the classifiers [3, 6]. Classifier effectiveness describes the classifiers' ability to classify a document in the right category. Three most often used methods to determine effectiveness applied in this study are precision, recall, and accuracy, based on probability technique. Table 1 shows the contingency table that is used to measure probability estimation for category c_i.

TABLE I. CONTINGENCY TABLE FOR CATEGORY c_i [3]

Category c_i		**Expert Judgement**	
		YES	***NO***
Classifier	YES	TP_i	FP_i
Judgement	NO	FN_i	TN_i

To determine precision, recall, and accuracy must first begin by understanding if the classification of a document was a true positive (TP), false positive (FP), true negative (TN), and false negative (FN). TP means the documents being classified correctly as relating to a category. FP determined as documents that is related to the category incorrectly. FN describes documents that is not marked as related to a category but should be. TN means documents that should not be marked as being in a particular category and are not.

a. ***Precision*** (π). *Precision*, π, is defined as

$P(\breve{\Phi}(d_x,c_i) = T|\Phi(d_x,c_i) = T)$, conditional probability of randomly chosen document d_x to be classified under category c_i . Precision explains ability of the classifiers to plce a document under the right category. i^{th} document's precision is calculated as:

$$\pi_i = \frac{TP_i}{TP_i + FP_i}$$

b. ***Recall*** (ρ) . *Recall*, ρ , is determined as $P(\Phi(d_x,c_i) = T|\breve{\Phi}(d_x,c_i) = T)$, the probability of decision is taken for a random document d_x be classified under the right category.

$$\rho_i = \frac{TP_i}{TP_i + FN_i}$$

c. Combining precision and recall may provide better analysis of classifier performance. This is called F-Measure:

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho}$$

where π denote precision, ρ for recall, and positive parameter β that represents the goal of evaluation task. β is given a value of 1 if both precision and recall are considered equally important. If precision is more important than recall, then $\beta = 0$. Conversely, if recall is more important than precision, the value of β is infinite.

d. ***Accuracy*** $(\boldsymbol{A})$. *Accuracy* is measured by
the following formula:

$$A_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

Accuracy is one of the methods commonly used as a measure for classification.
Basic techniques for estimating accuracy of classifiers are [11]:

a) ***Holdout***. Set of documents is partitioned into two parts, 70% as training documents for learning phase, and 30% as testing documents for classification phase.

b) ***k-fold Cross Validation***. This method randomly partitions the documents into k mutually exclusive subsets (called 'fold) S_1, S_2,... , S_k , each of approximately equal size. Model training and testing is performed $k$times, iteratively. At the i^{th}, subset S_i is reserved as the test documents while the remaining subsets are used as training documents. For example, at first iteration S_i will be the testing documents and S_2,S_3,... , $S_k$will be the training documents. Accuracy equals to the ratio of total appropriate classification from k iterations with total number of documents.

V. RESULT AND DISCUSSION

The experiment conducted using 19MclassTextWc text dataset, downloaded from http://weka.wikispaces.com/Datasets. For the algorithms comparison these aspects were involved, i.e. Precision, Recall, and F-Measure, accuracy, and classifier model building time. Chart at Fig. 2 shows the comparison of algorithms on the basis of Precision, Recall, and F-Measure obtained over the six algorithms. It can be seen that naïve Bayes classifier achieve the highest effectiveness values, respectively 0.854, 0.833, and 0.832 for Precision, Recall, and F-Measure. SVM produces very close performance compared to decision tree and regression-based classifiers

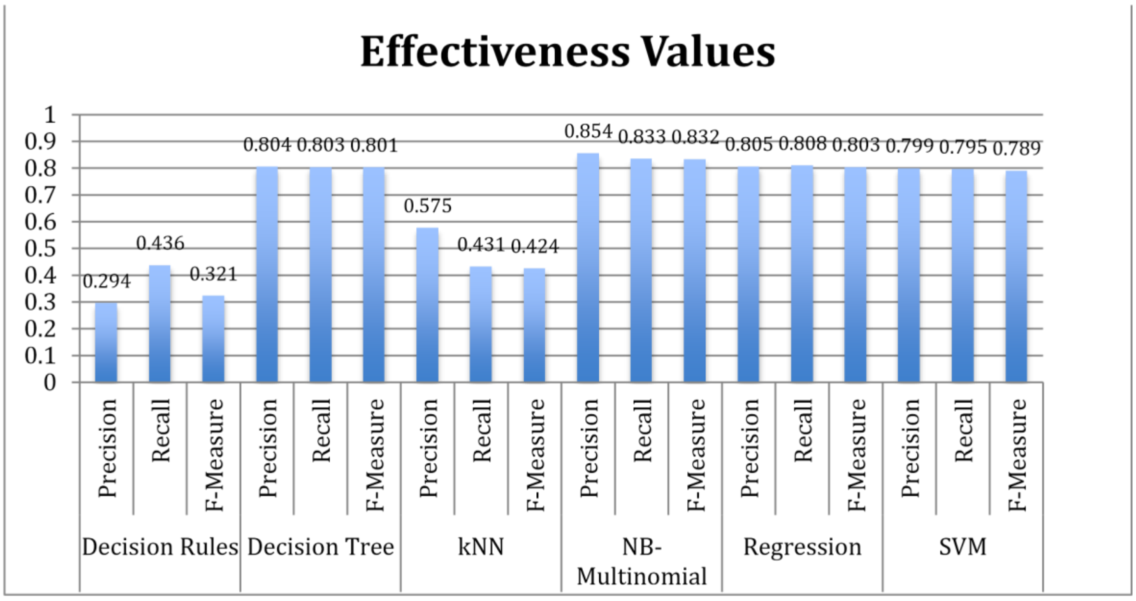

Figure 2. Average Classifier Effectiveness Values

TABLE II. CLASSIFIER ACCURACY

Datasets	**Decision Rules**	**Decision Tree**	**k-NN**	***Naïve Bayes***	**Regression**	**SVM**
fbis	35.73%	72.84%	51.48%	76.90%	79.37%	78.40%
la1s	44.91%	76.65%	44.35%	88.23%	70.84%	84.24%
la2s	46.67%	77.14%	42.80%	89.79%	68.17%	86.73%
oh0	35.09%	80.46%	21.53%	89.03%	73.78%	81.95%
oh5	24.73%	82.24%	21.68%	86.71%	76.91%	77.45%
oh10	26.19%	72.95%	8.38%	81.24%	65.91%	74.86%
oh15	27.71%	74.81%	18.73%	83.79%	70.21%	72.73%
re0	57.45%	75.27%	62.17%	80.39%	82.25%	75.47%
re1	39.89%	79.96%	51.12%	83.34%	79.72%	74.29%
tr11	45.89%	77.29%	48.79%	84.78%	86.96%	74.15%
tr12	43.13%	82.75%	40.58%	83.07%	83.07%	74.44%
tr21	77.38%	79.17%	67.56%	63.39%	86.31%	79.46%
tr23	62.75%	91.67%	54.90%	71.57%	92.16%	74.02%
tr31	58.25%	93.85%	64.83%	94.61%	96.44%	92.13%
tr41	44.76%	90.89%	51.25%	94.42%	94.19%	87.02%

tr45	40.43%	91.16%	41.88%	83.04%	90.14%	81.16%
wap	30.19%	65.64%	39.87%	81.09%	73.21%	82.18%

TABLE III. TIME TO BUILD A MODEL

Datasets	**Decision Rules**	**Decision Tree**	**k-NN**	***Naïve Bayes***	**Regression**	**SVM**
oh5	0.96	17.91	0	0.01	84.55	1.27
oh10	1.18	32.99	0	0.01	108.81	1.59
oh15	1.66	26.99	0	0.01	80.88	1.07
re0	6.63	48.88	0.01	0.03	117.35	3.19
re1	7.19	100.23	0	0.01	385.01	4.74
tr11	1.13	13.82	0	0.01	36.7	0.64
tr12	1.73	7.4	0	0.01	25.4	0.41
tr21	1.35	13.31	0	0.02	27.57	0.7
tr23	0.65	2.58	0	0.01	9.99	0.25
tr31	11.38	31.47	0.01	0.05	130.98	2.44
tr41	4.86	26.66	0.01	0.02	126	1.84
tr45	4.1	20.49	0	0.02	114.97	0.14
wap	54.45	335.36	0	0.02	577.19	7.33

TABLE IV. COMPARISON OF ACCURACY AND TIME TO BUILD A MODEL

Algorithms	**Accuracy (%)**	**Precision**	**Recall**	**F-Measure**	**Time (second)**
Decision Rules	43.6	0.294	0.436	0.321	10.9
Decision Tree	80.28	0.804	0.803	0.801	103.72
k-NN	43.05	0.575	0.431	0.424	0.01
Naïve Bayes	83.26	0.854	0.833	0.832	0.03
Regression	80.57	0.805	0.808	0.803	242.22
SVM	79.45	0.799	0.795	0.789	3.42

The comparison of algorithms on the basis of accuracy is presented in Table II. Data presented in Fig. I matches with the ones in Table II in many cases. Directly proportional to the evaluation of precision, recall, and Fmeasure, Table III shows that naïve Bayes classifier has the highest accuracy rate among the six classifiers. The average accuracy of naïve Bayes is 83.25%, followed by regression-based, decision tree, SVM, decision rules, and k-NN.

Another measure that is obtained form the experiment is the amount of time taken to build the classifier models (see Table III). It can be seen that the average time required by k-NN classifiers is the smallest (fastest), only 0.01 seconds. In contrast, regression-based classifiers take a long time to build a text classifier models. The average amount of time to accomplish building the model is 242.22 seconds.

In Table IV we try to conclude the relation between classifier effectiveness values with amount of time taken to build classifier models. Both decision rules and k-NN have poor classification performance. Compare to kNN, decision rules has the lowest in terms of precision, recall, and F-measure. Yet, its accuracy is higher than k-NN's. SVM can reach high effectiveness performance in average of time 3.42 seconds for building a classification model. In terms of time, decision tree and regression-based algorithm require a huge amount of time to build classification model. However, they can classify the documents well. Overall, results of the experiment indicate that naïve Bayes algorithm is superior among the six algorithms, assessed from the aspects of effectiveness and time. It requires small amount of time to build the model with high accuracy.

VI. CONCLUSION

This study compared performance of six machine learning based classification algorithms, namely decision rules, decision tree, k-NN, naïve Bayes, regression-based, and SVM. Comparison is based on time and four classifier effectiveness measurements: precision, recall, F-measure, and accuracy. After the experiment and analysis of the results following conclusions were drawn:

1. Decision rules and k-NN performance are lack. Decision rules' effectiveness is the lowest, but better in accuracy compare to k-NN.
2. The algorithms which can build classifiers with high effectiveness rate are naïve Bayes, decision tree, regression-based, and SVM.
    a. SVM is able to classify the documents well in small amount of model building time.
    b. Decision tree and regression-based algorithm have an equally good performance in classifying multi-class text documents, with average precision, recall, and F-measure values more than 0.8, as well as accuracy rate which is more than 80%. Yet, they are weak in time to build the classifier models.
    c. Experiment result shows naïve Bayes has the highest effectiveness values, as well as spent small amount of time to build the classifier models.
3. Regarding the time taken to build classifier model, k-NN is the fastest, while regression-based is the slowest. Using the chosen datasets, k-NN can build a model in average of 0.01 second. Regression-based requires average of 242.22 seconds to build a model. The greater the dataset, the longer it takes for regression-based to build a classifier model.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350. [4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.