Automatic document metadata extraction using support vector machines

Conference Paper · June 2003

DOI: 10.1109/JCDL.2003.1204842 · Source: IEEE Xplore

6 authors, including:

Hui Han

24 PUBLICATIONS **1,043** CITATIONS

SEE PROFILE

Zhenyue Zhang

Zhejiang University

66 PUBLICATIONS **2,836** CITATIONS

SEE PROFILE

Edward Alan Fox

Virginia Polytechnic Institute and State University

609 PUBLICATIONS **9,989** CITATIONS

SEE PROFILE

Automatic Document Metadata Extraction using Support Vector Machines

Hui Han[1] C. Lee Giles[1,2] Eren Manavoglu[1] Hongyuan Zha[1]
[1]Department of Computer Science and Engineering [2]The School of Information Sciences and Technology
The Pennsylvania State University University Park, PA, 16802
{hhan,zha,manavogl}@cse.psu.edu giles@ist.psu.edu

Zhenyue Zhang
Department of Mathematics, Zhejiang University
Yu-Quan Campus, Hangzhou 310027, P.R. China
zyzhang@math.zju.edu.cn

Edward A. Fox
Department of Computer Science, Virginia Polytechnic Institute and State University
660 McBryde Hall, M/C 0106, Blacksburg, VA 24061
fox@vt.edu

Abstract

Automatic metadata generation provides scalability and usability for digital libraries and their collections. Machine learning methods offer robust and adaptable automatic metadata extraction. We describe a Support Vector Machine classification-based method for metadata extraction from header part of research papers and show that it outperforms other machine learning methods on the same task. The method first classifies each line of the header into one or more of 15 classes. An iterative convergence procedure is then used to improve the line classification by using the predicted class labels of its neighbor lines in the previous round. Further metadata extraction is done by seeking the best chunk boundaries of each line. We found that discovery and use of the structural patterns of the data and domain based word clustering can improve the metadata extraction performance. An appropriate feature normalization also greatly improves the classification performance. Our metadata extraction method was originally designed to improve the metadata extraction quality of the digital libraries Citeseer[17] and EbizSearch[24]. We believe it can be generalized to other digital libraries.

1 Introduction and related work

Interoperability is crucial to the effective use of Digital Libraries (DL) [19, 23]. The Open Archive Initiatives Protocols for Metadata Harvesting (OAI-PMH) is critical for the process, facilitating the discovery of content stored in distributed archives [7, 18]. The digital library CITIDEL (Computing and Information Technology Interactive Digital Educational Library), part of NSDL (National Science Digital Library), uses OAI-PMH to harvest metadata from all applicable repositories and provides integrated access and links across related collections [14]. Support for the Dublin Core (DC) metadata standard [31] is a requirement for OAI-PMH compliant archives, while other metadata formats optionally can be transmitted.

However, providing metadata is the responsibility of each data provider with the quality of the metadata a significant problem. Many data providers [13, 4] have had significant harvesting problems with XML syntax and encoding issues, even leading to unavailability of service [18]. In fact, some digital libraries have no metadata to harvest (some search engines have little or no metadata), or metadata that is not OAI compliant, e.g., CiteSeer [17]. Non-compliant metadata must be either automatically wrapped to work with the OAI protocol, or manually encoded. Building tools for automatic document metadata extraction and representation will therefore significantly improve the amount of metadata available, the quality of metadata extracted, and the efficiency and speed of the metadata extraction process.

Several methods have been used for automatic metadata extraction; regular expressions, rule-based parsers, and machine learning are the most popular of these. In general machine learning methods are robust and adaptable and, theoretically, can be used on any document set. Generating the labeled training data is the rather expensive price that has

Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL'03)
0-7695-1939-3/03 $17.00 © 2003 IEEE

to be paid for learning systems. Although regular expressions and rule-based systems do not require any training and are straightforward to implement, their dependence on the application domain and the need for an expert to set the rules or regular expressions causes these methods to have limited use. Machine learning techniques for information extraction include symbolic learning, inductive logic programming, grammar induction, Support Vector Machines, Hidden Markov models, and statistical methods. Hidden Markov models (HMMs) are the most widely used generative learning method for representing and extracting information from sequential data. However, HMMs are based on the assumption that features of the model they represent are not independent from each other. Thus, HMMs have difficulty exploiting regularities of a semi-structured real system. Maximum entropy based Markov models [20] and conditional random fields [16] have been introduced to deal with the problem of independent features.

Recent work by Chieu [5] suggests that the information extraction task also can be addressed as a classification problem. Encouraged by their success in handling high dimensional feature spaces for classification problems [12, 9], we investigate Support Vector Machines (SVMs) for metadata extraction. Related work includes Kudoh et al using the SVM method for chunk identification, Mcnamee et al using a SVM for named entity extraction [22, 15, 29], and Pasula et al using relational probability models to solve identity uncertainty problems [10].

This paper discusses a machine learning method for automatic metadata extraction. The reported extraction results are based on experiments conducted on research papers. Most of the directly indexable information (e.g., authors' names, affiliations, addresses, and the title of the paper) are gathered in the header of a research paper. The header [27] consists of all the words from the beginning of the paper up to either the first section, usually the introduction, or to the end of the first page, whichever occurs first. In the experimental results section we illustrate the dominance of the introduced SVM-based metadata extraction algorithm over the well-known HMM based systems [27]. We also introduce a method for extracting individual names from the list of authors within the same framework and present a new document metadata extraction method using SVM classification, combining chunk identification. A new feature extraction method and an iterative line classification process using contextual information also are presented.

The remainder of the paper is organized as follows: section 2 describes the problem and dataset; section 3 presents our metadata extraction method, together with the cross validation results on 500 training headers; section 4 presents the experiment result of our metadata extraction algorithm on the test dataset; section 5 discusses the aspects to be improved and planned future work.

2 Problem definition and dataset

The Dublin Core has been widely used as a metadata standard and defines 15 elements for resource description: Title, Creator, Subject, Description, Contributor, Publisher, Date, Type, Format, Identifier, Source, Relation, References, Is Referenced By, Language, Rights and Coverage. However, this is only a basic set of metadata elements and is used by OAI-PMH for "minimal" interoperability. Extending document metadata through information on both authors (such as affiliation, address, and email), and documents (such as publication number and thesis type), would provide greater representation power. It also would help in building unified services for heterogeneous digital libraries, while at the same time enabling sophisticated querying of the databases and facilitating construction of the semantic web [3]. Seymore et al defined 15 different tags for the document header [27] to populate the Cora search engine [21], 4 of which are the same as those in the Dublin Core. Two of the remaining tags, introduction and end of page, are functional rather than informative, indicating the end of the header. Leaving out the functional tags, we adopt their format as extended metatags for research papers. We further propose to define affiliation as part of the address, instead of an exclusive tag. Table 1 is a short explanation of the extended metatags and the mapping to Dublin Core metadata elements.

Figure 1 is an example of meta-tagged document header. Document metadata extraction also can be viewed as labeling the text with the corresponding metatags. Each metatag corresponds to a class. Lines 22 and 25 are multi-class lines containing chunks of information from multiple classes. We define a chunk of information as consecutive words that belong to the same class. Line 22 and 25 contain the chunks of 5 classes: email, web, affiliation, address, and note. All the other lines contain information belonging to one class only and are therefore called single-class lines.

We use the labeled dataset provided by Seymore et al [27] to test our method of metadata extraction. The dataset contains 935 headers of computer science research papers, with 500 of those belonging to the training set and the remaining 435 headers belonging to the test set. The training set includes a total of 10025 lines and 23557 word tokens whereas there are 8904 lines and 20308 word tokens in the test set. These headers are text files converted from the pdf and ps files. Each line ends with a carriage return and the line break marks $+L+$ are provided by the dataset for identification.

The document headers are semi-structured. We observe that among total 10025 lines from 500 training headers, the majority (9775 lines, 97.51%) are single-class lines and only 250 (2.49%) lines are multi-class lines. Even after removing the abstract section which is mostly single-class

Table 1. Extended metatags and their mapping to Dublin Core metadata elements

Extended Metatag	DC Element	Explanation
Title	Title	Title of the paper
Author	Creator	The name(s) of the author(s) of the document
Affiliation		Author's affiliation
Address		Author's address
Note		Phrases about acknowledgment, copyright, notices, and citations
Email		Author's email address
Date		Publication date
Abstract Introduction	Description	An account of the content Introduction part in the paper
Phone		Author's phone number
Keyword	Subject	The topic of the content of the document
Web		URL of Author's webpage of the document
Degree		Language associated with thesis degree
Pubnum		Publication number of the document
Page		The end of the page

lines, multi-class lines still account for only 4.98% of all lines. Classifying each line into one or more classes thus appears to be more efficient for meta-tagging than classifying each word. Table 2 lists the class distributions of the lines from the 500 training headers.

The predicted tags for previous and next lines are also good indicators of the class(es) to which a line belongs. For instance, an abstract has consecutive lines uninterrupted by lines of other classes, and title lines usually come before author lines. Making use of such contextual information among lines we feel will increase the line classification performance.

We propose a third algorithm for processing the lines predicted to contain chunks of information from multiple classes. Since each chunk has consecutive words, we consider extracting metadata from the multi-class lines as the problem of seeking the optimal chunk boundaries. Recognition of individual author names within multi-author lines can also be considered as the problem of seeking the right chunk boundary, in this case between the author names. For example, does the line "Chungki Lee James E. Burns" refer to two authors "Chungki Lee" and "James E. Burns," two authors "Chungki Lee James" and "E. Burns," or one author "Chungki Lee James E. Burns"?

Based on the structural patterns of the document headers,

1:<title> Stochastic Interaction and Linear Logic +L+ </title>
2:<author> Patrick D. Lincoln John C. Mitchell Andre Scedrov +L+ </author>
3: <abstract> Abstract +L+
4:We present stochastic interactive semantics for prepositional linear +L+
...
22:<email> jcm@cs.stanford.edu </email> <web> http://theory.stanford.edu/people/jcm/home.html </web> <affiliation> Department of Computer Science, Stanford University, </affiliation> <address> Stanford, CA 94305.</address> <note> Supported in part +L+
23:by an NSF PYI Award, matching funds from Digital Equipment Corporation, the Pow-ell Foundation, and Xerox Corporation; and the Wallace F. and Lucille M. Davis Faculty +L+
24:Scholarship. +L+ </note>
25:<email> andre@cis.upenn.edu </email> <web>http://www.cis.upenn.edu/∼andre </web> <affiliation> Department of Mathematics, University of Pennsylvania, </affiliation> <address> Philadelphia, PA 19104-6395. </address> <note> Partially supported by +L+
26:NSF Grants CCR-91-02753 and CCR-94-00907 and by ONR Grant N00014-92-J-1916. Sce-drov is an American Mathematical Society Centennial Research Fellow. +L+ </note>

Figure 1. Example 1 labeled document header and metadata. Each line starts with the line number.

we decompose the metadata extraction problem into two sub-problems – (1) line classification and (2) chunk identification of multi-class and multi-author lines. Accurate line classification is a critical step, since it directly affects the performance of the chunk identification module.

3 Metadata Extraction Algorithm

This section describes two important aspects of our work, SVM classification and feature extraction. The metadata extraction algorithm is discussed in detail, together with the corresponding ten-fold cross-validation result on the 500 training headers. Performance is evaluated using accuracy, precision, recall, and F measure.

3.1 Support Vector Machine Classification

Support Vector Machine is well known for its generalization performance and ability in handling high dimension data. Consider a two class classification problem. Let $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a two-class training dataset, with x_i a training feature vector and their labels $y_i \subset$ (-1, +1). The

Table 2. Class distribution among 10025 total lines from 500 training header

Class No.	Class Name	Number of Lines	Percentage
1	Title	832	8.3%
2	Author	724	7.2%
3	Affiliation	1065	10.6%
4	Address	629	6.3%
5	Note	526	5.2%
6	Email	336	3.4%
7	Date	182	1.8%
8	Abstract	5007	50.0%
9	Introduction	326	3.3%
10	Phone	61	0.6%
11	Keyword	142	1.4%
12	Web	38	0.4%
13	Degree	169	1.7%
14	Pubnum	116	1.1%
15	Page	166	1.7%

SVM attempts to find an optimal separating hyperplane to maximally separate two classes of training samples. The corresponding decision function is called a classifier. The kernel function of an SVM is written as $K(x_a, x_b)$ and it can be an inner product, Gaussian, polynomial, or any other function that obeys Mercer's condition [30, 6].

We choose the Gaussian kernel for the SVM and base our experiment on the software SVM_light [11]. We set the parameter gamma (-g), the spread of the Gaussian kernel as 0.1, and all other parameters set by SVM_light. We extend the SVM to multi-class classifiers in the "One class versus all others" approach, i.e., one class is positive and the remaining classes are negative.

3.2 Feature Extraction

Most of the previous work on information extraction uses word-specific feature representations [27, 15, 29]. Recent research on the topic suggests that line-specific features also could be useful [20].

We make use of both **word** and **line**-specific features to represent our data. Each line is represented by a set of word and line-specific features.

We design a rule-based, context-dependent word clustering method explained below for word-specific feature generation, with the rules extracted from various domain databases and text orthographic properties of words (e.g. capitalization) [26]. Word clustering methods group similar words and use the cluster as a feature. Distributional clustering methods have shown significant dimensionality reduction and accuracy improvement in text classification [2, 28, 8]. While distributional clustering needs to use labeled training data, our rule-based method relies on the prior knowledge embedded in domain databases.

We collect the following databases to gather apriori knowledge of the domain:

- Standard on-line dictionary of Linux system
- Bob Baldwin's collection of 8441 first names and 19613 last names
- Chinese last names
- USA state names and Canada province names
- USA city names
- Country names from the World Fact Book [1], and
- Month names and their abbreviations

We also construct domain databases, i.e., word lists from training data for classes: affiliation, address, degree, pubnum, note, abstract, keyword, introduction, and phone. Words and bigrams that appear frequently in the lines of each class mentioned are selected to enter these word lists. Frequency thresholding is used to define the list size [32]. The abstract class word list contains one word "abstract" and the affiliation class list contains words shown in Table 3.

We then cluster words and bigrams based on their membership in the domain databases and their text orthographic properties. The words and bigrams in the same cluster are represented by a common feature, which we call word-specific feature. For example, an author line *"Chungki Lee James E. Burns"* is represented as *"Cap1NonDictWord: :MayName: :MayName: :SingleCap: :MayName:"*, after word clustering.

Such word clustering shows significant improvement in our experiment of classifying lines (details will be given in another paper). A reason is that the word cluster statistics give a more robust estimate than the original sparse word statistics [2, 28].

We define the weight of a word-specific feature as the number of times this feature appears in the sample (line).

The following is the list of line-specific features we believe to be useful for line classification. In particular, feature ClinePos is found to be very important in correct classification of title lines.

CsenLen Number of the words the line contains.

ClinePos The position of the line, i.e., line number.

CDictWordNumPer The percentage of the dictionary words in the line.

CNonDictWordNumPer The percentage of the non-dictionary words in the line.

CCap1DictWordNumPer The percentage of the dictionary words with first letter capitalized in the line.

CCap1NonDictWordNumPer The percentage of the non-dict words with first letter capitalized in the line.

CdigitNumPer The percentage of the numbers in the line.

We also have a feature for representing the percentage of the class-specific words in a line. CaffiNumPer is the percentage of the affiliation words in the line and CaddrNumPer, CdateNumPer, CdegreeNumPer, CphoneNumPer, CpubNumPer, CnoteNumPer, and CpageNumPer are the percentage of the address words, date words, degree words, phone words, publication number words, note words, and page number words, respectively. We assign weight to the line-specific features according to their definition.

Table 3. Affiliation Class Word List

DF Value	Word	DF Value	Word
325	University	37	Laboratory
221	Department	34	Technology
111	Univ	33	Dept
77	Institute	27	Systems
47	Research	26	School
39	Sciences	26	Center

However, our experiments show that SVM doesn't handle well the case when different features have very different ranges of values. For example, the feature "CsenLen" could have a weight of 40, while the line-specific feature **CdictWordNumPer** weight is over the range [0, 1]. Features with large scale may dominate the features with small weight. Therefore, we use the $\|X\|_\infty$ to normalize the feature weight and increase the classification performance as shown in the next section.

3.3 Line Classification Algorithms

The following is a two-step algorithm for classifying text lines into a single class or multiple classes. The two components are independent line classification followed by contextual line classification.

3.3.1 Independent line classification

In the first step, feature vectors are generated based on the feature extraction methodology described in the previous section. After removing the features with data frequency values < 3, we get feature vectors with 1100 dimensions on average for ten-fold cross validation. A feature vector is labeled as class C if the corresponding line contains words belonging to class C. Training feature vector set for class C is generated by collecting all the feature vectors with label C as positive samples and all the rest as negative; the same procedure applies to all classes. Note that a feature vector could have multiple labels and thus can belong to multiple training feature vector sets. 15 classifiers are then trained on the 15 labeled feature vector sets. Test lines are classified into one or more classes if their feature vectors are scored positive by the corresponding classifier. This process is called independent line classification (also shown in Figure 2), since each line is classified independently.

Table 4. Word-specific feature set

Feature	Explanation
:email:	using regular expression match
:url:	using regular expression match
:singleCap:	a capital letter like M or M.
:postcode:	such as PA, MI
:abstract:	abstract
:keyword:	key word, key words, keyword, keywords
:intro:	introduction
:phone:	tel, fax, telephone
:month:	a word in the month list
:prep:	at, in, of
:degree:	a word or bigram in the degree domain word list
:pubnum:	a word or bigram in the publication number domain word list
:notenum:	a word or bigram in the note domain word list
:affi:	a word or bigram in the affiliation domain word list
:addr:	a word or bigram in the address domain word list
:city:	a word or bigram in the city name list
:state:	a word or bigram in the state name list
:country:	a word or bigram in the country name list
:mayName:	a word in one of the 3 name lists
:Cap1DictWord:	a dictionary word with first letter capitalized
:DictWord:	small case dictionary word
:NonDictWord:	small case non dictionary word
:Dig[3]:	a number of three digits
The word-specific feature considers text orthographic properties, e.g., BU-cs-93 is converted to :CapWord2-LowerWord2-Digs2:	

Table 5 lists the ten-fold cross-validation results on the training dataset for the independent line classification algorithm. Figure 3 shows the F measure of independent line classification before and after normalization using ten-fold cross-validation on 500 training headers. Due to space limitations, we are not able to report our results for precision, recall, and accuracy. The effect of normalization is a significant improvement in performance. Normalization is especially important in identifying the rare classes, such as class 5 (note), 11 (keywords), and 12 (web). Consider class 5 "note" as an example, the positive note samples occupy 5.3% (53 out of 1001.5 averaged for each fold of ten-fold cross validation) of all test samples. Without normalization,

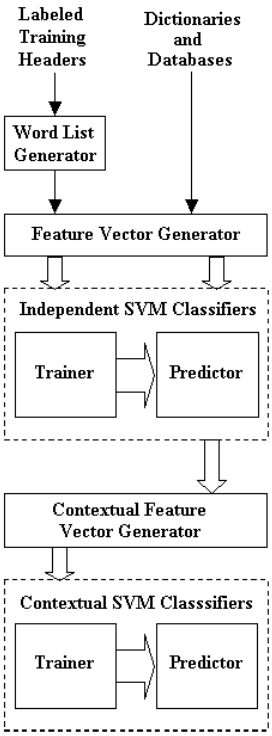

Figure 2. Overview of Line Classification Training Module.

the note classifier classifies all testing samples into non-"note" classes. Thus, the recall for class 5 "note" is zero and the precision is infinite. Normalization appears to increase the importance of features in the class "note", which then enhances "note" samples for the "note" classifier.

3.3.2 Iterative contextual line classification

The second step makes use of the sequential information among lines discussed in section 2 to improve the classification of each line. We encode the class labels of N lines before and after the current line L as binary features and concatenate them to the feature vector of line L formed in step one, independent line classification. A contextual line classifier for each metatag is then trained based on these labeled feature vectors with additional contextual information. Line feature vectors for testing are extended the same way. Their neighbor lines' class labels are those predicted by the independent line classifier. Test lines are then reclassified into one or more classes by the contextual line classifiers. This contextual line classification is repeated such that in each iteration, the feature vector of each line is extended by incorporating the neighbor lines' class label information predicted in the previous iteration. The procedure converges when the percentage of lines with new class labels is lower than a threshold. The threshold value is set to 0.7% in our experiments, and N is chosen to be 5. Ramshaw et al show the positive effect of a similar iterative algorithm on transformation-based learning for rule-selection [25].

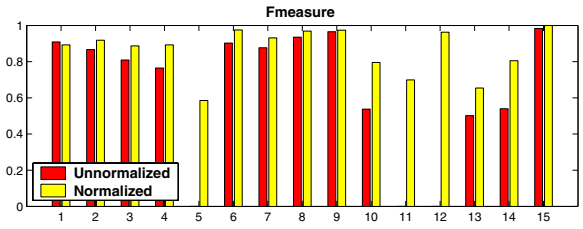

Figure 3. F measure of the independent line classification before and after normalization. X axis - class number; Y axis - F measure.

The contextual information we use for line classification is encoded by the binary features P_{ij} if the previous *i*th closest line belongs to class *j* and N_{ij} if the next *i*th closest line belongs to class *j*, with $i \in (1..5)$ and $j \in (1..15)$. We found that choosing P_{ij} and N_{ij} to be 0.5/0, instead of 1/0 achieves better line classification performance, based on the experiment on the training dataset. This is because the line feature values are already normalized into the range [0, 1]. Choosing the midpoint of this range as the weight for up to 150 ($15*10$) contextual features is a type of normalization and is found to be more effective.

Figure 4 shows the performance evaluated by the F measure in each round of the iterative contextual line classification. As expected, the performance is stabilized within the first 10 iterations. It also shows that the first two rounds are responsible for most of the performance improvement. This behavior suggests two iteration steps can be used instead of waiting for absolute convergence. Table 6 lists the results achieved for each of the 15 classes when the iterative procedure converges. The small sample sizes of the class – degree, note, phone, keyword, and publication number – as shown in Table 6 may account for their poor classification performance. Seymore et al report the same phenomenon on the class – degree, note and publication number – using HMM model [27].

3.4 Extract metadata from multi-class lines

After classifying each line into one or more classes, we now extract metadata from each multi-class line based on the predicted class labels for this line. As discussed the metadata extraction task from multi-class lines is turned into the chunk identification task. Chunk identification of

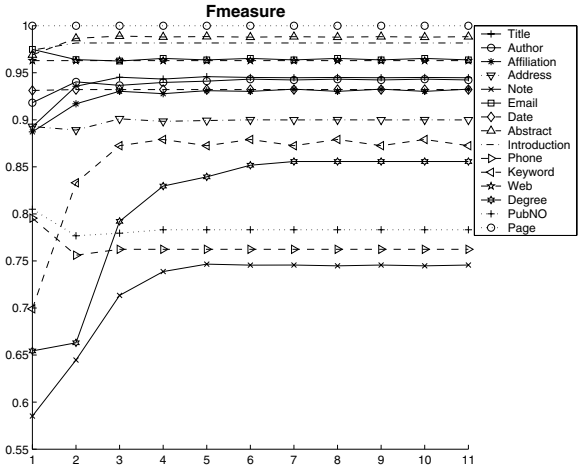

Figure 4. F measure in each round of the iterative contextual line classification. X axis - iteration round; Y axis - F measure.

Table 5. Independent line classification performance.

Class Name	Precision	Recall	F measure	Accuracy
Title	89.6%	88.9%	89.3%	98.2%
Author	94.2%	89.6%	91.8%	98.8%
Affiliation	93.8%	84.2%	88.7%	97.7%
Address	93.9%	85.1%	89.3%	98.8%
Note	82.3%	45.4%	58.5%	96.6%
Email	97.6%	97.4%	97.5%	99.8%
Date	97.2%	89.4%	93.1%	99.8%
Abstract	96.1%	97.7%	96.9%	96.9%
Introduction	98.8%	96.0%	97.4%	99.8%
Phone	93.8%	69.1%	79.5%	99.8%
Keyword	95.2%	55.2%	69.9%	99.3%
Web	100%	92.8%	96.3%	99.9%
Degree	86.0%	52.8%	65.4%	98.9%
Pubnum	91.7%	71.8%	80.5%	99.6%
Page	100.0%	100.0%	100.0%	100.0%

an N-class line is analogous to finding $N-1$ chunk boundaries in the line. Punctuation marks and spaces between words are candidate chunk boundaries.

Table 7 shows that 86% of the multi-class lines in training data are two-class lines. We search for the optimal chunk boundary which yields the maximum difference between the two chunks. Independent line classifiers are applied to calculate the difference between chunks.

Every punctuation mark and space can be a candidate chunk boundary for two-class lines. We consider only punctuation marks as candidates if two or more punctuation marks are used in the line; otherwise we try each punctuation mark and space. Assuming that each class has only one chunk in the line, two-class chunk identification is to find the optimal chunk boundary.

"The Ohio State University, Columbus, OH 43210-1277" is an example of two-class line of affiliation and address. Each comma is a candidate chunk boundary.

We call the affiliation classifier as classifier 1 and the address classifier as classifier 2. The classifiers we use here are the SVM line classifiers trained by single-class lines of the training dataset. We consider each chunk as a short line.

Definitions:

P_1 the classification score of chunk P by classifier 1;

P_2 the classification score of chunk P by classifier 2;

N_1 the classification score of chunk N by classifier 1;

N_2 the classification score of chunk N by classifier 2;

P_{12} = P_1 - P_2; N_{21} = N_2 - N_1;

PN_1 = P_1 - N_1; PN_2 = P_2 - N_2;

We choose the optimal chunk boundary as the punctuation mark or space yielding the maximal $P_{12} * N_{21}$. Chunk P is classified into class 1 if $PN_1 > 0$, and (1) $PN_1 * PN_2 < 0$ or (2) $PN_1 * PN_2 > 0$ and $\|PN_1\| = max(\|PN_1\|, \|PN_2\|)$, class 2 otherwise.

This two-class chunk identification algorithm results in an accuracy of 75.5% (160 out of 212 two-class lines from training samples). Accuracy here is defined as the percentage of the lines whose chunk boundaries are correctly predicted versus the total number of two-class lines. (This is the lower boundary of the accuracy.)

Many N-class ($N > 2$) chunk identification tasks may be simplified to two-class chunk identification tasks. For instance, using the positions of email and URL in the line, we may simplify the three-class chunk identification tasks as two-class chunk identification tasks. The position of the email address in the following three-class line "*International Computer Science Institute, Berkeley, CA 94704. email: aberer@icsi.berkeley.edu. Supported by Schweizerische Gesellschaft zur Forderung der Informatik und ihrer Anwendungen*" is a natural chunk boundary between the other two classes.

We are exploring more general multi-class chunk identification techniques.

3.5 Recognize authors in the multi-author lines

We consider the author lines with less than 4 words as *single-author lines* and the author lines with 4 or more words as *multi-author lines*. We further define a multi-author line where the authors are separated by spaces only

Table 6. Performance (%) of contextual line classification iteration algorithm when converges and the F measure increase than that of the independent line classification

Class Name	Precision	Recall	F measure (Increase)	Accuracy
Title	93.9	95.0	94.5(5.2)	99.1
Author	97.3	91.4	94.2(2.4)	99.2
Affiliation	96.4	90.3	93.3(4.5)	98.6
Address	93.6	86.7	90.0(0.71)	98.8
Note	86.4	65.6	74.6(16.0)	97.6
Email	98.9	94.0	96.4(-1.1)	99.8
Date	97.2	89.5	93.2(0.1)	99.8
Abstract	98.5	99.2	98.8(1.9)	98.8
Introduction	100.0	96.4	98.2(0.8)	99.9
Phone	98.3	62.3	76.2(-3.3)	99.7
Keyword	96.7	79.5	87.2(17.3)	99.7
Web	100.0	92.8	96.3(0.0)	99.9
Degree	91.4	80.5	85.6(20.1)	99.3
Pubnum	97.3	65.5	78.3(-2.2)	99.6
Page	100.0	100.0	100.0(0.0)	100.0

Table 7. The distribution of the multi-class lines in 500 training headers

N-Class	Number of Lines	Percentage
2	212	84.8%
3	33	13.2%
4	4	1.6%
5	1	0.4%

as *space-separated multi-author line*. Similarly, a multi-author line where the authors are separated by punctuation marks is defined as *punctuation-separated multi-author line*.

We extract a total of 326 multi-author lines from the training dataset as the dataset for our experiment on recognizing authors from the multi-author lines. Among the 326 multi-author lines, 227(69.6%) lines are punctuation-separated and 99(30.4%) are space-separated. Based on the different characteristics punctuation-separated multi-author lines and space-separated multi-author lines possess, we choose the following different strategies for either case.

3.5.1 Chunk identification in punctuation-separated multi-author lines

As we discussed before, to recognize each name from the multi-author lines is to identify chunk boundaries between author names. It is obvious that the spaces and punctuation marks between words are the candidate chunk boundaries. The problem now becomes classifying each space or punctuation mark as chunk boundary or not. We consider only the punctuation mark in the line as the candidate chunk boundary if there are two or more punctuation marks in the line; otherwise, we examine each space and punctuation mark. The dictionary word "and" is considered as a punctuation mark. The spaces next to a punctuation mark are ignored.

We design the feature vector for each space and punctuation mark using both the raw features of the punctuation mark itself such as "," or "&", and the contextual features listed in Table 8. We also convert each word of the line into a 5-tuple $< FN, LN, L, FC, D >$. Each element of the 5-tuple is defined as follows.

FN: 1 if the word is in the first name list, 0 otherwise.

LN: 1 if the word is in the last name list, 0 otherwise.

L: 1, 2 or 0, indicates the word is of one letter, two letters, or more than two letters, respectively.

FC: 1 if the word is capitalized, 0 otherwise.

D: 1 if the word is a dictionary word, 0 otherwise.

We use the attributes defined in the above tuple to represent the contextual feature (8) in Table 8 in the converted format. The motivation is that if the closest word to a punctuation mark appears only on the first name list, or only on the last name list, it helps to classify if this punctuation mark is the right chunk boundary. For example, if "Leonidas Fegaras, David Maier" satisfies this pattern "[10010(First name)] [01011(Last name)], [10011(First name)] [00010(Last name)]", it will be reasonable to classify the comma as the right chunk boundary. However, the big overlap between the first name list and the last name list makes such feature representation of each word ineffective.

We find from the stepwise feature selection that the dominating features in classifying chunk boundary are the punctuation marks themselves. Therefore in implementation, we design simple heuristic rules to make use of the punctuation marks to extract each name from the punctuation-separated multi-author line.

Table 9 lists the chunk identification performance on punctuation-separated multi-author lines. The evaluation is based on the percentage of punctuation marks classified correctly.

3.5.2 Chunk identification in space-separated multi-author lines

Space-separated multi-author lines do not have any explicit information for chunk boundary recognition, unlike punctuation-separated lines. The valid patterns for author

Table 8. Contextual features for each candidate chunk boundary in punctuation-separated multi-author line

No.	Feature
1	The number of total punctuation marks of the same kind in the line
2	The position of this punctuation mark
3	The number of words before this punctuation mark
4	The number of words after this punctuation mark
5	The number of words between the previous and the current punctuation mark
6	The number of words between the current and the next punctuation mark
7	The ratio of the number of words before and after this punctuation mark
8	The previous and next 5 words in converted feature representation

Table 9. Chunk boundary identification performance of punctuation-separated multi-author lines

Accuracy	Precision	Recall	F measure
93.31	82.38	96.65	88.95

names are the source of information in this case. [Mary(Full Name)] [Y.(Name Initial)], for instance, cannot be a valid name.

The algorithm for extracting names from space-separated multi-author lines has four steps. Step 1, generate all potential name sequences for the space-separated multi-author lines based on the valid patterns of names that we define in Table 10. Step 2, design the feature vector for each potential name sequence. We manually label each potential name sequence as 1 or -1 by checking each name in this sequence from the web. Step 3, train a SVM name sequence classifier by the labeled training samples. Step 4, if the test space-separated multi-author line has only one potential name sequence, it is the predicted name sequence. Otherwise, classify each of its potential name sequences. The name sequence with the highest score is predicted as the correct name sequence.

For example, the line "Alan Fekete David Gupta Victor Luchangco Nancy Lynch Alex Shvartsman" has three potential name sequences (Figure 5). We generate three reasonable sequences, with each name separated by $\diamond$. The "1" and "-1" in front of each name sequence identifies the sequence as a positive sample or a negative sample. The number at the beginning of each sequence is the classification score. The first sequence achieves the highest score and is predicted correctly.

Table 10. The valid patterns of a name. "F"- Full Name; "F^-" - Full Name with hyphen, e.g., Jon-hey; "I" - Name Initial; "s" - lower case word

Pattern Class	Patterns
1	$(F\|F^-)F, (F\|F^-)(F\|F^-)F$ $(F\|F^-)(F\|F^-)(F\|F^-)F$ e.g., Yu-Chee Tseng
2	$(F\|F^-)IF, (F\|F^-)IIF, (F\|F^-)IIIF$ e.g., Dhabaleswar K. Panda
3	IF, IIF e.g., C. L. Giles
4	$I(F\|F^-)F$
5	$(F\|F^-)ssF$ e.g., Th.P. van der Weide

The feature vector designed for each name sequence is based on the following features. Let us assume L is a line that contains M names, n_1, n_2 through n_M. For name n_i ($1 \leq i \leq M$) that has N_k words, we define the following five features.

$Form_{i,j}$ the form of the j^{th} word of n_i, $Form_{i,j} \in \{F, F^-, I, s, o\}$. "$o$" - others.

$Pos_{i,j}$ the position of the j^{th} word of n_i in the line.

$FN_{i,j}$ is equal to 1 if the j^{th} word of n_i is only in the first name list, 0 otherwise.

$LN_{i,j}$ is equal to 1 if the j^{th} word of n_i is only in the last name list, 0 otherwise.

$NonDic_{i,j}$ is equal to 1 if the j^{th} word of n_i is a non-dictionary word, 0 otherwise.

The feature $Form_{i,j}$ has non-numerical values such as "F", "I" or "s". We enumerate each of these name patterns and assign these values as the weights of the corresponding features.

We generated all the potential name sequences expanded from the 99 space-separated name sequences as the name sequence dataset. We achieve a classification accuracy of 90.9% for ten-fold cross validation. Since we pick the potential sequence with the highest score for each unknown name sequence, the accuracy is the ratio of the correct predictions to the total number of name sequences, which is 99 in this case.

Using SVM supervised learning to classify name sequences helps find the implicit regularities that could have been missed by the manual inspection. A regularity discovered from the training data is: hyphenated names such as Jon-hey are not likely to be the last name.

Classification Score	Class label	Potential name sequences
1.6398636	1	Alan Fekete ⋄ David Gupta ⋄ Victor Luchangco ⋄ Nancy Lynch ⋄ Alex Shvartsman
0.8996393	-1	Alan Fekete ⋄ David Gupta ⋄ Victor Luchangco Nancy ⋄ Lynch Alex Shvartsman
0.0061073704	-1	Alan Fekete ⋄ David Gupta Victor ⋄ Luchangco Nancy ⋄ Lynch Alex Shvartsman

Figure 5. Example of potential name sequences

4 Experimental results

Performance is evaluated by precision, recall, F measure, and accuracy as described below.

Overall evaluation: The overall word classification accuracy for the header is the percentage of the header words that are tagged with the words' true labels.

Class-specific evaluation: We define *A* as the number of true positive samples predicted as positive, *B* as the number of true positive samples predicted as negative, *C* as the number of true negative samples predicted as positive and *D* as the number of true negative samples predicted as negative. The sample may refer to the line in the line classification task and refer to the word when evaluating the final metadata extraction performance.

$$Precision = \frac{A}{A+C} \quad Recall = \frac{A}{A+B}$$

$$Accuracy = \frac{A+D}{A+B+C+D}$$

$$Fmeasure = \frac{2Precision*Recall}{Precision+Recall}$$

We apply the metadata extraction method discussed earlier, with the parameters chosen from ten-fold cross-validation on 500 training headers and 435 test headers. Our method achieves an overall accuracy of 92.9%, better than 90.1% reported by Seymore et al. Table 11 compares our method with the HMM method of multi-state L+D model from Seymore et al on the classification performance for each class, except two functional classes "introduction" and "end of page". However, we are unable to obtain the class-specific accuracy method used by Seymore et al at the time we submit this paper. Therefore, we also list class-specific precision and recall for more effective evaluation.

We present below the Example 2 document header with its true labels (Figure 6) and predicted labels (Figure 7) by our metadata extraction algorithm. We also present the labels (Figure 8) our algorithm predicted for the Example 1 header shown in Figure 1 of section 2. The bold fonts indicate the predicted labels different from the true labels. Both examples show the good performance of our algorithm on labeling the single-class lines, and recognizing the individual authors from the multi-author lines. Line 6 in Figure 7 and line 22 in Figure 8 also show the good performance of our two-class chunk identification algorithm. The only difference between our algorithm's predictions and the original labels is line 7. Although we count this as a false prediction (in our evaluation), the original label for this line "note" can be argued itself. The line contains two email addresses. Therefore it could be labeled as email just as well. This kind of uncertainty of labels is rare, though. Figure 8 shows the direct impact the line classification has on the chunk identification performance. Wrongly classifying the five-class line 22 in Figure 8 as the four-class line, causes the further incorrect chunk identification. Wrongly classifying the five-class line 25 as a single class line "note", also disables the further chunk identification algorithm. A reason that line 25 is wrongly classified as single-class line, is because our contextual line classification algorithm in Section 3.3.2 over weighs the contextual information of the "note" text from line 22 to line 26.

Table 11. Comparison on the performance(%) of metadata extraction using HMM and SVM evaluated based on words. A - Accuracy; P - Precision and R - Recall

Class	HMM(A)	SVM(A)	SVM(P)	SVM(R)
Title	98.3	98.9	94.1	99.1
Author	93.2	99.3	96.1	98.4
Affiliation	89.4	98.1	92.2	95.4
Address	84.1	99.1	94.9	94.5
Note	84.6	95.5	88.9	75.5
Email	86.9	99.6	90.8	92.7
Date	93.0	99.7	84.0	97.5
Abstract	98.4	97.5	91.1	96.6
Phone	94.9	99.9	93.8	91.0
Keyword	98.5	99.2	96.9	81.5
Web	41.7	99.9	79.5	96.9
Degree	81.2	99.5	80.5	62.2
Pubnum	64.2	99.9	92.2	86.3

5 Discussion and future work

This paper describes a classification-based method using Support Vector Machines (SVM) for metadata extraction. These initial results achieve nominally better results than Hidden Markov Model based methods. This occurs

1:<title> THE CORAL USER MANUAL +L+
2:A Tutorial Introduction to CORAL +L+ </title>
3:<author> Raghu Ramakrishnan Praveen Seshadri Divesh Srivastava +L+ </author>
4:<author> S. Sudarshan +L+ </author>
5:<affiliation> Computer Sciences Department, +L+
6:University of Wisconsin-Madison,</affiliation><address> WI 53706, U.S.A. +L+ </address>
7:<note>The authors' e-mail addresses are fraghu,divesh, praveeng@cs.wisc.edu; sudarsha@research.att.com.+L+</note>

Figure 6. Example 2 document header with the true labels.

1: chunk(1) - <title> - THE CORAL USER MANUAL
2: chunk(1) - <title> - A Tutorial Introduction to CORAL
3: chunk(1) - <author> - Raghu Ramakrishnan
chunk(2) - <author> - Praveen Seshadri
chunk(3) - <author> - Divesh Srivastava
4: chunk(1) - <author> - S. Sudarshan
5: chunk(1) - <affiliation> - Computer Sciences Department,
6: chunk(1) - <affiliation> - University of Wisconsin-Madison
chunk(2) - <address> - WI 53706, U.S.A.
*7: chunk(1) - <***email***> - The authors' e-mail addresses are fraghu,divesh,praveeng@cs.wisc.edu; sudarsha@research.att.com.*

Figure 7. Example 2 document header labeled by SVM metadata extraction algorithm.

because we use apriori information of the structural pattern of the data, feature extraction based on domain specific databases, an appropriate normalization technique, and an iterative correction procedure. In addition, the method we propose for extracting individual names from a list of author names has good performance. We believe that our results indicate a promising classification-based method for information extraction.

There are some aspects of our method that could still be improved. The line classification performance limits the further multi-class line chunk identification performance as shown in Figure 8. We will add the functionality to correct the errors caused by the line classification algorithm. Some chunks such as an integrated name may be broken into two lines occasionally. In this case, the multi-class chunk algorithm may make the incorrect decision. We will combine some of the consecutive lines of the same class to minimize the corresponding errors. Currently we assume each line has only one chunk for each class. This is not appropriate even though it is rare for a class to have multiple chunks of

1:chunk(1) - <title> - Stochastic Interaction and Linear Logic
2:chunk(1) - <author> - Patrick D. Lincoln
chunk(2) - <author> - John C. Mitchell
chunk(3) - <author> - Andre Scedrov
3:chunk(1) - <abstract> - Abstract
4:chunk(1) - <abstract> - We present stochastic interactive semantics for propositional linear
...
22:chunk(1) - <**note**> *- jcm@cs.stanford.edu*
chunk(2) - <web> - http://theory.stanford.edu/people/jcm/home.html)
chunk(3) - <affiliation> - Department of Computer Science, Stanford University
chunk(4) - <**address**> *- Stanford, CA 94305.* **Supported in part**
23:chunk(1) - <note> - by an NSF PYI Award , matching funds from Digital Equipment Corporation, the Pow-ell Foundation, and Xerox Corporation; and the Wallace F. and Lucille M. Dav is Faculty
24:chunk(1) - <note> - Scholarship.
25:chunk(1) - <**note**> *- andre@cis.upenn.edu*
http://www.cis.upenn.edu/∼andre Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104-6395. **Partially supported by**
26:chunk(1) - <note> - NSF Grants CCR-91-02 753 and CCR-94-00907 and by ONR Grant N0001 4-92-J-1916. Scedrov is an American Mathematical Society Centennial Research Fellow.

Figure 8. Example 1 document header labeled by SVM metadata extraction algorithm.

the same class in one line. It is worthwhile to explore more general multi-class chunk identification techniques.

In addition to extracting the taggable metadata from the header part of the research papers, we will apply text summarization techniques, such as Zha's [33], to extract the implicit metadata subject and description. This will have the potential for generating a hierarchical metadata representation of the document. We also will intend to develop a robust and accurate wrapper for bibliographies and to define and extract metatags for metadata as well as for equations and figures.

6 Acknowledgments

We acknowledge Cheng Li and Guangyu Chen for useful comments on the first draft of the paper. We acknowledge the valuable comments from reviewers. We would like to acknowledge the support from NSF NSDL 0121679, partial support from the Special Funds for Major State Basic Research Projects of China (project G1999032800), and partial support from Lockheed-Martin.

References

[1] The world factbook. *http://www.cia.gov/cia/publications/factbook/*, 2002.

[2] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proc. of SIGIR-98*, pages 96–103, 1998.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.

[4] T. Brody. Celestial - Open Archives Gateway. *http://celestial.eprints.org/*, 2002.

[5] H. L. Chieu and H. T. Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Proc. 18th National Conference on Artificial Intelligence (AAAI 2002)*, pages 786–791, 2002.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[7] H. de Sompel and C. Lagoze. The Open Archives Initiative Protocol for Metadata Harvesting, January 2001.

[8] I. Dhillon, S. Manella, and R. Kumar. A divisive information theoretic feature clustering for text classification. *Machine Learning Research (JMLR)*, 2002.

[9] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. 7th International Conference on Information and Knowledge Management*, pages 148–155, November 1998.

[10] H.Pasula, B.Marthi, B.Milch, S.Russell, and I.Shpitser. Identity uncertainty and citation matching. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2002.

[11] T. Joachims. Making large-scale Support Vector Machine learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[12] T. Joachims. A statistical learning model of text classification with Support Vector Machines. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proc. SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 128–136, 2001.

[13] A. Kent. OAI Harvester Crawling Status. *http://www.mds.rmit.edu.au/∼ajk/oai/interop/summary.htm*, 2001.

[14] D. Knox. CITIDEL: Making resources available. In *Proc. 7th Annual Conference on Innovation and Technology in Computer Science Education*, pages 225–225, 2002.

[15] T. Kudoh and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proc. of CoNLL-2000 and LLL-2000*, 2000.

[16] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, 2001.

[17] S. Lawrence, C. L. Giles, and K. Bollacker. Digital Libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.

[18] X. Liu. Federating heterogeneous Digital Libraries by metadata harvesting. *Ph.D. Dissertation, Old Dominion University*, December 2002.

[19] C. C. Marshall. Making metadata: A study of metadata creation for a mixed physical-digital collection. In *Proc. 3rd ACM International Conference on Digital Libraries*, pages 162–171, June 1998.

[20] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598, 2000.

[21] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal Volume 3*, pages 127–163, 2000.

[22] P. Mcnamee and J. Mayfield. Entity extraction without language-specific resources. In D. Roth and A. van den Bosch, editors, *Proc. of CoNLL-2002*, pages 183–186, 2002.

[23] A. Paepcke, C.-C. K. Chang, H. Garcia-Molina, and T. Winograd. Interoperability for Digital Libraries worldwide. *Communications of the ACM*, 41(4):33–43, 1998.

[24] Y. Petinot, P. B. Teregowda, H. Han, C. L. Giles, S. Lawrence, A. Rangaswamy, and N. Pal. eBizSearch: An OAI-Compliant Digital Library for eBusiness. In *Proc. the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2003. In this proceeding.

[25] L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In D. Yarovsky and K. Church, editors, *Proc. 3rd Workshop on Very Large Corpora*, pages 82–94, 1995.

[26] P. Schone and D. Jurafsky. Knowlege-free induction of inflectional morphologies. In *Proc. of the North American chapter of the Association for Computational Linguistics (NAACL-2001)*, 2001.

[27] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *Proc. of AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[28] N. Slonim and N. Tishby. The power of word clusters for text classification. In *Proc. 23rd European Colloquium on Information Retrieval Research (ECIR)*, 2001.

[29] K. Takeuchi and N. Collier. Use of Support Vector Machines in extended named entity. In D. Roth and A. van den Bosch, editors, *Proc. 6th Conference on Natural Language Learning (CoNLL-2002)*, 2002.

[30] V. Vapnik. *Statistical Learning Theory*. Springer Verlag, New York, 1998.

[31] S. Weibel. The Dublin Core: A simple content description format for electronic resources. *NFAIS Newsletter*, 40(7):117–119, 1999.

[32] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proc. ICML-97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

[33] H. Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–120, August 11-15 2002.

Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL'03)
0-7695-1939-3/03 $17.00 © 2003 **IEEE**