ornl

OAK RIDGE
NATIONAL
LABORATORY

LOCKHEED MARTIN

ORNL/TM-13354

Multi-Sensor Text Classification Experiments—A Comparison

V. R. Dasigi
R. C. Mann
V. Protopopescu

MANAGED AND OPERATED BY
LOCKHEED MARTIN ENERGY RESEARCH CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL-27 (3-96)

ORNL/TM-13354

Computer Science and Mathematics Division

MULTI-SENSOR TEXT CLASSIFICATION EXPERIMENTS - A COMPARISON

V. R. Dasigi[+]
R. C. Mann
V. Protopopescu

[+]Department of Computer Science and Information Technology, Sacred Heart University, Fairfield, CT. 06432-1000.

DATE PUBLISHED -January 1997

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
Lockheed Martin Energy Research Corp.
for the
U.S. Department of Energy
under Contract Number DE-AC05-96OR22464

CONTENTS

ABSTRACT

In this paper, we report recent results on automatic classification of free text documents into a given number of categories. Our method uses multiple sensors to derive informative clues about patterns of interest in the input text, and fuses this information using a neural network. Encouraging preliminary results were obtained by applying this approach to a set of free text documents from the Associated Press (AP) news wire. New free text documents have been made available by the Reuters news agency. The advantages of this collection compared to the AP data are that the Reuters stories were already manually classified, and included sufficiently high numbers of stories per category. The results indicate the usefulness of our new method: after the network is fully trained, if data belonging to only one category are used for testing, correctness is about 90%, representing nearly 15% over the best results for the AP data. Based on the performance of our method with the AP and the Reuters collections we now have conclusive evidence that the approach is viable and practical. More work remains to be done for handling data belonging to the multiple categories.

1 Introduction

The goal of this project is to automatically classify free text documents into a given number of categories. The approach itself was motivated in part by the Gene Recognition and Analysis Internet Link (GRAIL) system developed at the Oak Ridge National Laboratory, because of the similarity in problem characteristics. GRAIL is a pattern recognition system for identifying rarely occurring protein-encoding segments of DNA sequence in higher eukaryotes, which may span tens to hundreds of "sparse" kilobases [Uberbacher and Mural, 91] [Xu, et al., 94]. A multi-layer feed-forward neural network receives inputs from several sensors that measure different characteristics of the signals or data sets to be analyzed. The net acts as a classifier and assigns the input pattern to a given number of classes. The net is trained using a set of known data patterns that are representative of the application domain.

Similarly, in the problem of classifying large amounts of text, not any single clue gives enough confidence for proper classification, and when using multiple clues, it is not known how these should be integrated into a decision. There has been some work in information retrieval (IR) on how to best combine information from different algorithms. Multiple algorithms can be combined in at least two possible ways: (i) combining multiple (aspects of) *representation of the same input* into a pattern-matching system (often called *information/sensor fusion*; in the IR literature, this method is often called *query combination*, and (ii) combining *results* from multiple pattern matchers (often called decision fusion; in the IR literature the term *data combination/fusion* is often used) [Belkin, et al., 94] [Dumais, 96] [Towell, et al., 95]. For this study, we have used neural nets as parameterized mappings that allow for fusion of higher level clues extracted from free text. (This is like query combination, except that we perform classification, rather than retrieval.)

In our past work [Dasigi and Mann, 95], we tested our method using stories from the Associated Press (AP) news wire [Harman, 93]. We used multiple sensors, the primary sensor based on Latent Semantic Indexing (LSI) [Deerwester, et al., 90], to derive informative clues about patterns of interest in the input text, and fused this information using a neural network. A term-document matrix represents the information contained in a collection of documents as a matrix of term frequency vs. document. In LSI, this large and sparse term-document matrix is reduced into three smaller matrices (one of which is simply a diagonal matrix) by singular value decomposition (SVD). The diagonal matrix yields the singular values from which the dominant ones are selected. Thus, instead of representing documents by thousands of possible terms, LSI allows for a document to be represented by a substantially smaller number of "factors" that are supposed to capture the "significant" term-document associations. This is done by linear transformations of the much longer term vector, using the constituent matrices that result from the SVD of a *reference matrix* [Dasigi and Mann, 96].

A reference matrix is the term-document matrix of a *reference library / collection* of documents. A reference library is the collection of documents that "adequately" represents all concepts of interest. The SVD computations are performed on such a reference library only once, and the transformations mentioned above essentially project *any* new document into the "concept space" represented by the reference library.

Section 2 describes the general approach. We re-checked and expanded our previous experiments perfoemd with the AP data; these results are presented in Section 3. Although the results were encouraging, a careful analysis indicates, however, that additional work was required to ensure the proper training of the fusion network. In order to provide further training, we turned to a new collection of data, made available by the Reuters news agency. The advantage of this collection, as compared to the AP data, comes from the fact that the Reuters stories were already manually classified, and included sufficiently high numbers of stories per category. Another major difference between the new and old data is that the Reuters stories have anywhere between zero to five categories assigned to each of them, while the AP documents were assigned to one and only one category each. Details of the Reuters data are presented in detail in Section 4. The programs developed to support experimentation with the new data are described in Section 5.

Since there were too many categories for the Reuters data, we combined several of them into single higher level categories, as suggested by the data developers. Experiments were conducted with the data (described in Section 6), and the results were consistent with the results for the AP data and expectations based on our previous analyses. There have been a few interesting surprises, which are discussed in Section 7.

2 Background: A Multi-Sensor Neural Net Approach

We started from the assumption that a GRAIL-like system [Uberbacher and Mural, 91] [Xu, et al., 94] would be very effective in classifying and filtering of English text documents. Namely, it is hypothesized that the eventual system, being neural network-based, would be capable of integrating in a systematic way existing and new approaches as required by the application. The GRAIL-type system can integrate different kinds of sensors, e.g., statistical and syntactic sensors as well as simple keyword sensors, and other standard techniques already in use.

Specifically we focused on two main goals. The first goal was to create input to a neural network that is LSI-based, so that the size of the neural net could be kept reasonably small, and the net therefore could be trained without much difficulty. The second goal was to assess the impact of additional sensors that can be added easily to the neural net input, to improve the overall performance.

Actually, the developers of LSI indicate that a query may be viewed as a pseudo-document and may be represented by a vector of a chosen number of factors, so it may be placed in the same vector space as regular document vectors [Deerwester, et al., 90]. This is done as follows. First a reference term-document sparse matrix X is derived from the library of documents that are of interest. This matrix is split into three matrices by SVD [Forsythe, et al., 77], so that

$X = T.S.D'$

Here, X is a tXd matrix, where t is the numbers of distinct terms (word roots) and d is the number of documents in the reference collection. The order of T is tXk, that of S, which is a *diagonal* matrix, is kXk, and that of D is dXk, where k is the chosen number of factors. (LSI research indicates that a choice of k around a hundred is very effective.) Now, the pseudo-document vector D_Q corresponding to a 1Xt query vector Q may be derived simply as:

$D_Q = Q.T.S^{-1}$

In this work, we use this same idea to squash any 1Xt document vector into a 1Xk vector that serves as input to the neural network. *The only care that must be exercised is to make sure that the reference term-document matrix that is used as the starting point is one that "adequately" represents all concepts of interest.* Note that this requirement is no more stringent than would be required in the standard LSI approach. Some more details can be found in [Dasigi and Mann, 95].

3 Experiments and Results with AP Data

As mentioned before, LSI is the primary sensor used in our work, but its utility is limited by the *reference library* of documents. Thus, to enlarge the domain of applicability, there is an important need to complement or at least supplement this sensor. Information from the different sensors is integrated into a decision by a neural network. The LSI-based sensor is used to achieve the necessary dimensionality reduction and thereby render the approach feasible. The additional sensors should be sensitive to *new* words and patterns in the input or otherwise relate to the output categories.

The purpose of the second logical sensor currently used in this work is to allow for simple keyword profiles to be considered in the classification. Each category profile is simply a set of keywords characterizing that particular category. There is one input to the neural network from this logical sensor, corresponding to each category. Each such input simply represents what fraction of the terms in the given document match the category profile. Inclusion of more sophisticated algorithms is the subject of ongoing research, but as it stands, the information supplied by this second sensor is, to a degree, different from and independent of that from the first LSI-based sensor.

We focused on a number of AP news wire stories from the standard TIPSTER collection [Harman, 93]. The collection contains AP news wire stories for two full years, tens to hundreds of stories per day. The purpose of the multi-sensor neural net is to classify the news stories into one of ten ad hoc categories: accidents and natural disasters, crime, terrorism, politics and government, business and finance, science and technology, culture, weather, obituary, and a catch-all general / miscellaneous category. For the documents used for training and testing purposes, the categories of the news story documents were manually determined, because category information is not encoded in the TIPSTER collection. This turned out to be a bottleneck.

Our experiments involved a comparison of performance among five approaches, one involving the classic LSI and four based on neural network training. The first approach uses the original LSI algorithm, modified to perform classification by first identifying the document from the reference library that best matches the input document, and then looking up the category of the reference document. Of the neural network methods, two involve just a single sensor and two involve two sensors each. The first single-sensor method (version 1a) uses an LSI-based dimensionality reduction, which also constitutes one of the two sensors in both the two-sensor methods. The second single-sensor method (1b) employs the simple category profiles, as already explained. The two-sensor methods also use a second sensor based on category profiles. The two methods differ in that in one of them the category profiles are directly used (2a), while in the other, we use the results of SVD from the category profiles (2b).

The goal in comparing the last two approaches was to see whether the results would be substantially different. The initial expectation was that there should not be any substantial difference, but we wondered whether LSI would be able to capture any hidden associations between the category profiles and the documents. This was expected because the case with category profiles is somewhat similar to that of terms in documents. In general, both the term-document matrix and the profile-document matrix have multiple non-zero entries in any given row or column.

Out of the nearly five hundred documents used for training, about three-quarters constituted the "reference library" for *all* LSI/SVD operations. The number of SVD "factors" used in this work was 112. We used simple feedforward nets with back propagation, with the delta rule for learning and the *tanh* transfer function. The classification systems were tested in different configurations: two single-sensor versions and two two-sensor versions. One single-sensor version (1a) employed the LSI-based term frequency sensor comprising 112 input units and 9 hidden units. The other single-sensor version (1b) used the simple category profile sensor, with 10 input units (one corresponding to each category profile) and 5 hidden units. Both two-sensor versions used the LSI-based term frequency sensor as the first sensor (corresponding to 112 input units), and a second sensor based on the category profiles (corresponding to 10 more input units). The first two-sensor version (2a) employed a sensor identical to that of version 1b for its second sensor, and 10 hidden units. The other two-sensor version (2b) used an SVD-based version of the sensor in version 1b as the second sensor, and also used 10 hidden units. All configurations used 10 output units, one for each category.

3.1 Analysis of the Results with AP Data

When implemented alone, the LSI method yielded somewhat surprising results. Indeed, although the performance of LSI in classifying the *known reference library* documents was a perfect 100%, the percentage of correct results dropped to 54% when *new documents* (that is, those outside the reference library) were used. For this method, there was just a single experiment, because there was no training involved and, consequently, there was only one non-trivial test set, that of the documents outside the reference library.

For the four neural network experiments, the percentages of correct results as cited represent the peak performance that did not get any better with more iterations. Since there were only a limited number of inputs, several different experiments were constructed by cross validation. We cross validated the available data by generating a dozen pairs of files, each pair containing a training file (90% of data) and a test file (10% of data). The same pairs of data sets were used to test all four neural net configurations, and are referred to by the same data set numbers across the different experiments.

Tables 1 and 2 summarize the main results of the neural network experiments, which are discussed below.

Single-sensor neural nets: Twelve sets of data were used in each experiment. With each training set, to achieve best performance, the neural net was trained for a number of iterations varying between 16,000 and 128,000 for version 1a, and between 64,000 and 256,000 for version 1b. On the test set, the correctness percentage ranged from a minimum of 58% to a maximum of 72% for version 1a and from 6% to 78% for version 1b. When the training set itself was used as a data set, the performance was between 76.05% to 80.7% correct for version 1a and between 7.44% to 64.4% for version 1b (contrasted to 100% in the

Single-Sensor Neural Network Classifiers						
Data	Term LSI Sensor			Category Profile Sensor		
Set No.	No. of Iterations	Correct on Test Data	Correct on Training Data	No. of Iterations	Correct on Test Data	Correct on Training Data
1	48K	58%	80.93%	224K	70%	64.19%
2	48K	68%	78.14%	256K	66%	64.42%
3	64K	72%	77.21%	128K	12%	8.60%
4	16K	62%	76.74%	64K	30%	24.19%
5	96K	70%	77.44%	64K	34%	23.72%
6	48K	62%	78.14%	64K	22%	26.28%
7	128K	62%	80.23%	160K	36%	28.84%
8	64K	58%	77.91%	64K	14%	10.00%
9	32K	66%	77.21%	64K	20%	25.35%
10	32K	60%	76.98%	128K	6%	7.44%
11	48K	62%	78.37%	64K	22%	28.60%
12	48K	68%	76.51%	160K	78%	62.09%

Table 1: Results of classification using neural nets with an LSI-based term frequency sensor input and a simple category profile sensor input

LSI-only method). That very large standard deviation is associated with version 1b is not surprising if we take into account the overall performance of this sensor which is rather poor (around 30% on the average, with more iterations in general).

Two-sensor neural nets: Again, with each training set, to get the peak performance, the neural net was trained for between 32,000 and 128,000 iterations for version 2a and between 48,000 and 144,000 iterations for version 2b. When the category profiles were directly used for the second sensor (version 2a), the correctness percentage on the test sets ranged from a minimum of 58% to a maximum of 76%. When the training set itself was used as a data set, the performance was between 80.47% to 85.11% correct.

When the results of SVD from the category profiles were used for the second sensor (version 2b), the correctness percentage ranged between 56% and 72% on the test sets and between 82.33% and 85.81% for the training sets. Comparing these results, we notice that the first fusion method (2a) yielded a slight improvement in performance over version 1a (7-8% on the average), while the second fusion method (2b) yielded practically no improvement, although it typically required more iterations than the first fusion method (2a).

Two-Sensor Neural Network Classifiers						
Data	Fusion Version 2a			Fusion Version 2b		
Set No.	No. of Iterations	Correct on Test Data	Correct on Training Data	No. of Iterations	Correct on Test Data	Correct on Training Data
1	64K	58%	85.12%	144K	56%	84.42%
2	32K	70%	83.26%	80K	70%	83.26%
3	80K	76%	84.65%	80K	72%	84.19%
4	64K	66%	84.19%	48K	62%	83.95%
5	80K	70%	84.19%	64K	62%	84.42%
6	64K	68%	82.33%	64K	64%	82.33%
7	32K	68%	85.12%	80K	64%	83.49%
8	32K	68%	83.72%	80K	62%	85.81%
9	128K	72%	85.35%	80K	66%	82.33%
10	80K	64%	86.05%	128K	60%	83.95%
11	32K	68%	82.09%	64K	68%	83.02%
12	64K	74%	84.19%	48K	66%	83.26%

Table 2: Results of classification using two-sensor neural nets, the first with a simple category profile second sensor and the second with an LSI-based category profile sensor

The modest improvement in the first case is due to the fact that the second sensor is not very powerful to begin with. We expect that if the two sensors have comparable individual performances and provide relatively independent information, their fusion should - in general - yield more substantial improvements. However, we also expect a saturation or even decrease in performance to set in, if both sensors have extreme (too high or too low) individual performance.

The lack of improvement in the second case may be related to the number of iterations, the size of the test sets, etc. However, we believe that the main cause for this behavior is the fact that by applying a SVD to the results of the second sensor, one makes it "too parallel" to the first one, thereby stripping an already poor classifier of its discriminating power. A substantially different reason could also contribute to this behavior: the SVD algorithms used in this case might not have been best suited for the case where one dimension of the matrix is only 10.[1]

[1]The size of the term-document matrix for the reference library was 10025X380, but the size of the category profile-document matrix was only 10X380.

4 The Reuters Data

The Reuters-22173 text categorization collection is copyrighted by Reuters, Ltd., and is distributed freely *for research purposes only.* Copyright for additional annotations and a number of auxiliary files resides with David D. Lewis and the Information Retrieval Laboratory at University of Massachusetts.[2] The size of the corpus is about 25 MB, representing 22,173 Reuters news wire stories. The stories are pre-classified and pre-analyzed. Thus, in addition to the 25 MB of raw text, many additional files are provided, that contain information such as, what categories, if any, have been assigned to each document, what words are present in each document (story) with what frequency, what multi-word noun-phrases occur in each document with what frequency, etc. Many details about the collection can be found in [Lewis, 92], and also in the `README` file associated with the collection.

The collection contains 22,791 different words and 29,496 different phrases indexed to the documents. These numbers are rather high, and part of the reason is "words" corresponding to the different numeric values that occur in the documents, special symbols, etc. have all been treated as genuine words. It is up to the system that processes this information to use them appropriately. The documents come with preassigned classification, with about 135 topic categories and another 539 other kinds of categories. The 135 categories may further be grouped into 8 higher level categories, which we use in this work. The categories and their statistical distribution are shown in Table 3. Table 4 shows the distribution of documents with different numbers of category assignments.

The files most important for the purposes of this work are described here, along with their formats. The `corpus` directory contains files with names of the form `initial.frmt0000012345`, where 12345 is replaced by the number of the first document in the file. There are 45 such files, all of which except the very last one containing 500 documents each, with the last one containing just 173 documents. Each document has a header followed by the actual text, with the header including, together with some irrelevant information, the date, time and category codes.

The file `cd1/churcht1w2/n.kixf` contains single words and word identifiers (ids) and other related information. This file would not be needed unless the actual word strings themselves are of interest, because most information is encoded in terms of the unique numeric word ids. Each line corresponds to one word, and has six fields. Only the second and third fields are of interest for our purposes, since they represent the unique numeric word id and the word string itself, respectively. The file `cd1/churcht1w2/df2nsb/n.ixf` simply

[2]The authors thank the providers of the Reuters-22173 collection for making it available. It can be obtained at URL *ftp://ciir-ftp.cs.umass.edu/pub/reuters1.*

No.	Category Name	Category Code	Documents in this Category	Documents only in this Category
1	Money/Foreign Exchange	MONEY-FX	827	323
2	Shipping	SHIP	308	163
3	Interest Rates	INTEREST	530	293
4	Economic Indicators	ECON	1,312	1,136
5	Currency	CURR	306	26
6	Corporate	CORP	6,561	6,466
7	Commodity	COMM	1,941	1,769
8	Energy	ENERGY	796	638

Table 3: Document distribution among the various categories

Number of documents with 0 category assignments	10,509
Number of documents with 1 category assignment	10,814
Number of documents with 2 category assignments	788
Number of documents with 3 category assignments	57
Number of documents with 4 category assignments	5

Table 4: Document distribution according to the number of assigned categories

lists each of the 22,791 unique word ids, where each line has two fields: the first field may be ignored and the second field is the unique word id. These ids appear in increasing order.

The phrase files have so far not been used in this project, but might be valuable. The file `cd1/churchp1p1/n.kixf` contains multi-word phrases, most of which are noun phrases, along with their ids. The format is similar to that of `cd1/churcht1w2/n.kixf`. The file `cd1/churchp1p1/df2tk2/n.ixf` lists each of the 19, 496 phrases, along with their ids, with the ids appearing in increasing order as is the case with word ids.

The file `cd2/churcht1w2-docs/df2nsb-all/wdf.trn` lists word-document associations. Each line contains a word id, followed by a document id, followed by an integer representing the number of occurrences of the specified word in the specified document. The word ids are listed in increasing order and for any given word id, the document ids appear in increasing order. The file `cd2/churchp1p1-docs/df2tk2-all/wdf.trn` lists phrase-document associations, with a couple of fields extraneous for our purpose. The second field in each line is the phrase id, the fourth field the document id and the last (fifth) field is a floating point number representing the frequency. The lines ordered first by the phrase id and next by the document id.

The file `cd1/rcat/n.ixf` specifies the numeric ids for each category name. The first field may be ignored. The second field is the unique category id. The third field has two parts separated by a colon (:), the left part is the broad category id (TOPICS, PLACES, PEOPLE, ORGANIZATIONS and EXCHANGES), and the right part the actual category name (e.g., DLR, GNP, LEAD, OAT, etc.). Our work focuses only on the TOPIC categories - 135 of them broadly regrouped into 8 higher level categories. These TOPIC category ids are listed one per line, following an extraneous field value, in the file `cd1/rcat/topics/n.ixf`. Finally, the file necessary for supervised learning and verification is the file that contains document category associations: `cd2/rcat-docs/all-all/rc-sorted.trn`. Each line of this file contains three fields: the first field is ignored; the second field is the unique id of a category (could be any category, not necessarily a TOPIC category), and the last field is a document id. The file is sorted by the category ids first and within each category id, by the document ids.

5 The Preprocessing Programs

Programs have been written to work with the new format of Reuters data. For use by the SVD routines, described in [Dasigi and Mann, 95], a `Rmatrix` input file is needed to be created in the *compressed column storage format.* The `ccs` program creates the file, but it also does other things, much like the `convert` program described in [Dasigi and Mann, 95]. It generates two executable files, `Rneural`[3] and `RLSIcat`, the executable logic of which is expected to be specified in the files `Rneural.c` and `RLSIcat.c`. The temporary files `%@+3.c` and `%@+4.c` are created respectively from these files by adding some data definitions corresponding to the ids of the words that occur in the reference library documents. (The original `.c` files are left unchanged.) These temporary files are compiled, the executable generated, and the temporary files deleted.

The purpose of `Rneural` is to generate a training or test input for the neural network, into a file called `RNEU.nna`, where the suffix nna is required by the NeuralWorks package. Information on the documents on which this neural net data is based is already in the file `cd2/churcht1w2-docs/df2nsb-all/wdf.trn`. The program asks for the number of the first (sn) and last (en) document to be classified (actually it always starts with the first

[3]Most programs written to deal with the Reuters data start with the prefix letter `R`, which refers to Reuters. This convention distinguishes them from the programs that had been written to work with AP data. Most of them have same or similar names, except for the single-letter prefix.

document and processes en-sn+1 documents), and also for the number of desired factors. This latter input value corresponds to the number of singular values (and vectors) used in computations, and is required for predictable control. This is necessitated by the fact that the SVD programs supplied in SVDPACKC do not give the user precise control on the number of singular values computed. The specified number of the more significant values are used from the file `RT.SINV`, which should have been created by the modified SVDPACKC package as described in [Dasigi and Mann, 95]. This program also assumes the availability of a `categories` file[4], which is used to create the training output (or reference output for testing). The program also repeatedly generates and later deletes a temporary file named `%@+3`.

`RLSIcat` can be used to classify a sequence of input document files using an LSI-based method, for comparison against the performance of the multi-sensor neural network. The LSI approach, as described in [Deerwester, et al., 90], was originally intended for document *retrieval*, rather than *classification*. To perform classification, we modified it as follows. First, the document from the reference library that matches an input document best is identified. Next, the category of the reference document is looked up in the `categories` file and is reported as the category for the input document in question.

As with `Rneural`, information on the documents which need to be classified using LSI is already in the file `cd2/churcht1w2-docs/df2nsb-all/wdf.trn`. `RLSIcat` also asks for the numbers of the first and last document to be classified, and also for the desired number of "factors". It uses the two files `RT.S` and `Rlav2` in computing the best matching reference document. Further, the program asks for the name of the "document magnitudes" file, that must already have been created by the `Rmags` program. This program repeatedly generates and later deletes a temporary file named `%@+4`.

The program `Rgensensor2.c` creates a program that generates a second input vector (viewed as a second sensor) for the neural network. It assumes the existence of the file `Rsensor2.c`, which represents the executable portion of the second logical sensor for the neural net (to be described shortly). It asks for the number of categories, and also assumes the existence of as many *category profile* files as the number of categories. These files are supposed to contain a number of keywords which constitute a profile for the respective

[4]This file is to specify the categories for each of the document files, in the format of the document number (ignored), followed by a tab, followed by an 8-bit binary-coded decimal value corresponding to the category numbers assigned to the document number, followed by a tab-separated list of the actual category numbers themselves. Since there are 8 categories, an 8-bit encoding would do. For example, the encoding 17 specifies that the document belongs to categories 5 and 1. This list must be immediately preceded by a line starting with a hyphen.

category. These files are assumed to have the names cat#.basic, where # stands for the category number. It also creates a temporary file (%@+5.c by adding data corresponding to the category profiles to the logic of the sensor2.c file, compiles it to create the executable sensor2 and then deletes the temporary file. It uses the cd1/churchtlw2/n.kixf file to relate the unique word ids to their word strings.

The program sensor2 asks for the starting and ending file numbers, which define the range of the input document file numbers. he file cd2/churchtlw2-docs/df2nsb-all/wdf.trn is used to identify all the words that occur in a document. It creates an output file named RNEU.S2, in which each line is simply a vector (corresponding to an input document), whose size equals the number of categories. The components of the vector represent what fraction of the terms in the document correspond to the words in the profile. The match for the "general" category is negatively related to the best match for any of the other categories.

The program Rneural3.c creates a combined neural net input vector from RNEU.nna and the RNEU.S2 files. Recall, from the preceding description of the Rneural program, that the RNEU.nna file contains LSI-based input vector and training output (or test reference) values. It actually asks for the two input file names and the desired name for the combined file. The latter file is created as the output.

Other utility programs have been written, as well. The program Rresults interprets the neural net output for multiple category documents. It has four command line inputs for specifying the number of categories, the names for the input and output files and the threshold to be used in interpreting the results. Any value at or above the threshold is treated as a value of 1 and smaller values taken to be 0. The program results, as described in [Dasigi and Mann, 95], interprets neural net output with single category documents. The program sepsingle writes all those records of documents that belong to exactly one category from the input file to the output file. The file names are specified as command line input parameters. A frequency distribution of the categories (in terms of single category documents) is written out to a file named sepsingle.out.

6 Experiments and Results with Reuters Data

A "reference library" of the first 1,500 documents was chosen for the LSI implementation. The distribution of these documents is shown in Table 5. The SVD computations were performed using the same techniques discussed in [Dasigi and Mann, 95]. Since documents could be assigned to anywhere between 0 and 4 categories, all experiments were first conducted by focusing on single category documents, that is those documents with one and only one category assigned to them, and then on the entire document set.

Number of documents with no category assignment	611
Number of documents assigned to category 1	46
Number of documents assigned to category 2	16
Number of documents assigned to category 3	65
Number of documents assigned to category 4	75
Number of documents assigned to category 5	20
Number of documents assigned to category 6	550
Number of documents assigned to category 7	146
Number of documents assigned to category 8	27

Table 5: Distribution of the 1,500 reference library documents

First, classification was performed using LSI alone, with no learning. This was done by identifying the document that best matches the document given to be classified, and then declaring its category or categories as the "answer". If *any* of these categories matches *any* of the categories assigned to the document in question or if *both* documents have no category assigned to them, the answer is declared correct. When documents assigned to exactly one category are used, both these sets are of size 1, so in a certain sense the problem is easier when multiple categories are involved, although one would normally expect the multiple category cases to be harder. This "anomaly" is reflected in Table 6.

Parameter	Single Category Case	Multiple Category Case
Total documents	10,814	22,173
Documents in Reference Library	837	1,500
Percent correct on reference library	100.00%	100.00%
Percent correct outside reference library	65.86%	67.87%
Percent correct overall	68.50%	70.04%

Table 6: LSI classification results for documents with exactly a single category assignment and those with multiple category assignments

As with AP data, neural net experiments were conducted with both a single sensor and two sensors on the Reuters data, as well. Experiments were conducted with the entire data set, as well as the single category data set. Because of the availability of a large number of documents, no cross-validation was deemed necessary. Further, with thousands of documents in the training and the test sets, the experimentation took time comparable to cross-validation with multiple sets in the case of the AP data!

Once again, the same reference library was used as above, namely, the first 1,500 documents from the data. Also, the largest 112 factors were used in the computations. In both categories of experiments (single category and multiple category data sets), the first 4,000 documents were used for training and the remaining documents were used for testing. This means that in experiments with the single category documents, 6,814 documents comprised the test cases, and with the entire data set, 18,173 documents were used for testing. Both single-sensor and two-sensor experiments were conducted, with the second sensor the same as before, namely, one based on simple category profiles.

When data with multiple category assignments are considered, the question of interpreting the output of the neural net arises. This is a more complex situation than the previous one, because in this latter case, the category corresponding to the largest output is declared to be the desired classification. In the multiple category case, a threshold has to be chosen. This threshold was chosen to maximize the performance on the training set output, and then used to interpret the test data output. If the category (-ies) assigned after the interpretation match(es) *any* of the correct categories, the classification is considered to be correct. All the results related to these experiments are tabulated in Table 7.

Parameter	Single Category Case		Multiple Category Case	
	Single-Sensor	Two-Sensor	Single-Sensor	Two-Sensor
Training Documents	4,000	4,000	4,000	4,000
Test Documents	6,814	6,814	18,173	18,173
Input Units	112	120	112	120
Hidden Units	9	10	9	9
Output Units	8	8	8	8
Iterations	48K	112K	144K	80K
Threshold	N.A.	N.A.	-0.338	-0.213
Correct with Training Data	91.13%	92.70%	83.48%	84.13%
Correct with Test Data	89.90%	90.84%	72.51%	72.93%

Table 7: Summary of neural net experiments on single category and multiple category data, with single- and two-sensor networks

7 Analysis of Results and Further Work

In comparing the results of Table 6, that is classification results on the Reuters data using LSI alone, with those for the AP data, one observes that those in Table 6 are clearly better, by almost 15% over the best results for test data. This better performance is perhaps attributable to a larger, and consequently more likely to be representative, reference library.

A quick comparison between Tables 6 and 7 reveals that for both the single category data and multiple category data, the results with learning are consistently better than those without learning (that is when LSI was used by itself). In the case of the single category data, the results improved substantially, that is from about 68.5% up to about 90%. There is practically no difference in the performance of the net with training and test data. Somewhat disappointing is the result that the addition of the second sensor results in only a marginal 1-2% improvement in the performance.

For multiple category data, the details are a bit different. Learning does improve the results, but only marginally, that is from about 70% to about 72-73% on the test data. On the training data, there is more improvement, but this fact is not really of any practical utility. The disparity between the performance on the training and test data is striking. It has been observed that very marginal improvements (about 0.1% for every 16K or so) in performance were possible with more still iterations through the backpropagation algorithm, which is not of much practical use. As in the single category case, the second sensor seems to have resulted in only a very marginal improvement (0.5-1%).

These experiments enable us to make some conclusive remarks for the single category data. As expected, more training data yields significant improvements. Yet, if one of the sensors has extreme performance (in this case, about 90% correctness), the second sensor, especially a relatively poor one, such as the one used here, does not seem to add much value. This result is, again, consistent with our initial expectations.

We cannot compare the results of the multiple category data with prior experience since the AP data did not have multiple category assignments. Initially, it was expected that multiple category data would be hard to deal with. That, indeed, turned out to be the case; yet, the results seem to point in the right direction.

Plans for future work include:

- *More systematic experimentation to analyze the impact of uneven distribution of categories among the documents*: An uneven distribution of categories in the training data tends to bias the network. The categories occurring more often influence the weights in the network more, resulting in a bias that favors them (minimizing false negatives for those categories). One possible consequence of this bias is that documents belonging to other less frequent categories might get incorrectly classified into one of the frequent categories (false positives).

- *Interpretation of the results of multiple category data in terms of precision and recall for each class*: Precision and recall are measures of false positives and false negatives, respectively. While correctness percentage is an adequate measure for data that are known to belong to single categories, for multiple category data precision and recall could provide valuable additional information.

- *Evaluation of the impact of the training set size*: Comparison of our results with the Reuters data against the results with the AP data reveals that a larger training set can positively influence the performance of the neural net. It would be interesting to find out a threshold, if any, beyond which the improvement is negligible.

- *Evaluation of the impact of the reference library size*: The more concepts the reference library covers, the more effective the LSI-based sensor would be. Thus, the size of the reference library is another important factor, like the training set size, that can improve the overall performance of the system. One interesting issue is that of identifying an optimum combination of these parameters. A larger reference library would have a term-document matrix that is larger in both dimensions, and therefore, increases the time for the SVD computations.

- *Evaluation of the impact of the number of factors used in LSI computations*: Intuitively, the LSI factors are ordered in terms of their decreasing ability to capture siginificant associations. Since the last few / several / many factors are the smaller and less significant values, it is expected that the usefulness of the factors would be limited after a certain number. An optimum combination of the number of factors and the reference library size would be useful to identify. It is worth noting that the SVD computations take a longer time as the number of factors goes up.

- *Automatic synthesis of category profiles*: Zhou and Dapkus came up with a method to identify the significant terms for given predefined topics [Zhou and Dapkus, 95]. If the categories are thought of as predefined topics, their method could be turned into a method to synthesize the profiles automatically.

- *Other sensors*: Other sensors could be based on n-grams, phrases, clustering, natural language analyses, etc. Reuters data already has support for another kind of "sensor", namely, noun phrases in the documents. It should be kept in mind that a sensor that performs comparable to LSI would be needed in order to expect significant improvement.

- *Data combination (Decision fusion) from different networks using single sensor each*: Other IR researchers have worked on data combination methods that fuse the results from different approaches [Towell, et al., 95]. It would be worth comparing the effectiveness of query combination (sensor fusion) and data combination (decision fusion).
- *Other possible interpretations of the raw results for the multiple category data*: In our experiments, the raw results from the neural networks have been interpreted by choosing a threshold that maximized the performance on training data (and using the same threshold to interpret the raw results with the test data, as well). Other interpretations and other ways of handling the information are possible and warrant further study.

Acknowledgments

The first author wishes to thank the Oak Ridge Institute for Science and Education for the summer fellowship that made this work possible. Thanks are also due to Prof. Mike Berry of the Computer Science Department at the University of Tennessee, Knoxville and his students for distributing the SVDPACKC software, and to Dr. Chuck Glover of the Intelligent Systems Section (ISS) at ORNL for help with the NeuralWorks package and general help with neural nets. Dr. Donald Lewis's help with questions the Reuters data and related questions is very much appreciated.

References

[Belkin, et al., 94] Belkin, N. J., P. Kantor, C. Cool, and R. Quatrain, Combining Evidence for Information Retrieval. In Harman, D. (Ed.), *The Second TREC Conference*, NIST Special Publication 500-215, pp. 35-44, 1994.

[Dasigi and Mann, 95] Dasigi, V. and R. C. Mann, Toward a Multi-Sensor-Based Approach to Automatic Text Classification, *ORNL/TM-13094*, Oak Ridge National Laboratory, October, 1995.

[Dasigi and Mann, 96] Dasigi, V. and R. C. Mann, Toward a Multi-Sensor Neural Net Approach to Automatic Text Classification, to appear in *Proc. of IFIP-96, International*

Federation for Information Processing World Conference on Advanced Information Technology Tools, Canberra, Australia, September, 1996.

[Deerwester, et al., 90] Deerwester, S., S. Dumais, G. Furnas, T. Landauer, and R. Harshman, Indexing by Latent Semantic Analysis *Journal of the American Society for Information Science*, 41(6), pp. 391-407, 1990.

[Dumais, 96] Dumais, S., Combining Evidence for Effective Information Retrieval, *Working Notes of AAAI Spring Symposium of Machine Learning in Information Access*, pp. 26-30, March, 1996.

[Forsythe, et al., 77] Forsythe, G., M. Malcolm, and C. Moler, *Computer Methods for Mathematical Computations* (Chapter 9: Least Squares and the Simgular Value Decomposition), Prentice Hall, Englewood Cliffs, NJ, 1977.

[Harman, 93] Harman, D., Overview of the First TREC Conference, *Proc. of SIGIR-93*, pp. 36-47, 1993.

[Lewis, 92] Lewis, D., Representation and Learning in Information Retrieval, Ph.D. dissertation, University of Massachusetts, Amherst, February, 1992.

[Towell, et al., 95] Towell, G., E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird, Learning Collection Fusion Strategies for Information Retrieval, *Proc. Twelfth Annual Machine Learning Conference*, Lake Tahoe, July, 1995.

[Uberbacher and Mural, 91] Uberbacher, E. and R. Mural, Locating Protein-Coding Regions in Human DNA Sequences by a Multiple Sensor-Neural Network Approach, *Proc. Natl. Acad. Sci. USA*, 88, pp. 11261-11265, 1991.

[Xu, et al., 94] Xu, Y., R. Mural, M. Shah, and E. Uberbacher, Recognizing Exons in Genomic Sequence using GRAIL II, *Genetic Engineering, Principles and Methods*, Plenum Press, 15, June, 1994.

[Zhou and Dapkus, 95] Zhou, J. and P. Dapkus, Automatic Suggestion of Significant Terms for a Predefined Topic, *Third ACL Workshop on Very Large Corpora*, MIT, Cambridge, June 1995.

INTERNAL DISTRIBUTION

1.	J. Barhen	14.	N. Peterfreund
2.	J. R. Einstein	15.	S. Petrov
3.	C. W. Glover	16-20.	V. Protopopescu
4.	H. E. Knee	21.	N. S. Rao
5.	R. W. Lee	22.	M. Shah
6.	S. McKenney	23.	R. F. Sincovec
7-11.	R. C. Mann	24.	E. C. Uberbacher
12.	E. M. Oblow	25.	Y. Xu
13.	C. E. Oliver	26.	CSMD Reports Office
11.	V. Olman	27.	Laboratory Records-RC
12.	M. Parang	28.	Document Reference Section
13.	L. E. Parker	29.	Central Research Library
		30.	ORNL Patent Office

EXTERNAL DISTRIBUTION

31-32. Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831.
34-38. Professor V. R. Dasigi, Department of Computer Science and Information Technology, Sacred Heart University, Fairfield, CT 06432-1000.