Data Extraction as Text Categorization: An Experiment with the MUC-3 Corpus

David D. Lewis
Center for Information and Language Studies
University of Chicago
Chicago, IL 60637
lewis@orange.uchicago.edu

INTRODUCTION

The data extraction systems studied in the MUC-3 evaluation perform a variety of subtasks in filling out templates. Some of these tasks are quite complex, and seem to require a system to represent the structure of a text in some detail to perform the task successfully. Capturing reference relations between slot fillers, distinguishing between historic and recent events, and many other subtasks appear to have this character.

Other of the MUC-3 subtasks, however, appear amenable to simpler techniques. In particular, whenever a slot filling task involves making choices among a relatively small set of prespecified alternatives, the potential exists for modeling this task as *text categorization* [10]. Text categorization systems treat a document as a single unit, and make a decision to assign each document to zero, one or more of a fixed set of categories.

Text categorization techniques have played a role in previous text extraction systems, both for screening out documents which it is not desirable to process, and for directing documents to category-specific extraction routines [2, 9, 15]. The role of categorization in these systems was relatively limited, however. Analyses of the behavior of these systems have given relatively little attention to how categorization operated or to what factors influenced categorization performance. Nor had performing data extraction solely by categorization techniques seen much attention before MUC-3.

The study reported here is an initial exploration of the issues involved in using categorization as part of the extraction process. As such it has two main goals. One goal is provide a measure of the difficulty of the extraction task being performed by the MUC-3 systems. By evaluating the degree to which various text to template mappings can be captured statistically as word to filler mappings, we gain a sense of where the linguistic and domain knowledge embedded in the NLP systems is having the most effect.

A second goal is to identify aspects of the extraction task that categorization might plausibly aid, or even take primary responsibility for. With respect to this second goal, this paper provides an additional data point to be considered along with the results of several official MUC-3 sites which made greater or lesser use of categorization techniques.

In order to most clearly discover what characteristics of the extraction task impact categorization, we ran an experiment using an "off-the-shelf" categorization system, with access to no additional language processing capability or knowledge base. The technique investigated was a well-understood, domain independent statistical procedure. It should be pointed out that the method used is only one of a variety of methods, some quite knowledge intensive, that have been used in previous work on text categorization [12].

The results presented here are not official MUC-3 results, and should be considered tentative. This paper is only a small step toward an understanding of the role of categorization methods in data extraction.

VIEWING MUC-3 AS TEXT CATEGORIZATION

For a data extraction task to be viewed as text categorization, it must involve choosing among a finite, preferably small, set of categories that a document might be assigned to. We therefore need to consider which MUC-3 subtasks might fit this description.

The MUC-3 task can be broken into two main parts: deciding how many templates should be generated for a document, and filling the slots within the generated templates. Deciding the number of templates could conceivably be modeled as categorization, given some upper bound on the number of allowed templates. (A more profitable view, if one wanted to use machine learning techniques for this subtask, might be to use multiple regression or some similar technique.) We chose not to address this task directly, though indirectly it is dealt with by the rules we use to generate templates from a binary categorization.

Many of the slot filling tasks also could not be modeled as classifying a document with respect to a fixed set of categories. For instance, all slot filling tasks that involve extracting strings from the text allow a range of answers that cannot be defined in advance. One might force string filling into a categorization mode by viewing the task as one of deciding, for each substring in a document, whether that substring falls into the class of fillers of a particular slot. In this paper, however, we chose to consider only the classification of entire document texts into categories.

The slots that appeared most amenable to categorization techniques were the *set fill* slots. The following slots were considered to be set fill slots in the MUC-3 documentation:

```
0.  MESSAGE ID
3.  TYPE OF INCIDENT
4.  CATEGORY OF INCIDENT
7.  PERPETRATOR: CONFIDENCE
10. PHYSICAL TARGET: TYPE(S)
13. HUMAN TARGET: TYPE(S)
14. TARGET: FOREIGN NATION(S)
15. INSTRUMENT: TYPE(S)
16. LOCATION OF INCIDENT
17. EFFECT ON PHYSICAL TARGET(S)
18. EFFECT ON HUMAN TARGET(S)
```

Of these, the MESSAGE-ID slot is assigned straightforwardly without reference to the article content. The foreign nation (14) and location (16) slots, are set fill slots only under the assumption that the set of relevant locations is known and fixed. This is appropriate for the MUC-3 evaluation, but might be problematical in an operational setting. In any case, filling of these slots is to such a large extent a function of recognizing particular names in the text that it appeared categorization would have little to say about these slots.

This left eight slots (3, 4, 7, 10, 13, 15, 17, and 18) which could plausibly be treated by categorization. Each of these slots has a finite set of possible fillers. The decision to fill a particular slot for a particular document can therefore be modeled as deciding whether the document fits into one or more of the 88 categories defined by combinations of these slots and their fillers. Even here, however, several factors, such as the necessity of distributing slot fillers among multiple templates, cause filling these slots to depart from a traditional categorization model. These issues are discussed in the next section.

FILLING SET FILL SLOTS WITH A STATISTICAL CATEGORIZER

To investigate the use of categorization techniques for filling set fill slots, we used the Maxcat system [11]. This is a suite of C programs and Unix shell scripts supporting machine learning using large,

sparse feature sets, such as are found in text retrieval and text categorization. Datasets are stored using sparsely nondefault matrices. A variety of programs for structural and arithmetic manipulation of these matrices are included, and several machine learning algorithms have been implemented using them.

In the following we discuss the processing of MUC-3 training and test documents, and how categorization and template generation were performed.

Modeling and Preparing the MUC-3 Data

The training data available for the MUC-3 task was a set of 1400 document texts (1300 original training documents and the interim TST1 testset of 100 documents), along with one or more completed templates for each document. The 100 documents and corresponding templates from the second MUC-3 testset (TST2) were used for testing.

Maxcat requires all data it operates on to be represented in the form of feature vectors. We therefore viewed the complete MUC-3 data set as consisting of two sets of 1500 feature vectors, two vectors for each document. One set of features, the predictor features, specified the presence or absence of each English word from the corpus in each document. We defined a word as a sequence of alphanumeric characters separated by whitespace or punctuation. By this definition, there were a total of 16,755 distinct word types in the 1500 documents, and thus 16,755 predictor features. Most predictor features took on a value of 0 (not present) for most documents, and the Maxcat system took advantage of this by storing feature vectors in inverted file form. Information on the number of occurrences and positions of words was not used in this experiment.

The second set of feature vectors specified the presence or absence of each of the 88 slot/fixed set filler combinations in the templates for each document. As with words, information about multiple occurrences of a filler, in one or multiple templates for a document, was omitted since the particular categorization method used did not take make use of this information. The program which converted the training templates into feature vectors uncovered approximately 100 errors in these templates. Most of these were obviously typographical errors in filler names, and we corrected these by hand before training.

It should be noted that while the separation between training and test documents was strict from the standpoint of the machine learning and evaluation procedures, it was not maintained implementationally. The entire set of training and test documents were processed together for the purpose of defining the complete set of word features that would be used. Strict isolation, to official MUC-3 standards, could have been handled with the Maxcat software, given some additional coding and a loss in efficiency.

Categorization Method

Categorization was performed by a Bayesian classification strategy previously investigated for text categorization [7, 13] as well as for text retrieval [18]. The strategy treats the categorization problem as one of estimating $P(F_j = 1|D_m)$, i.e. the conditional probability the slot/filler pair F_j should be assigned given a particular document description, D_m. The formula used, suggested by Fuhr [4] for text retrieval, estimates $P(F_j = 1|D_m)$ by:

$$P(F_j = 1) * \prod \left(\frac{P(W_i = 1|F_j = 1)}{P(W_i = 1)} * P(W_i = 1|D_m) + \frac{P(W_i = 0|F_j = 1)}{P(W_i = 0)} * P(W_i = 0|D_m) \right)$$

$P(F_j = 1)$ is the prior probability that slot/filler combination F_j occurs in some template for the document. $P(W_i = 1)$ is the prior probability that word W_i is an appropriate indexing term for the document, while $P(W_i = 0)$, which is $1 - P(W_i = 1)$, is the prior probability that it is not an appropriate indexing term for the document. $P(W_i = 1|F_j = 1)$ is the conditional probability W_i should be an indexing term for the document given that F_j appears in a template for the document, and similarly for $P(W_i = 0|F_j = 1)$.

$P(W_i = 1|D_m)$ is the probability that word W_i should be considered an indexing term, given that the document has the description D_m. For purposes of this experiment, we let $P(W_i = 1|D_m)$

be 1.0 if W_i had one or more occurrences in the document, and 0.0 if it did not. However, better performance is likely to be obtainable with nonbinary estimates of $P(W_i = 1|D_m)$ [1, 16, 17]. All other probabilities were estimated from the 1400 training documents. The expected likelihood estimate was used to partially compensate for small sample errors [6].

The main categorization program took as its input a set of matrix files storing the above probabilities, along with a matrix specifying the word occurrences in the 100 test documents. It produced a matrix of estimates of $P(F_j = 1|D_m)$ for each of the 100 test documents and each of the 88 slot filler pairs F_j. The problem then remained of making a yes/no decision for each F_j, D_m pair on the basis of these 8800 probability estimates.

The obvious strategy of setting a single threshold on the estimated probabilities and assigning to each document all slot/filler pairs which meet this threshold will not, unfortunately, work. The problem is that the probability estimates produced by the above formula vary widely between fillers. This is due to the different predictor features used for each filler, and the fact that the assumptions about independence of features made by the above model are not met in reality.

The result is that a different threshold is necessary for each category. Tying these thresholds to a single parameter that can be varied for a system is a difficult problem and has not received much attention in previous research on text categorization. For the current experiment, we used the following ad hoc strategy. The categorization software multiplies the estimate of $P(F_j = 1)$ from the training corpus by 100 (the number of test documents) and by a constant k. The resulting product provides an estimate, $N(F_j = 1)$, of the number of test documents that F_j should be assigned to. For each F_j, the software assigned the slot/filler pair F_j to the $N(F_j = 1)$ test documents with the highest estimated values of $P(F_j = 1|D_m)$. The parameter k controls the tradeoff between recall and precision, with higher values of k leading to more slot/filler pair assignments, higher recall and, usually, lower precision.

As implemented, this strategy requires processing test documents in batch mode. An incremental alternative would be to apply the classification function to the original training set and check the relative position of the test document's score within the set of scores it produces on the training set. Even better would be to explicitly model the distribution of scores on the training set and derive a decision theoretic assignment rule with a cost parameter.

Feature Selection

Recall that, by our very simple definition of a word, the MUC-3 corpus contained 16,755 distinct word types. In theory, all of these words could be used as features for each categorization, with the actual quality of each feature being compensated for in the conditional probability estimates. In practice, the "curse of dimensionality" [3] limits the number of features that can be used effectively by any statistical estimation method. The more parameters which need to be estimated by a learning procedure, the more samples which the procedure must see in order to avoid fitting noise in the data. The relationship between the appropriate number of features and the appropriate number of samples varies with feature quality, the nature and amount of noise in the data, appropriateness of the statistical model being fit, and other concerns. For text categorization problems, Fuhr has suggested that 50-100 samples per parameter are necessary [5].

With only 1400 training documents, it is clearly inappropriate to use 16,755 features. Maron [13] and Hamill [7] suggested, but did not try, information theoretic measures for selecting a subset of words to be used in predicting each category. Such measures are widely used for feature selection in pattern recognition and machine learning. We chose to use $I(W_i; F_j)$, the *system mutual information* [8] between assignment or nonassignment of F_j and assignment or nonassignment of W_i, as a feature quality measure:

$$I(W_i; F_j) = \sum_{b=0,1} \sum_{c=0,1} P(W_i = b, F_j = c) log_2 \frac{P(W_i = b, F_j = c)}{P(W_i = b) * P(F_j = c)}$$

Notice that system mutual information is symmetric with respect to F_j and W_i. However, the above formula is sometimes rewritten to emphasize the information gained about the feature we desire to predict (F_j in our case) based on the feature we can observe (W_i):

$$\left(\sum_{c=0,1} -P(F_j = c) log_2 P(F_j = c)\right) - \left(\sum_{b=0,1} P(W_i = b) \sum_{c=0,1} -P(F_j = c|W_i = b) log_2 P(F_j = c|W_i = b)\right)$$

In the latter form the measure has been referred to as *information gain* [14].

For each category (slot/filler pair), the words with the top d values on mutual information were selected as features. Differing values of d were investigated as described below.

It should be noted that information-based feature selection itself requires estimating probabilities, and so also is affected by the curse of dimensionality. While we have 1400 samples to estimate each of the mutual information values, if the filler and/or the word was very infrequent in the training set we will have few or no positive instances available for the estimation, leading to inaccurate values of mutual information. There's little to be done about low frequency fillers, but we did investigate omitting low frequency words from consideration as features, as described below.

Binary Categorization and Template Generation

For each test document, the categorization procedure outputs a length 88 binary vector indicating whether each slot/filler pair should be assigned to that document. It is then necessary to map from such a vector into one or more templates. This was done by a program that examined the 88 categorization decisions and did the following:

1. If all 88 categorization decisions were "NO", it considered the document irrelevant and generated an single irrelevant template (all slots filled with "*").
2. If there were one or more positive categorization decisions, but no positive categorization decisions for the TYPE OF INCIDENT slot, a single ATTACK template was generated, and all other slots in the template were filled according to the positive categorization decisions.
3. If there were one or more positive categorization decisions for TYPE OF INCIDENT, we generated one template for each TYPE OF INCIDENT filler for which there was a positive decision. Each of these templates got a copy of every non-TYPE OF INCIDENT filler for which there was a positive categorization decision.

In addition to the above procedure, we encoded the rules for legal fillers from the MUC-3 documentation into the program. If a filler suggested by a positive categorization was in fact not legal for the type of incident of a template, the program omitted it from the template. No other domain knowledge or reasoning was used in generating the templates from the binary categorization.

Since the categorizer output only binary categorization decisions, cross references were never produced in templates, nor was more than one copy of the same filler produced in any slot. This constraint imposed an upper bound of 50% on the system's recall for most slots, since most slots allowed only partial credit in the absence of cross references.

Evaluation

Two methods of evaluation were used. One was a simple Unix shell script, ebc0, which computes microaveraged recall, precision, fallout, and overlap measures for a binary assignment of categories to documents [10]. It operates on two matrices specifying, respectively, the binary categorization decisions made by the categorizer, and the correct binary categorization decisions extracted from the key templates. Since this program was fast and did not require human intervention it was used to evaluate all the experiments. However, it overestimates performance on the MUC-3 task, since it does not take into account multiple copies of fillers in the key templates.

The other form of evaluation was, of course, to generate the MUC-3 templates as described above and evaluate them using the MUC-3 scoring program. This was done for only a few of the runs, due to the time required for interactive scoring. In interactive scoring, I allowed partial matches

Recall/ Precision k	Number of Features (d)				
	5	10	20	40	80
0.1	.03/1.0*	.02/.82	.02/.82	.03/1.0	.02/.91
0.2	.07/.82*	.07/.79	.07/.82	.07/.79	.07/.82
0.5	.19/.69	.20/.71*	.19/.70	.19/.69	.18/.64
1.0	.34/.57	.37/.60*	.35/.57	.35/.58	.35/.57
1.5	.44/.46	.48/.51*	.46/.49	.45/.48	.45/.48
2.0	.52/.41	.55/.43*+	.55/.43	.54/.42	.53/.41
3.0	.64/.34	.66/.34	.65/.34	.67/.35*	.65/.34
5.0	.76/.26	.76/.26	.76/.26	.77/.26*	.76/.26
10.0	.87/.17	.86/.17	.87/.17	.88/.17*	.86/.17
15.0	.90/.14	.90/.14	.90/.14	.91/.14	.90/.14
20.0	.92/.12	.92/.12	.92/.12	.92/.12	.92/.12
25.0	.92/.11	.93/.11	.93/.11	.94/.11*	.94/.11

Table 1: Microaveraged recall and precision under binary categorization model. MUC-3 scoring program SFO rows for *'ed entries appear in Table 2. Complete MUC-3 scoring output for the +'ed entry appears in Table 3.

only when the categorizer produced exactly the same filler as one of the fillers in the key template, but omitted a cross-reference string. Note that the maximum possible recall on most slots was 50% (partial credit on every filler) since they required a cross-reference for full credit.

Matches were allowed against optional and alternative fills whenever possible. If the same fill occurred multiple times in the key template, the categorizer's fill was allowed to match each of these occurrences. Partial credit was not allowed in any other circumstance, and interactive full credit was not used.

RESULTS

Figure 1 shows microaveraged recall and precision results from ebc0 for 60 parameter combinations. Five variations on the number of features were explored, as well as 12 variations on the parameter k that trades off recall and precision. An additional 60 runs were made requiring that a word occur in at least 4 training documents to be a feature, vs. requiring it occur in at least 2 training documents. The resulting differences were insignificant, however, so only the cases where at least 2 document occurrences were required are reported in Table 1.

For certain of the parameter settings with superior tradeoffs between recall and precision, we generated a template file from the binary categorization and evaluated the resulting categorizations using the MUC-3 scoring program. (The latest version of scoring program available June 5, 1991 was used.) The scoring program's summary for all set fill slots (the SFO row) for these parameter settings are presented in Table 2, and their recall precision points in Table 1 are marked with a *. As we can see, performance on the MUC-3 evaluation measures is lower than is estimated by ebc0. This is due to the limitation to partial credit, presence of multiple templates with the same incident type, and the inclusion of location slots in the MUC-3 set fill summary.

An example of the complete MUC-3 scoring output for parameter settings $d = 10$ and $k = 2.0$ is shown in Table 3.

DISCUSSION

The results of the above experiments have not yet been analyzed in detail. In this section I attempt, however, to discuss some of the more noticeable results. I particularly examine the performance of feature selection.

d, *k*	POS	ACT	COR	PAR	INC	IPA	SPU	MIS	NON	RC	PR	OV	FL
5, 0.1	532	22	9	9	4	9	0	510	479	2	61	0	0
5, 0.2	533	63	19	22	17	22	5	475	474	6	48	8	0
10, 0.5	550	143	47	52	25	52	19	426	477	13	51	13	0
10, 1.0	581	303	82	110	49	108	62	340	488	24	45	20	0
10, 1.5	602	447	108	149	65	147	125	280	488	30	41	28	1
10, 2.0	612	604	128	174	85	171	217	225	467	35	36	36	2
40, 3.0	618	871	164	237	91	235	379	126	441	46	32	44	2
40, 5.0	648	1300	187	302	84	298	727	75	407	52	26	56	4
40, 10.0	668	2156	210	358	71	354	1517	29	340	58	18	70	7
40, 25.0	672	3084	218	385	43	382	2438	26	345	61	13	79	12

Table 2: MUC-3 summary scoring. SFO row only. d is number of features and k is the tradeoff parameter. I omit the ICR column, which is 0 for all runs.

	POS	ACT	COR	PAR	INC	IPA	SPU	MIS	NON	RC	PR	OV	FL
TI	115	153	92	0	0	0	61	23	3	80	60	40	
ID	111	0	0	0	0	0	0	111	4	0	*	*	
IT	115	92	73	0	19	0	0	23	0	63	79	0	1
C	85	100	48	0	16	0	36	21	11	56	48	36	36
IP	94	0	0	0	0	0	0	94	54	0	*	*	
OP	68	0	0	0	0	0	0	68	55	0	*	*	
PC	71	66	0	12	21	12	33	38	34	8	9	50	8
PTI	55	0	0	0	0	0	0	55	76	0	*	*	
PTN	38	0	0	0	0	0	0	38	77	0	*	*	
PTT	56	78	0	27	8	27	43	21	73	24	17	55	3
HTI	147	0	0	0	0	0	0	147	22	0	*	*	
HTN	93	0	0	0	0	0	0	93	22	0	*	*	
HTT	148	137	0	83	13	83	41	52	11	28	30	30	4
TN	16	0	0	0	0	0	0	16	103	0	*	*	
IT	26	24	7	3	2	0	12	14	79	33	35	50	1
IL	115	0	0	0	0	0	0	115	0	0	*	*	
PE	37	47	0	23	1	23	23	13	81	31	24	49	3
HE	58	60	0	26	5	26	29	27	75	22	22	48	3
MO	1186	757	220	174	85	171	278	707	609	26	40	37	
M/M	1448	757	220	174	85	171	278	969	780	21	40	37	
AT	1448	1039	220	174	85	171	560	969	1593	21	30	54	
SFO	612	604	128	174	85	171	217	225	467	35	36	36	2

Table 3: MUC-3 summary scoring for run with $d = 10$, $k = 2.0$. I omit the ICR column, which is 0 for all runs.

Overall Results

The proportional assignment strategy was successful, in that varying k allowed a smooth tradeoff to be made between recall and precision. The break even point between recall and precision was roughly 50/50 when evaluated under a binary categorization model and around 35/35 under the MUC-3 scoring for set fill slots.

Performance varied considerably between different set fill slots as can be seen, for example, in Table 3. Not surprisingly, performance was best for the two slots, TYPE OF INCIDENT and CATEGORY OF INCIDENT, which are tied most closely to the overall content of the article rather than to single references within the article. Performance for the other six set fill slots was roughly equivalent and fairly low, with break even points between 10 and 30.

Also not surprisingly, recall and precision for a filler was strongly related to the frequency of the filler on the training set. Fillers which had few occurrences on the training set were, of course, assigned infrequently under our categorization strategy, but when they were assigned they were usually incorrect.

Feature Selection

Feature selection by the mutual information measure worked surprisingly well, considering the challenge of choosing from such a large set of potential features, most of which had relatively few positive instances. As expected, feature selection was much more accurate for fillers which occurred frequently on the training set than for the more rare fillers. In Table 4 we list, in alphabetical order, the top 10 features selected for the four most frequent fillers, as well as for two fillers that had only a single appearance apiece on the training set. Each word was required to occur in at least 2 training documents to be a feature.

The selected features show a number of interesting properties of the feature selection algorithm. The features selected for BOMBING (an incident type filler) fit one's intuitions of what good features should look like. They are words that are associated with bombing and, in a semantic analysis system, would undoubtedly be mapped to concepts closely related to bombing.

The features selected for MURDER (another incident type filler) are a mixture of words which are in general related to the concept of murder, and words connected to a particular murder that is widely discussed in the MUC corpus. The influence of this incident, the murder of 6 Jesuit priests, is even more evident in the features selected for CIVILIAN (a human target type filler). Since we would not expect these features to be good predictors for future articles, this suggests a need to consider the composition of training sets carefully when machine learning or statistical methods are used in text processing.

Another interesting phenomenon shows up in the features selected for TERRORIST ACT (a category of incident filler). We see some contentful words, but also a number of grammatical function words (WE, ALL, CAN). These clearly result from stylistic characteristics of a large class of articles reporting terrorist acts, rather than to the content of those articles. A "stoplist" is often used to screen out function words in text retrieval systems, but in the MUC-3 domain the use of such stylistic clues may in fact be helpful.

For fillers such as KIDNAPPING THREAT and WATER with only one document occurrence, features were selected that occurred in that single document, and one other. These are poor predictors of the fillers, since their single cooccurrence is often an accident. Most fillers fell between the two extremes presented in the table, and received a mixture of good and poor features.

The number of features used, d, was a smaller influence on performance than expected. As Table 1 shows, performance was relatively constant across a range of feature set sizes from 5 to 80. However, if we look at Table 5, which gives only those recall precision points from Table 1 which are not strictly dominated by some other point, a bit of a trend becomes clear. We can see that the optimal number of features increases as the category assignment parameter increases. This means, not surprisingly, that more features should be used if higher recall is desired.

BOMBING (199 docs)	*CIVILIAN* (375 docs)	*MURDER* (386 docs)
ATTACK	ELLACURIA	ASSASSINATION
BOMB	INJURED	CRIME
CAR	JESUIT	DEATH
DESTROYED	JESUITS	ELLACURIA
DAMAGE	KIDNAPPED	JESUIT
DYNAMITE	MURDERED	JESUITS
EXPLODED	PRIESTS	KILLED
EXPLOSION	SIX	MURDER
POLICE	TWO	MURDERED
INJURED	UNIVERSITY	PRIESTS
TERRORIST ACT (528 docs)	*KIDNAPPING THREAT* (1 docs)	*WATER* (1 docs)
ALL	ARGUETA	ALLEY
BOMB	BENITEZ	BOOTH
BOGOTA	BLANCO	DESCRIPTIONS
CAN	CRISTIANO	EYEWITNESS
DAMAGE	GILBERTO	OSORNO
DYNAMITE	PALO	PIPES
EXPLODED	PERQUIN	SKETCHES
MUST	THREATING	SPORTS
POLICE	TROOP	TNT
WE	VELIZ	125

Table 4: Predictor features selected for six fillers.

Recall/ Precision	Number of Features *(d)*				
k	5	10	20	40	80
0.1	.03/1.0*			.03/1.0	
0.2	.07/.82*		.07/.82		.07/.82
0.5		.20/.71*			
1.0		.37/.60*			
1.5		.48/.51*			
2.0		.55/.43*	.55/.43		
3.0				.67/.35*	
5.0				.77/.26*	
10.0				.88/.17*	
15.0				.91/.14	
20.0	.92/.12	.92/.12	.92/.12	.92/.12	.92/.12
25.0				.94/.11*	.94/.11

Table 5: Nondominated recall/precision pairs from Table 1.

CONCLUSIONS

The above results provide a measure of the extent to which the MUC-3 task can be viewed as one of sorting documents into categories based on statistical word associations. Only a subset of slots, the set fill slots, can easily be treated in this way. Of those slots, high performance was reached only on the two most associated with overall document content. However, a nontrivial level of performance was achieved on all the set fill slots, and provides an interesting point of comparison with knowledge-based techniques. An information theoretic method of choosing words to predict slot/filler pairs was shown to achieve reasonable results, and studying the output of this method raised some interesting questions about the composition of the MUC-3 corpus.

The Maxcat categorization software used was the result of about 6 months of programming effort. However, applying Maxcat to the MUC-3 task required only about a week's effort on the author's part, plus an additional two weeks' effort by a staff programmer to write the software for mapping MUC-3 texts and templates into and out of the format used by Maxcat. This suggests that categorization techniques may be of practical use in situations where an extraction system for set fill slots must be brought into operation in a short period of time. In addition, the relatively high performance on those slots related to overall document content suggests that categorization may have an important role to play with respect to similar slots in any extraction system.

Acknowledgments

Thanks to Peter Shoemaker for writing the software to map the MUC-3 data into and out of Maxcat format. Thanks also to Bruce Croft and Abe Bookstein for useful discussions on statistical categorization techniques and information theory.

References

[1] Croft, W. B. Experiments with representation in a document retrieval system. *Information Technology: Research and Development*, 2:1–21, 1983.

[2] DeJong, G. An overview of the FRUMP system. In Wendy G. Lehnert and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1982.

[3] Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, 1973.

[4] Fuhr, N. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.

[5] Fuhr, N. and Buckley, C. Probabilistic document indexing from relevance feedback data. In *Thirteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–61, 1990.

[6] Gale, W. A. and Church, K. W. Poor estimates of context are worse than none. In *Speech and Natural Language Workshop*, pages 283–287, San Mateo, CA, June 1990. DARPA, Morgan Kaufmann.

[7] Hamill, K. A. and Zamora, A. The use of titles for automatic document classification. *Journal of the American Society for Information Science*, pages 396–402, 1980.

[8] Hamming, R. W. *Coding and Information Theory.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[9] Jacobs, P. S. and Rau, L. F. SCISOR: Extracting information from on-line news. *Communications of the ACM*, 33(11):88–97, November 1990.

[10] Lewis, D. D. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*. Defense Advanced Research Projects Agency, Morgan Kaufmann, February 1991.

[11] Lewis, D. D. *Representation and Learning in Information Retrieval.* PhD thesis, University of Massachusetts at Amherst, 1991. In preparation.

[12] Lewis, D. D.; Croft, W. B.; and Hayes, P. J. Statistical and knowledge-based text categorization. In preparation., 1991.

[13] Maron, M. E. Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8:404–417, 1961.

[14] Quinlan, J. R. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[15] Sahin, K. and Sawyer, K. The Intelligent Banking System: Natural language processing for financial communications. In Herbert Schorr and Alain Rappaport, editors, *Innovative Applications of Artificial Intelligence*, pages 43–50. AAAI Press, Menlo Park, CA, 1989.

[16] Salton, G. and McGill, M. J. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, New York, 1983.

[17] Turtle, H. *Inference Networks for Document Retrieval.* PhD thesis, Computer and Information Science Department, University of Massachusetts at Amherst, October 1990.

[18] van Rijsbergen, C. J. *Information Retrieval.* Butterworths, London, second edition, 1979.