Combining Classifiers in Text Categorization

Leah S. Larkey and W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
{larkey,croft}@cs.umass.edu

Abstract

Three different types of classifiers were investigated in the context of a text categorization problem in the medical domain: the automatic assignment of ICD9 codes to dictated inpatient discharge summaries. K-nearest-neighbor, relevance feedback, and Bayesian independence classifers were applied individually and in combination. A combination of different classifiers produced better results than any single type of classifier. For this specific medical categorization problem, new query formulation and weighting methods used in the k-nearest-neighbor classifier improved performance.

1 Introduction

Past research in information retrieval has shown that one can improve retrieval effectiveness by using multiple representations in indexing and query formulation [27] [19] [3] [11] and by using multiple search strategies [5] [24] [7]. In this work, we investigate whether we can attain similar improvements in the domain of text categorization by combining different representations and classification methods.

Our domain is the automatic assignment of ICD9 codes to dictated inpatient discharge summaries. There is a great deal of interest in automating the assignment of categories such as ICD9-CM and other codes to patient medical records. Many hours of human effort go into this task. Because this coding determines reimbursement, it is important to accomplish this task as easily and as accurately as possible.

The most common approaches use a large corpus of previously coded documents to infer codes for new documents. Many different algorithms have been used for text categorization, including k-nearest-neighbor algorithms [6, 26, 18], Bayesian independence classifiers [13], relevance feedback [21], and rule-induction algorithms from machine learning, like decision trees [2, 16]. These categorization algorithms have been applied to many different subject domains, usually news stories, but also physics abstracts [9], and medical text [29].

We are following several of these approaches to automatic coding, all of them incorporating INQUERY, a probabilistic information retrieval system based on an inference net model [27]. Each possible code is a category, and we want to determine whether documents belong in each category, or more generally, the probability that a document belongs in each category. We use three different classification techniques, a k-nearest-neighbor [6] approach using the belief scores from INQUERY as the distance metric, Bayesian independence classifiers [13], and relevance feedback [21]. At some time in the future, we may also experiment with direct lookup in the ICD-9-CM manuals (Alphabetic Index and TabularList).

Within each classification method, we experiment with ways to optimize performance that are specific to the characteristics of the ICD9 coding problem and the kinds of discharge summaries that make up our collection. Each classification method lends itself to different kinds of variations on representations. In k-nearest-neighbor, test documents are queries, so we experiment with various forms of structuring the query using INQUERY's query operators. Because the discharge summaries contain large amounts of text that are not relevant to the coding task, we have incorporated a method for differentially weighting sections that provide the most diagnostic evidence, described in [12]. For the Bayesian and relevance feedback classifiers, the documents are represented by a small set of features (terms, phrases), and they are selected by slightly different criteria. We do not try to make representations consistent across classifiers. Instead we take advantage of the diversity of the representations when the classifiers are combined.

These classification techniques yield a ranked list of codes (categories) for each document. A purely automatic coder would need cutoff criteria for which codes should actually get assigned. Lewis [14] has argued that in evaluating a classification system, one should use effectiveness measures based on estimates of class membership rather than measures based on rankings, like recall-precision. We do not take this last step of going from a score to a yes/no decision, partly because the correct number of codes for a document can range from 0 to 15, and partly because of the way these classifiers are likely to be used. We envision these classifiers being used in an interactive system which would display the 20 or so top ranking codes and their scores to an expert user. The user could choose among these candidates, possibly with the aid of other software which could display information from the ICD-9-CM manuals. We use several different measures in addition to average-precision, which we believe to measure how well our classifiers would perform in such a semi-automatic coder.

Submitted to SIGIR-96, draft as of March 20, 1996
This material is based on work supported by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623 and on work supported by NRaD Contract Number N66001-94-D-6054.

1.1 K-nearest-neighbor Classifier

The k-nearest-neighbor classification scheme attempts to retrieve those already-coded documents which are most similar to the to-be-coded document, and assign codes based on the codes of the retrieved documents. The already-coded documents make up an INQUERY database, and the to-be-coded coded documents (also referred to as test documents) are queries attempting to retrieve similar documents from the database. A similar approach has been used for other classification tasks and is sometimes referred to as *memory-based reasoning* [18] [26]. Our approach is similar to that of Yang and Chute [29] except that we use INQUERY rather than cosine similarity for the similarity metric. We go beyond their work in representing the document as a structured query, and in combining k-nearest-neighbor with other classifiers. We also incorporate into the weighting scheme some additional independent information that is specific to this data set. We consider not only whether a category (code) is assigned to a retrieved document, but also whether that category is the principal diagnosis code for the retrieved document.

1.2 Bayesian Independence Classifier

The Bayesian independence classifier was first proposed by Maron [17] as a way to estimate a probability that a category or key word should be assigned to a document, given the presence of "clue words" in the document. Various improvements to Maron's approach have been explored by other researchers [8, 15, 13]. We adopt a form of classifier very close to one used by Lewis [13].

Highlights of this probabilistic model are the following: A small set of features is selected separately for each code. Independent binary classifiers are trained for each ICD9 code. Bayes theorem is used to estimate the probability of category membership for each category and each document. Probability estimates are based on the co-occurrence of categories and features (stopped and stemmed terms) in the training corpus, and assume that the features are independent.

Within this framework, we train a separate binary classifier for each ICD9 code, using a manually labelled training corpus of discharge summaries.

1.3 Relevance Feedback

Relevance feedback has typically been used in information retrieval to improve existing queries. Usually, an original query is submitted to an information retrieval system. From the documents retrieved, the user indicates a (usually small) set of relevant documents. The original query and terms from the indicated relevant documents are combined to produce a new query which is better at ranking relevant documents over nonrelevant documents. Term weights in the new query depend upon the occurrence of the terms in relevant and nonrelevant documents [21, 23].

Although the original query typically plays an important role in relevance feedback, it does not have to. For this categorization problem, we start with a null query for each category, and a set of documents of known status, that is, whether they are relevant (have the code) or nonrelevant (do not have the code), and use relevance feedback to generate a query.

Relevance feedback is like the Bayesian independence classifier in its broad outline. A small set of features is selected separately for each code, and a query is trained for each code. Both feature selection and the weights on the features in the query are based on the co-occurrence of features and codes in the training set.

PRINCIPAL DIAGNOSIS: 1. OSTEOARTHRITIS OF THE LEFT HIP
SECONDARY DIAGNOSIS: 2. WOLFF-PARKINSON-WHITE SYNDROME
PROCEDURES: Left total hip replacement (uncemented), 2-2-93.
HISTORY OF PRESENT ILLNESS: The patient is a 54 year old white male with a 9 month history of left hip pain. He has noted a severe limitation of ambulation over this period of time and presently is limited to non reciprocal stairs and short distances. He has trouble getting out of a chair as well as a car. The examination and radiographs ... confirmed bilateral hip osteoarthritis with left greater than right. He is admitted for an elective left total hip replacement. He has donated three units of autologous blood.
PAST MEDICAL HISTORY: Notable for osteoarthritis as noted above and WPW syndrome. PAST SURGICAL HISTORY: Notable for tonsillectomy at age 3 and bilateral hammer toe corrections. MEDICATIONS ON ADMISSION: At the time of admission, the patient was on Ferrous Sulfate 325 mg po t.i.d. ALLERGIES: NKDA.
PHYSICAL EXAMINATION: HEENT examination was within normal limits. The lungs were clear. The cardiac examination revealed no murmurs. The abdomen was benign. The extremity examination revealed a left antalgic gait with no lurch. There was negative bilateral Trendelenburg sign. His range of motion of both hips are as follows: flexion is 90 bilaterally and extension was -10 degrees bilaterally. He had abduction to only 5 degrees bilaterally and adduction of 30 degrees bilaterally. His external rotation was 5 degrees and internal rotation was 0 degrees bilaterally. His knees and ankles had full range of motion. Distal sensory motor examination was intact. Distal pulses were intact.
LABORATORY DATA: The patient's admission hematocrit was 38.1. Electrolytes were within normal limits. Coagulation factors were normal. Sed rate was 11.
HOSPITAL COURSE: The patient underwent a left total hip replacement on 2-2-93. Postoperatively, he was transferred to the floor in stable condition. His hematocrit immediately postoperative was 38 and trended down to a hematocrit of 34. His postoperative course was notable for quick progression in physical therapy and he was discharged on 2-9-93. He was anticoagulated in routine fashion postoperatively and at discharge his PT was 13.8 with iron of 1.6. Vascular ultrasound and x-rays were taken prior to discharge and the results were not available at the time of this dictation. He was to continue on 6 weeks of coumadinization and follow up with Dr. ... at that time.
MEDICATIONS ON DISCHARGE: ... DISPOSITION: To home.

D715.95 Osteoarthrosis, unspecified whether generalized or localized, involving pelvic region and thigh

D426.7 Anomalous atrioventricular excitation

Figure 1: Example discharge summary and its codes

2 Method

2.1 The Corpus

The corpus consists of 11,599 dictated inpatient discharge summaries, divided into a training set of 10,902 documents, a test set of 187 documents, and a tuning set of 510 documents. We are using the discharge summaries rather than the entire patient medical record, because this is the part of the medical record that has been computerized.

The discharge summaries range from around 100 to nearly 3000 words in length with a mean length of 633 words. Each document has between one and 15 ICD-9 codes assigned to it, with a mean of 4.43 codes per document. 90% of the documents have fewer than 9 codes. The first ICD9 code is the principal diagnosis (DX) code. The ordering of the additional codes is not necessarily indicative of importance.

A sample document can be seen in Figure 1. Note that the codes and text following the codes is not indexed in the

Ranked list of retrieved documents

Doc	$belief_i$	Principal Dx code	Other codes for retrieved doc_i			
3580	.4320	715.35	996.4			
5997	.4301	715.95				
7059	.4300	715.35	428.0	041.10	458.9	490
1040	.4298	720.0	424.0	592.0	533.90	
4556	.4295	715.35	276.1	458.9	278.0	
6476	.4294	715.35	276.5	796.3		
...	...	...				

↓

Ranked list of retrieved codes

Code	#	Sco_c	Description of code
715.35	10	7.71	osteoarthrosis, localized not spec...
*715.95	5	3.85	osteoarthrosis, unspecified whether
.			generalized or localized ...
428.0	4	1.71	congestive heart failure
401.9	4	1.70	unspecified essential hypertension
....	...	...	

Figure 2: Ranked list of retrieved docs, and derived ranked list of retrieved codes for test doc

database, and are not included in the test documents.

In style, the discharge summaries are fairly typical of hospital discharge summaries. Most of the documents in the corpus follow a standard medical document chronology, usually consisting of an assessment, history of present illness, past medical history, physical examination, laboratory examination, hospital course, and disposition. Some documents include operations and procedures. A small proportion of the documents are aberrant in format, or were very short addenda to other documents. No effort was made to screen these out of the corpus, or to attach addenda to other documents that they may belong with. The documents were produced by a large number of practitioners and were consequently heterogeneous in linguistic style and in the way the sections were labeled.

Automatic coding of such documents is particularly difficult because there is so much free form text in each document, much of it is not relevant or only indirectly relevant to the coding task, and the portion of text relevant to each code is not explicitly associated with its code in any way.

2.2 K-nearest-neighbor classifier

The training collection of 10,902 discharge summaries was indexed and built into an INQUERY database, using the normal stop list and Porter stemmer. The test documents were turned into queries. In the baseline condition, these queries were the full free-text of the discharge summaries, which were stopped and stemmed as part of the query process.

K-nearest-neighbor classification consists of two steps, exemplified in Figure 2. In the first step, a query (test document) is submitted to the database of manually coded discharge summaries. The INQUERY retrieval engine returns a list of those discharge summaries from the database which are most similar to the test discharge summary to be coded. Each retrieved document has an associated belief score, and the list is ranked by this score. Each code found in a retrieved document is a candidate for assignment to the test document.

The second step in assigning codes to the test document is to assign a score to each code in each retrieved document, based on the belief score for the retrieved document. These scores allow us to rank-order the codes proposed for the test document.

Our preliminary studies showed that the optimal number of documents to retrieve for each test document was 20. In all subsequent work this number remained fixed at 20.

A major problem in this paradigm is how to assign a score to a candidate code for a test document, based on the codes and scores assigned to retrieved documents. We have experimented with several different ways of assigning scores to candidate codes for each test document. The simplest and most obvious method is to use as a code's score the number out of the twenty retrieved documents that have that code assigned to it, but this produces too many ties. Instead, we sum the belief scores of the retrieved documents assigned that code, weighting the scores before summing, i.e.

$$Score_c = \sum_i (belief_i \cdot w_{ic})$$

where i ranges over the retrieved documents, $Score_c$ is the test document's score for code c, $belief_i$ is the belief score for retrieved doc i, and w_{ic} is the weight for code c in document i.

We have tested several different weighting methods for determining w_{ic}, which are discussed in [12]. However, the best method used a very simple weighting scheme (referred to as *Princ* in the results section), which depended only upon whether a code was the principal diagnosis for the retrieved document, as follows:

$$w_{ic} = \begin{cases} w_P & \text{if } c \text{ is the principal DX code for doc } i \\ 1 & \text{if } c \text{ is a nonprincipal DX code for doc } i \\ 0 & \text{if } c \text{ is not assigned to doc } i \end{cases}$$

w_P could range from 1 to 3, and was tuned on a tuning set of 255 documents which was separate from the corpus and the test set.

Besides manipulating document-score weighting, we experimented with query formulation, turning the 187 test documents into structured queries using `#wsum` (weighted sum) and `#sum` operators, as in Figure 3. Two subtasks, described in detail in [12], made up this part of the research: identifying document sections, and tuning the weights on the sections. Sections were identified heuristically. Weights were tuned using the tuning set divided into two sets with 255 documents each. We used a hill-climbing algorithm [6], and accepted each successive change in weights that improved the first tuning set without hurting performance on the second tuning set.

2.3 Bayesian Independence Classifiers

A set of 1068 classifiers were trained, one for each code that occurred 6 or more times in the training data, using the training corpus of 10902 labelled discharge summaries. Some of these codes have a large number of training examples (the most frequent code occurred in 2364 of the 10902 training documents), but most do not. Obviously, the number of examples of a code in the training set will have a large effect on the quality of the classifier that can be trained from the examples.

First, the documents were stopped and stemmed using the standard stop list and (Porter) stemmer in the INQUERY system. The resulting stemmed terms were the potential features for the classifiers. Second, up to 40 features

```
#wsum(1.0
1.5 #sum ( PRINCIPAL DIAGNOSIS: 1.  OSTEOARTHRITIS OF THE
LEFT HIP
SECONDARY DIAGNOSIS: 2. WOLFF-PARKINSON-WHITE SYNDROME
)
1.0 #sum (PROCEDURES: Left total hip replacement (uncemented),
2-2-93. )
1.5 #sum (HISTORY OF PRESENT ILLNESS: The patient is a 54 year
old white male with a 9 month history of left hip pain . He has
noted a severe limitation of ambulation over this period of time and
presently is limited to non reciprocal stairs and short distances. He
has trouble getting out of a chair as well as a car. The examination
and radiographs by ... confirmed bilateral hip osteoarthritis with
left greater than right. He is admitted for an elective left total hip
replacement. He has donated three units of autologous blood. )
1.0 #sum (PAST MEDICAL HISTORY: Notable for osteoarthritis as
noted above and WPW syndrome . PAST SURGICAL HISTORY: No-
table for tonsillectomy at age 3 and bilateral hammer toe correc-
tions. MEDICATIONS ON ADMISSION: At the time of admission, the
patient was on Ferrous Sulfate 325 mg po t.i.d. ALLERGIES: NKDA.
) ...)
```

Figure 3: Example weighted sum query for k-nearest-neighbor classification

(stemmed terms) were chosen for each classifier (code) according to mutual information [28], subject to the following constraints: Terms must have length >1, they cannot begin with a digit, they must contain at least one alphabetic character, they must co-occur at least two times with the code. Forty terms were obtained for most codes. The exceptions were codes with few training examples, where fewer than forty terms met the criteria. Preliminary experiments showed that increasing the number of features above 40 per code did not improve performance.

Classifiers were trained according to the probabilistic model described by Lewis [13], which was derived from a retrieval model proposed by Fuhr [8]. The model supports probabilistic indexing [8], however we implement a simplified version in which only estimates of 0 or 1 are used for the probability that a document has a feature. The model also considers features which are absent in the test document, which many models do not. (See [12] for more detail.)

The classifier yields an estimate of the log probability that a code is assigned to a test document. We do not attempt to determine a threshold and make a binary membership decision. Instead, we produce a ranked list of code candidates for each test document, ordered according to this probability. This output is comparable to that produced by the k-nearest neighbor classifier, facilitating the comparison between them, and their combination.

In order to compare more directly the Bayesian classifier with the k-nearest-neighbor classifier, we recomputed the k-nearest-neighbor results based only on the 1068 codes that occur 6 or more times in the training data rather than the 3261 codes that occur 1 or more times.. In practical terms, we pretend that the list of codes for a (test or training) document includes only those codes which occur 6 or more times in the training corpus. We removed any test documents whose principal diagnosis code was removed by this restriction.

2.4 Relevance Feedback

A set of 1068 queries was trained using relevance feedback, one for each ICD9 code occurring 6 or more times in the training corpus. The relevance feedback algorithm was essentially the same as that used in TREC4 [1] and is more fully described there. Relevance feedback began with null queries. First, 40 terms were chosen by comparing their occurrences in relevant and non-relevant training documents. A weighted sum query was built from these 40 terms with weights from the Rocchio formula applied to INQUERY's weighting scheme. Finally, the weights were adjusted using an iterative technique similar to that of Buckley and Salton and others [4, 20].

The relevance feedback classifier is very much like the Bayesian classifier. In our instantiation of the two approaches there are two major differences, concerning the use of term frequency and the use of terms that don't occur in relevant training documents.

Our Bayesian classifier considers only whether a term occurs or does not occur in a document, not how often the term occurs in the document. This classifier ignores term frequency both in feature selection and in training the classifier. The relevance feedback classifier also does not consider term frequency in feature selection, but it does use term frequency in determining weights for the terms in the trained query.

The Bayesian classifier chooses terms by mutual information, which means it can select terms which are strongly associated with documents in the class, or terms that are strongly associated documents that are not in the class. For example, the term "male" is selected as one of the features for a leiomyoma of the uterus, and the classifier gives this term a high negative weight. If the term "male" occurs in the document, it counts strongly against this diagnosis. The relevance feedback classifier selects only terms that are strongly associated with documents that are in the class.

2.5 Combining Different Classifiers

The k-nearest-neighbor classifier was combined with each of the other classifiers in linear combinations (weighted sums) in several different ways to test 2-way combinations of classifiers. For each code c, the 2-way combination score for a given test doc is:

$$Score_c = k \cdot score_{knn,c} + (1-k) \cdot score_{other,c}$$

where $Score_c$ is the test document's combined score for code c. The component $score_{knn,c}$ is a function of the test document's k-nearest-neighbor score for code c. The component score $score_{other,c}$ is a function of the test document's score for code c on either the Bayesian or relevance feedback classifier. Component scores could be based either on ranks or on scores output by the individual classifiers, and scores could be normalized.

2.5.1 Ranks

For a given test document, rank-based component scores for code c was determined as follows for each component classifier:

$$score_{component,c} = \begin{cases} N - rank_{component,c} & \text{if ranked} \\ 0 & \text{otherwise} \end{cases}$$

Recall that the k-nearest-neighbor method yields a candidate list of codes for each test document. This does not include all possible codes, but only those codes which were in the top 20 retrieved documents. In contrast, the Bayesian and relevance feedback classifiers give a score for each possible code (class) for each test document. Codes that were not k-nearest-neighbor candidates for a document were given a score of zero for $rank_{knn,c}$. Furthermore, in all the combinations below, performance was better if the k-nearest-neighbor candidate lists included only codes which occurred

	Component		Component	
1	KNN	rank	Bayesian	rank
2	KNN	rank	Bayesian	rank of norm score
3	KNN	score/20	Bayesian	normalized score
4	KNN	rank	RF	rank
5	KNN	score/20	RF	score

Table 1: Components of 5 2-way combination classifiers

in two or more retrieved documents. For this reason, scores for codes which occurred in only one retrieved document were also set to zero before combination with Bayesian or relevance feedback candidate lists.

2.5.2 Normalization of component scores

The k-nearest-neighbor and Bayesian scores had to be normalized to fall in a range between 0 and 1 for combination. Relevance feedback scores already fell in this range, so they did not need to be normalized. K-nearest-neighbor scores were divided by 20, and Bayesian scores were divided by the maximum score for that code, that is, the score that would have been attained for a hypothetical document that had all the terms which had larger coefficients for presence of the term than for absence of the term, and which did not have any terms which had larger coefficients for absence of the term than for presence of the term. Note that normalization by the maximum possible score for the code changes the ranks of code candidates for each document, because each code is normalized by a different quantity.

2.5.3 Combination Conditions

The component scores are summarized in Table 1 for each of the five 2-way combinations tested. The parameter k above was tuned separately for each of the five combinations, using one of the 255 document tuning sets. Values ranging from .1 to .9 in steps of .1 were tested. This optimization process was carried out separately for the full code classifiers and for the category classifiers.

Combinations 1, 2, and 3 merged the k-nearest-neighbor and Bayesian classifiers. Combination 1 used scores based on the ranks of the codes assigned to each document. Combination 2 was similar, but the Bayesian rank was based on the normalized score described above. Combination 3 used normalized scores rather than ranks. Combinations 4 and 5 merged the k-nearest-neighbor and Relevance Feedback classifiers. Combination 4 was based on ranks, and Combination 5 was based on scores.

Combination 6, not shown in Table 1, is a 3-way combination of the k-nearest-neighbor score/20, the normalized Bayesian score, and the relevance feedback score. We tested all possible triples of coefficients ranging in tenths from .1 to .9 in which the coefficients summed to one. These tests used the same tuning set of 255 documents that the 2-way combinations were tuned on.

2.6 Measuring effectiveness

2.6.1 Five measures

We report five measures of coding accuracy. These measures reflect the success at getting all the codes as high as possible in the list of candidates without considering a cutoff for acceptance.

Average 11 point precision. Precision and recall have been standard measures of retrieval effectiveness in information retrieval [22]. When the task is retrieval, these measures are computed from the ranked list of documents retrieved for each query. For each such list, and each possible stopping point on the list, one can measure *precision* - the proportion of retrieved documents that are relevant to the query - and *recall* - the proportion of all the relevant documents that are retrieved. Average precision is computed across precision values obtained at n evenly spaced recall points (0, 10%, etc.).

In a categorization task, one can use the same measures, recall and precision, in the same way, on the list of documents ranked by their score on the classifier. Being in the category is analogous to being relevant.

In this study, we compute recall and precision on the list of codes ranked for each test document, rather than the list of documents ranked for each classifier (code). A "relevant" code is one which should be assigned to the test document. This is a natural way to analyze the output of the k-nearest-neighbor classifier. It is a less natural way to analyze the output of the Bayesian and relevance feedback classifiers, but it allows us to compare the performance of the three classifiers and combine them in simple ways.

Top candidate. Proportion of cases where the test document's principal diagnosis (first) code is top candidate in the list of codes ordered by $Score_c$.

Top 10. Proportion of cases where the test document's principal diagnosis code is in the top 10 candidates.

Recall 15. Recall level in the top 15 candidates, that is what proportion of all the correct codes for the document appear in the top 15 candidates. Fifteen was chosen because it is the largest number of codes that can be assigned to a document. Therefore, it is the smallest candidate list where recall could potentially be 100%.

Recall 20. Recall level in the top 20 candidates, that is what proportion of all the correct codes for the document appear in the top 20 candidates. Twenty was chosen because it is a reasonable number of codes for an interactive coder to display.

2.6.2 Full codes vs categories.

The five measures above can be based on full codes or categories. ICD9 codes have two parts, a major category (before the decimal point) and a subcategory (additional digits after the decimal point). Although a completely automatic coder would have to assign full codes including subcategories, we have included some measures that reflect partial success. Therefore, we report the three measures above for two different scoring schemes. *Full Codes* means that the whole code with subcategory had to match to be counted as correct. *Categories* means that only the category – the part of the code before the decimal point – had to match to be counted as correct.

3 Results

3.1 K-nearest-neighbor Results

Table 2 shows k-nearest-neighbor performance on the five measures described above for the baseline, for the best docu-

Full Codes

	Average Precision		Principal code is top candidate		Principal code in top 10		Recall at 15		Recall at 20	
Base	37.5		24.1		59.4		52.8		55.4	
Princ	38.5	+2.7	30.5	+26.7	65.2	+9.9	53.5	+1.5	56.6	+2.2
Wsum	41.3	+10.2	36.4	+51.1	69.0	+16.2	55.8	+5.7	58.7	+6.0
Sec	42.6	+13.6	38.5	+60.0	72.2	+21.6	57.6	+9.1	61.6	+11.3
Categories										
Base	48.7		42.2		74.9		65.4		69.0	
Princ	50.6	+3.8	49.7	+17.7	78.1	+4.3	66.4	+1.5	69.0	+0.0
Wsum	53.5	+9.7	54.0	+27.8	81.8	+9.3	68.2	+4.3	72.8	+5.6
Sec	54.0	+10.7	55.1	+30.4	84.0	+12.1	69.7	+6.6	72.9	+5.7

Table 2: K-nearest-neighbor coding performance

ment-score weighting condition, and for the weighted sum condition with equal weights on all sections, and for the weighted sum condition with tuned weights on each section. The weighted sum conditions included the document score weighting. These results summarize ranked lists of codes that could include any of the 3261 ICD9 codes that occurred in the training corpus. The table also shows percentage increase over the baseline for the nonbaseline conditions.

3.1.1 K-nearest-neighbor baseline accuracy

The rows labeled *Base* in Table 2 show performance for the baseline condition.

Average 11-point precision for full codes in the baseline condition is 37.5%. The principal code was the top candidate in 24.1% of the cases, and was in the top ten in 59.4% of the cases. Recall is 52.8% at the first 15 candidates, and 55.4% at 20 candidates. When we score categories rather than full codes, the average precision is 48.7%. The principal category is the top candidate in 42.2% of the cases, and is in the top 10 in 74.9% of the cases. Recall is 65.4% at 15, and 69.0% at 20.

3.1.2 Document-score weighting

The best value for the principal diagnosis code weight (w_P) was 1.8. Note that this was the value that maximized average precision. A value of 3 would have maximized the top candidate measure. However, in all of the tuning experiments reported in this paper, we maximized average precision in the tuning set, since this is the only measure that summarizes the performance of the full ordering of codes.

As can be seen in the Table 2 in the row labeled *Princ*, this weighting scheme produced a 2.7% increase in average precision over the baseline, a 26.7% increase in the top candidate measure, a 9.9% increase in the top 10 measure, and very small increases in recall 15 and recall 20 measures. A similar pattern is seen with category scores.

Note that principal DX weighting has a large effect only on the measures involving the principal DX code. This weighting scheme primarily moves the principal DX code higher on candidate lists, and does not greatly affect the other codes.

3.1.3 Structured queries

Table 2 also shows the results when the test document is converted into a query which is a weighted sum of sections. Formulating the query as a weighted sum with weights of 1, combined with a principal DX weight of 1.8 (*Wsum* condition) produces a 10.2% improvement in average precision over the baseline, a 51.1% increase in the top candidate measure, and a 16.2% increase in the top 10 measure. Combining the optimal section weights found in the tuning procedure with the best principal DX weight (*Sec* condition) yields a 13.6% improvement in average precision, a 60% increase in the top candidate measure, and a 21.6% increase in the top 10 measure. A similar pattern is seen with category scores.

It is interesting that the `#wsum` version of the documents is such an improvement over the flat free-text version, even before the sections are differentially weighted. The improvement is probably due to the length normalization INQUERY performs at each `#sum` node, which has the effect of giving more weight to short sections and less weight to long sections.

3.2 Bayesian Results

Table 3 shows the Bayesian and k-nearest-neighbor results on the test data restricted to codes that occur six or more times, and restricted to test documents whose principal diagnosis code was not eliminated by this frequency criterion. This set has 157 test documents in it and tests 1068 different codes. Note that the k-nearest-neighbor data in this table have been restricted to the same subset of codes and documents. For this reason, the baseline k-nearest-neighbor scores in this table are substantially higher than the baseline in in Table 2.

Note also that the category scoring is done differently for the Bayesian and relevance feedback classifiers. To get scores for category assignments, classifiers were trained for categories. To make the k-nearest-neighbor conditions comparable, they were rerun as if the training and test documents had only category scores assigned to them.

Although the k-nearest-neighbor and Bayesian results are not significantly different in average precision, they do show some striking differences in the other measures. The k-nearest-neighbor classifier is better at getting correct codes, and particularly the principal diagnosis code, to the top of the candidate list, but the Bayesian classifier is better at getting more codes onto the list. This can be seen to a certain extent in Table 3, in that the Bayesian classifier is much worse than k-nearest-neighbor in the TopCand measure, about the same in the top 10 measure, and better in the Recall 15 and Recall 20 measures. This pattern is more apparent when one examines the precision at 11 recall levels, in Table 4. The k-nearest-neighbor classifier is much better at low recall levels, and the Bayesian classifier is much better

Full codes

	Average Precision		Principal code is top candidate		Principal code in top 10		Recall at 15		Recall at 20	
KNN 6	48.9		45.9		80.9		63.2		67.1	
Bayes 6	47.5	−2.8	35.7	−22.2	81.5	+0.8	68.8	+8.9	74.7	+11.3
RF 6	42.1	−13.9	34.4	−25.0	81.5	+0.8	63.0	−0.4	67.1	+0.0

Categories

	Average Precision		Principal code is top candidate		Principal code in top 10		Recall at 15		Recall at 20	
KNN 6	55.2		56.0		84.6		70.0		73.1	
Bayes 6	53.3	−3.6	41.2	−26.5	85.7	+1.3	75.1	+7.3	79.0	+8.1
RF 6	51.0	−7.5	39.6	−29.4	85.2	+0.7	69.2	−1.2	74.3	+1.6

Table 3: Performance of Bayesian and Relevance Feedback Classifiers - codes occurring ≥6 times

Precision and % change 157 queries					
Recall	KNN 6	Bayes 6		RF 6	
0	81.0	72.7	−10.1	71.0	−12.3
10	79.4	71.8	−09.7	69.0	−13.2
20	74.5	66.0	−11.4	64.1	−14.0
30	65.9	57.7	−12.5	56.2	−14.8
40	56.2	51.7	−08.0	46.1	−17.9
50	53.0	50.2	−05.2	43.9	−17.2
60	37.3	39.7	+06.4	31.0	−16.8
70	27.3	32.9	+20.4	25.5	−6.7
80	24.1	29.7	+23.6	20.9	−13.0
90	19.7	25.4	+29.2	17.8	−9.7
100	19.6	25.1	+28.3	17.7	−9.7
avg	48.9	47.5	−02.8	42.1	−13.9

Table 4: Precision at 11 standard recall points for Bayesian and Relevance Feedback Classifiers

at high recall levels.

3.3 Relevance Feedback Results

The rows labeled *RF 6* in Table 3 show the relevance feedback results in comparison with the k-nearest-neighbor and Bayesian classifiers, also restricted to codes that occur six or more times in the training corpus. Overall performance is substantially worse than that of the k-nearest-neighbor and Bayesian classifiers. It scores low where each of the other classifiers scores low, but does not score high where they score high. Average precision is lower than that of the k-nearest-neighbor and Bayesian classifiers, the top candidate measure is comparable to the Bayesian classifier, that is, much lower than k-nearest-neighbor . The relevance feedback classifier is comparable to the k-nearest-neighbor classifier on the recall 10 and recall 15 measures, that is, substantially lower than the Bayesian classifier.

3.4 Results - Combination Classifiers

Table 5 shows the results of all five 2-way combinations of the k-nearest-neighbor and other classifiers in comparison with the best versions of the individual classifiers. It is striking that all the combinations perform much better than the individual classifiers. Although the individual relevance feedback classifier was quite a bit worse than the Bayesian classifier, the relevance feedback combination classifier performed as well as or better than the Bayesian combination classifier. Combinations involving normalized scores were better than combinations involving ranks.

Consequently, when we tested combinations of all three classifiers, we used only scores. K-nearest-neighbor scores were divided by 20 and Bayesian scores were normalized by the maximum possible score that classifier. Relevance feedback scores were not normalized. The optimal set of coefficients was .3 for the k-nearest-neighbor classifier, .1 for the Bayesian classifier, and .6 for the relevance feedback classifier. As can be seen in Table 5, this 3-way combination was better than any of the 2-way combinations in all measures.

4 Discussion

Combining a k-nearest-neighbor classifier with another classifier yielded a substantial improvement in accuracy over either classifier alone, and the combination of all three classifiers was the best of all. Detailed analyses of the outputs of each classifier showed that they had somewhat complementary strengths and weaknesses. The k-nearest-neighbor classifier was good at getting the principal DX code at the top of the list of candidates, probably because of the principal DX weighting. It was also good at getting other codes to the top of the list (good at low recall levels). The other classifiers were worse at getting correct codes to the top of the list. The Bayesian classifier was better than the k-nearest-neighbor and relevance feedback classifiers at getting correct codes onto the list, that is it was better at high recall levels.

It is somewhat surprising that the relevance feedback combination classifier was as good or slightly better than the Bayesian combination classifier, given that the relevance feedback classifier alone was substantially worse than the Bayesian classifier alone. It is also suprising that the optimal 3-way combination had such a higher weight on the relevance feedback component (.6) than on the Bayesian component (.1). An examination of the codes assigned to individual documents suggested a possible explanation for this pattern. There were several documents on which neither individual classifier (k-nearest-neighbor or relevance feedback) did well, but the combined classifier did very well. An examination of the candidate lists of codes for these cases showed that the k-nearest-neighbor and relevance feedback classifiers proposed very different codes for these documents. For a code to appear high on the list for the combined classifier, it must occur moderately high on both lists. Only the correct codes did so.

We have confirmed our hypothesis that using multiple classifiers improves classification performance, just as using multiple retrieval methods improves retrieval effectiveness.

4.1 Amount of training data

All the results so far are based on codes which have at least 6 examples in the training corpus. Six examples is a small number of cases to base our training on, and we believe the

Full codes

	k	Average Precision		Principal code is top candidate		Principal code in top 10		Recall at 15		Recall at 20	
KNN		48.9		45.9		80.9		63.2		67.1	
Bayes		47.5	−2.8	35.7	−22.2	81.5	+0.8	68.8	+8.9	74.7	+11.3
1	.3	52.0	+6.4	40.8	−11.1	80.3	−0.8	68.1	+7.7	72.9	+8.7
2	.5	53.9	+10.3	38.9	−15.3	80.3	−0.8	67.9	+7.5	72.6	+8.1
3	.6	55.3	+13.0	46.5	+1.4	86.0	+6.3	72.1	+14.1	76.4	+13.9
RF		42.1	−13.9	34.4	−25.0	81.5	+0.8	63.0	−0.4	67.1	+0.0
4	.7	53.7	+9.9	44.6	−2.8	82.2	1.6	67.5	+6.8	71.8	+7.1
5	.3	55.6	+13.8	46.5	+1.4	87.9	+8.7	71.5	+13.1	75.7	+12.8
6 (3-way)		57.0	+16.6	46.5	+1.4	91.1	+12.6	73.2	+15.9	77.6	+15.6
Categories											
KNN		55.2		56.0		84.6		70.0		73.1	
Bayes		53.3	−3.6	41.2	−26.5	85.7	+1.3	75.1	+7.3	79.0	+8.1
1	.5	59.6	+8.0	51.7	−7.8	87.9	+3.9	74.7	+6.8	80.0	+9.4
2	.4	59.9	+8.5	44.5	−20.6	86.8	+2.6	75.0	+7.2	78.8	+7.8
3	.6	62.1	+12.6	57.1	+2.0	90.7	+7.1	77.4	+10.6	81.1	+11.0
RF		51.0	−7.5	39.6	−29.4	85.2	+0.7	69.2	−1.2	74.3	+1.6
4	.8	57.2	+3.6	56.0	+0.0	85.2	+0.7	71.8	+2.6	77.2	+5.6
5	.3	63.1	+14.2	57.1	+2.0	91.2	+7.8	79.0	+12.8	82.4	+12.7
6 (3-way)		65.0	+17.7	59.9	+6.9	91.2	+7.8	80.0	+14.2	83.9	+14.8

Table 5: Performance of all combination classifiers - codes occurring ≥6 times

results would improve with more training data. To illustrate the effects of amount of training data we could restrict the data to codes which met a training minimum frequency criterion. Unfortunately, this would not give a clear picture of the effects of amount of training, because the number of training cases would be confounded with the number of codes in the test. For example, for the test restricted to codes with 100 or more training examples, precision would be based on ranked lists of 89 codes. For the test restricted to codes with 6 or more training examples, precision would be based on a ranked lists of 1068 codes reflecting a choice among 1068 rather than 89 codes, a more difficult task.

Figure 4 shows the 3-way combination data partitioned in a way that avoids this confounding. The test documents have been grouped by the frequency of the principal diagnosis code for the document. Precision is still computed using the ranked lists of 1068 codes. The data point at frequency 6 includes the documents whose principal diagnosis code occurs between 6 and 12 times in the training data. The data point at frequency 13 includes the documents whose principal diagnosis code occurs between 13 and 24 times in the training data, etc.

Figure 4 shows a rapid rise in average precision as the frequency in the training data rises from 6 to 25, then it rises more slowly. Clearly performance is better when each code has 25 or more training examples.

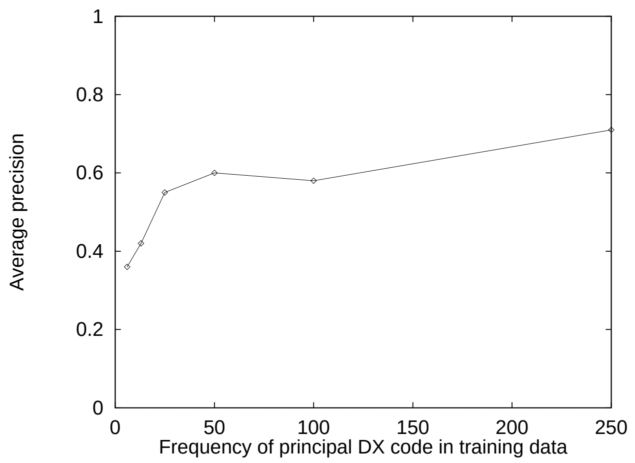

Figure 4: Average precision as a function of frequency of principal DX code

4.2 Comparison with other research

How do these results compare to other attempts at automatic coding and categorization in the medical domain? Researchers at the Mayo Clinic [29] have used a method called ExpNet which is very similar to our k-nearest-neighbor classifier and which yields performance very similar to that of our k-nearest-neighbor classifier when applied to a problem with similar parameters.

Yang and Chute report categorization performance on two different data sets, one for surgical reports in which the classes were ICD9 categories, and one for a set of MED-

LINE documents. Although their surgical report task was more like ours in content, the task was very different. The average text had only nine words, and needed to be associated with one code. There were many duplicate texts, and a total of 281 codes were trained. On this easy tasks, average precision was 88%. Recall that our data set contained texts averaging 633 words in length, had 3261 (different) codes, with an average of 4.4 codes per text. Yang and Chute's MEDLINE data set was somewhat comparable to ours, averaging 168 words per document, 17 categories per document, and a total of 4020 different codes. Their performance of 35% was very similar to the 37.5% attained by our baseline k-nearest-neighbor classifier. Our improvements to the k-nearest-neighbor classifier brought the performance up to 42.5%, and the 3-way combination classifier was at 57% average precision.

Performance in this task is still far from the level required for unsupervised automatic coding. However, this system could form a component of a computer-aided coding system. It could present a list of codes as candidates to be checked by an expert coder. As Table 5 indicates, this system would get the principal DX code as the top candidate 46.5% of the time, it would have the principal DX code in the top 10 candidates 91.1% of the time, and it would have 77.6% of the correct codes in the top 20 candidates. Such an interactive system has proven useful to human experts in indexing physics abstracts [9] and may be useful in coding patient records.

4.3 Future Directions

Our next step is to take advantage of yet another level of structure in these documents. Our associates are using NLP techniques to tag phrases in the discharge summaries with five subtypes each of diagnoses and signs or symptoms [25]. Our hypothesis is that performance will be improved by giving more weight to these items in k-nearest-neighbor classification, or to consider such phrases as candidate features along with the single terms we now use in the Bayesian or relevance feedback classifiers.

The Bayesian and relevance feedback classifiers could be possibly enhanced by training two levels of classifiers. The first level classifiers would assign categories of codes (the code without the subcategories after the decimal points). The second level would choose the best subcategory for each code. This approach is motivated by the observation that the candidate lists often contained many codes of the same category, pushing other correct codes lower on the list. This is not surprising, since codes for related conditions have very similar evidence. A classifier which was trying to distinguish a code from other codes in the same category could be more discriminating than a classifier trying to distinguish a code from all the others. Another advantage of a two-level classifier would be to capture co-occurrence patterns among different classifers, as Fuhr, et. al. do in the AIR/X system [9]. Our current models lose this information because classifiers for each code are completely independent of each other.

5 Acknowledgments

We would like to thank David Aronow for his help in categorizing the section titles in the documents. I would also like to thank David Fisher, Fang-Fang Feng, and Stephen Soderland for the NLP tagging. Thanks to James Allan for the relevance feedback algorithms. This work is supported by by ARPA contract N66001-94-D-6054. Stephen I. Gallant also contributed to this work, supported by National Cancer Institute Grant 1 R43 CA 65250-01 to Belmont Research Inc. Data are courtesy of Brigham and Women's Hospital, Boston, MA.

References

[1] James Allan, Lisa Ballesteros, James P. Callan, W. Bruce Croft, and Zhihong Lu. Recent experiments with INQUERY. In Donna K. Harmon, editor, *The Fourth Text REtrieval Conference (TREC-4)*, Gaithersburg, MD, 1996. National Institute of Standards and Technology, special publication. To appear.

[2] Chidinand Apte, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, July 1994.

[3] N. Belkin, C. Cool, W. Bruce Croft, and James P. Callan. The effect of multiple query representations on information retrieval system performance. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, 1993.

[4] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357, 1995.

[5] W. B. Croft, T. J. Lucia, J. Cringean, and P. Willett. Retrieving documents by plausible inference: An experimental study. *Information Processing and Management*, 25(6):599–614, 1989.

[6] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[7] Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In Donna K. Harmon, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 243–252, Gaithersburg, MD, 1994. National Institute of Standards and Technology, special publication 500-215.

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

[9] Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Konstadinos Tzeras, and Gerhard Knorz. AIR/X - a rule-based multistage indexing system for large subject fields. In *Proceedings of the RIAO '91*, pages 606–623, Barcelona Spain, April 1991.

[10] Donna K. Harmon, editor. *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, 1995. National Institute of Standards and Technology, special publication 500-225.

[11] J. Katzer, M.J. McGill, J.A. Tessier, W. Frakes, and P. DasGupta. A study of the overlap among document representations. *Information Technology: Research and Development*, 1:261–274, 1982.

[12] Leah S. Larkey and W. Bruce Croft. Automatic assignment of ICD9 codes to discharge summaries. Technical Report IR-64, University of Massachusetts Center for Intelligent Information Retrieval, 1995.

[13] David Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50, 1992.

[14] David Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–254, 1995.

[15] David D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, University of Massachusetts, 1992.

[16] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, University of Nevada, Las Vegas, 1994.

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

[18] Brij Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65, 1992.

[19] T. B. Rajashekar and W. Bruce Croft. Combining automatic and manual index representations in probabilistic retrieval. *Journal of the American Society for Information Science*, 6(4):272–283, May 1995.

[20] S. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In Harmon [10].

[21] J.J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System - Experiments in Automatic Document Processing*, chapter 14. Prentice Hall, Englewood Cliffs, 1971.

[22] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.

[23] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[24] Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In Harmon [10].

[25] Stephen Soderland, David Fisher, Jon Aseltine, and Wendy Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.

[26] C. Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.

[27] Howard Turtle and W. Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

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

[29] Yiming Yang and Christopher G. Chute. An application of Expert Network to clinical classification and MEDLINE indexing. In *Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care*, pages 157–161, 1994.