Automatic Assignment of ICD9 Codes To Discharge Summaries

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

Three different types of classifiers were implemented for assigning ICD9 codes automatically to dictated inpatient discharge summaries: A k-nearest-neighbor, relevance feedback, and Bayesian independence classifers. In the k-nearest-neighbor paradigm, a test document is treated as a query against a collection of already-coded training documents. Experiments were carried out to optimize the means of selecting and ranking candidate codes for the test document, based on the scores associated with the retrieved documents. Another line of investigation within the k-nearest-neighbor paradigm determined how best to turn a test discharge summary into a structured query to maximize the chance of retrieving documents with the correct codes for the test document. A combination of different classifiers produced better results than any single type of classifier.

1 Introduction

A great deal of human time and effort is expended in assigning codes of various types to patient medical records. Because this coding determines reimbursement, it is important to accomplish this task as easily and as accurately as possible. At the Center for Intelligent Information Retrieval(CIIR) at UMass Amherst, we are working with several medical organizations on information retrieval problems. We present work in progress on automatically assigning ICD9 codes to dictated inpatient discharge summaries.

We are following several different approaches to automatic coding, all of them incorporating INQUERY, a probabilistic information retrieval system based on an inference net model [1]. These approaches all attempt to use a prelabeled (coded) corpus of discharge summaries to infer codes for new discharge summaries. Determining what codes should be assigned to a document can be seen as a classification problem. Each possible code is a class or category, and we want to determine whether documents belong in each class, or more generally, the probability that a document belongs in each class. We use three different classification techniques, a k-nearest-neighbor [2] approach using the belief scores from INQUERY as the distance metric, Bayesian independence classifiers [3], and relevance feedback.

At some time in the future, we may also experiment with direct lookup in the ICD-9-CM manuals (Alphabetic Index and TabularList).

Past research in information retrieval has shown that one can improve retrieval effectiveness by using multiple representations in indexing and query formulation [4, 5, 1] and by using multiple search strategies [6]. In this work, we investigate whether we can attain similar improvements in the domain of text categorization by combining different representations and classification methods.

The classification methods each lend themselves 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 a large number of terms that are not relevant to the coding task, we are incorporating several different methods for selecting and giving extra weight to words, phrases, and document sections that provide the most diagnostic evidence. These methods include natural language processing to identify diagnosis and symptom-related phrases in the documents [7], and heuristics to divide the summaries into fields that represent different sections of the documents. 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 in order to get the benefit of the multiple 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. 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. Instead, we envision these classifiers being used in an interactive program 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.

2 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.

A sample document can be seen in Figure 1. Note that the codes and text following the codes is not included in the documents in the database, or in the test documents.

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 other codes is not necessary indicative of importance.

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,

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: At the time of discharge, the patient was on Percocet 1-2 q 3 prn, Coumadin 5 mg po q d until directed otherwise. DISPOSITION: To home.
ICD9 Codes for this discharge summary:

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

laboratory examination, hospital course, and disposition. Some have 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.

3 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* [8, 9]. Our approach is similar to that of Yang and Chute [10] 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.

The major questions our k-nearest-neighbor research has addressed so far are:

- How to assign a score to a candidate code for a test document, based on the codes and scores assigned to retrieved documents.
- How best to turn a test document into a structured query, to maximize the chance of retrieving documents with the correct codes.

In addition, we have experimented with field-based indexing and retrieval, and with using natural language processing (NLP) to aid the automatic coding process by tagging items as negated, as diagnoses, or as symptoms.

3.1 Method

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 stripped of their codes and presented one at a time to INQUERY as queries. It should be emphasized that the queries in this paradigm are the full free-text of the discharge summaries, which are then stopped and stemmed as part of the query process. This information retrieval step retrieves 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

Ranked list of retrieved documents

Doc	$rank_i$	$belief_i$	Principal DX code	Other codes for retrieved doc_i				
3580	1	.4320	715.35	996.4				
5997	2	.4301	715.95					
7059	3	.4300	715.35	428.0	041.10	458.9	490	V70.7
1040	4	.4298	720.0	424.0	592.0	533.90		
4556	5	.4295	715.35	276.1	458.9	V43.6	278.0	
6476	6	.4294	715.35	276.5	796.3			
...	...	...	...					

↓

Ranked list of retrieved codes

Code	#	$Score_c$	Description of code
715.35	10	7.7077	osteoarthrosis, localized, not specified ...
*715.95	5	3.8535	osteoarthrosis, unspecified whether generalized or localized ...
428.0	4	1.7080	congestive heart failure
401.9	4	1.7057	unspecified essential hypertension
....	...	...	

Table 1: Ranked list of retrieved docs, and derived ranked list of retrieved codes for test doc

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.

These two steps are exemplified in Table 1.

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.

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 start by summing the belief scores of the retrieved documents assigned that code, weighting the scores in various ways 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,
w_{ic} is the weight for code c in document i.

3.1.1 Weighting Methods

We have tested several different weighting methods for determining w_{ic}:

Baseline. Weight is 1 if the code is assigned to the retrieved document, 0 otherwise. In other words, $Score_c$ is the sum of the scores of the retrieved documents assigned the code.

Rank weights. The weight is a function of the rank of the document in the list of retrieved documents. We tried various linear functions of the rank.

Up-weight principal diagnosis (principal DX) code. We experimented with up-weighting the principal diagnosis code of each retrieved document by giving it a weight w_P ranging from 1 to 3, that is,

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

w_P was tuned on the tuning set.

Icf weights. The weight is determined by the frequency of the code in the collection. We experimented with an inverse code frequency (*icf*) analogous to the standard inverse document frequency (*idf*) [1], computed by: $\log\left(numdocs/cf\right) + 1$, where *numdocs* is the total number of documents in the collection and *cf* is the number of docs assigned code *c*.

Normalized icf weights The *icf* weights are normalized by the log of the total number of docs, producing weights that range from 0 to 1: $\log\left(numdocs/cf\right) / \log\left(numdocs\right)$.

3.1.2 Structured queries

For the baseline condition and for testing the weighting schemes above, each test document was stripped of its ICD9 codes and input in full text form. In addition, we tested various ways of representing the document to make use of known structure in the document. We reasoned that certain sections found in some of the documents (for example *PRINCIPAL DIAGNOSIS:*) would be more relevant to diagnosis and hence to code assignment than other sections. INQUERY's flexible query language allowed us to formulate each query as a weighted sum of the sections of the documents. Two subtasks made up this part of the research: identifying document sections, and tuning the weights on the sections.

Sections were identified heuristically. This is the only part of the processing that was not completely automatic. Although not all documents had the same sections, and the sections were labeled in various ways, they were usually identified by a title in upper case and terminated by a colon. We used the Unix tools `flex` and `awk` to list the titles in reverse frequency order. All the titles above a threshold were categorized under one of ten section types: *ADD* (addendum), *ADMIN* (administrative), *DISCH* (disposition), *DX* (diagnosis),

<DX> PRINCIPAL DIAGNOSIS: 1. OSTEOARTHRITIS OF THE LEFT HIP SECONDARY DIAGNOSIS: 2. WOLFF-PARKINSON-WHITE SYNDROME < /DX>

<OR> PROCEDURES: Left total hip replacement (uncemented), 2-2-93. < /OR>

<HPI> 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. < /HPI>

<PMH> 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. < /PMH> ...

Figure 2: Example test document with section tags

HOSP (hospital course), *HPI* (history of present illness), *LAB* (laboratory examination), *OR* (operations and procedures), *PE* (physical examination), *PMH* (past medical history).

A **`flex`** script was written to recognize the high-frequency titles using regular expressions, and to add tags marking the beginnings and endings of each section. Then we iteratively examined the untagged sections, and made the regular expressions more general to encompass more variations on the titles. We stopped when most of the remaining untagged sections belonged to the same category as the preceding tagged section and then modified the algorithm to include any untagged material in the section preceding it.

Figure 2 shows a portion of the example document with section tags.

To weight sections differentially, we used INQUERY's **`#wsum`**(weighted sum) and **`#sum`** operators, as in Figure 3.

The weights in the baseline weighted sum condition were all equal to 1.

Weights were tuned using the tuning set divided into two sets with 255 documents each. We used a hill-climbing algorithm [2], and accepted each successive change in weights that improved the first tuning set without hurting performance on the second tuning set.

3.1.3 Field Specific Retrieval

To investigate whether field-specific retrieval would improve classification, a fielded version of the database of discharge summaries was created. Each section type was indexed as a different field. To query the fielded database, queries were formulated by replacing the **`#sum (...)`** operator in the weighted sum query above with a **`#field (fieldname ...)`** operator, indicating that the retrieval system should look for that material only inside the corresponding field in the training documents.

3.1.4 Natural Language Processing

In addition to the section tagging, text was also tagged with "without" (WO) tags, in cases like:

```
#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: 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. ) ...)
```

Figure 3: Example test document as a weighted sum query

...patient denied <WO>fevers< /WO>, <WO>chills< /WO> , ...

This tagging was carried out using a simple finite automaton incorporating rules for the scoping of negation and a lexicon of medical words. The aim was to avoid spurious retrievals of cases with positive mentions of items occurring in WO fields of the test documents. We tried many different ways of using the WO tags, including leaving the WO sections out of the queries, downweighting them, and indexing them in their own fields in the database and using field operators in the queries.

3.2 Measuring effectiveness

3.2.1 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 [11]. 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.

3.2.2 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.

3.3 Results

Table 2 shows k-nearest-neighbor performance on the five measures described above for the baseline and best document-score weighting conditions, and for the baseline and best section weighting conditions. The table also shows percentage increase over the baseline for the nonbaseline conditions.

3.3.1 K-nearest-neighbor baseline accuracy

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

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

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. 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.

3.3.2 Document-score weighting

Of all the methods of determining w_{ic} described in section 3.1.1, only upweighting the principal DX code was better than the baseline. None of the other document-score weighting methods produced any improvement.

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, and a 9.9% increase in the top 10 measure. A similar pattern is seen with category scores.

Note that principal DX weighting causes a larger increase in the top candidate measure than in average precision or in the top 10 measure. This weighting scheme moves the principal DX code to the top of the list more than it gets correct codes onto the list of candidates.

3.3.3 Structured queries

Table 2 shows the results when the test document is converted into 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 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.

3.3.4 NLP tags and fields

None of the methods of using the WO tags improved the results. No improvements occurred using field operators and the fielded database, either for sections or for WO tags.

3.4 Discussion: Generality of the document structure analysis

Taking advantage of the section structure of the documents afforded a substantial gain in retrieval accuracy. Since the labelling of sections was a heuristic step requiring a few weeks of tweaking a set of regular expressions, one could question the general value of this approach. However, the section labelling program would continue to successfully tag new documents like the old ones. At a site where the format of discharge summaries was more standardized, or in a database where the sections were already in different fields, this step could be more completely automated.

4 Bayesian Independence Classifiers

Deciding whether to assign a given ICD9 code c to a document can be conceived as a text categorization problem: should the document be placed in the class of code c type documents or not?

We trained a large number of binary classifiers, one for each code, using the large, prelabeled corpus of discharge summaries as the training set. The Bayesian independence classifier, described more fully below, uses Bayes theorem to estimate the probability of category membership for each category and each document. The probability estimates are based on the co-ocurrence of codes and features (terms) in the training set, and assume the features are independent. 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.

The form of the Bayesian classifier used here [3] does not consider term frequency, but only whether a term occurs in the data.

4.1 Method

A set of 1068 classifiers were trained, one for each code that occurred 6 or more times in the training data. 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 forty features (stemmed terms) were chosen for each classifier (code) according to mutual information [12], 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 did not improve performance.

The probabilistic model described by Lewis [3] was used for training the classifiers. Our classifiers are all binary - a document either has a code or it doesn't have the code, so the model takes the following form:

$$P(C|Doc) \quad = \quad P(C) \cdot \prod_i \left(\frac{P(A_i|C) \cdot P(A_i|Doc)}{P(A_i)} + \frac{P(\overline{A}_i|C) \cdot P(\overline{A}_i|Doc)}{P(\overline{A}_i)} \right)$$

where:

$$\begin{aligned}
P(C) &= \text{Prior probability any doc is in class } C \\
P(A_i) &= \text{Probability any doc has feature } A_i \\
P(\overline{A}_i) &= \text{Probability any doc does not have feature } A_i \\
P(A_i|Doc) &= \text{Probability that the test doc has feature } A_i \\
P(\overline{A}_i|Doc) &= \text{Probability that the test doc does not have feature } A_i \\
P(A_i|C) &= \text{Probability that docs in class C have feature } A_i \\
P(\overline{A}_i|C) &= \text{Probability that docs in class C do not have feature } A_i
\end{aligned}$$

We estimate

$$P(A_i|Doc) = \begin{cases} 1 & \text{if the test doc has feature } A_i \\ 0 & \text{if the test doc does not have feature } A_i \end{cases}$$

and use the following log probability as a score for code c:

$$Score_c = \log(P(C)) + \sum_i \begin{cases} \log\left(P(A_i|C)/P(A_i)\right) & \text{if the test doc has feature } A_i \\ \log\left(P(\overline{A}_i|C)/P(\overline{A}_i)\right) & \text{if the test doc does not have feature } A_i \end{cases}$$

and the following estimates:

$$P(C) \quad = \quad (n_r + .5)/(N + 1)$$

$$
\begin{aligned}
\frac{P(A_i|C)}{P(A_i)} &= \frac{(df_{rel} + .5)/(n_r + 2)}{(n_t + .5)/(N + 1)} \\
\frac{P(\overline{A}_i|C)}{P(\overline{A}_i)} &= \frac{1 - (df_{rel} + .5)/(n_r + 2)}{1 - (n_t + .5)/(N + 1)}
\end{aligned}
$$

where N is the total number of documents, n_r is the number of relevant documents, that is, the number of documents with code c, n_t is the number of documents with the feature A_i, and df_{rel} is the number of relevant documents that have the feature A_i.

The model yields an estimate of the log probability that a code is assigned to a 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.

The results presented for k-nearest-neighbor performance in Table 2 include all 3261 ICD9 codes that occur in the training corpus, regardless of how many training examples existed for these codes. (Only 789 of these codes are correct assignments in in the test data.) In fact, some of the codes had very few training examples, as few as 1. We do not know the minimum number of examples we need to train on to get a reasonable Bayesian classifier, be we decided to restrict our analyses to codes that occur at least six times in the training data. There are 1068 such codes.

In order to compare the Bayesian classifier with the k-nearest-neighbor classifier, we also computed the k-nearest-neighbor results based only on codes that occur 6 or more times in the training data. 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.

4.2 Results

The Table 3 show 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 classifier. 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

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										
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

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 at high recall levels.

5 Relevance Feedback

Relevance feedback is another approach to training a classifier for an ICD9 code. It is similar to the Bayesian approach in that we train a classifier for each code or category based on the co-occurrence of codes and terms. In this case, the classifier is a query, which is run against a database of test documents. A successful query retrieves documents that should be assigned the code with higher scores than documents that should not be assigned the code.

The relevance feedback classifier is very much like the Bayesian classifier. 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 terms 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.

Precision and % change 157 queries					
Recall	KNN 6	Bayes		RF	
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

5.1 Method

The relevance feedback algorithm was essentially the same as that used in TREC4 [13] and is more fully described there. Relevance feedback began with null queries. First, terms were chosen by comparing their occurrences in relevant and non-relevant training documents. Weights were assigned using the Rocchio formula applied to INQUERY's 2.1 weighting scheme. Finally, the weights were adjusted using a technique similar to that of Buckley and Salton and others [14, 15].

5.1.1 Term selection

For each ICD9 code, all terms occurring in the relevant documents (training documents with that code) were identified and ranked by their relative occurrences in the relevant and a subset of the non-relevant documents (documents without the code) in the large training corpus of discharge summaries, that is, by:

$$\frac{df_{rel}}{n_r} - \frac{df_{nonrel}}{n_{nr}}$$

where df_{rel} is the total number of relevant documents containing the term, df_{nonrel} is that count in non-relevant documents, n_r is the number of relevant documents, and n_{nr} is the number of non-relevant documents. The number of nonrelevant documents that went into the training was $min(n_{n_r}, 15 \cdot n_r)$.

The top 40 terms in this ranking were chosen, and a weighted sum query was built from these terms, the weights from the Rocchio formula:

$$\beta \cdot \frac{1}{n_r} \sum_{rel} belief - \gamma \cdot \frac{1}{n_{nr}} \sum_{nonrel} belief$$

where $\beta = 2$, $\gamma = \frac{1}{2}$, and the belief for term t in doc d was calculated by the formula:

$$0.4 + 0.6 \cdot (0.4 \cdot \min(1, \frac{200}{maxtf_d}) + 0.6 \frac{\log(tf_{t,d} + 0.5)}{\log(maxtf_d + 1)}) \cdot \frac{\log((n_t + 0.5)/N)}{\log(N+1)}$$

where $tf_{t,d}$ is the occurrences of term t in document d, $maxtf_d$ is the largest number of times any term occurs in documents d, n_t is the number of documents in the collection containing term t, and N is the total number of documents in the collection,

5.1.2 Weight adjustments

Weights were adjusted by an iterative procedure which tried to optimize the performance of the query on the training set. For each term in the query, terms weights were adjusted one a time and the slightly modified query was evaluated. Each change was retained only if it improved effectiveness on the entire training set. Weights were adjusted in 5 passes, with factors of 2.0,1.5, 1.25, 1.125, and 1.0625. In each pass, each term was potentially reweighted by $w_{new} = w_{prev} \cdot pass\_factor$. A pass was terminated when no term's reweighting improved the results.

5.2 Results

The rows labeled *RF 6* in Table 3 show the relevance feedback results in comparison with the k-nearest-neighbor and Bayesian classifiers. The test is restricted to codes that occur six or more times in the training corpus in the same way that the k-nearest-neighbor and Bayesian data were. 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.

6 Combining Different Classifiers

6.1 Method

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. $score_{knn,c}$ is a function of the test document's k-nearest-neighbor score for code c. $score_{other,c}$ is a function of the test

Condition	Component Score		Component Score	
1	KNN	rank	Bayesian	rank
2	KNN	rank	Bayesian	rank of normalized score
3	KNN	score/20	Bayesian	normalized score
4	KNN	rank	Relevance Feedback	rank
5	KNN	score/20	Relevance Feedback	score

Table 5: Components of 2-way combination classifiers

document's score for code c on either the Bayesian or relevance feedback classifier. The functions are described in more detail in sections 6.1.1 and 6.1.2 below.

The component scores are summarized in Table 5 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 a normalized score which is described in section 6.1.2 below. 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 5, 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.

6.1.1 Ranks

For a given document, each rank-based component (k-nearest-neighbor , Bayesian, or relevance feedback) score for code c was determined as follows:

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

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

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
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

Table 6: Performance of 2-way combination classifiers - codes occurring $\geq$6 times

6.1.2 Normalization of component scores

For the combinations using scores rather than ranks, the scores had to be normalized. The k-nearest-neighbor and Bayesian scores were normalized in different ways to fall in a range between 0 and 1. 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.

6.2 Results

Table 6 shows the results of all five 2-way combinations of the k-nearest-neighbor and other classifiers comparison with the best versions of the individual classifiers. It is striking that all the combinations perform much better than the individual classifiers. It is particularly surprising that the the relevance feedback combination classifier performs as well as or better than the Bayesian combination classifier, although the relevance feedback classifier alone was quite a bit worse than the Bayesian classifier alone. Combinations involving normalized

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
Bayes Combo	55.3	+13.0	46.5	+1.4	86.0	+6.3	72.1	+14.1	76.4	+13.9
RF Combo	55.6	+13.8	46.5	+1.4	87.9	+8.7	71.5	+13.1	75.7	+12.8
3 Way Combo	57.0	+16.6	46.5	+1.4	91.1	+12.6	73.2	+15.9	77.6	+15.6
Categories										
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
Bayes Combo	62.1	+12.6	57.1	+2.0	90.7	+7.1	77.4	+10.6	81.1	+11.0
RF Combo	63.1	+14.2	57.1	+2.0	91.2	+7.8	79.0	+12.8	82.4	+12.7
3 Way Combo	65.0	+17.7	59.9	+6.9	91.2	+7.8	80.0	+14.2	83.9	+14.8

Table 7: Summary of best classifiers - codes occurring $\geq$6 times

scores were better than combinations involving ranks.

Consequently, when we tested combinations of all three classifiers, we used k-nearest-neighbor scores normalized by dividing by 20, Bayesian scores normalized by the maximum possible score each classifier, and non-normalized relevance feedback scores. Scores from three classifiers were combined in the same way as scores from 2 classifiers. 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 7, this three way combination was better than any of the 2-way combinations in all measures.

7 Discussion

7.1 Combining Classifiers

Table 7 shows the performance of the best classifiers of each type on all the measures described in section 3.2.1.

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 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.

7.2 Improving the results with more training data

Performance in this task is 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 7 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.

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 training cases to base our training on, and we believe the results would improve with more training data. To illustrate the effects how more training data would improve the results, we performed a series of tests using the 3-way combination classifier in which we restricted the data to codes which met a minimum frequency criterion in the training data. Figure 4 shows how average precision improves as the minimum frequency is varied from 6 to 500.

Figure 4 does not give the clearest possible picture of the effects of amount of training, however, because the number of training cases is confounded with the number of codes in the test. That is, when we look at codes occurring 100 or more times in the data, we are computing precision based on a ranked list of 89 codes. When we consider codes occurring 6 or more times in the data, we are computing precision over the ranked list of 1068 codes for each document, reflecting a choice among 1068 rather than 89 codes, a more difficult task.

Figure 5 shows the data partitioned in a way that avoids the confounding in Figure 4. The test documents have been grouped by the frequency of the principal diagnosis code for the document but 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 5 shows a rapid rise in average precision as the frequency in the training data rises

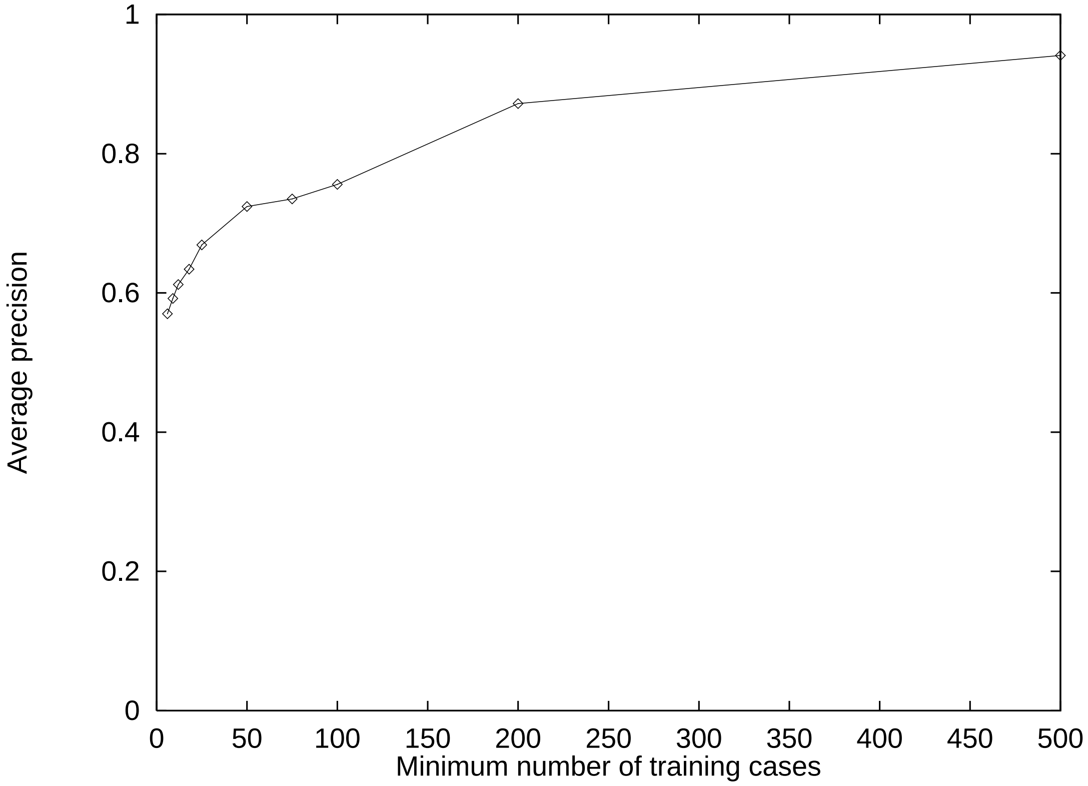

Figure 4: Average precision as a function of minimum number of training examples

from 6 to 25, then it rises more slowly. Clearly performance is better when each code has 25 or more training examples.

How do these results compare to other attempts at automatic coding and categorization in the medical domain? Researchers at the Mayo Clinic [10] 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 MEDLINE 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, their average 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. The improvements to our k-nearest-neighbor classifier brought the performance up to 42.5%, and the three way combination classifier was at 57% average precision, greatly exceeding their results.

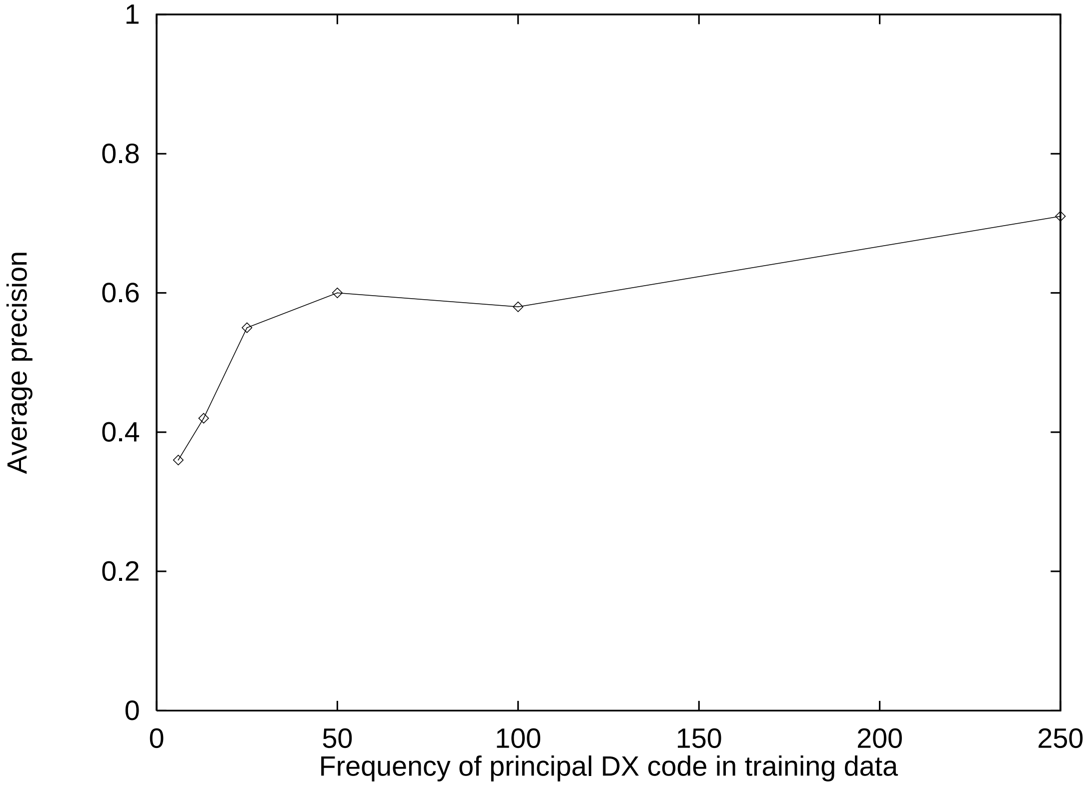

Figure 5: Average precision as a function of frequency of principal DX code

7.3 Future Directions

Our next step in the k-nearest-neighbor approach 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 [7]. Our hypothesis is that performance will be improved by giving more weight to the items selected in this way.

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 would 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 method would be to obtain the text of the ICD9-CM Tabular List and Alphabetic Index, and to automate the lookup procedures used by manual coders. This method would be particularly useful in just the cases where the other two categorization methods would fail — the codes for which there is too little (or no) training data.

This technique would confront us with a vocabulary mismatch problem which was not a major part of the k-nearest-neighbor and other categorization techniques mentioned above. In our work so far, we have been matching test documents against a corpus of training documents of exactly the same kind. They have a varied vocabulary, but there is no systematic difference between the training and test documents. In contrast, the vocabulary used in the discharge summaries is systematically different from the controlled vocabulary of the ICD9 descriptors. Our preliminary research has shown this to be a serious problem, and we are now experimenting with using the UMLS Metathesaurus to alleviate the mismatch problem.

8 Acknowledgments

I 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] H. Turtle and W. B. Croft, "Evaluation of an inference network-based retrieval model," *ACM Transactions on Information Systems*, vol. 9, pp. 187–222, July 1991.

[2] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis.* New York: John Wiley & Sons, 1973.

[3] D. 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*, pp. 37–50, 1992.

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

[5] N. Belkin, C. Cool, W. B. Croft, and J. 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*, pp. 339–346, 1993.

[6] W. B. Croft, T. J. Lucia, J. Cringean, and P. Willett, "Retrieving documents by plausible inference: An experimental study," *Information Processing and Management*, vol. 25, no. 6, pp. 599–614, 1989.

[7] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert, "Crystal: inducing a conceptual dictionary," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, (Montreal, Canada), August 1995.

[8] B. Masand, G. Linoff, and D. 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*, pp. 59–65, 1992.

[9] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM*, vol. 29, pp. 1213–1228, Dec. 1986.

[10] Y. Yang and C. 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*, pp. 157–161, 1994.

[11] G. Salton, *Automatic text processing: The transformation, analysis, and retrieval of information by computer.* Reading, MA: Addison-Wesley, 1989.

[12] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Francisco, CA: Morgan Kaufmann, 1988.

[13] J. Allan, L. Ballesteros, J. P. Callan, W. B. Croft, and Z. Lu, "Recent experiments with INQUERY," in *The Fourth Text REtrieval Conference (TREC-4)*, (Gaithersburg, MD), National Institute of Standards and Technology, special publication, 1996. To appear.

[14] C. Buckley and G. Salton, "Optimization of relevance feedback weights," in *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 351–357, 1995.

[15] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *The Third Text REtrieval Conference (TREC-3)*, (Gaithersburg, MD), National Institute of Standards and Technology, special publication, 1995.