In the *Journal for Experimental and Theoretical Artificial Intelligence* 7(1), pp. 271-302, 1995.

Inductive Text Classification for Medical Applications

Wendy Lehnert, Stephen Soderland, David Aronow*,
Fangfang Feng, Avinoam Shmueli

Center for Intelligent Information Retrieval
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003
lehnert@cs.umass.edu

*Harvard Community Health Plan
Brookline, MA 02146

Abstract

Text classification poses a significant challenge for knowledge-based technologies because it touches on all the familiar demons of artificial intelligence: the knowledge engineering bottleneck, problems of scale, easy portability across multiple applications, and cost-effective system construction. Information retrieval (IR) technologies traditionally avoid all of these issues by defining a document in terms of a statistical profile of its lexical items. The IR community is willing to exploit a superficial type of knowledge found in dictionaries and thesaurae, but anything that requires customization, application-specific engineering, or any amount of manual tinkering is thought to be incompatible with practical cost-effective system designs. In this paper we will challenge those assumptions and show how machine learning techniques can operate as an effective method for automated knowledge acquisition when it is applied to a representative training corpus, and leveraged against a few hours of routine work by a domain expert. We describe a fully implemented text classification system operating on a medical testbed, and report experimental results based on that testbed.

1. Introduction

The AI community has always spanned the full spectrum of theoretical/applied research activities. Anyone who is interested in making computers act intelligently must be interested in all forms of useful intelligence and useful intelligence is necessarily defined in terms of useful activities. Although AI research is not always application-driven, it inevitably benefits from a clear understanding of tasks that people do, and an imaginative vision of the things that people can't do but wish they could.

In recent years, a number of a technologies have increased the availability of online text materials. Increasingly powerful storage media such as CD-ROMs and optical hard drives have made it practical to archive large amounts of text electronically. High speed modems and a plethora of Internet services have made it both possible and practical to download text that was previously controlled by publishers, or oftentimes previously unavailable for general distribution. Optical character recognition and speech recognition threaten to increase our stores of electronic text by yet another order of magnitude. For those of us who are "plugged in," the intelligent management of online text is almost certainly one of the things that we all wish we were handling better.

Language and text are human inventions but they are intimately tied to both cognitive apparatus and social interactions. Biological evolution has kept pace with the basic needs of our species, but those needs are changing rapidly in a technological environment. Our biological mechanisms are sometimes characterized as being operational on a "human scale." For example, people do most of their speaking and language hearing in the context of one-to-one social interactions. We consequently find it difficult to attend to more than one thing at a time, and psychologists have studied resource-limited processing in an effort to understand human processing limitations in greater detail. Text is typically viewed as a variant on spoken language that has been frozen in time. Text is consequently a linear medium just like spoken language. A single-channel, narrow bandwidth language processor may be all we ever need when our language is confined to social interactions and the morning newspaper, but the limitations of human scale are painfully apparent when we think about channel surfing across 500 broadcasts or when we pick up an 800-page guide to all the Internet servers. Most of us have come to know The Library of Congress as a repository of written text that is thought to be of some long-lasting value. In the 21st century we may very well adopt the Library of Congress as a standard unit of measure for large amounts of electronic text, as in, "This electronic B-board distributes .2 LCs of text each year." We are clearly headed for a world that was never designed with the limitations of human scale in mind.

Massive amounts of data characterize the essential challenge of intelligent text management. We need to index large collections of static text, filter dynamic streams of incoming text, and extract useful information from potentially relevant text. Any technology that attempts to address these challenges must be evaluated in terms of both scale-up and portability concerns. Issues of scale address the amount of text that can be effectively managed, as well as throughput rates for ongoing text management. Portability refers to the ease or difficulty with which one successful system can be transformed into a slightly or significantly different system, depending on the needs and interests of different users. Successful language processing technologies for online text management must grapple with scale and portability issues in

order to be serious candidates for practical system deployment. This is one area of AI where ignoring the demands of practical applications is tantamount to ignoring the primary challenge of the next century.

Researchers in the field of information retrieval (IR) have been highly sensitive to issues of scale and portability all along. The IR enterprise has traditionally occupied a computer science niche in applied systems: I have yet to meet an AI researcher who describes their primary specialty as IR. Indeed, IR researchers view documents in terms of statistical properties first and foremost. Information in IR is a statistical abstraction in the tradition of Shannon. Semantic content is something that people can extract from documents, but it is nothing that traditional IR systems attempt to characterize.

Since rich semantic representations and their intelligent manipulation are two cornerstones of AI, it cannot surprise us to find AI researchers and IR researchers steering clear of one another. Historical traditions have kept these two communities cleanly separated, and until now, there was never much reason to bemoan such a state of affairs. But the world of online text is suddenly expanding (perhaps exploding) and stretching out into the professional activities of an increasingly larger population. Information retrieval is no longer something that only affects academics conducting on-line literature searches. There are many opportunities for imaginative answers to pressing problems associated with online text, and we need to look at a number of potential solutions before we make any facile assumptions about what technologies are likely to win out in the long run.

There is still a considerable gap between most natural language processing (NLP) technologies and the requirements of scalability and portability. But a few NLP researchers are beginning to address these issues, and the prospects for hybrid technologies are always intriguing. Other areas of AI may also be prepared to produce scalable and portable solutions to specific problems traditionally associated with IR.

In this paper we will explore the application of a machine learning technique to the problem of text classification. More specifically, we will show how inductive decision trees can be used in a text classification system. We will explore some variations on the basic theme, and we will assess the performance of our resulting prototypes. This work addresses needs in health care delivery, but can be applied to any number of text classification scenarios where relatively sensitive discriminations based on professional expertise are needed to produce correct classifications.

1.1 Automated Text Analysis and the Computerized Patient Record

Computerized information management is increasingly crucial for the smooth operation of health care delivery services. At a relatively mundane level, we expect to find computerization of general office tasks such as word processing, scheduling, billing and accounting. At a somewhat more ambitious level, we need to address problems associated with the storage and retrieval of medical records. At the present time, computers provide us with capabilities that are effectively limited to the operations of a highly efficient and compact file cabinet. The basic unit of information is the patient's medical record. The standard medical record operations of storage and retrieval, reading,

printing, and browsing are greatly facilitated by desktop computers and networked workstations.

However, this level of computerized assistance offers no real advantage in terms of the substance and meaning of a medical record's content. With the exception of a very few centers where computerized patient records have been embraced enthusiastically and are an integral part of the everyday delivery of medical care, the ability to use clinical information in electronic form has yet to be realized [Aronow and Coltin 1993; Dick and Steen 1991]. Massive on-line collections of medical files are nevertheless accumulating, and the information in such collections holds a tremendous potential for epidemiological research and improved health care management. To realize this potential, new technology is needed which operates on the information content inside these medical records. Ongoing research in information retrieval and natural language processing is relevant to this challenge, and is needed to convert textual data into a manageable information source.

One area that is particularly well-suited for massive online text analysis involves the health care management of chronic conditions. Chronic health conditions are characterized by long periods of stability, interspersed with periodic exacerbations that can lead to acute attacks, followed by a return to the base line status present prior to the exacerbation, or adaptation to a new relatively stable baseline.

A large corpus of medical records holding descriptions of data pertaining to such cycles can offer insight into the factors that cause exacerbations and/or the signals that indicate the onset of an exacerbation. Harvard Community Health Plan (HCHP) has been using automated medical record keeping from the time of its founding 25 years ago. The staff model division of the health maintenance organization currently has more than 350,000 active members, each of whose entire medical record is maintained on-line [Schoenbaum and Barnett 1992]. An example of an encounter note in the HCHP medical record follows:

```
---------------------------------------------------------------------
LASTNAME<30>,FIRSTNAME<30>
S/R HEADER: 10/12/93
               STATUS REPORT PREPARED 4:46 PM 10/12/1993


LASTNAME<30>,FIRSTNAME<30>                          KEN-000-00-00-A
AGE: 11 YRS.   FEMALE   DOB: xx/xx/xx               FAM #: xxxxxxxxxxx
HOME: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx              GRP: 1400 EFF: 2/1/92
EMERG #: NONE
CURR ADDR: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                                    BENEFIT LEVEL: INTEGRATED CLASSIC
  PRIMARY MD: XXXXX                 PRIMARY RN: XXXXX
---------------------------------------------------------------------
  PARENTAL STATUS: SINGLE           BIRTH ORDER: 2 OF 2
  NAME OF GUARDIAN: XXXXXX          MOTHER'S OCCUPATION: HOME

ENCOUNTER---12/3/92

OBJECTIVE DATA
WEIGHT 199 LBS
TEMP RECTAL 98.4
```

```
PROBLEMS
G992  COUGH
      PRESUMED WHOOPING COUGH CONTINUES TO COUGH.VOMITS.HAS WHOOP
      AFTER COUGHING CAUSES BAD H.A.PE: CHEST CLR.STARTED NOV. 3
      NP SWAB FOR FA & CULTURE BLD FOR AGGLUTINING

TESTS
A156  WHITE BLOOD COUNT  7.46 K/CUMM  (4.50-11.00)
  ACC #:AB00000000

B176 #A  PERTUSSIS SEROLOGY  * MCG/ML  (0.0-5.0)  STL
     STATE LAB RESULT : BORDETELA PERTUSSIS NOT FOUND
  ACC #:AB00000000

THERAPIES
B111  ERYTHROMYCIN 333 MG # 42 1 TAB TID X 2 WK NRF

SITE:   KENMORE
TYPE:   SAME DAY

XXXXXX X. XXXXXX, MD/XXX
-------------------------------------------------------------------------
```

Working with researchers at the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts at Amherst, an HCHP representative identified the domain of pediatric asthma as a good prototype testbed for text analysis technologies to support health care quality improvement. In particular, a text classification task was proposed in which a fully automated system would be responsible for analyzing patient encounter notes to determine which encounters occurred in response to asthma exacerbations.

This paper describes a text analysis system designed to handle fine-grained text classification problems of this type. Experimental results based on the HCHP testbed will be presented, along with detailed descriptions of key algorithms. The **FIGLEAF** (**FI**ne-**G**rained **LE**xical **A**nalysis **F**acility) system, employs (1) a text marking interface to capture domain knowledge from a domain expert, (2) domain-specific dictionary construction, (3) keyword recognition based on a hash function rather than standard stemming and morphology routines, and (4) an inductive decision tree that is trained on a representative text corpus.

1.1 Clinician/Patient Encounter Notes

Whenever a patient meets with a clinician, that clinician typically generates a written record describing the encounter. Sometimes these reports are scribbled onto a piece of paper, while others are dictated for later transcription. In any case, these notes contain potentially valuable information, both for the ongoing care of an individual patient and for future studies based on aggregate data across a large patient population. Some of these reports are stored online, as part of the computerized patient record. But unlike records that can be reduced to forms with fixed fields and expected data values, patient encounter notes cannot be totally codified without loss of information. In free-form text, clinicians are free to remark on any number of observations or concerns that may arise during the meeting.

As more hospitals and HMOs acquire collections of encounter notes online, we have an opportunity to monitor and improve health care delivery by extracting useful information from the encounter notes. Although the manual extraction of data from a large number of encounter notes may be prohibitively expensive, automated information extraction from text and automated text classification technologies can provide a cost-effective alternative. Quality improvement of patient care is an ongoing concern in health care, and one that can benefit greatly from natural language and text processing technologies.

1.2 Encounter Notes and Natural Language Processing Challenges

As text styles go, encounter notes present some special problems that are not normally encountered in published text. When a clinician writes an encounter note with pen and paper, a highly telegraphic form of language may be used. There are often very few (if any) proper sentences, and abbreviations are frequently used for virtually every word containing more than six letters. None of these abbreviations will be found in a dictionary: they are highly idiosyncratic and can vary widely from provider to provider. Although encounter notes preserve a written record of great importance, they are traditionally private notes used by clinicians for later reference - the clinician writing an encounter note is normally not worried about how intelligible the encounter note is to other people, as long as it makes sense to the original author. This makes the analysis of encounter note documents by computer especially challenging.

To illustrate the use of idiosyncratic abbreviations, consider the following examples taken from our pediatric asthma corpus:

CL (CLEAR)	PK (PEAK)
CONG (CONGESTED)	PROBS (PROBLEMS)
DIST (DISTRESS)	PT (PATIENT)
F/U (FOLLOW UP)	RESP (RESPIRATORY)
INCR (INCREASED)	VENT (VENTOLIN)
MEDS (MEDICATIONS)	V (VERY)
NOC (NIGHT)	WHZS (WHEEZES)
NEB (NEBULIZER)	3D (3 DAYS)

One might try to compile a dictionary of such abbreviations based on a large collection of encounter notes, but any such compilation is likely to omit highly idiosyncratic usages. An encounter note dictionary would also need to be based on a large collection of encounter notes with comprehensive domain coverage, since many terms are likely to arise only in certain narrow contexts. For example, "wheezing" is frequently seen in asthma case notes but may be found only rarely in other contexts. It could be very debilitating to omit a term that appears with high frequency in a narrow subset of encounter notes when those are the encounter notes being analyzed. An encounter note dictionary would also require constant updating since medical care and treatment programs evolve over time. For example, there is now a vocabulary specific to AIDS which did not exist ten years ago. Language specific to new test technologies and new treatment plans can also spring into existence overnight. Given the shifting nature of health care practices, the vocabulary of encounter notes is probably much more fluid and transient than the vocabulary of a standard English dictionary.

In addition to the problem of medical-specific vocabulary and abbreviations, hand-written encounter note reports often contain numerous misspellings, and their transcriptions tend to include a large number of typos. These linguistic complications, while easy for a human to negotiate, further complicate the general problem of lexical recognition by machine since spelling correction algorithms rely on an online dictionary of recognizable words and their morphological variants.

Hand-written encounter notes are also problematic because they frequently contain telegraphic sentences. Encounter notes are most likely to contain full sentences when the clinician is dictating the report. When clinicians write their reports by hand, the language tends to contain many disjointed phrases with highly concise and minimal descriptions. These telegraphic texts are also more likely to contain abbreviations throughout, making it even more difficult to segment phrases correctly. Even when there are only phrases and no sentences, it is still important to segment the text into coherent fragments. Correct fragment boundaries for telegraphic text are every bit as difficult as sentence boundaries in general — possibly more so.

The following encounter note excerpts demonstrate these difficulties:

```
FLARE UP ON MEDS,MILD DIFFUSE EXP WHZ,ALMOST
CLEAR AFTER BRONK

IMPROVED CONTINUES ON SLOPHYLLIN 250 MG TID,
ALUPENT 10MG TID, WILL STOP ALUPENT

WILL NOT COMPLY WITH PEFR BUT CLEAR WITH
I:E=1:1.

CLEARED W/BRONCH X1,NO MEDS X1YR

SEEN EARLIER, CLEARED W/ 1 NEB. SENT HM ON
ALUPENT, 2 TSP. TID. AWOKE 3 A.M., SL WHEEZY
AGAIN. RR, APPROX 40. GAVE 1 TSP, 1 A.M.

DOING WELL. REV ASTHMA, MED PALN PER DR SMITH'S
9/23 ENC. MOM FOLLOWING. PF ABOVE 300 W/DECR IN
BECLOVENT. MOM TO DR SMITH TOMORROW & DISC. POSS
D/C BECLOVENT
```

Traditional dictionaries, grammars, spelling correction algorithms, and morphological routines are all unlikely to succeed with texts such as these. Encounter notes are more likely to benefit from a corpus-driven approach which enables us to construct an application-specific dictionary from representative documents.

1.3 Text Classification in the Pediatric Asthma Domain

Our first experiments in fine-grained text classification (FTC) were conducted on a corpus of encounter notes in the domain of pediatric asthma. Working in conjunction with Harvard Community Health Plan, we identified a text classification problem that was relevant to ongoing quality improvement

concerns and activities within Harvard Community Health Plan. The problem was to identify all encounter notes that recorded a specific exacerbation, and separate those from the remaining encounter notes describing routine follow-up visits and other encounters that were not motivated by an exacerbation.

An asthma exacerbation is a change in the patient's usual baseline asthma condition to a more severely ill condition. A mild or easily managed attack may or may not require immediate medical attention. A severe exacerbation can be life threatening. Since all of the encounter notes collected for pediatric asthma patients describe symptoms of asthma, it is a non-trivial problem to single out only those reports that resulted from specific exacerbation episodes. We include some excerpts from sample encounter notes in the pediatric asthma domain to illustrate the problem:

```
PROBLEMS
G100  ASTHMA
      UNABLE TO CLEAR COMPLETELY W/VENTOLIN VIA NEB,EPI,ALUPENT
      VIA NEB.BEST PEAK FLOW 150.COMFORTABLE,BUT STILL W/SOME
      WHEEZING.CONSULT DR XXXXX.

P328 M FAMILY PROBLEM
      FATHER W/PANCREATIC CANCER

THERAPIES
D151  EPINEPHRINE(ADRENALIN) 0.3CC NOW
I111  PREDNISONE 60MG STAT PER DR XXXXX
J001  BRONCHODILATOR VENTOLIN VIA NEBULIZER NOW VENTOLIN 1 1/2 TABS TID
      PER DR XXXXX
J109  THEOPHYLLINE PREP (SLO-BID) SLOBID 200MG TID PER DR XXXXX
P215  ATROVENT INHALER (IPRATROPIUM BROMIDE) PUFFS 2 QID PER DR XXXXX

SITE:   KENMORE
TYPE:   SAME DAY
```

This medical record excerpt contains free text which is typical of the telegraphic writing style often found in these records. To get a sense of the range of writing styles found in free text fields, we present the following transcribed dictation:

```
PROBLEMS
G100 #D ASTHMA
    FIRSTNAME<19> PRESENTS TODAY ON A REFERRAL BASIS FOR EVALUATION OF
    HER ASTHMA REGIMEN.  SHE IS A 11-YEAR-OLD GIRL WITH A SEVEN YEAR
    HISTORY OF ASTHMA.  HER PAST MEDICAL HISTORY INCLUDES ONE
    HOSPITALIZATION, NO INTENSIVE CARE UNIT ADMISSIONS, SEVERAL
    EMERGENCY ROOM VISITS, THE MOST RECENT BEING TWO MONTHS AGO WHICH
    WAS THE FIRST ONE SINCE 1986.  HER PRESENT ASTHMA REGIMEN IS
    VENTOLIN ONE OR TWO PUFFS BY METERED DOSE INHALER OCCASIONALLY
    WITH AN AEROCHAMBER ON A PRN BASIS.  THERE IS SOME AMBIGUITY
    ABOUT HOW MUCH SHE IS USING.  HER INHALERS ARE LASTING ONLY THREE
    WEEKS SAYS MOM BUT THEY FIGURE THAT SHE USES ONE OR TWO PUFFS
    ABOUT EIGHT TIMES PER WEEK.  HER SYMPTOMS ARE MORE PRONOUNCED AT
    NIGHT THEN IN THE DAYTIME.  SHE HAS NIGHTTIME AWAKENING WITH
    COUGH MORE THEN FOUR TIMES A WEEK AND SHE IS INTOLERANT OF EVEN
    MILD AEROBIC EXERCISE.  SHE CANNOT CLIMB THREE FLIGHTS OF STAIRS
    WITHOUT BECOMING WINDED.  FAMILY HISTORY IS REMARKABLE FOR
    ASTHMA IN FATHER AND ENVIRONMENTAL HISTORY CONTRIBUTES TWO
    PARENTS WHO SMOKE CIGARETTES IN THE HOUSE.  STUFFED ANIMALS ON THE
```

BED AND NON-ENCASED BEDDING.

PHYSICAL EXAMINATION: TODAY, REMARKABLE FOR SOME MILD DIFFUSE POLYPHONIC WHEEZING. HER I TO E RATIO WAS ONE TO ONE. THERE WERE NO RETRACTIONS NOR USE OF ACCESSORY MUSCLES OF RESPIRATION. HER THORACIC AND ABDOMINAL RESPIRATORY EXCURSIONS WERE FULL AND EQUAL. PEAK EXPIRATORY FLOW RATE WAS 280 LITERS PER MINUTE WITH A PREDICTED OF 340.

I HAVE MADE THE FOLLOWING RECOMMENDATIONS. I HAVE ASKED MOTHER TO SEE THAT SHE AND FATHER BOTH CONTACT THEIR INTERNIST FOR HELP IN SMOKING CESSATION. IN ADDITION, I HAVE TOLD THEM THAT THEY SHOULD NOT PERMIT THEMSELVES TO SMOKE IN THE HOUSE. I THINK IT IS CLEAR AT THIS POINT THAT FIRSTNAME<19>'S ASTHMA IS ONGOING. I HAVE STARTED HER ON INTAL TWO PUFFS FOUR TIMES A DAY, THREE AT A MINIMUM BY AEROCHAMBER SPACER. SHE HAS BEEN INSTRUCTED IN THE USE OF THE SPACER AND INSTRUCTED TO ALWAYS USE THE SPACER. IN ADDITION, FOR THE NEXT TWO WEEKS SHE WILL USE HER VENTOLIN METERED DOSE INHALER TWO PUFFS PRECEDING THE INTAL. SHE WILL ALSO MEASURE HER PEAK FLOW ON A TWICE A DAY BASIS BEFORE INHALERS AND GRAPH THESE OUT. I WILL SEE HER IN TWO TO FOUR WEEKS AND HOPEFULLY ESTABLISH A COLOR CODED SELF-MANAGEMENT SCHEME. AT THIS POINT, I HAVE ALSO ASKED THEM TO PURCHASE MATTRESS ENCASEMENTS AND INSTRUCTED THEM ON THE IMPORTANCE OF REMOVING STUFFED ANIMALS FROM THE BED AND WASHING THE BEDDING, PREFERABLY WITH HOT WATER AND HOT DRYER.

The shorter more telegraphic write-up presented above is difficult for a lay person to classify, but it does describe an exacerbation. The longer transcription does not describe a recent asthma exacerbation in spite of the fact that a hospitalization is mentioned and an exacerbation was known to have taken place two months ago.

Discriminations of this type can be effectively handled by information extraction systems [Lehnert and Sundheim 1991, Riloff and Lehnert 1994]. Unfortunately, information extraction systems need to be customized for each new application, and these customization efforts are both labor-intensive and expensive.

Our research has been aimed at the establishment of FTC algorithms that reside somewhere in between the traditionally robust IR technologies and the more exacting and computationally intensive information extraction technologies. If we only want to classify texts, we shouldn't need to build a complete information extraction system designed to pull specific pieces of information out of each text. But we might be able to use some of the ideas associated with information extraction in order to achieve greater precision on FTC tasks. It is important to develop a generic technology which can be readily ported across applications without intensive application-specific knowledge engineering or manual coding efforts. To meet this challenge, we must capture the domain expertise needed to classify these texts correctly in a fully-automated or almost fully-automated fashion.

2. The FIGLEAF Text Classification System

FIGLEAF tackles the problem of fine-grained text classification as a knowledge acquisition problem. While standard information retrieval systems are designed to classify documents using generic algorithms that operate well on a wide range of user queries and information needs, FIGLEAF is designed to exploit application-specific text signatures based on representative text corpora. The tools for extracting application-specific information are completely generic and can be used to construct any number of text classification systems. But a new text classification system must be constructed for each new application in order to take full advantage of the text signatures that operate in each new domain.

Because FIGLEAF constructs customized systems for each new application, we can separate all FIGLEAF operations into two phases: the training phase and the runtime phase. The training phase must be completed before any runtime processing can take place. We will briefly review FIGLEAF's basic operations, and then present a more detailed discussion of each one.

FIGLEAF During Training

[1] Sorting and Dictionary Creation: A domain expert (e.g. a physician) sorts a set of training documents into relevant and irrelevant text subsets. In the case of pediatric asthma, the relevant texts describe specific exacerbation episodes and the irrelevant texts do not. Any phrases in the document which helped the domain expert reach a relevance decision are marked using a point-and-click text marking interface. All highlighted character strings highlighted within the interface become dictionary entries.

[2] Feature Generation: All dictionary entries collected by the text marking interface are mapped to a feature vector using a lexical hash function. The feature list generated by the hash function is called the *master dictionary*.

[3] Feature Extraction and Bigram Generation: Each document in the training corpus is analyzed by a dictionary lookup routine using the master dictionary. Information-rich phrases consisting of one or more adjacent dictionary entries are extracted from each training text. Each such phrase is then broken up into single words and word bigrams in order to create training instances for an inductive decision tree.

[4] Decision Tree Construction: For each document in both the relevant and irrelevant training sets, all the extracted words and word bigrams are passed to the ID3 decision tree algorithm [Quinlan 86] as positive training instances (if the text is relevant) or negative training instances (if the text is irrelevant). ID3 constructs a tree using the training instances provided. Once a d-tree (decision tree) has been built, pruning techniques are applied to achieve optimal performance.

FIGLEAF at Runtime

[1] Feature Extraction: Using the master dictionary acquired during training, each new text is examined for known dictionary entries, and key

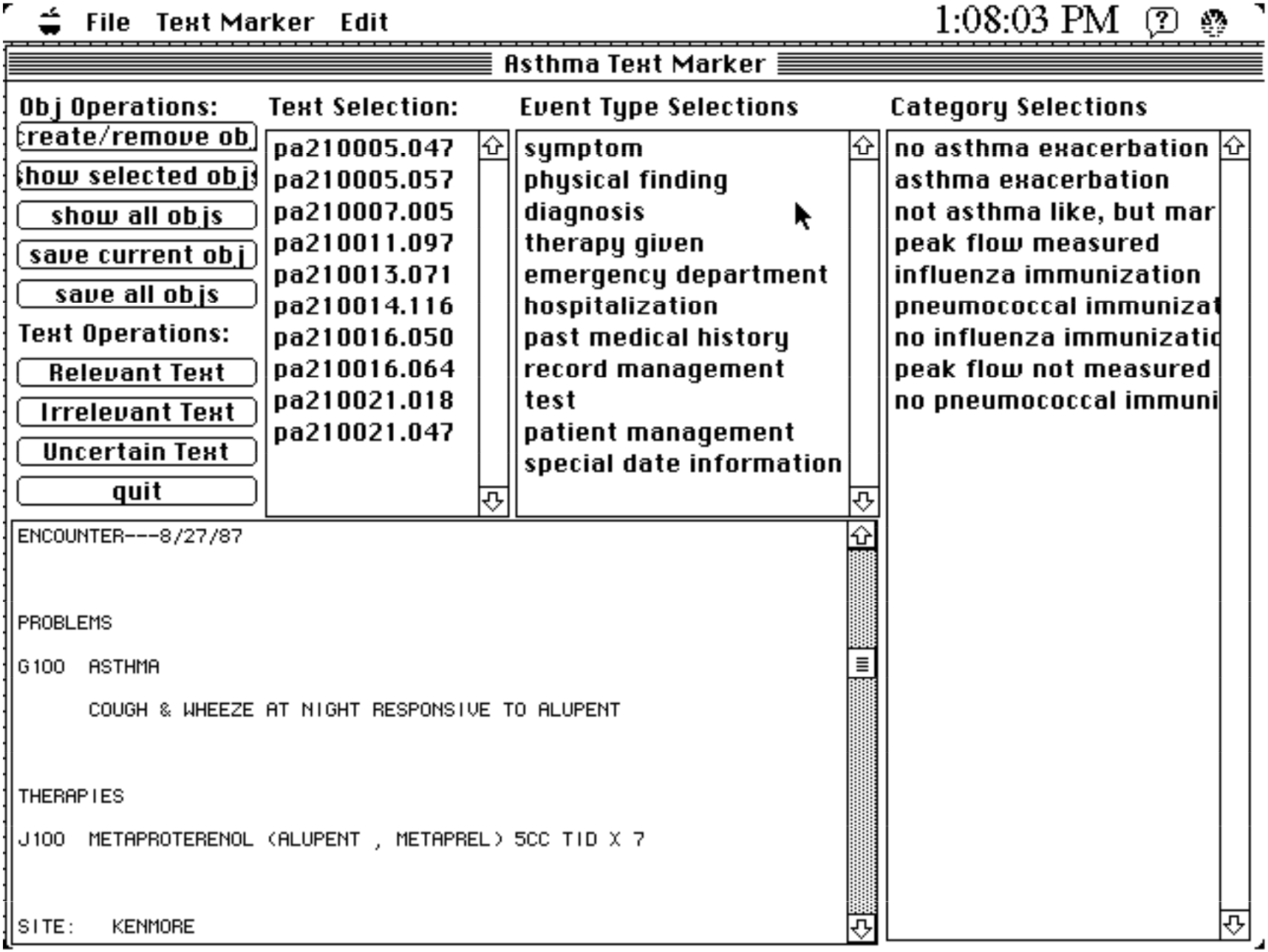

Figure 1: A screen snapshot of the text marking interface

phrases are extracted. The resulting phrases are broken up into features which can be processed by the d-tree.

[2] Rank Listing the Documents: The d-tree returns confidence levels associated with the classification of each feature vector. These numbers allow us to rank list the input documents according to the strength of their relevancy judgements.

We will now describe each of these basic steps in greater detail.

2.1 Dictionary Construction Using Representative Text Corpora

Application-specific dictionary construction is the first step in the construction of an FTC system using FIGLEAF. We do not need a comprehensive dictionary that contains every word in every training document, but we do need to recognize important keywords that contribute to the relevancy judgements made by human domain experts. The application-specific dictionary created by FIGLEAF is basically a collection of important keywords that we expect to see over and over again.

All knowledge engineering occurs within our text marking interface, a graphical user interface implemented in MCL 2.0 for the Macintosh (see Figure 1). The interface is given a collection of training texts, and each one is displayed by the interface for examination by a domain expert. The domain expert is asked to process each text in two ways.

First, a judgement must be recorded concerning the relevance or irrelevance of a text. We note that these judgements are not always straightforward and may be subject to coding errors. Two experts may disagree on the relevancy of any given document, and the same expert may evaluate the same document differently at different times. Inconsistencies in the training data will impair system performance, so it is important to take coding errors into consideration when evaluating FIGLEAF's test classification performance[1]. Even though computers are not subject to the fatigue factors that contribute to human coding errors, we still need good training data in order to obtain the best possible machine performance. If the training data is poor, FIGLEAF cannot be expected to operate as well as it might otherwise.

Next, the domain expert is also asked to mark any phrases in the training document that contributed to each relevancy judgement. Unlike the knowledge engineering that accompanies the design of expert systems, no discussion or explanation of the expert's reasoning is needed: we only need to collect the important phrases that contributed in some way to the decision process. The interface collects all highlighted text fragments and saves these annotations in memory. Some highlighted strings from the pediatric asthma testbed include phrases such as:

---

[1] It may not be reasonable to expect higher precision from a computer than we can obtain from humans. When multiple coders can't classify documents consistently, it may be difficult to assess correctness for the documents in question, and absolute performance guidelines suffer accordingly.

Started wheezing Mo.
seen at children's E.W., unable to speak
rattles & whzs no exp whz

Document sorting and text marking could be handled manually, but the use of a point-and-click interface is faster than manual notation, and allows us to make the most efficient use of a domain expert's time. The physician who marked up the pediatric asthma testbed estimates that it took about 10 seconds to process an irrelevant document (tagging it as irrelevant) and 120 seconds to process a relevant document (tagging it as relevant and marking all relevant phrases). A testbed containing 250 relevant and 250 irrelevant documents would therefore require a minimum of 9 hours on the part of a domain expert. This estimate presumes the existence of 500 usable documents evenly divided between relevant and irrelevant — an ideal but less than realistic starting point for testbed preparation — and does not take into consideration time for breaks, second thoughts, or corrections of any kind. Depending on the difficulty of the classification task and the learning curve of the domain expert, a more realistic estimate for the preparation of a testbed of 500 documents may be closer to 20 hours.

In applications where the target documents are homogeneous in structure, no additional information is taken from the text marking interface. But the encounter notes collected by HCHP are structured with subfields that contain different types of information, some of which are relevant to the text classification task and some of which are not.

For example, every document starts with a serial number followed by a header, a problems field, a therapies field, site and type fields. Additional optional fields may contain information describing a discharge diagnosis and date of discharge. The record also includes personal patient details such as age, address, benefit level, and some administrative codes[2]. We may find objective data such as height, weight, and temperature. A date field records the date of the patient encounter. The site field specifies the encounter location, and the type field describes the type of the encounter. For example, "same day", "telephone", "hospital arrangement", or "urgent care" are all used to describe encounter types.

Most of the free text in our HCHP corpus is found in the problems section and the therapies section. Here we can find short, abbreviated sentences or extensive amounts of text transcribed from dictation. For administrative reasons, every paragraph in these sections is preceded by an HCHP code tag. The most common problem code in our pediatric asthma testbed is a code for asthma, but other codes may appear instead. The codes used in the therapies section refer to specific medications, with usage instructions recorded in text. For example,

```
take 1 tsp. PO ID X5-7dys   x2   ref ventolin
5mg/ml   1   1/4 dropperful/   3cc   ns   x2.
```

It is important to note that the classification system underlying these various codes is not sufficient, by itself, to separate exacerbations from non-exacerbations. If the code taxonomy had been defined in a way that would

[2] All personal identifiers were removed from our training and testing corpora by HCHP to preserve patient privacy.

make this particular discrimination, our work would have already been done for us by the individuals generating the codes.

For applications where the target documents are structured into fixed segments, it makes sense to monitor the text marking behavior of a domain expert, and design text filters which isolate those segments of the training texts where key words are highlighted during text marking. If a text segment is never marked by a domain expert throughout the course of the entire development corpus, we assume that the contents of that field are irrelevant for our purposes and will only add noise to our training data.

It is relatively easy to design manual text filters which reduce noise in this manner. In the pediatric asthma testbed, all text which is not preceded by a problem or therapy code is removed from our training corpus. In this case, these fields harbor most of the open-ended text, but our filtering decisions are based only on data given to us from text annotations. We would preserve heavily codified fields if the domain expert indicated that information in these fields was useful in making a relevancy judgement.

2.2 Dictionary Encoding with Lexical Signatures

For most NLP systems, lexical representations are typically driven by generic dictionary entries. Words are thought to consist of a root form or "stem" plus a morphological variant (which might be null). Because there are some general rules which apply to plural noun forms and verb tenses, it is computationally reasonable to encode canonical stems and then use recognition routines to map lexical items found in an input text into recognizable canonical forms.

Unfortunately, our encounter notes contain highly idiosyncratic vocabulary, so it is not a good idea to depend on a generic dictionary which may not be able to provide the recognition capabilities we need. In order to obtain adequate dictionary coverage, we construct an application-specific dictionary based on phrases captured in the text marking interface. We can expect this dictionary to be relatively small (no more than 1000-2000 entries including all the morphological variants), so there is no compelling need to invoke the stemming routines used in most NLP applications. At the same time, it is probably advantageous to recognize morphological variants in order to maximize the impact of small training sets.

FIGLEAF can recognize many common morphological variants without any knowledge of word stems by using a lexical hash function. The hash function converts lexical items into symbolic codes called *lexical signatures* (or L-SIG codes) without the use of a dictionary. Each input word is checked for the presence of certain characters and the number of instances of those characters. Some vowels are ignored unless they occur at the beginning of the word, as well as consonants found in common noun and verb suffixes (e.g. "s", "d", "n" "g" are all ignored). The algorithm also checks for doubled consonants so that a past tense verb such as "stopped" will hash to the same L-SIG as "stop."

Dictionary construction is accomplished by computing the L-SIG codes for each lexical item highlighted in the text marking interface. A list of these codes is saved along with each lexical item that was mapped to the code. Some codes are associated with a single lexical item and others pick up a number of

related items. Figure 2 shows some L-SIG codes and the lexical items found in the pediatric asthma testbed that map to those codes.

```
W55-34-98 |WHEEZE/COUGH|WHEEZE&COUGH
I7-2-195 |INFREQUENT
Z1-1-35 |13TH
A2-1-3 |ARE|AIR
W143-1-7 |WHEEXING
A2-42-1 |ALUP|ALUPEN
F35-1-3 |FRCD
T1-3-5 |TYPE:|TSP
R1-14-15 |RESULT|RESULTS
V1-5-110 |VOMITING|VOMITED
P1-21-12 |PREDNISOLONE|PROLONGED
C30-7-6 |CONSIDERABLE|CONSIDERABLY
A2-5-5 |ADMITTED|ASMITTED
D1-1-2 |DO|DOING|DOSE|DOSES
W11-11-1 |WK|WEEKEND|WKS|WKS.|WKS,|WEEKS
W11-17-7 |WHEEZE|WHZ|WHEEZES|WHEEZING|WHEEZ,|WHEEZING,|WHZ,|WHEEZES,
E1-1-1 |E|E:I|END
&1-1-1 |&
I1-1-1 |I|IN|I&G|I/E|IS|I111|I114|I345|I&E|I+E
W11-1-35 |WITH
T1-1-175 |TIGHT
C10-7-3 |CLEAR|CLEARED|CLEAR,|CLEARING|CLEARING,|CLEARS
S2-2-1 |SUNDAY
A4-5-35 |ASTHMA|ASHTMA,
```

Figure 2: Some sample L-SIG codes and associated lexical items

As these examples show, standard morphological variants are collapsed to the same L-SIG when they are encountered during training (see C10-7-3). But L-SIG representations are also capable of recognizing some common typos, misspellings, and impromptu abbreviations as well. For example, A4-5-35 picks up "ashtma" as a variant on "asthma" because the hash function is largely insensitive to the order of characters. This allows us to recognize a common class of typos that result from the transposition of two characters. In a similar way, some misspellings due to incorrect vowel substitutions will be mapped to the same L-SIG codes because the hash function ignores specific vowels unless they appear as the first character in the word. We also see how common and idiosyncratic abbreviations can be picked up when the abbreviation is derived by dropping out vowels (see W11-11-1 and W11-17-7).

As effective as the L-SIG codes are, they are not perfect. The hash function can also collapse lexical items that are completely unrelated (see A2-1-3, P1-21-12, D1-1-2 and I1-1-1). This may weaken FIGLEAF's ability to recognize significant phrases accurately, but other aspects of FIGLEAF's design operate to lessen the effects of L-SIG confusion. We will see how FIGLEAF can compensate for L-SIG conflation errors in the next section.

We do not have extensive experience with L-SIG codes, but they appear to be operating adequately on the pediatric asthma testbed. We note that some adjustments can be made to the hash function if we determine that a class of conflation errors are significant in a given application. For example, I1-1-1 conflates the HCHP codes I111, I114, and I345 because the current version of the hash function ignores all numbers. If we determined that these HCHP

codes were useful discriminating features, we could easily alter the hash function to differentiate the HCHP codes. The master dictionary for our pediatric asthma testbed contains 563 L-SIG codes derived from 630 marked training texts. Most of these L-SIG codes are associated with only a single lexical item.

It would be ideal to use L-SIG codes that require no tuning as we port FIGLEAF from one application to another. However, the hash code function can be easily monitored by a quick visual inspection of the L-SIG dictionary, and simple modifications can be made to sensitize the hash code function if it is found to be producing some undesirable I/O behavior on some identifiable class of lexical items. No such manual tuning was used for any of the experiments reported here.

2.3 Phrasal Bigrams for Training and Runtime Algorithms

The ID3 inductive decision algorithm can be trained to make binary classification decisions when confronted with novel test data. In order to train a decision tree (d-tree), it is first necessary to encode training and test data in terms of a finite feature vector. The design of this feature vector is critical to the success of trainable d-trees. Predictive features that correlate strongly with classification decisions are preferable to features which operate as sources of noise. But it is not always easy to know which features are predictive and which are not.

FIGLEAF automates the design of a feature vector by basing all vector representations on its master dictionary. If the master dictionary contains 500 L-SIG codes, then FIGLEAF will use a vector containing 500 binary features. Single words and short phrases will trip only a few of these features, resulting in very large sparse vectors[3]. Note that when multiple phrases are represented by a single vector, a feature vector cannot preserve positional information associated with disjoint words or phrases.

Two different methods for mapping texts into d-tree vector representations have been investigated in the context of FIGLEAF. The first is the *voting method* for representing phrases. The voting method maps each phrase to a single ID3 training instance.[4]

Under the voting method, multiple phrases are mapped to multiple feature vectors and the d-tree produces a classification for each feature vector. For most documents, some phrases might be classified by the d-tree as positive (exacerbation) instances while other phrases might be classified as negative (non-exacerbation) instances. To reconcile such disagreements, each phrase is granted a single vote and different voting thresholds were tested.

Although resolution by voting sounds workable at first glance, this approach has an inherent flaw. In particular, the voting method is overly pluralistic.

---

[3] A special version of ID3 has been designed to manipulate such vectors efficiently [Soderland and Lehnert 1994].

[4] We will speak loosely of mapping words and phrases into training instances or feature vectors, but to be accurate, all words and phrases are first mapped into L-SIG codes and then the L-SIG codes are used as vector elements.

The one-instance/one-vote principle assumes a uniform reliability across all text fragments, but some fragments are clearly more reliable than others. Consider, for example, a document which contains the two phrases,

- **asthma**
- **acute bronchitis**

The phrase "asthma" carries no useful information gain, since nearly every document in the training corpus contains it. Even so, the phrase will be classified one way or another by the d-tree. On the other hand, "acute bronchitis" refers to an asthma exacerbation and this phrase is strongly associated with reliable text classifications. But the one-instance/one-vote approach treats all phrases as equals regardless of their reliability. ID3 has no a priori method of assessing the reliability of its classification decisions.

The *bigram method* avoids voting problems by converting each document into a single training/testing instance. This is done by concatenating all keyword fragments into one long segment. Because word order is more important for longer fragments than for shorter fragments, we can preserve at least some word order information by capturing word adjacencies in word pairs. In the bigram method the list of all text fragments recognized by the master dictionary is converted into a list of all single keywords and all adjacent keyword pairs.

Suppose a document contains the sentence, "Her Mom said that she is not wheezing or coughing at night". Assume *not*, *wheezing*, *coughing*, *at*, and *night* are keywords recognized by the master dictionary. If no other keywords are extracted from any other sentences, then the complete representation for this text will be *not wheezing coughing at night* and ID-3 will receive the instance:

```
not/wheezing/coughing/at/night/  ...
not-wheezing/wheezing-coughing/coughing-at/at-night
```

Bigrams preserve word order at a local level and the bigram method requires ID3 to operate on larger feature vectors. According to some preliminary investigations, it appears that the bigram method yielded better performance than the voting method. We have therefore abandoned the voting method in all subsequent experiments.

Note that a trigram method is also possible, preserving a larger window of adjacency information. However, in moving from a bigram to a trigram representation, the number of possible single, double, and triple-term combinations probably overwhelms any expected information gain. The bigram method appears to be providing us with an adequate and practical representation on which to build d-trees for text classification.

One limitation of ID3 d-trees is that each tree node tests for only one feature at a time (see section 2.4 for more on ID3). When we can identify sets of conceptually related features with the assistance of a domain expert, it makes sense to consider an entire set of related features at once. We found that combining several individual terms or bigrams into *meta-features* improved generalizations by the d-trees. The instance encoding can include several meta-features in addition to the usual single term and bigram features.

In section 3.2.3 we will describe an experiment using an extended vector representation including meta-features. A particularly useful meta-feature for the asthma domain identifies medications administered for acute asthma exacerbations together with methods for administering the medication ("nebulizer", "intravenous", etc.) and phrases associated with medication effects ("clear after", "clear with", "better after", etc.). The meta-feature MEDICATION assumes a value equal to the number of medication instances found in a text. Two or more medication instances in a text provides very strong evidence that the text describes an exacerbation: a single medicine instance is less reliable but nevertheless suggestive of an exacerbation.

Meta-features can also be derived statistically from analysis of the training documents. Features that correlate strongly with relevant documents can be combined into a relevance meta-feature, and those correlated with irrelevant documents into an irrelevance meta-feature. Two thresholds are used to select candidate features, (1) frequency of occurrence in training documents, and (2) relevancy-rate (the proportion of occurrence in relevant documents). Several versions of the same meta-feature can be defined which differ only by their thresholds. If all the variants are added to the vector representation, ID3 will identify the most predictive variant as it builds a decision tree. In this way, ID3 can be used to find the best thresholds for statistically-oriented meta-features.

2.4 Applying Decision Trees to Text Classification Problems

The categorization decision is carried out by an ID3 decision tree. A major task for FIGLEAF is the proper construction of these d-trees for each new domain and FTC application. In this section we will describe the operation of ID3 as it is used by FIGLEAF.

2.4.1 Inductive Decision Trees

ID3 is a supervised machine learning algorithm that automatically derives a decision tree from a set of training instances once each instance is tagged with its correct classification. A fully trained decision tree can then be used to classify previously unseen instances from a test set.

In building a decision tree, ID3 tabulates how often each feature is associated with positive instances and negative instances in the training set. When FIGLEAF uses the bigram encoding method, the feature set consists of all L-SIG codes plus any pairs of adjacent L-SIG codes found in a training instance. ID3 uses an information gain metric to select a feature that most effectively partitions the training instances [Quinlan 1986]. This is often a feature that may not seem intuitively obvious.

For example in many of our pediatric asthma decision trees, ID3 selects an L-SIG code generated by numbers as a root node. Documents that include numbers were relevant 60% of the time, and most of the numbers referred to prescriptions or medical test results. Texts that did not include numbers were relevant only 24% of the time. This turned out to be the most useful feature for separating positive and negative instances in the first partition. ID3 then recursively selects features to further partition each partition.

The implementation of ID3 used by FIGLEAF returns a confidence value rather than simply making a binary decision whether a new instance is positive or negative. Each node in the tree represents a partition of positive and negative training instances. The proportion of positive training instances to negative training instances provides us with useful information concerning the reliability of each partition.

2.4.2 FIGLEAF Decision Trees and Text Representations

Once FIGLEAF has trained a d-tree using ID3, it is ready to process new encounter notes and produce classification decisions based on the d-tree output. Each new encounter note is represented by a feature vector based on the master dictionary of L-SIG codes and L-SIG bigrams as previously described. FIGLEAF will traverse the d-tree, taking various branches based on the feature vector representing the test instance, until it halts at a terminal node. The proportion of positive instances at this node gives a confidence value for the test instance.

Consider the d-tree shown in Figure 3, which has been pruned to eliminate branches from nodes with partition size below 200 instances. The full tree would have 109 internal nodes, 110 leaf nodes and maximum depth of 36. The numbers in parentheses at each node indicate the number of training instances and the proportion of positive instances to all instances encountered during training. A total of 630 training instances were used to construct this tree, of which 47% were positive. The branch where the feature <number> has value "false" leads to a node with 233 instances, 24% of which are positive. If the next feature, BRONKOSOL, is also false the partition is reduced to 225 instances, only 21% positive.

A test document that contains a number, does not contain the term MONTH, and contains the term EPINEPHRINE, will arrive at a node representing 29 positive training instances and will be given a confidence of 1.00. A text that does not have <number>, does not have BRONKOSOL, but has a term such as 10MG will reach a node representing 25 training instances of which 60% are positive. This is a terminal node at this level of pruning and a confidence of 0.60 is returned. With no pruning, further tests would be made to partition those 25 instances until a leaf node is reached with only positive or only negative instances.

2.4.3 Tree Pruning Heuristics

Pruning can make a significant difference in the final performance of a decision tree. Near the leaves of the decision tree, partitions become small and classifications become less reliable due to a problem known as overfitting the data. The feature selected to discriminate among a small number of instances will often reflect accidental characteristics of the data and have little predictive power. If a partition has nine positive instances and only one negative, any feature particular to that one negative instance might be selected and that feature can easily turn out to be pure noise.

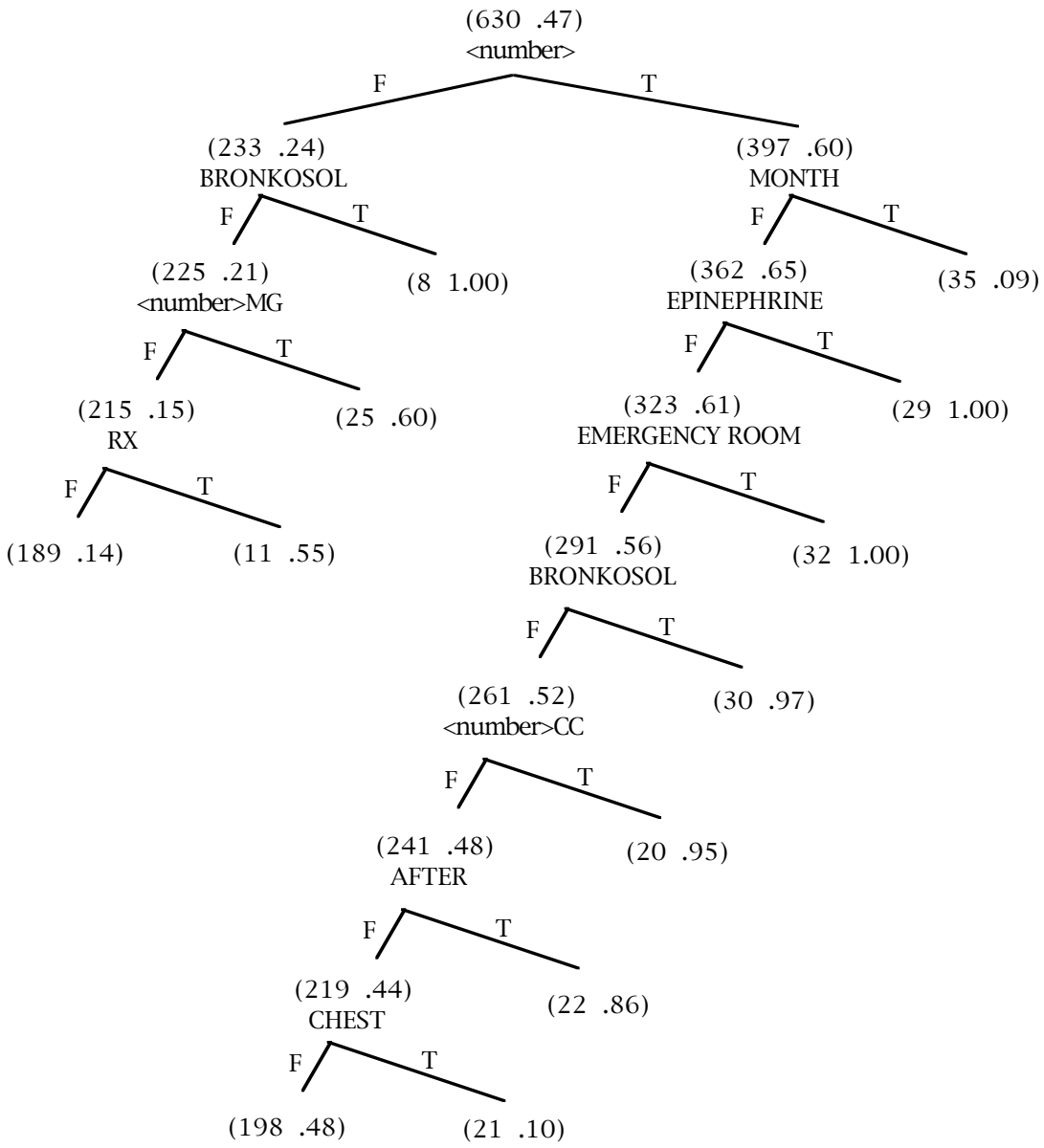

Figure 3: A pruned decision tree for pediatric asthma

We avoid the problem of overfitting by pruning the tree using two thresholds: a *pruning threshold* and a *branching threshold*. To understand the utility of these two thresholds, consider the example in Figure 3. The tree has two main branches coming from the root, which divides the training instances into two nearly equal partitions. Each main branch has a series of tests, each for a feature that is typically true for only a few instances. This splits off a few instances and leaves the main branch with a lightly smaller partition and a slightly different percentage of positive instances. For the pediatric asthma d-trees we found that the instances that are split off tend to be positive instances, gradually lowering the percentage of positive instances in the main branch.

A pruning level of 200 cuts off the tail of each main branch. The training instances that are cut out are those still on the main branch after a long list of tests, which means they had false feature values for each test after the root node. After a number of these tests, there are only a small percentage of positive instances left in the main branches. A point is reached where further testing doesn't help and the remaining instances are treated as indistinguishable.

A branching level of 20 has a very different effect. Suppose we traverse the tree until a feature is tested that has the value true. When that positive branch is taken, the confidence level can jump dramatically. For example, if the branch leads to a partition where all the instances are positive, the instance will jump to 100% confidence. In most cases, branches that result in high confidence lower in the tree lead to partitions with less than 20 instances. If we set the branching threshold to 20, these branches will never be considered. The confidence level of the main branch is used instead. In this way, the highest confidences go to instances that terminated higher in the tree, where high confidence levels are more meaningful.

Pruning and branching thresholds are empirically set for each tree using ten-fold cross validations. The training set is divided into ten subsets, and each 10% training subset is tested by a tree built from the remaining 90% of the training instances. A full ten-fold cross validation is done for a particular setting of pruning and branching thresholds to evaluate this setting. We evaluated performance for a range of settings spread evenly throughout the space of possible pruning and branching thresholds to find optimal tree pruning.

To evaluate our d-trees, we assess performance on the basis of recall and precision. Recall and precision are two independent metrics traditionally used to assess the performance of IR systems. Recall refers to the percentage of relevant documents that are classified as relevant, and precision refers to the percentage of documents classified as relevant which were correctly classified. In practice, most systems exhibit a trade-off between recall and precision: an improvement along one dimension typically forces a drop in performance along the other dimension. Depending on the target application, it may be desirable to buy high precision at the price of recall, or one may prefer to optimize recall and settle for low precision.

Settings that maximize overall average precision may not be the same as settings that maximize precision at low recall or at high recall. We can manipulate the pruning thresholds by selecting evaluation functions that emphasize different performance strengths. Based on our experiments with the pediatric asthma testbed, it appears that moderate pruning and branching levels yield the best precision at high recall levels. The best precision at low recall levels tends to come from large pruning or large branching settings.

We hope that the settings established during our cross-validation runs give us optimal or nearly optimal performance on our training sets. If the training sets are representative of runtime input, we can trust these settings to give us the best possible performance during runtime discriminations. However, each time we retrain a new tree on a new training set, we have to establish new pruning thresholds in order to use the tree effectively. An empirical search over the space of all possible threshold settings is needed to arrive at these settings.

2.4.4 Document Ranking

Confidence levels returned by the d-tree can be used to arrange a set of test documents into a list that ranks them from the most relevant to the least relevant. This ranking will incorporate some nondeterminacy since several test instances may terminate at the same node in the d-tree and therefore have identical confidence levels, particularly in a pruned tree.

Figure 4 shows how a test set of 141 instances were ranked by the tree shown in Figure 3 with a pruning threshold of 200 and a branching threshold of 20. The first four instances were true for <number>, false for MONTH, and true for EPINEPHRINE. As the second entry in the first line indicates, each of these four test set instances were correctly classified as positive instances. The next eight instances were each true for <number>, false for MONTH, false for EPINEPHRINE, and true for the bigram EMERGENCY ROOM. Although all the instances encountered during training with this feature profile had been marked as positive (32 documents according to Figure 3), one of eight such documents encountered in the test set turned out to be irrelevant.

Looking at lower confidence levels in Figure 4, we can see that a confidence level of .48 characterizes documents that are as likely to be irrelevant as relevant, and confidence levels below 10% are largely reliable as an indication of irrelevance.

#inst	#pos	#neg	conf.	- - - - - - - - - - - features - - - - - - - - - - -
4	4	0	1.00	F1 ~F2 F3
8	7	1	1.00	F1 ~F2 ~F3 F4
6	6	0	.97	F1 ~F2 ~F3 ~F4 F5
10	8	2	.95	F1 ~F2 ~F3 ~F4 ~F5 F6
4	3	1	.86	F1 ~F2 ~F3 ~F4 ~F5 ~F6 F7
6	4	2	.60	~F1 ~F5 F8
51	25	26	.48	F1 ~F2 ~F3 ~F4 ~F5 ~F6 ~F7 ~F9
1	0	1	.24	~F1 F5
3	2	1	.15	~F1 ~F5 ~F8 F10
32	6	26	.14	~F1 ~F5 ~F8 ~F10
1	0	1	.10	F1 ~F2 ~F3 ~F4 ~F5 ~F6 ~F7 F9
15	1	14	.09	F1 F2

F1 = <number>
F2 = MONTH
F3 = EPINEPHRINE
F4 = EMERGENCY ROOM
F5 = BRONKOSOL
F6 = <number>CC
F7 = AFTER
F8 = <number>MG
F9 = CHEST
F10 = RX

Figure 4: Confidence Levels in a Ranked List of Test Documents

The performance of this tree at these pruning thresholds can be graphed as a recall-precision (R/P) curve. Of the first *n* instances in the ranked list, recall is determined by the percentage of positive instances from the entire test set, while precision is determined by the percentage of those *n* instances that are positive. In this example the first 12 documents include 11 relevant out of a total of 66 relevant documents in the entire test set. This gives a recall-

precision point of 16.7 recall and 91.7 precision. Of the first 18 documents 17 are relevant, producing a recall-precision point of 25.8 recall and 94.4 precision.

After computing the recall-precision points for each set of ranked instances, linear interpolation is used to generate precision values at eleven evenly spaced recall levels: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

3. Experimental Results

In order to evaluate the performance of FIGLEAF after d-tree training and pruning, we have conducted a number of experiments on blind test sets taken from the pediatric asthma testbed. To keep our test results consistent with the recall/precision curves used in IR, we found it necessary to rank list all the test documents from the "most relevant" to the "least relevant" as described in section 2.4.4.

Any system that can rank list a set of test documents can be viewed in terms of an R/P (recall/precision) curve based on that rank list. To establish the curve, one need only compute the recall and precision percentages associated with the top N-ranked documents as N ranges from 1 to the size of the test set (see section 2.4.4). R/P curves are usually sampled at 10% intervals along the test set ranking in order to establish the general shape and position of the R/P tradeoff. R/P curves are then graphed by representing recall percentage along the *x*-axis and precision percentages along the *y*-axis. A system that is making completely random guesses would be expected to produce a flat curve with precision equal to the percentage of relevant documents present in the test set (the precision baseline). A system that has correctly ranked all relevant documents above all irrelevant documents will maintain 100% precision for all values of x. Most systems exhibit their highest levels of precision for low values of x, and then drop down to baseline precision levels for high values of x. A rough indication of overall performance can be obtained by computing the average precision value.

All training and testing was conducted on four DECstations running Unix. FIGLEAF was implemented in Unix Awk and Allegro Common Lisp. The text marking interface runs on a Mac II and was implemented with the MCL 2.0 Toolkit. FIGLEAF was designed and implemented at the Natural Language Processing Laboratory of the University of Massachusetts.

3.1 Training and Testing

The pediatric asthma testbed consists of 630 encounter notes that were classified and marked by a physician, along with 141 additional documents that were classified by the same physician but not marked. It is important to note that this testbed consists of encounter notes that are narrowly characterized by descriptions of pediatric asthma patients. They do not all describe exacerbations, but those that don't are "near misses" for asthma exacerbation. Appropriate applications for FTC will be characterized by input documents that are narrowly circumscribed in a similar manner. Coarse-grained retrieval algorithms are generally needed to generate good "ballpark" document streams for FTC systems. L-SIG encodings and d-tree sensitivities are predicated on assumptions about reduced vocabulary requirements and predictable signatures in both relevant and irrelevant documents. If the input documents

going into an FTC system are too open-ended, we would not expect to see strong performance even if the system were trained on large amounts of representative text samples. But as long as an FTC system can be pipe-lined to pick up where a coarse-grained IR system leaves off, then we can hope to improve on the precision of the first-pass IR system which must make its first cut on document collections that are unconstrained.

To conduct a single test run on the full testbed we shuffled the marked and unmarked texts and then selected 141 random documents for the test set. The remaining 630 documents were designated as the training set. We ensured that a similar ratio of relevant and irrelevant documents was present in both the training and test sets. The training set contained 294 relevant documents (47%) and the test set contained 65 relevant documents (46%).

For each test run, we (1) construct a master dictionary on the basis of the training corpus, (2) train a single d-tree, and (3) use 10-fold cross validation to establish pruning thresholds. In fact, we establish one set of thresholds for optimal precision at lower recall levels (the 10-30 thresholds), a second set of thresholds for optimal precision at higher recall levels (the 80-100 thresholds), and a third set of thresholds for optimal precision across all recall levels. Once the pruning thresholds have been found, we are ready to run FIGLEAF on the test set. In order to keep the test set completely blind, we conduct all cross-validation runs on the training set only.

3.1.1 Effects of Increased Training

Our first experiment was designed to track FIGLEAF's performance as a function of increased amounts of training. To obtain overall average precision values in response to different amounts of training, we first created four successively smaller training sets which were nested subsets of the full 630 document training set. For each of these different training sets we constructed a new master dictionary, trained a new d-tree, and established three new sets of pruning thresholds as described above.

Figure 5 shows the overall precision averages that were obtained during cross-validation as the training set grew toward the target training set of 630 documents.

	------ number of training docs ------				
	126	252	378	504	630
Best Overall Precision	70.78	71.91	74.61	77.25	78.11
Best Precision at 10-30 rec	79.78	88.11	86.77	91.69	91.02
Best Precision at 80-100 rec	52.14	55.40	57.10	59.29	59.04

Figure 5: Average precision during cross-validation

As these precision averages show, performance improved steadily up to the 80% training set, but no significant improvements were found in moving from the 80% training set (504 documents) to the complete training set (630

documents). This suggests that we may have reached training saturation at about 500 documents.

To complete the experiment, we used the pruning thresholds obtained during cross-validation on the blind test set in order to see how well performance during cross-validation predicts performance on the blind test set. Figure 6 shows average precision values on the blind test set.

	------ number of training docs ------				
	126	252	378	504	630
Best Overall Precision	75.16	82.95	73.39	80.65	77.84
Best Precision at 10-30 rec	90.48	96.68	95.07	97.76	82.39
Best Precision at 80-100 rec	56.52	61.04	60.57	60.51	59.74

Figure 6: Average precision on the blind test set

Comparing the precision values in Figures 5 and 6, we see that the cross-validation runs tend to underpredict performance on the blind test set (with the exception of the 10-30 thresholds at 100% training). In general, these precision values tend to vary no more than about 10 percentage points. With a different blind test set, we might have seen the cross-validation runs coming in higher than the blind test set. There is no reason to expect that blind test sets will always outperform cross-validation runs, especially since the cross-validation runs already represent average performance over 10 distinct test sets. When the cross-validation predictions are consistent with performance levels on a blind test set, we can conclude that the training set documents are highly representative of the test set documents. Variations in performance between the cross-validation runs and the test set runs probably indicate that the test set is skewed in some way relative to the training set. Another factor that may contribute to variance is the sensitivity of ID3 to order effects. We will investigate the presence of statistical deviations due to ID3 order effects in a later experiment.

The saturation effect seen during cross-validation is also visible on the blind test set, and with a pronounced drop in performance on the 10-30 thresholds with 100% training. This suggests that cross-validation data alone can reliably detect training saturation as a training set grows.

3.1.2 Manipulating Recall/Precision Trade Offs

In our next set of experiments, we ran a single d-tree under three different sets of pruning thresholds in order to see how much we could control the R/P tradeoff by manipulating pruning thresholds. Figure 7 shows the resulting R/P curves for the thresholds designed to maximize precision at the lowest (10%-20%-30%) recall levels and then the highest (80%-90%-100%) recall levels. For this experiment we used the 504 document training set (from the first experiment), and collected R/P data from the 141-document test set.

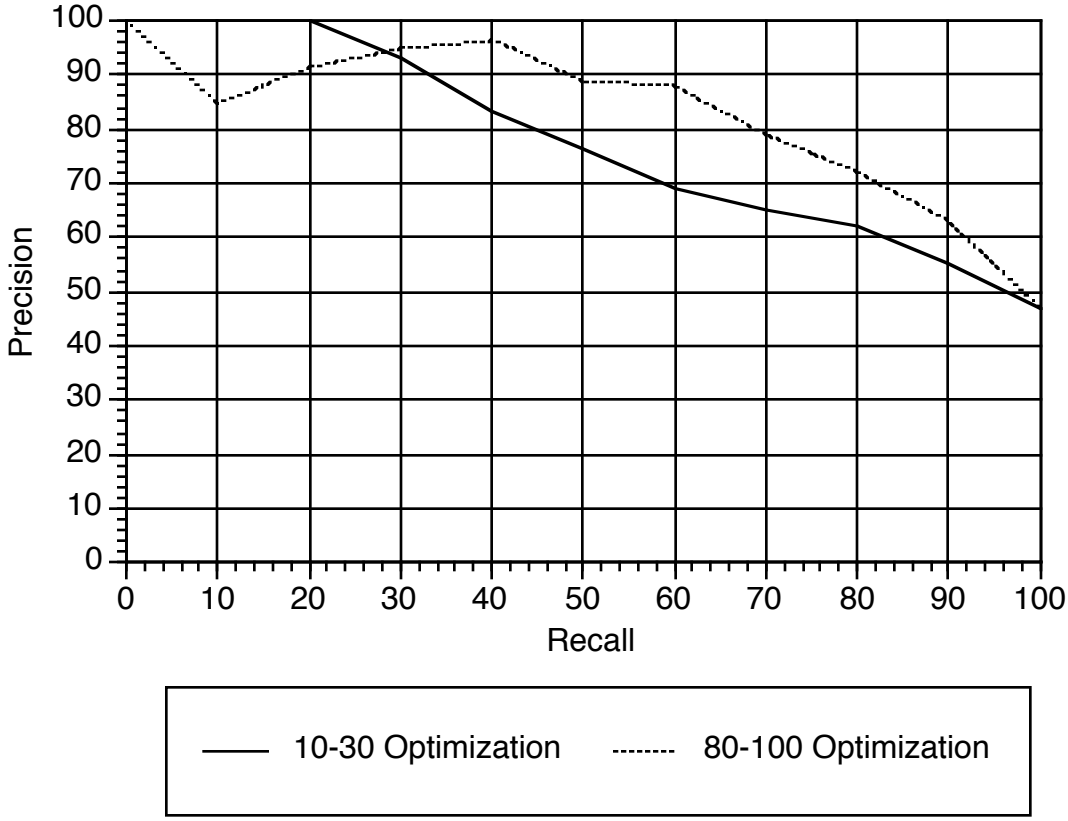

Figure 7: Two R/P curves based on different pruning thresholds

The 10-30 thresholds are particularly effective at maintaining highly reliable precision among the documents near the top of the ranked list. The top 13 documents ranked by FIGLEAF under these settings were all relevant (at the 20% recall level), and the top 21 documents contained only 1 false hit (at the 30% recall level). Although less reliable at the highest levels of the ranked list, the 80-100 thresholds were also very effective at mid-recall ranges, with only 4 false hits in the top 37 documents (the 50% recall level).

We note that the relatively low precision (85%-92%) found in the 80-100 runs at the lowest recall levels are the effects of noise. At 10% recall, we are dealing with a small number of documents (6 or 7), so any errors at this recall level tend to be magnified. In general, R/P curves are smoother and more reliable as recall levels grow unless we are working with test sets that are large enough to reduce noise at the lower levels of recall.

3.2 Additional Experiments

Our first two experiments raised questions about statistical variations in our test data and suggested additional R/P manipulations through the use of pruning heuristics. We will address both of those issues in greater detail here.

3.2.1 ID3 Order Effects

ID3 is somewhat sensitive to the order of training data when building a d-tree. If more than one feature has an equal information metric, the choice of feature at a node will be influenced by the order of the training instances. These order effects may influence performance on the blind test set, particularly if the effects are seen high in the tree. We are careful to shuffle the positive and negative training instances in order to minimize these order effects, but the effect is still a potential problem.

In order to see how much variation we may be getting from order effects, we have conducted a collection of 10 test runs in which we reshuffled the training data for each training session. We then ran a 10-fold cross-validation on each resulting d-tree in order to obtain the best possible pruning thresholds, and then tested each tree on the same test set of 141 items used in our earlier experiments. The pruning thresholds were optimized for the best average precision over all recall levels. These test runs were based on a training set of 504 documents and a test set of 267 (the same 504-document training set used in section 3.1.2 with the remaining 267 documents going into the test set).

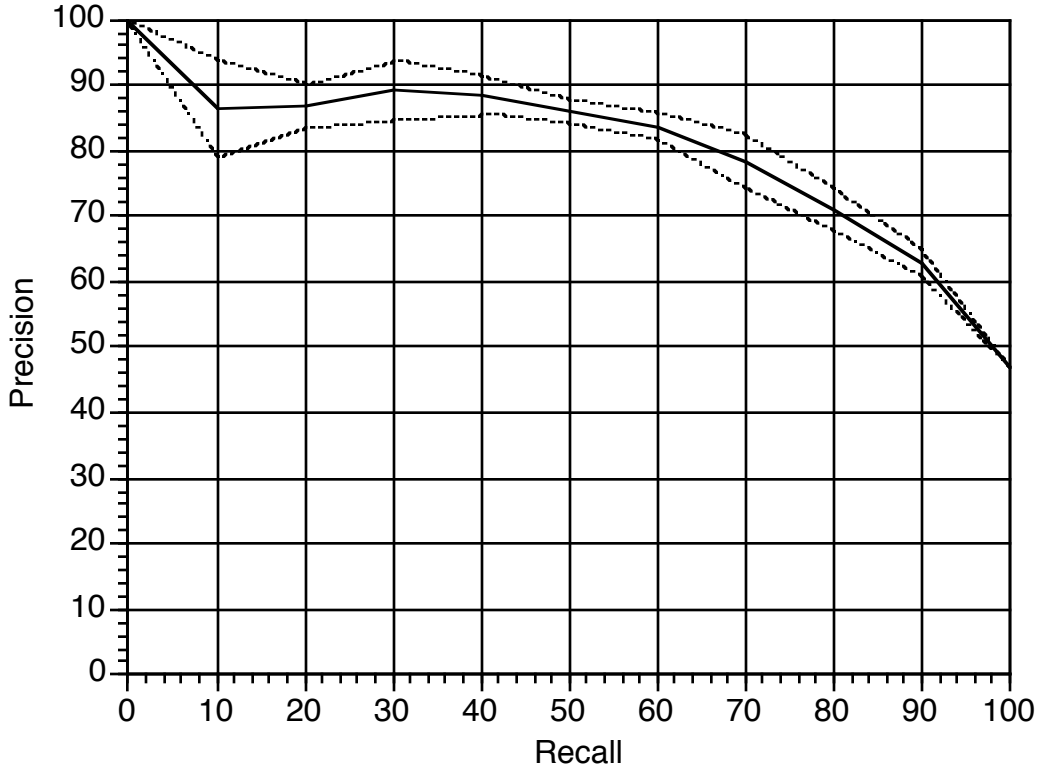

Figure 8: The effects of different training orders on 10 d-trees

The resulting R/P curves show that there can be as much as a 14 percentage point difference at any given recall level between two different test runs. However, the overall average precision between any two R/P curves does not vary by more than about 3 percentage points. Figure 8 shows an aggregate R/P curve computed by averaging all the data points from all the R/P curves for each of our 10 reshuffled test runs. The average overall precision for this

aggregate R/P curve is 79.94. The dotted lines show the span of one standard deviation at each recall level. As expected, deviations are greater at the lowest levels of recall because of the noise factor described in section 3.1.2.

If we really wanted to squeeze our d-trees for an extra percentage point or two, we might be able to maximize performance by choosing one d-tree over another, but these results suggest that order sensitivities in ID3 are probably not the best place to look for ways to strengthen FIGLEAF's performance.

3.2.2 Split Threshold Runs

In section 3.1.2 we saw how it is possible to manipulate R/P curves by changing pruning thresholds. Different evaluation metrics can be used during cross validation in order to shape the performance of our final d-tree. Attempts to optimize precision at different levels of recall in the R/P curve have been somewhat successful, in spite of the difficulties associated with extrapolated threshold settings (see section 2.4.3).

Given our success with threshold manipulations, it seemed reasonable to look at test runs where different thresholds could be used to generate different portions of the ranked document list. To investigate this, we ran a test set with one set of pruning thresholds designed to optimize precision at low recall levels when we rank list the top 17% of the test set, and then we changed to a different set of pruning thresholds to rank the remainder of the test set.

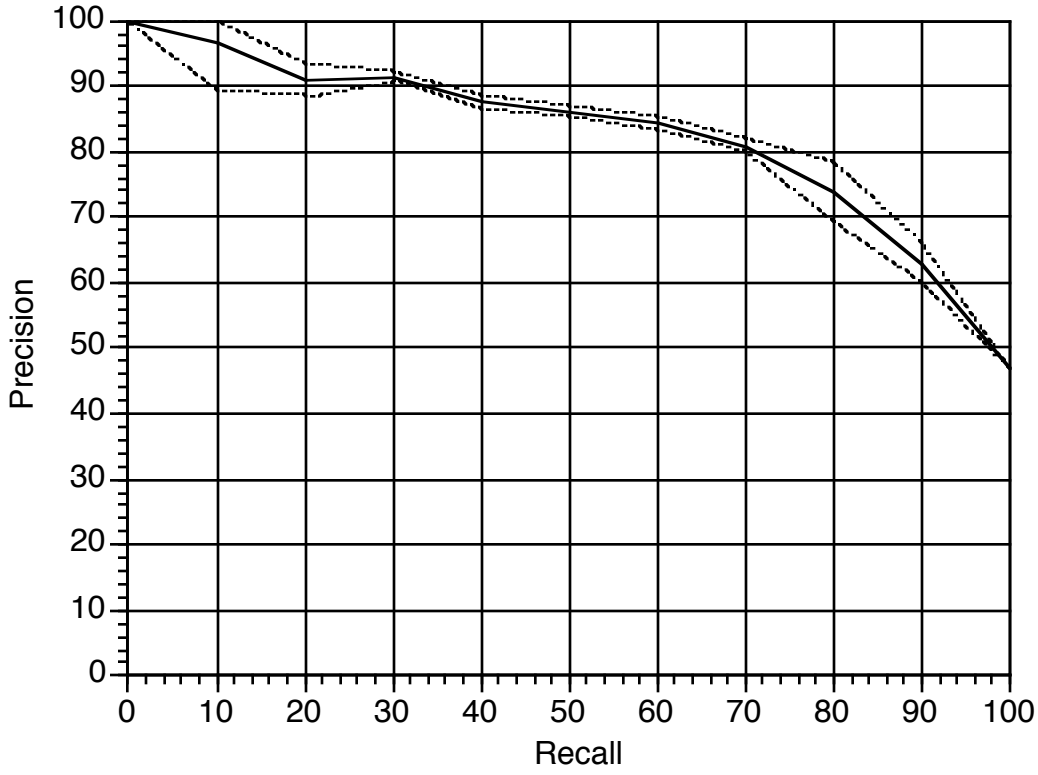

Figure 9: Randomized test runs with two sets of pruning thresholds

Figure 9 shows an average R/P curve along with one standard deviation as described in section 3.2.1. These test runs were based on the same training and test sets used in section 3.2.1. A cross-validation run was conducted to obtain pruning thresholds optimized for low recall levels and different pruning thresholds optimized for high recall levels as described in section 3.1.2.

As expected, the split-threshold runs produced the strongest overall precision of all our test runs thus far (81.93 on average). This opens up a promising area for additional experiments since trees can be tuned for local optimizations in different ways. Although we have only looked at two optimizations (low recall and high recall), it may be advantageous to work with a 3-way split that includes a third set of thresholds designed to optimize mid-range recall levels. Split-threshold runs appear to be a promising area for future investigations.

3.2.3 Adding Meta-Features

The results reported in sections 3.2.1 and 3.2.2 were based on trees built with single-term and bigram features, but without meta-features. As explained in section 2.3, meta-features are formed from a list of terms and bigrams, and the meta-feature records how many of these terms or bigrams are found in a document. Meta-features therefore require knowledge engineering in order to identify useful abstractions, and this in turn represents additional overhead in the system development cycle. In this experiment we will assess the performance gain associated with a small number of meta-features designed by the physician who annotated the HCHP asthma text corpus.

We incorporated three meta-features into an extended vector representation. One meta-feature was for terms related to asthma MEDICATION (see section 2.3), one meta-feature identified terms describing SYMPTOMS, and the final meta-feature grouped terms associated with HOSPITALIZATION or emergency room activities.

While MEDICATION, SYMPTOMS, and HOSPITALIZATION provide positive evidence for an asthma exacerbation, it is more difficult to engineer meta-features that might be strongly correlated with irrelevant documents. To find a good negative meta-feature, we used statistics from the training documents to automatically generate the IRRELEVANT meta-feature. Various thresholds were used to select terms and bigrams for different versions of this meta-feature such as IRRELEVANT-15-5 which included features that occurred in at least 15 training documents, with no more than 5% of them in relevant documents. Six versions of IRRELEVANT, with frequency threshold of 10 or 5 and relevancy-rate threshold of 3, 5, or 7, were included in the encoding of each instance. ID3 selected the version with highest information metric and ignored the others, sometimes selecting different versions in different subtrees.

Statistical analysis was also used to second-guess the domain expert's terms and bigrams. Frequency thresholds of 0, 5, 10, and 15, and relevancy-rate thresholds of 0, 85, 90, and 95 were used to create several versions of each meta-feature. MEDICATION-0-0 was the full list of 19 terms and bigrams related to asthma medication, while the more restrictive MEDICATION-0-90 eliminated six of them, and MEDICATION-10-95 had only three terms. When all distinct versions of this meta-feature were included in the instance representation,

ID3 selected MEDICATION-0-90 as the root node and did not include the other versions in the tree.

Figure 10 shows how the addition of these meta-features improved performance on the 267-document test set after training on 504 texts.

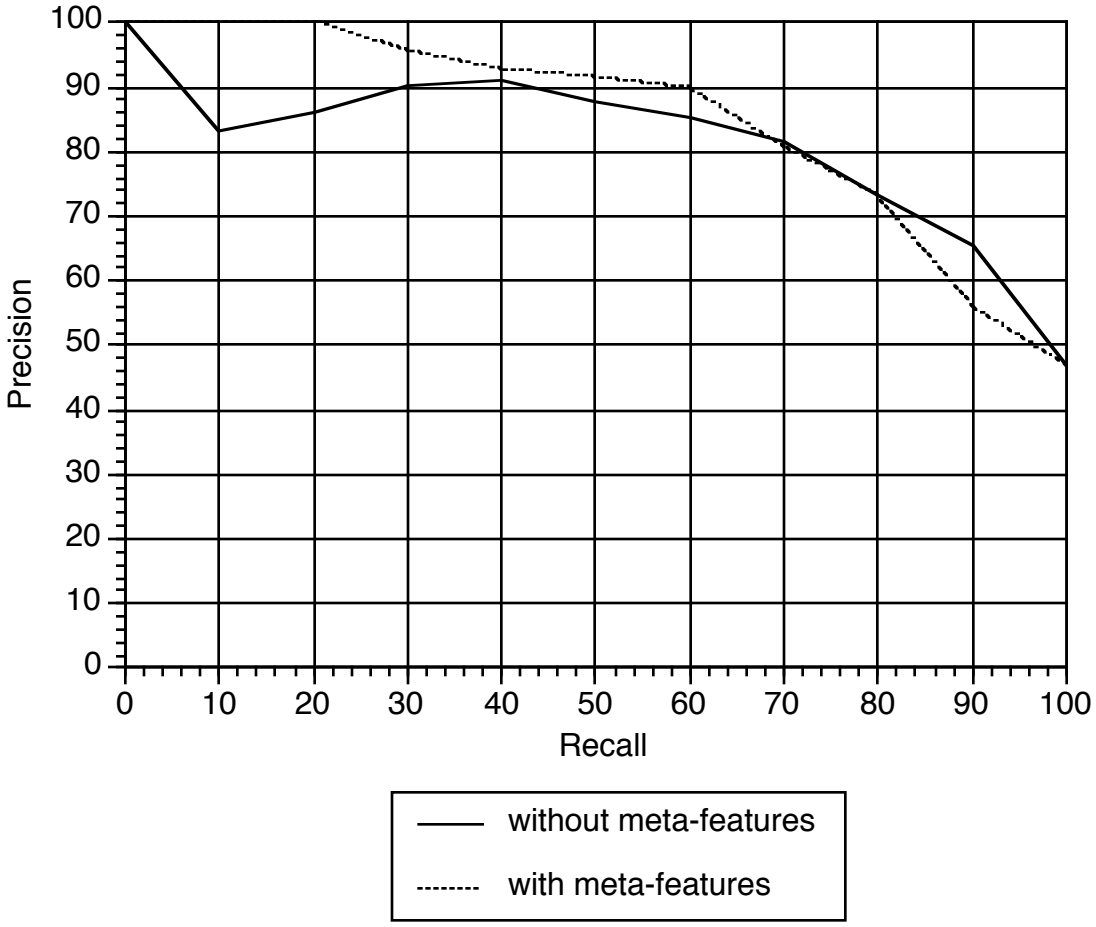

Figure 10: Improving precision with meta-features

A d-tree trained with meta-features is fundamentally different from a d-tree trained without meta-features. These different trees incorporate independent patterns of evidence based on the different vector representations available to them. The tree with meta-features may assign a high confidence level to an instance because it has one MEDICATION feature and no IRRELEVANT features. The tree without meta-features may test a long list of features such as DAYS, EMERGENCY-ROOM, NEB, WHEEZE, Q12H, NO-WHEEZE, and so forth before assigning a confidence level to the same instance. The confidence level from the two trees do not always agree and can disagree dramatically from time to time.

We have discovered that two d-trees are better than one when the trees are based on different vector representations. Averaging the confidence levels produced by a tree with meta-features and a tree without meta-features results in a more robust and noise-tolerant classification algorithm. Figure 11 shows three R/P curves from trees trained on 504 documents and tested on 267 documents. In addition to the two curves shown in Figure 10, we now add a

third curve based on averaged confidence levels. As we can see, adding meta-features improved performance, and combining confidence levels from two d-trees (one with and one without meta-features) produced some additional improvement at higher levels of recall.

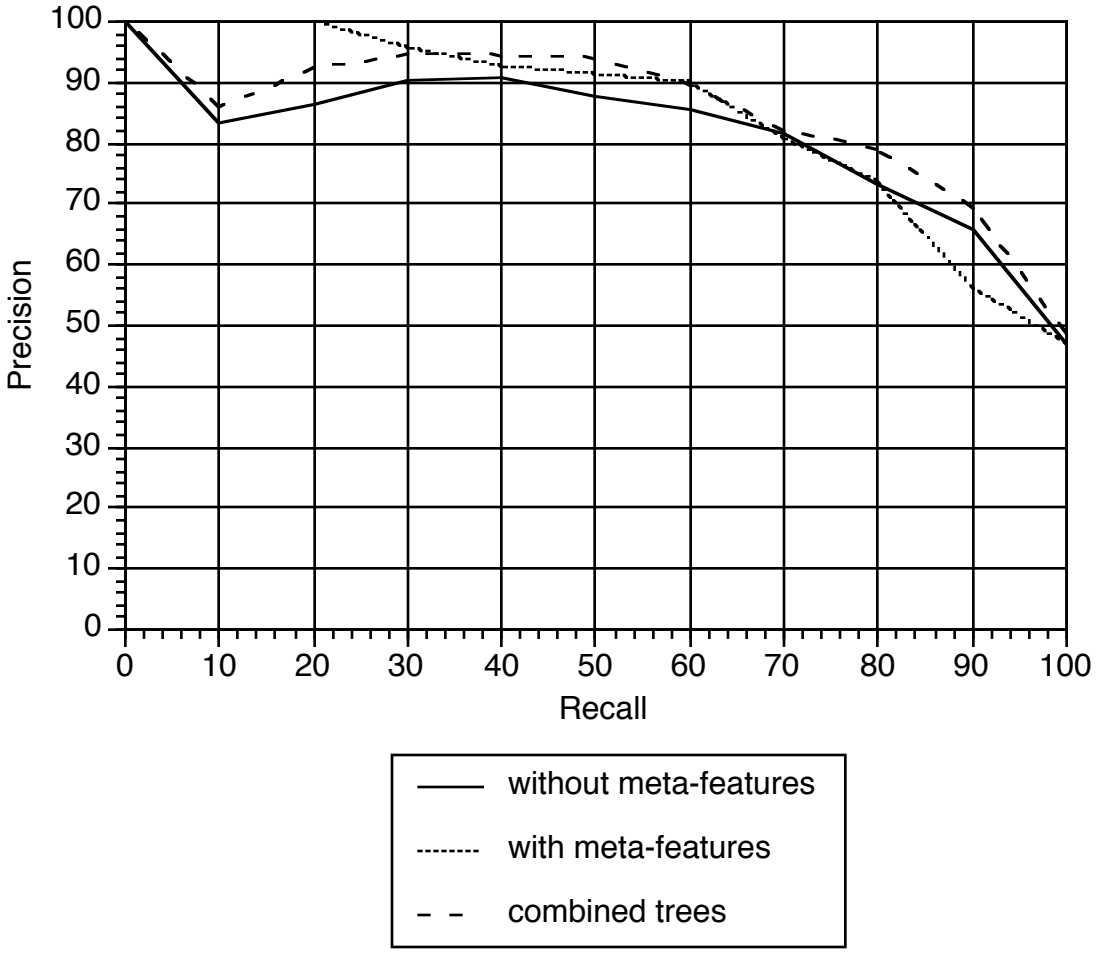

Figure 11: Averaging confidence levels from two d-trees

3.3 Performance Issues

In the course of these experiments we have encountered a number of difficulties that are known to impair performance when they are not handled adequately. Although we have described these issues in various places throughout the paper, we will enumerate them once again and summarize our observations about each one.

3.3.1 Coding Reliability by Domain Experts

Coding reliability on the part of humans is actually a double-edged sword when we try to evaluate an automated text classifier. On the one hand, any trainable system that relies on human coding to drive a supervised learning algorithm requires reliable training data in order to operate effectively. On the other hand, low degrees of coder consistency result in lower performance

levels for human beings. So if human performance is a benchmark for automated systems, fallibility among humans results in an easier performance target for the automated systems. For maximal system impact, we want a classification task that can be reliably handled by skilled humans (highly experienced or very careful coders), but which is nevertheless very difficult for most people.

Although we do not have any data on human coding for the asthma exacerbation task, other classification and indexing studies indicate that even the most experienced human coders will tend to produce coding errors [Green and Wintfeld 1993]. Error rates are affected by the complexity of the task, the expertise of the coder, and the ratio of documents that fall into grey areas with respect to domain definitions. When manual coding is part of an ongoing job responsibility, error rates will also be exacerbated by fatigue factors, distractions on the job, and pressure to maintain adequate throughput rates.

Inductive machine learning technologies are particularly attractive for tasks where no domain definition may be available for training purposes, or it may be very difficult to engineer a comprehensive knowledge base for a variety of reasons (e.g. cost constraints, lack of available expertise, difficulties with manual knowledge engineering, issues of scale). ID3 is effectively extracting domain knowledge from an annotated text corpus in a fully automated manner, and this enables FIGLEAF to acquire domain expertise that is never provided to the system in terms of explicit rules and heuristics.

3.3.2 Reliable L-SIG Codes

As we explained earlier, L-SIG codes can sometimes conflate lexical items that are completely unrelated. Although L-SIG error rates may be interesting in their own right, we have not studied them at length because the design of FIGLEAF tends to obviate any L-SIG weaknesses.

For example, we see that DAYS and DEAD both map to the same L-SIG code in the master dictionary. This looks potentially dangerous until you find out that DAYS occurred 206 times in the training corpus and DEAD occurred only once. The overwhelming statistical bias in favor of DAYS effectively erases this particular ambiguity in the L-SIG codes. As long as the training corpus is representative of the runtime input, L-SIG collisions like this one have negligible impact.

Indeed, the nodes that carry the greatest information gain tend to come from feature nodes that are not conflated by colliding L-SIG codes. For example, L-SIG codes for medicine names tend to be reliable because these names are usually long and longer words are encoded more reliably under L-SIG codes than shorter words.

But suppose a potentially important feature such as an abbreviation for an asthma medication, PRED, is conflated with a high frequency word of no significance, such as PER. Then the corresponding L-SIG code will tend to occur in both relevant and irrelevant texts, and the d-tree will not recognize any strong information gain in connection with this feature. FIGLEAF will therefore fail to recognize the importance of PRED, but it will also fail to generate false hits on the basis of the conflation alone. The L-SIG collision does not actively mislead FIGLEAF when PRED|PER appears at runtime, but the

d-tree has probably failed to exploit a discrimination with strong information gain, and FIGLEAF has missed an opportunity to weigh the presence of the PRED feature as strong evidence in favor of an exacerbation classification.

We plan to examine this situation more carefully to see if FIGLEAF is losing potentially predictive features because of L-SIG encoding problems. If we do determine that classification errors are being made because of L-SIG encoding errors, a manual realignment of the master dictionary can be undertaken to correct any conflated features. A visual inspection of the dictionary will reveal all conflated features. New codes can be inserted into the dictionary in order to create needed discriminations and the dictionary lookup routine can be patched to check for potential encoding problems based on the manual insertions. Since the master dictionary is small and the number of conflated entries represents a small percentage of the full master dictionary, this manual coding effort is a relatively minor undertaking.

We need to investigate the role of L-SIG errors in the overall performance of FIGLEAF, but if we determine that L-SIG errors are harmful, then there is a straightforward solution that can be easily implemented. Note that a manual adjustment to the master dictionary is feasible only because we're dealing with very small dictionaries (typically less than 1000 entries). FTC applications are expected to hinge on a small number of predictive keywords and phrases because we are dealing with a relatively narrow universe of text to begin with. Severely restricted FIGLEAF dictionaries give L-SIG codes a fighting chance, and then inductive d-trees help to weed separate reliably predictive codes from badly conflated ones.

We will acquire a better understanding of this balance between coarse lexical encodings and inductive discriminations when we apply FIGLEAF to more FTC applications. The utility of L-SIG codes must be assessed in the functional context of the larger FIGLEAF text classification system.

3.3.3 Effective D-Tree Threshold Settings

As we saw in section 2.4.3, there are a number of difficulties associated with tree pruning heuristics. If we cannot conduct an exhaustive search over the full space of possible threshold values, then we need to employ a heuristic search strategy which may or may not yield optimal settings. For an application where optimal precision is very important, it may be worth the extra cycles to conduct an exhaustive search despite the computational overhead.

Searches for pruning thresholds are computationally expensive because a full ten-fold cross validation must be conducted to evaluate each threshold setting that is considered. Given the expense of the cross-validation design, there is a compelling need for useful heuristics to guide the search process. Although heuristic search has been studied extensively in general, the problem of finding good pruning thresholds for d-trees might constitute a special case of heuristic search. We know that local maxima are a problem for hill-climbing algorithms in general, but we don't know where such maxima are most likely to appear in a search space of pruning thresholds. If there are any regularities in these search spaces, we might be able to exploit those regularities in order to arrive at reliable and practical solutions for the threshold optimization problem.

Note that in our experiments we divided the training corpus into a large portion of 630 documents reserved for training and a smaller portion of 141 documents which we set aside as a blind test set. This was done only for the purposes of our experiments, so we could obtain test data on a set of texts that were not used in any way for d-tree training. If we were building an application system, we would want to use the complete training corpus for the cross validation runs as well as the final d-tree training. This would not compromise our threshold settings in any way - it would just prevent us from running any experiments on a fully blind test set.[5]

3.3.4 Order Sensitivities in ID3

It is important to preserve the original training data files when a machine learning algorithm is sensitive to order effects in the training data. We can duplicate any test run for all the experiments described in this paper, as long as we preserve the original training files in order to maintain the order of the training data presented to ID3. But it is understandably worrisome to know that a particularly good d-tree might never be duplicated if a training file has to reconstructed and the order of the training data is not guaranteed to duplicate the order of some earlier version of the training file.

To assess the potential variation in FIGLEAF performance due to order effects, we designed the randomized shuffle experiment described in section 3.2.1, and then we generated multiple runs in order to see the same standard deviations for the split threshold experiment in section 3.2.2. As Figures 7 and 8 demonstrate, it is possible to see variations between randomized data runs as great as 14 percentage points at the noisy (low recall) end of the R/P curve, with expected variations no greater than 8 or 9 points throughout the remainder of curve.

We have explained how noise contributes to greater deviations at the low recall end of the spectrum because of the very small numbers associated with those computations. But it is interesting to note that a different factor contributes to larger deviations at the high recall end of the curve as well. These deviations are due to the d-tree's increasingly weak discriminations as confidence levels degrade for instances near the high recall end. In every test set there will inevitably be instances that are very difficult to discriminate. Those instances will be placed near the end of the list, after the easily identifiable positive instances that make up perhaps 70 or 80 percent of the recall. Precision necessarily declines as we enter this weakly discriminated section of the list. It is theoretically possible for a d-tree to guess each test instance correctly at the highest levels of recall, but the probability of this is very small for any non-trivial number of instances. So the standard deviations seen at the high recall end of an R/P curve reflect how much guesswork is taking place late in the list.

---

[5] Note that the test runs conducted within the cross-validation design are each being run on a test set that is blind with respect to that particular partition. So these test runs are each blind relative to their own training partition. The training sets underlying each tree constructed during cross-validation do overlap with one another, but there are no overlaps between training sets and test sets.

With noise of one kind or another entering into the low recall and high recall ends of an R/P curve, we expect to see our most meaningful data in the middle regions of the R/P curve. The standard deviation data demonstrates this with minimal deviations in the mid-recall range. This increased reliability in the middle of the curve is especially visible in Figure 9, where the split-threshold run attempts to combat the effects of noise operating at both ends of the curve. The recall levels ranging from 30% to 70% are characterized by standard deviations which range from 0.7 to 1.1 percentage points — a remarkably tight behavioral range for 10 separate d-trees.

Since the data in the mid-recall ranges is the most meaningful data in an R/P curve, and the standard deviations for that data are so minimal in our split-threshold runs (which produce our best test results), it seems that we have little to fear from order effects in the training data. Any tree that we train and run under a split-threshold design appears to be operating at most 2 percentage points beneath the best tree we are likely to generate from the same training data. So if we want to strengthen the overall performance of FIGLEAF, it seems that ID3 order effects are not the best place to look for potential improvements.

3.3.5 Final Document Rankings

We have not been able to locate any published papers that discuss the problem of converting d-tree output into ranked instance lists, although ranked instance lists could be of general interest for many d-tree applications. Ranked listings of test set instances do not appear to be a conventional output format outside of IR, but there is no reason in principle why that should be the case. When a d-tree is used to rank list a set of test instances, we can see how the d-tree organizes its test data from the strongest positive instance to the weakest (in the opinion of the d-tree). An absolute cutoff threshold for distinguishing positive instances from negative instances can then be inserted at any point along the ranked spectrum.

If absolute thresholds are needed for a specific application, one could always use a cross-validation analysis of the training data to identify cutoff values that separate system output into positive and negative instances in a fully automated manner. Given a set of instances which have already been tagged as positive and negative, and a ranked listing of these instances (produced by a d-tree), one need only evaluate R/P percentages for each successive sublist of the full list (starting from the head) in an effort to find an optimal dividing line between positive instances at the head end, and negative instances at the tail end. R/P percentages can be optimized according to a bias for recall, a bias for precision, or no bias at all, depending on the requirements of the target application.

Document rankings therefore suggest a useful strategy for automatically computing d-tree cutoff thresholds when an application requires an absolute judgement on runtime data. Ranked lists are also easy to convert into the standard R/P curves, and these provide a convenient visual standard for performance comparisons. There is no reason why R/P curves should be restricted to information retrieval applications. In fact, the information extraction community has found this convention useful for evaluating the performance of their complex systems, in spite of the very different I/O

requirements associated with information extraction and information retrieval. Any d-tree application which can be analyzed in terms of correct answers and incorrect answers is amenable to an evaluation in terms of ranked instance lists and R/P curves.

While the utility of ranked output will be more useful for some applications than others, the problem of converting d-trees output into ranked lists is an interesting problem of great generality. The technique used by FIGLEAF may not be the best possible solution, in which case significant performance gains might be obtained by addressing the ranking problem more effectively. We are surprised to see that the machine learning community does not appear to be promoting standard algorithms for handling this problem — there is a clear need for good algorithms that tackle this challenge.

4. Conclusions

The FIGLEAF system was designed to provide us with fine-grained text classification capabilities that can be obtained in a fully-automated fashion given a collection of annotated training documents. We acknowledge the importance of domain knowledge and document content for applications that require highly sensitive classification discriminations, but we also want to avoid the knowledge engineering bottleneck that traditionally characterizes knowledge-oriented approaches to text comprehension [Lehnert 88].

FIGLEAF addresses this apparent knowledge engineering conundrum by exploiting the knowledge of a domain expert who effectively teaches FIGLEAF how to classify texts by example. Each text that is marked and categorized by the domain expert provides FIGLEAF with an example of the problem being solved correctly. The text annotations direct FIGLEAF's attention to important text signatures, and enable FIGLEAF to construct a vocabulary of important terms that influence classification decisions. Once FIGLEAF has amassed enough training examples, it organizes this knowledge inductively in a decision tree and is then prepared to apply the knowledge on novel input.

FIGLEAF is therefore a knowledge-based system, but one which avoids the knowledge-engineering bottleneck by the use of a text marking interface. This provides a system developer with a reasonable balance between competing system requirements, while satisfying important key design desiderata. FIGLEAF is designed to:

- attain text classification performance levels that duplicate human performance levels as measured by precision and recall.

- provide a technological solution to text classification problems that can be ported from one application to another in a cost-effective manner.

- benefit from the expertise of domain experts while avoiding the knowledge-engineering bottleneck.

- enable a developer to manipulate the recall and precision tradeoff according to the needs of specific applications, and do so in a systematic and principled manner.

- handle idiosyncratic writing that may be characterized by telegraphic sentence fragments, ad hoc abbreviations, and extremely esoteric language.

- operate efficiently and without unreasonable computational requirements.

We have demonstrated the operation of FIGLEAF in the context of a specific medical application where we have had an opportunity to investigate each of these design challenges. Our first goal of duplicating human performance levels has not been explicitly addressed by this paper: we need to collect baseline data for human performance in order to evaluate FIGLEAF relative to human capabilities. We nevertheless suspect that there is room for improvement and we have identified some likely places to find performance gains in our discussion of performance issues. In the meantime, FIGLEAF is producing performance levels that are probably adequate for many text classification applications associated with text routing and filtering.

The remainder of our design desiderata appear to be well in hand, and require additional investigations with a range of text classification applications. FIGLEAF is designed to enhance existing IR technologies by leveraging its trainable expertise against the generality of retrieval algorithms which operate on large unconstrained collections of text. FIGLEAF assumes a universe of text that is "in the right ballpark" with respect to its target text categories. FIGLEAF is effective at fine-grained text classification in part because of this assumption: we would not expect FIGLEAF to scale up to large unconstrained testbeds of text. This limitation is not a serious problem if we can use standard IR queries to operate as a first-pass filter (the coarse-grained retrieval), and then bring in FIGLEAF to operate as a second-pass filter (the fine-grained retrieval).

FIGLEAF operates as a bridge between two classes of text processing technologies that are generally thought to be very distant from one another. Information retrieval technologies are typically restricted to algorithms that circumvent a deep analysis of meaning, because in-depth text comprehension is too slow and too expensive for practical application development. FIGLEAF attempts to acquire useful knowledge about a given application through the use of a training corpus and a shallow characterization of useful information understood by domain experts.

While FIGLEAF's underlying decision tree is driven by statistics and some generic tree building heuristics, those two mechanisms alone do not paint a completely accurate picture of FIGLEAF's domain knowledge: we must be careful to remember the role of the text marking interface. When training materials are collected from domain experts working with the text marking interface, we effectively engage in a shallow form of knowledge engineering. Shallow knowledge is not codified in specific rules or heuristics that are ever verbalized, but some knowledge is implicitly present in each training example that is passed along to FIGLEAF. This knowledge is easier for a domain expert to provide than the more explicit knowledge needed to build an expert system: no knowledge engineer needs to be on hand during the operation the text marking interface. So shallow knowledge is both easier to acquire and easier to incorporate into a text classification system.

At the same time, shallow knowledge may not be good for anything beyond the specific task for which is was acquired. The knowledge acquired by FIGLEAF to handle one application will not port to another application, it is probably not reusable in any sense, and it will probably not enable FIGLEAF to bootstrap itself up to more ambitious task orientations. FIGLEAF is designed to do what it is designed to do and nothing more.

In spite of these limitations, FIGLEAF demonstrates a viable design strategy for intelligent agents which address the needs of individual users who want to customize text filtering capabilities for text routing applications over the internet. Readily available IR services such as WAIS and GOPHER illustrate the basic functionality for coarse-grained retrieval technologies operating in a rich online text environment. Trainable systems designed to handle fine-grained retrieval represent a new retrieval paradigm that can be piggy-backed on top of the coarse-grained technologies to provide us with a new class of realistic text processing applications. FIGLEAF represents a first step in this direction by exploiting established technologies in artificial intelligence in a cost-effective manner. We are encouraged by the success of our preliminary experiments with FIGLEAF, and we plan to expand our repertoire of fine-grained text classification testbeds in an effort to assess the full generality of this new technology.

Acknowledgements

This work was conducted in conjunction with the National Center for Intelligent Information Retrieval at the University of Massachusetts which is supported by the National Science Foundation (under Grant No. EEC-9209623), the Commonwealth of Massachusetts, and a number of member industry partners. We are especially grateful to the Harvard Community Health Plan for their participation and the use of their medical documents: our experiments would not have been possible without their assistance. This research was also supported, in part, by the Harvard Community Health Plan Foundation. The claims and opinions put forth in this paper are nevertheless those of the authors alone, and do not represent the official policies or beliefs of any of our affiliated or supporting organizations.

Bibliography

Aronow, D.B. and Coltin K.L. (1993) Information Technology Applications in Quality Assurance and Quality Improvement, Part I. *Joint Commission Journal on Quality Improvement* 19(9):403-415

Dick R.S. and Steen E.B. (eds.) (1991) *The Computer-Based Patient Record: An Essential Technology for Health Care.* Washington DC: National Academy Press.

Green J., and Wintfeld, N. (1993) How accurate are hospital discharge data for evaluating effectiveness of care? *Medical Care*, 31:719-731.

Lehnert, W.G. (1988) Knowledge-Based Natural Language Understanding. in *Exploring Artificial Intelligence*. (ed: H. Shrobe) Morgan Kaufmann. San Mateo, CA. pp. 83-131.

Lehnert, W.G. and Sundheim, B. (1991) A Performance Evaluation of Text Analysis Technologies, *AI Magazine,* pp. 81-94.

Quinlan, J. R. (1986) Induction of Decision Trees, *Machine Learning 1*, 1986, pp. 81-106.

Riloff, E. and Lehnert, W.G. (1994) Information Extraction as a Basis for High-Precision Text Classification, *ACM Transactions on Information Systems*. Vol. 12, No. 3, pp. 296-333.

Schoenbaum, S.C. and Barnett G.O. (1992) Automated ambulatory medical records systems: An orphan technology. *International Journal of Technology Assessment in Health Care*. 8:598-609.

Soderland, S. and Lehnert, W.G. (1994) Corpus-Driven Knowledge Acquisition for Discourse Analysis, *Proceedings of the Twelfth National Conference for Artificial Intelligence*. AAAI/MIT Press (in press).