Acrophile: An Automated Acronym Extractor and Server

Leah S. Larkey, Paul Ogilvie, M. Andrew Price
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
Email: {larkey, pogil, maprice}@cs.umass.edu

Brenden Tamilio
School of Cognitive Science
Hampshire College
Amherst, MA 01002
Email: bat96@hampshire.edu

ABSTRACT
We implemented a web server for acronym and abbreviation lookup, containing a collection of acronyms and their expansions gathered from a large number of web pages by a heuristic extraction process. Several different extraction algorithms were evaluated and compared. The corpus resulting from the best algorithm is comparable to a high-quality hand-crafted site, but has the potential to be much more inclusive as data from more web pages are processed.

KEYWORDS: Acronyms, information extraction

INTRODUCTION
Acronyms are everywhere; we read and hear them but rarely think about them, except when we do not know what they mean. Every content domain has its own acronyms and abbreviations. In many of these areas, particularly those that are highly technical or bureaucratic, acronyms occur frequently enough to make it difficult for outsiders to comprehend text.

Many acronym and abbreviation dictionaries are available, both in printed form and on the World Wide Web. Some attempt to be all inclusive, others are specialized for particular domains. There are searchable databases and simple lists. Some general problems in building such collections, or any dictionaries, are getting comprehensive coverage, and keeping the collection current. New abbreviations continually come into use. To keep their dictionaries growing, some maintainers allow users to submit new acronyms and definitions. This openness, however, can result in poor-quality data.

Acrophile is an automated system that builds and serves a searchable database of acronyms and abbreviations using information retrieval techniques and heuristic extraction. It was developed and built by students during an NSF REU (Research Experience for Undergraduates) summer program. The current version, available on the web at http://ciir.cs.umass.edu/ciirdemo/acronym/, contains a set of acronyms and expansions that were extracted from a large static collection of web pages. The system can crawl the web for additional pages, extract additional acronym/expansion pairs, and collect them in a file. Periodically, the database can be rebuilt, incorporating the additional new pairs.

To appear in DL00.

Another important goal of this project was to evaluate the quality of our automatically-built acronym and abbreviation databases. We developed evaluation techniques to compare different extraction algorithms and to compare the quality of our automatically-built databases with manually collected databases.

Our evaluation goals were to test the following hypotheses:

1. It should be possible to use IR techniques and heuristic extraction to collect a set of acronyms and expansions which is at least as good and as comprehensive as carefully constructed manually built lists available on the web.
2. In order to collect as many correctly expanded acronyms as possible from an essentially unlimited corpus like the web, one should choose a strict algorithm that accepts few errors, even at the cost of missing some cases in specific documents. It should be possible to pick up those missed definitions from other contexts by processing more text, and the resulting lists should have higher precision than a similar-sized list produced by a less strict algorithm.
3. We should be able to increase the coverage of our collection more efficiently by searching for acronyms than by processing random pages.

Related Work
Many acronym and abbreviation dictionaries have been compiled and published in books and many lists are available on the web, such as Acronym Finder [1] and the World Wide Web Acronym and Abbreviation Server (WWWAAS) [17]. The Opaui Guide to Lists of Acronyms, Abbreviations, and Initialisms [13] has 124 links to acronym and abbreviation lists, some of them general, and some as specialized as the Dog fanciers acronym list [4] or the Mad Cow disease list [10].

All of these web-based lists appear to be built manually rather than by automatic extraction. The lists range in size from a few dozen items to over 127,000 acronym definitions in Acronym Finder [1]. The accuracy seems to vary widely. The primary problem with many large lists on the web is that they allow people to submit expansions. Some sites screen submissions carefully [12], others do not. As far as we can determine, no previous automatic extraction efforts have resulted in publicly searchable online databases of acronyms and none have received thorough evaluations. IBM advertises a tool for abbreviation extraction, *IBM Intelligent Miner for Text*, which allows corporations to process and categorize text documents [16]. Among the product's features is the ability to extract abbreviation phrases. But the paper does not present any information on the heuristics used, nor does it present data on the quality of the results.

Two small-scale acronym extraction projects have been described and received limited evaluation in unrefereed literature. AFP (Acronym Finding Program) [15] is an acronym extraction algorithm which considers strings of from 3 to 10 uppercase letters as an acronym, and looks for candidate expansions in windows of twice the number of letters in the acronym before and after the acronym. It only looks for matching letters occurring at word beginnings (or after hyphens) but allows some mismatch between the letters of the acronyms and the sequence of initial letters of the words. AFP was tested on 17 documents from the Department of Energy. It attained 93% recall and 98% precision on acronyms in this set with length of three or greater, 86% recall and 98% precision when two character acronyms were included.

TLA (Three Letter Acronyms) [18] was developed at the University of Waikato. It has no case requirements for acronyms, so that any token is a candidate acronym. The token is accepted if a matching sequence is found by taking up to three characters from adjacent words. TLA was evaluated on ten computer science technical reports, on which it obtained 91% recall and 68% precision. A newer approach by the same researchers uses compression models to identify acronyms and definitions [19]. This approach is less ad-hoc than a purely heuristic approach like ours, but requires a corpus of hand-marked training data.

None of these extraction systems have been used to process a large corpus of text and compile a searchable dictionary of acronyms.

Automated extraction projects for extracting non-acronym text relations bear some interesting similarities to the acronym problem. Several email extractors such as Atomic Harvester 98 [4] and EmailSiphon [6] can be found on the web. They crawl through every web page at a given site and extract every email address they can find, to compile lists to sell commercially. Extracting email addresses is simpler than finding acronyms and expansions because it does not require relating pairs of segments found in text. It is sufficient to search for the general pattern *username@location*. Higher accuracy can be gained by checking the suffix of an address for the existence of common domains such as .edu, .com, and .gov.

The processes of extracting hyponyms [9] and citations [8] are more similar to the acronym task in that they require extracting a relation from text. Hearst's hyponym extractor [9] finds pairs of noun phrases NP_1 and NP_2 such that NP_1 is a kind of NP_2, for example, nutmeg is a hyponym of spice. Her system finds hyponyms in text by looking for some simple patterns like "spices, such as nutmeg," "spices, including nutmeg and sage", or "such spices as nutmeg and sage." As we find for acronyms, these heuristics provide reliable but not foolproof methods of finding hyponyms. Hearst ran the extraction algorithm on an encyclopedia, and found many correct hyponyms which could be added to WordNet [1].

CiteSeer [8] is a system that extracts bibliographical citations and references. Like Acrophile, it uses a set of heuristics to index information extracted from web pages. CiteSeer searches for pages that might contain PostScript documents and keywords such as *PostScript* and *publication*. Once the documents are retrieved, the system verifies that they are legitimate publications by searching for the presence of a references section. When documents are parsed, the system saves the title, author, year of publication, page numbers, and citation tag, a shorthand abbreviation for identifying the cited paper in the text body. It uses a set of canonical rules, for example, that citation tags always appear at the beginning of references, author information generally precedes the title, and the publisher usually follows the title. The developers of CiteSeer provide a facility on the web called ResearchIndex where users can search for references and citation information [14].

Terminology

An acronym is "a word formed from the initial letters or parts of a word, such as *PAC* for *p*olitical *a*ction *c*ommittee." An abbreviation is "a shortened form of a word or phrase used chiefly in writing…" [3]. Thus, an acronym is an abbreviation whose letters are read as a word. This definition excludes abbreviations such as *FBI* and *NAACP*, which are pronounced by saying the individual letters in the abbreviation.

Our project covers a subset of abbreviations which is larger than the set of acronyms, but smaller than the set of all abbreviations. We include abbreviations that do not form words, as long as their letters come from the words in the phrase. We also include abbreviations with numbers such as *4WD*, and *3M*, although our expansion algorithms can only deal successfully with cases where the digit stands for a spelled-out number (*Four Wheel Drive*), or acts as a multiplier (*Minnesota Mining and Manufacturing*). It cannot handle cases like *Y2K*. We exclude abbreviations composed of letters that are not in the words (*lb*.), and abbreviations for single words rather than multiword phrases.

We use the term "expansion" for the phrase an acronym stands for.

In the remainder of this paper we describe the Acrophile system, then the acronym extraction algorithms; finally, we evaluate the algorithms and compare the resulting collections with some hand-crafted acronym collections on the web.

SYSTEM DESCRIPTION

The core of Acrophile is a large collection of acronyms and expansions, which was automatically extracted from web pages and indexed using Inquery 3.2, a probabilistic information retrieval system developed at the University of Massachusetts. Users can submit an acronym such as *IRS*, and see a list of expansions for that acronym, or they can submit words (such as *Internal Revenue* or *revenue*) and see the acronyms whose expansions contain those words. The system returns lists of acronyms and expansions, ranked by a quality score. One can also submit a URL to the acronym extractor and get a list of acronyms and expansions found on the page.

We first describe this collection and how it was created, then we describe the lookup system on the web.

Building and updating the database

Figure 1 outlines the process by which the database was created and how it can grow. A static collection of around 1 million (936,550) military and government web pages, comprising around 5 gigabytes of text, was processed in the manner illustrated in Figure 1.

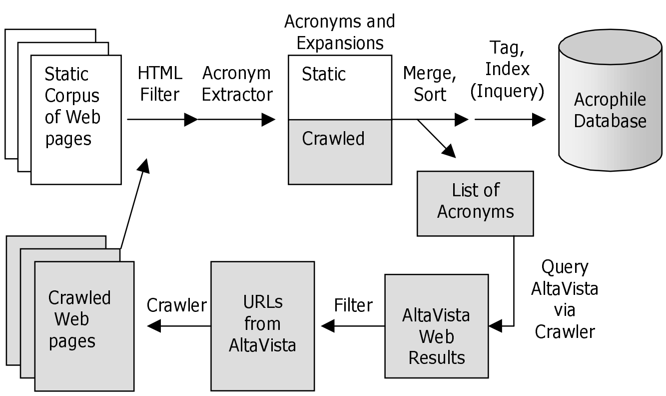

Figure 1: Building and updating Acrophile

First, a Perl script performs some simple filtering on the web pages, to remove all HTML tags. The resulting stream of text is fed into our acronym extractor, a C program using flex and yacc, which incorporates the best of the four algorithms we tested in the evaluation reported below. The acronym extractor produces a list of acronym and expansion occurrences. These pairs are marked to indicate whether they came from a parenthetical form such as *DUI (Driving under the Influence)*, information which is later used in computing a confidence rating for the expansion.

Acronym/expansion sets are then sorted and merged, accumulating counts of occurrences. Occurrences in parentheses are also counted separately. The output is tagged, creating pseudo-documents for indexing. Each pseudo-document has an acronym as its title and an expansion as its text. A confidence score is also placed in a tagged field. These data files are then indexed, with no stopping or stemming performed on acronyms or expansions. The indexing process creates a searchable Inquery database.

The shaded path through Figure 1 shows how the database can be expanded by crawling for web pages containing known acronyms. A list of acronyms is submitted as individual queries to AltaVista, using a modified version of Gnu's wget. For each query, we retrieve the top *n* matching pages, returned from AltaVista ten at a time. Each results page is piped through a hand-coded filter which attempts to remove all content except the URLs of the found documents. These URLs are then crawled in sets of 10 by another instance of the crawler. This crawling accumulates a new collection of web pages, which are processed like the static set, to extract an add-on set of acronym/expansion pairs. These can be added to the original set, and the database rebuilt.

The Search System

The search process is illustrated in Figure 2, below.

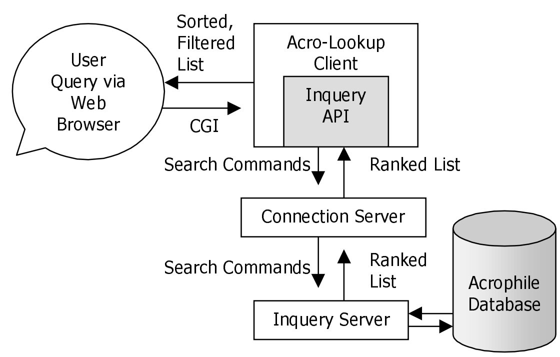

Figure 2: Searching for acronyms

The system uses a client/server architecture which could accommodate multiple servers across a network, although at present our client and (single) server run on the same Unix system. The user types an acronym or phrase into a text box on the Acrophile search page. This query is submitted via CGI to a custom Inquery web client developed for Acrophile. The client creates a network connection to the Inquery connection server, which issues search commands to an available Inquery server, which retrieves acronym/expansion pairs from the database. A ranked list is then returned through the connection server and back to the client. A confidence score is computed for each expansion, based on the stored occurrence counts. The list is sorted by this confidence score, filtered, and formatted for display in the user's web browser. The user may select how many expansions they would like to see.

Extraction Demonstration

In addition to the searchable acronym collection, the Acrophile splash page also contains a link to an online extraction demonstration, which accepts a URL from the user and extracts acronyms and expansions from the submitted document in real time. Currently, the results of this extraction are not added to the online database.

EXTRACTION ALGORITHMS

The Acrophile extraction algorithms use flex, a lexical analyzer, and yacc, a parser, to process a text document to extract acronyms. Expansions for the acronyms are found in the text using a combination of document context and canonical rules, which match patterns in which acronyms are commonly defined in standard written English.

We developed several different versions of extraction algorithms and tested four of them. All versions work on the general principle of hypothesizing that a sequence is an acronym if it fits certain patterns, and confirming it as an acronym if a plausible expansion for it is found nearby. For all four algorithms, some normalization is performed after extraction. Two acronyms are considered equivalent if they differ only in capitalization. Two expansions for an acronym are considered equivalent if they differ only in capitalization or in the presence or absence of periods, hyphens, or spaces.

Our four algorithms, called *contextual, canonical, canonical/contextual,* and *simple canonical*, differ in what patterns are taken to indicate potential acronyms, what forms expansions can be found in, and what text patterns indicate a possible acronym/expansion pair. The contextual, canonical, and canonical/contextual algorithms are all related and arose by modifying an earlier contextual algorithm. The simple canonical algorithm was designed independently to try a more limited approach that might yield higher precision on the acronyms it found. We did some initial tuning of algorithms based on their performance on a small pilot set of 12,380 Wall Street Journal articles from 1989.

The simple canonical algorithm (also called *simple*) is the strictest of the four. It finds only those acronym/expansion pairs which fit a small set of canonical forms, such as "expansion (ACRONYM)", or "ACRONYM or expansion". The contextual algorithm, on the opposite end of the strictness continuum, looks for an expansion in the vicinity of the potential acronym without requiring any canonical pattern ("or", parentheses, commas, etc.) indicating their relationship. The canonical/contextual and canonical algorithms fall in between the other two. The four algorithms are contrasted in Table 1, which lists their major characteristics. The columns of the table summarize the four different algorithms. The top half of the table lists properties of hypothesized acronyms. The bottom half covers properties of the expansions. All four algorithms are described below.

Finding Acronyms

The algorithms identify potential acronyms by scanning text for the patterns shown in the row labeled *Acronym Patterns* in Table 1. This row uses a pseudo-regular-expression notation in which superscript + indicates one or more occurrences of a symbol, * indicates 0 or more occurrences, numbered superscripts indicate a specific number or range of occurrences. U stands for an uppercase letter, L a lowercase letter, D a digit, S an optional final *s* or *'s*, {sep} is a period or a period followed by a space, and {dig} is a number between 1 and 9, optionally followed by a hyphen. Terms in square brackets are alternatives.

The contextual algorithm accepts acronyms that are all uppercase (*USA*), with periods (*U.S.A.*) or which have a sequence of lowercase characters either at the end of the pattern following at least three uppercase characters (*COGSNet*), or internally following at least 2 uppercase characters (*AChemS*). An uppercase pattern can also have any number of digits, anywhere.

The canonical/contextual and canonical algorithms accept a wider range of acronym patterns. They have less constraint on lower case sequences, to allow patterns like *DoD*. Slashes and hyphens are allowed in acronyms, to get patterns like *AFL-CIO* and *3-D*. Acronyms are not allowed to end with lower case characters except for *s,* and only 1 digit is allowed in an acronym.

The simple canonical algorithm takes a minimalist approach, excluding acronyms with digits, periods, and spaces. An acronym must begin with an uppercase letter, followed by zero to 8 upper or lowercase letters, slashes, or dashes, and ending in an uppercase letter.

Acronym Expansion

Contextual Algorithm. The contextual algorithm finds expansions by matching from the last character of the acronym to the front. It always saves the twenty most recent words scanned, so when a potential acronym is identified, it tries to find the expansion in this saved buffer. Otherwise, it looks for the expansion in the text following the acronym. It requires no canonical forms, so it can successfully deal with text like, "… is three dimensional. In 3D images…"

The expansion rules can refer to a list of 35 noise words like *and*, *for*, *of*, and *the*, which are often skipped in acronyms, as in *CIIR (Center for Intelligent Information Retrieval).* The algorithm tries to find a sequence of words such that the initial 1 to 4 characters from each non-noise word match the characters of the acronym, as in *Bureau of Personnel (BUPERS).* In addition:

- One initial character of a noise word can match an internal acronym character, as in *Department of Defense (DOD).*
- A noise word can be skipped, as in *Research Experience for Undergraduates (REU).*
- The initial character and the 4th, 5th, or 6th characters of potential expansion terms could be matched to acronym characters as in *PostScript (PS).* This is an attempt to simulate a crude morphemic decomposition, but without any knowledge of English prefixes.

ACRONYMS

	Contextual	Canonical/ Contextual	Canonical	Simple Canonical
Patterns for Acronyms	$(U\{sep\})^+$ e.g. U.S.A. U^+ e.g. USA $D^*U[DU]^*$ e.g. 3D,62A2A UUU^+L^+ e.g. JARtool $UU^+L^+U^+$ e.g. AChemS	$(U\{sep\})^{2\text{-}9}S$ e.g. U.S.A, U.S.A.'s $U^{2\text{-}9}S$ e.g. USA, USA's $U^*\{dig\}U^+$ e.g. 3D, 3-D, I3R $U^+L^+U^+$ e.g. DoD $U^+[/\text{-}]U^+$ e.g. AFL-CIO		$U[UL/\text{-}]^{0\text{-}8}U$ e.g. USA, DoD, AFL-CIO
Upper vs. **Lower Case**	First two chars must be U, then any number of L anywhere, but adjacent	L internal, or final s or 's DOD, DoD, DOD's		Must begin and end with U Can have L elsewhere DOD, DoD
Digits	Any number of digits, anywhere	Only 1, any nonfinal position 3M, 2ATAF		None
Spaces and periods	After capital letters	'.' Or ". +space" must be after each character. N.A.S.A, N. A. S. A.		None
/ or -	None – treated as space in tokenizing	1 interior of /,- CD-ROM,OB/GYN		Any number of /, - in interior CD-ROM, OB/GYN
Max length	None explicit	9 alphanumeric chars, plus any included punctuation or final s		10 characters including any punctuation
EXPANSIONS				
Noise words	Fixed list of 35	Fixed list of 40		None
Skip words	Only noise words	Noise words, or words following hyphens		Only first and last words have to match chars in acronym
Noise word chars	At most 1, only characters internal to the acronym			N/A
Prefixes	Yes, assumes any initial 3,4, or 5 chars may be a prefix			N/A
Chars from non-noise word	Up to 4. Greedy, prefers to take more	Up to 4. Not greedy, prefers to take fewer		Prefers to 1. Can take more if word starts with upper case
Canonical Definition	N/A	(Unordered) AC (Exp), Exp(AC) (Exp) AC, (AC) Exp AC or Exp, Exp or AC, AC stands for Exp AC {is} an acronym for Exp known as the AC Exp "AC", "AC" Exp		(Ordered) Expansion (ACRONYM) Exp or AC Exp, or AC Exp, AC AC (Exp) AC, Exp
Capitalization	Expansion can be all L	Canonical: can be all lower Contextual: only noise words can be lower, rest must be upper	Can be all lower case	Lower case allowed, but with stricter rules than upper case; each letter in acronym must be matched by a letter starting a word in the expansion.
Numbers	Spell out or multiply			No numbers

Table 1: Properties of acronyms and expansions for four different algorithms

The contextual algorithm scans for an expansion until another acronym pattern is encountered, wherein the old acronym is forgotten and the new one becomes the source for matching, or until the expansion is found or fails.

If a digit *n* is found in the acronym, the acronym receives some special handling. The algorithm tries replacing the digit and the following or preceding character with *n* repetitions of the character, as in *MMM* for *3M*. If it cannot find an expansion for this transformed acronym, it then tries matching the digit with the spelled out number, as in *three dimensional* for *3D*. Periods in acronyms are ignored in looking for expansions.

One of the major problems with the contextual algorithm was its greediness in trying to match more than one initial character from expansion terms. This would lead it to expand *NIST* as *National Institute of Standards*, taking the *t* from *Standards*, rather than as *National Institute of Standards and Technology*. A second problem, particularly with two letter acronyms, was the unacceptably high likelihood of finding a sequence of lower case words with a spurious match for the acronym, as in *story from* for *SF*.

Contextual-Canonical. The canonical/contextual algorithm is a modification of the contextual algorithm to address the above two problems. First, canonical rules were added to constrain when lower case words are accepted for expansions. Only if an acronym/expansion pair is found in a form in the row labeled *Canonical Definition* in Table 1, is a lower case expansion allowed. An expansion found via the contextual rules must be capitalized, except for noise words. Second, the algorithm tries conservatively, rather than greedily, to match multiple characters in an expansion term, addressing the problem illustrated with *NIST*, above. In addition, hyphens and slashes are allowed in acronyms, and are passed over silently in expanding them. If an expansion term is hyphenated, such as *Real-Time* from CRICCS (Center for *Real-Time* and Intelligent Complex Computing Systems), the algorithm can either treat *Real-Time* as two words, or as a single word, not requiring a *T* in the acronym.

Canonical. The canonical algorithm was derived from the canonical contextual, filtering the output list so that only acronym/expansion pairs that were found in canonical form were retained.

Simple Canonical. The simple canonical algorithm was an attempt to do away with most of the complexity of the contextual algorithm and its derivatives. Like the canonical algorithm, the simple canonical algorithm requires that the acronym be found in certain textual contexts, but it accepts fewer canonical patterns for acronym/expansion pairs, and fewer acronym patterns. The algorithm searches for the forms in the *Canonical Definition* row of Table 1 in the order they are listed.

When checking the validity of a potential expansion, the algorithm has a few acronym/expansion matching schemes. Each of these schemes recursively checks shorter expansions first. The matching schemes are performed as follows:

1) Uppercase strict: each letter in the acronym must be represented, in order, by an uppercase letter in the expansion. The expansion must begin with the first letter of the acronym.
2) Lowercase strict: each letter in the acronym must be represented, in order, by the first letter of a word in the expansion. The expansion must begin with the first letter of the acronym and must not contain uppercase letters.
3) Uppercase loose: the first word must begin with the first letter of the acronym and the last word must begin with a letter in the acronym. This scheme is extremely loose, and can result in expansions where some letters in the acronym are not matched at all.

The functions that check shorter expansions first remove words from the end of the expansion farthest from the acronym, then the functions call themselves with the modified expansion. Each function will remove a word from the beginning of the expansion if the expansion follows the acronym, or from the end of the expansion if the expansion precedes the acronym. If the shorter expansion passes the requirements, the algorithm returns the short expansion with the acronym as valid. For example, *Air Carrier Access Act (ACAA)* fits the pattern "expansion (ACRONYM)." Since *Air Carrier Access Act* passes the uppercase strict test, it is returned as the valid expansion for *ACAA*. While *Access Act* would pass the uppercase loose test for *ACAA*, it would not be returned because the uppercase strict test is performed first.

EVALUATION OF ALGORITHMS

In order to evaluate how well our algorithms correctly find all the acronyms that are explicitly defined in a set of documents, we use standard information retrieval measures. *Precision*, that is, *found correct/found total*, measures the accuracy of extraction, and *recall*, that is, *found correct/known correct*, measures the completeness of the extraction. For this evaluation we started with the 1M set, that is, the 936,550 military and government web pages that we processed for the Acrophile web database. From this set, we selected at random 170 pages that contained text and manually found all the acronyms with explicit definitions. These documents contain 353 defined acronyms, 10 with an ampersand or slash, and none with numbers or dashes. Variations in expansions that were accepted as correct were the omission or addition of an 's,' and differences in punctuation.

Table 2 shows recall and precision values for the four algorithms on the 353 acronyms in test set and on a subset containing the 328 acronyms of length three or higher.

	All Acronyms		**Length > 2**	
	Precision	Recall	Precision	Recall
Contextual	.89	.61	.96	.60
CanCon	.87	.84	.92	.84
Canonical	.96	.57	.99	.59
Simple	.94	.56	.99	.57

Table 2: Recall and precision on 170 sample docs

There were sixteen cases missed by all our algorithms because the expansion was too far (more than twenty words) away from the acronym. We do not expect any algorithm to get these, and other researchers do not include such cases [15][18]. The results excluding these cases can be seen in Table 3.

	All Acronyms		**Length > 2**	
	Precision	Recall	Precision	Recall
Contextual	.89	.63	.96	.63
CanCon	.87	.88	.92	.88
Canonical	.96	.60	.99	.61
Simple	.94	.59	.99	.60

Table 3: Excluding distances > 20 words

Precision is very high, especially on acronyms longer than two characters. Recall is considerably higher for the canonical contextual algorithm than the other three algorithms but with lower precision. As expected, the two canonical algorithms have lower recall but higher precision. The contextual algorithm has lower recall, and slightly higher precision than the canonical contextual algorithm, in a pattern indicating that a preponderance of its errors are on 2 letter acronyms. These results cannot be directly compared to the .93 recall and .98 precision found for acronyms longer than 2 characters in [15], and .91 recall and .68 precision in [18], and roughly .80 recall and .90 precision in [19], because these studies are based on different text, and possibly different criteria for correctness.

COMPARISON WITH HAND CRAFTED LISTS

Two web collections were chosen for the comparison. We tried to use the largest and best quality sites from which we could easily get and parse lists of acronyms and expansions. We used WWWAAS, the World Wide Web Acronym and Abbreviation server at University College Cork in Ireland [17] and Acronym Finder, Mountain Data Systems's acronym database [1]. From WWWAAS, the smaller of the two sites, we could extract the entire database by submitting a "." as a query. The output was converted from HTML to our format with lex. The items were not added to our database. For Acronym Finder, the larger site, we were not able to dump the entire database, but we were able to collect all the expansions for a test set to be described in the next section.

Size

First, Table 4 shows how our collection compares with the others in overall size. WWWAAS contains far fewer acronyms and expansions than our set. Acronym Finder contains more acronyms, but fewer expansions than we extracted from 1M set described above. Processing additional pages outside of the military and government domain would undoubtedly find more acronyms.

Algorithm	# Acronyms	# Exps	Avg Exps/Acro
Contextual	44,241	143,620	3.25
CanCon	51,726	161,686	3.13
Canonical	41,832	117,746	2.81
Simple	40,073	119,081	2.97
WWWAAS	12,108	17,753	1.47
Ac.Finder	60,000	127,000	2.17

Table 4: Number of acronyms and expansions extracted from 1M pages by each algorithm, and at 2 web sites

Evaluation method

To go beyond size and compare the correctness of different collections is much more difficult than comparing algorithms on a fixed set of data. A major challenge was in defining "correct." A usable criterion was to require that we could find the acronym in use on the web. By taking a random sample of 200 acronyms from each of our lists, we were able to determine that virtually all the acronyms in all the sets were real acronyms, that is, we were able to find them used as an acronym somewhere on the web. However, it looked as though some expansions might be erroneous, and we devised the following method to evaluate the accuracy of the set of expansions listed for an acronym.

The test samples of acronyms and expansions. We initially selected a sample of 55 acronyms for evaluating expansions. Forty acronyms were chosen to mimic the distribution of acronym length found in the small Wall Street Journal collection. Acronyms with length 2, 3, and 4 were generated randomly, while others were selected at random from a longer list of acronyms of that type. We added 5 acronyms containing numbers, 5 known to have a large number of expansions, and 5 with dashes or slashes.

For each of the 55 acronyms, we collected a pool of expansions from the two reference databases on the Web, and from our four algorithms, run on the 1M set. We also added all the additional expansions that came up in the crawling experiments discussed below. We later found that for 10 of the 55 acronyms, none of the systems found any expansions. These 10 were removed from the evaluation, leaving 45 acronyms in the test set.[1]

Criteria for correct expansions. Our criterion for a correct expansion was similar to that for a correct acronym, that is, that we could find at least one example on the web defining that expansion for that acronym. We hired evaluators to examine pages returned from an AltaVista [2] search for a query consisting of the acronym and the expansion. If they could find the acronym defined with the target expansion on any web page, using a list of explicitly defined criteria, it was accepted as correct. Otherwise, it was incorrect.

Scoring. We defined recall for this context as the number of correct expansions for an acronym found by one algorithm or system divided by the number of known correct expansions for that acronym found by all algorithms or systems evaluated. Similarly, we defined precision as the number of correct expansions for an acronym found by one algorithm or system, divided by the number of expansions, correct or incorrect, found by that algorithm or system. We then averaged across acronyms.

To obtain a range of recall/precision points, we ranked the expansions by a confidence score, which was a function of how many times the expansion was found for an acronym, and another factor which we found highly related to reliability – whether an occurrence is in one of the two canonical forms "expansion (ACRONYM)" or "ACRONYM (expansion)". Pilot research with the 1989 Wall Street Journal corpus showed that acronym/expansion pairs extracted from this frame were about five times more likely to be correct than pairs extracted from any other form. Therefore, we gave occurrences in this form more weight than

[1] Several patterns in our results make us doubt that our test set of 45 acronym is representative. First, the average number of expansions per acronym is much higher in the test set than in the complete set. We are in the process of judging a better corpus of 200 acronyms. This list includes most of the 45 acronyms from the present test set, plus acronyms chosen randomly from a list of acronyms found in the evaluated systems. These judgments will allow a more reliable evaluation.

occurrences in other forms by counting them as five occurrences.

An acronym's expansion with a count of 1 in a very large corpus is somewhat likely to be erroneous. Expansions with a count of 10 are much more likely to be correct and expansions with counts of 30 are even more likely to be correct. The higher the count we require, the better accuracy (precision) we can obtain. However, requiring higher counts also causes more legitimate expansions be missed. We can therefore get higher precision by requiring some threshold number of counts in order to accept an expansion for an acronym, but at the cost of lower recall. By varying this threshold, we obtain a range of recall-precision points for our evaluation below. The confidence scores are also used in the online search system, but they are transformed to $C/(C+2)$, in order to range between 0 and 1.

Note that weighting the count does not bias our measurements of recall and precision, it only affects how acronyms are grouped by confidence to get a range of recall/precision values.

Table 5 shows the total number of expansions found for the 45 acronym test set by each of the 4 tested algorithms and the two web sites. It also shows recall and precision. The contextual and canonical/contextual algorithms find the largest number of expansions for the test acronyms. Consistent with the analysis on the 170 documents, the simple and canonical algorithms have higher precision and lower recall. Acronym Finder has performance similar to our algorithms. A more complete picture of the situation can be seen in Figure 3.

Algorithm	# Exps	Precision	Recall
Contextual	1172	.75	.28
CanCon	1055	.76	.34
Canonical	573	.79	.21
Simple	344	.81	.25
WWWAAS	90	.84	.09
Acronym Finder	450	.76	.31

Table 5: Number of expansions, precision, and recall for each system, measured on 45 test acronyms

Figure 3 shows recall and precision curves for the four algorithms, evaluated on the 45 test acronyms whose expansions were all judged. The points on each curve show recall and precision at thresholds of 1, 2, 3, 4, 5, 10, 15, 20, 25, and 30, computed as described above. The recall and precision values in Table 5 correspond to the threshold 1 points on Figure 3. The graph shows that for all algorithms, it is possible to attain precision values in the .95-.97 range, but only at very low recall levels, that is, for the acronyms we have the most confidence in. The worst-performing algorithm is the contextual, with substantially lower values than the other values all along the recall precision curve. The canonical/contextual algorithm and the simple algorithm perform the best across most of the curve, except at the high recall end, where the canonical/contextual algorithm attains higher recall. In other words, the contextual rules of the canonical/contextual algorithm allow us to find more acronyms and/or expansions than we can find using canonical rules alone, but this non-canonical set also has more errors in it. The canonical algorithm falls between simple canonical and contextual algorithms in recall and precision.

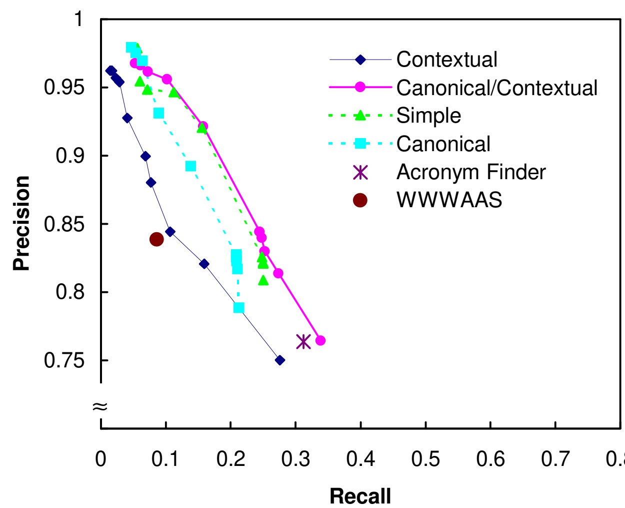

Figure 3: Recall and Precision on 45-acronym test

The unconnected points on Figure 3 show the recall and precision values we measured for the handcrafted web sites, on the 45 test acronyms. Each site contributes a single point to the graph rather than a curve, because we have no way to vary a threshold.

WWWAAS, the smaller site, falls at the low end of recall, with a recall of .09 and precision of .84. Although precision (.84) appears good, compared to the other values in Table 5 (all in the .70's), we see from the more complete recall-precision curves that this value is comparable to our worst-performing algorithm – the contextual – at a threshold of 3. Our best algorithm, the canonical/contextual, has recall of .25 at the comparable value of precision, and precision of .96 at the comparable recall level.

Acronym Finder, the larger site, had recall and precision values of .26 and .76, comparable to our best algorithm, the canonical/contextual, at a threshold of 1. These results confirm the hypothesis that our algorithms can create a corpus of acronyms and expansions that is comparable in quality to the best manually built site that we could evaluate.

Note that precision and especially recall values here are substantially lower than what we found in evaluating the extraction from 170 web pages. The difference is due to the different pool of expansions which were considered correct. We are certain that some of the acronym expansions we counted as incorrect were in fact correct, but were not found in the AltaVista search, resulting in lower precision.

The pool of correct expansions has an even larger effect in reducing recall. An expansion is counted as missed if any other evaluated system found the expansion, whether or not it was present in the set of documents input to the acronym extractors. This makes the set of correct expansions a moving target that grows the more we search. The crawling experiments below show that the same acronyms are used in many domains, and if we go beyond our military and government 1M set, more expansions will be found.

PROCESSING ADDITIONAL PAGES

This analysis addresses the extent to which we can find more expansions by searching the web for acronyms. We used the 55 test acronyms, submitted them as queries to AltaVista, and ran our extraction algorithms on the top 30 and 100 pages that were returned for each query. This process found many new expansions for the target acronyms. As an illustrative example, Table 6 shows all the expansions for the acronym *EWI*, as found by all the systems mentioned. WWWAAS does not appear in the table because it did not include the acronym *EWI*. The other manual site, Acronym Finder (AF) had three expansions listed, two correct and one incorrect. All four of our algorithms, run on the 1 million web pages, found the two correct expansions for *EWI* listed by Acronym Finder, and did not get the incorrect expansion. In addition, our algorithms found a third correct expansion, and all but the contextual algorithm found another incorrect expansion. The additional pages found by searching and crawling more than doubled the number of correct expansions. When 30 pages were processed for each acronym query, four new correct expansions and one incorrect expansion were found. When 100 pages were processed for each acronym query, another two correct expansions were found.

Expansion	**AF**	**Con**	**CanCon**	**Can**	**Simple**	**1M+30**	**1M+100**
Edison Welding Institute	+	+	+	+	+	+	+
Education With Industry	+	+	+	+	+	+	+
Electronic Warfare Intelligence	-						
Equal Width Increment		+	+	+	+	+	+
Explosive Waste Incinerator			-	-	-	-	-
Eijkman Winkler Institute						+	+
Elliott Wave International						+	+
European Wireless Institute						+	+
European Web Index						+	+
Edison Welding						-	-
Electro World Inc							+
Executive Women International							+

Table 6: Expansions for acronym *EWI*

In addition to finding more expansions for the target acronyms, extraction from the crawled pages found some new acronyms that had not been extracted before. For 1M+30, 318 new acronyms were found, and for 1M+100, 1120 new acronyms were found. None of these were the 55 acronyms targeted by the search.

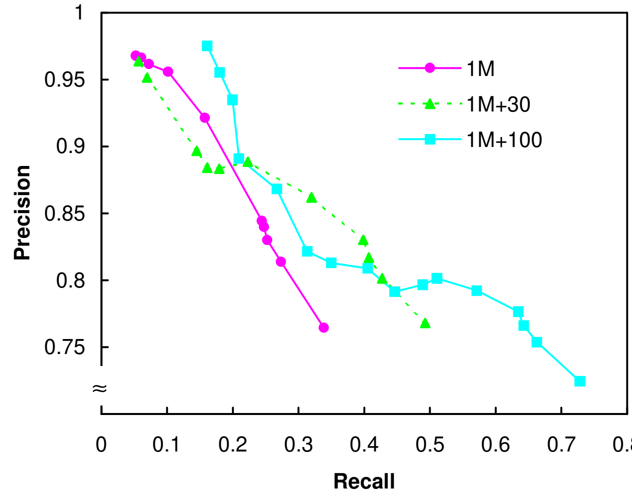

Figure 4: Adding source pages by searching for target acronyms

Figure 4 shows the recall precision curves for the canonical contextual algorithm, processing 30 crawled pages per acronym in addition to the basic 1M set, and 100 additional crawled pages per acronym, along with the old curve for the 1M set. This targeted crawling results in a huge increase in recall, without dropping precision except at the very highest recall levels – thresholds of 1. At a threshold of 2, precision (.75) is not appreciably lower than the precision for the 1M pages alone at a threshold of 1 (.76), but recall has almost doubled from .28 to .54.

As a control, we also measured the performance of the canonical contextual algorithm on comparably sized sets, consisting of the 1M set with the addition of either 55x30=1650, or 55x100=5500 randomly selected documents. We did not include these results on the graph in Figure 4, however. The results were so similar to that of the 1M set alone that they could not be seen separately on the graph.

CONCLUSIONS

We were able to build in a largely automated manner, a searchable dictionary of acronyms and expansions which rivals the quality of a good manually constructed dictionary of acronyms, by extracting acronyms and expansions from a large corpus of static web pages. We showed that we can increase the precision (accuracy) of our extraction by raising a threshold. Although this results in lower recall (coverage), we can increase recall by processing more pages. We can increase recall dramatically without loss of precision by processing web pages that are returned by a search for the acronyms that we have already found. This two-stage strategy results in a collection that is superior to any manually built database, and it can be kept up-to-date in an automated manner.

FUTURE WORK

Dynamic Extraction

Given the great efficiency and success of finding additional expansions for an acronym by searching for the acronym and extracting expansions from the top 100 web pages returned, we are planning to add a facility to do this online. This will not replace the static database, however. There are some acronyms which spell existing words (IS, TIDES) for which the acronym may not occur in the top 100 pages returned from a search.

We would like to have the system automatically crawl for pages containing known acronyms, to continue to find new expansions for our acronyms, and to discover new acronyms. We have found that our confidence scores get distorted by this process because the same web pages may be processed many times. Presently we remove duplicate URLs from the set of pages for one acronym, but we do not keep a master list to prevent processing the same page again in a later run.

HTML Parsing

Our simple method of ignoring material inside HTML tags could be improved. We lose several occurrences of acronym/expansion pairs defined within the ALT property of <IMG> tags, as in: <IMG SRC="image.gif" HEIGHT="50" WIDTH="50" ALT="Library of Congress (LOC)" >.

We also do not take advantage of the <ACRONYM> and <ABBR> tags, which allow a web author to declare acronyms and abbreviations as follows: <ACRONYM title="Rapid Eye Movement"> REM </ACRONYM> or <ABBR title="Y2K"> Year 2000</ABBR>. These tags are not yet in common usage, but if they become more widely used, we would want our extraction algorithms to be able to extract acronyms and abbreviations from them.

ACKNOWLEDGMENTS

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement numbers EEC-9209623 and EIA-9820309. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsor.

We thank Mike Molloy for information about Acronym Finder, and Morris Hirsch for an early version of the contextual algorithm. Thanks also to Don Byrd for his comments on a draft of this paper.

REFERENCES

1. Acronym Finder. http://www.AcronymFinder.com.

2. AltaVista. http://www.altavista.com.

3. *The American Heritage College Dictionary*, Third Edition. Boston: Houghton Mifflin Company, 1993.

4. Atomic Harvester. http://www.desktop-server.com/atomic.htm.

5. Dog fanciers acronym list. http://mx.nsu.ru/FAQ/F-dogs-acronym-list/Q0-0.html.

6. EmailSiphon is known by the evidence it leaves when it crawls archives for email addresses, purportedly for spamming purposes. See discussion in http://archives.list-universe.com/list-moderators/9802.

7. Fellbaum, Christiane. *WordNet: An Electronic Lexical Database*, Cambridge: MIT Press, 1998.

8. Giles, C. Lee, , Bollacker, Kurt D., and Lawrence, Steve. CiteSeer An Automatic Citation Indexing System, in *Digital Libraries 98*, New York: ACM Press, 1998, pp. 89-98.

9. Hearst, Marti. Automatic Acquisition of Hyponyms from Large Text Corpora, in *Proceedings of the Fourteenth International Conference on Computational Linguistics* (Nantes, France, July 1992).

10. Mad Cow disease list. http://www.maff.gov.uk/animalh/ bse/glossary.html.

11. MetaCrawler. http://www.metacrawler.com.

12. Molloy, Michael (Acronym Finder), personal communication. February, 2000.

13. Opaui Guide to Lists of Acronyms, Abbreviations, and Initialisms (http://spin.com.mx/~smarin/acro.html).

14. ResearchIndex. http://www.researchindex.com.

15. Taghva, Kazem, and Gilbreth, Jeff. Recognizing Acronyms and their Definitions. Technical Report 95-03, ISRI (Information Science Research Institute) UNLV, June, 1995. http://www.isri.unlv.edu/ir/publications/Taghva95-03.ps

16. Tkach, Daniel, ed. Text Mining Technology: Turning Information into Knowledge. IBM White Paper, 1998. http://www.software.ibm.com/data/miner/fortext/download/whiteweb.html.

17. World Wide Web Acronym and Abbreviation Server (WWWAAS). http://www.ucc.ie/cgi-bin/acronym.

18. Yeates, Stuart. Automatic extraction of acronyms from text. In *Proceedings of the Third New Zealand Computer Science Research Students' Conference*. Hamilton, New Zealand, April 1999, University of Waikato, pages 117-124. http://www.cs.waikato.ac.nz/~syeates/pubs/acroPaper.ps.gz

19. Yeates, Stuart, Bainbridge, David, and Witten, Ian. Using Compression to identify acronyms in text. Submitted to Data Compression Conference, DCC2000.