S-RAD: A Simple and Robust Abbreviation Dictionary

Eytan Adar
HP Laboratories, Palo Alto, CA
eytan@hpl.hp.com

Abstract

The Simple and Robust Abbreviation Dictionary (S-RAD) provides an easy to implement, high performance tool for the construction of a biomedical symbol dictionary. We describe and evaluate the algorithms of the system and apply it to the MEDLINE document set. The resulting dictionary represents a useful tool to researchers and provides programmers with a mechanism to disambiguate abbreviation symbols automatically.

1. Introduction

In the past few years there has emerged a significant interest in the use of existing textual material to augment the results of traditional biological and bioinformatics experimentation. This includes the use of text sources to determine gene-gene [Jen01][Ada02] and protein-protein interactions [Bla99], improving homology searches [Cha01] as well as dozens of other applications [Yan02][Cha02].

One of the main challenges in the analysis of text is the disambiguation of certain terms and symbols. This is especially difficult given the bias of scientists to abbreviate terminology. For example, while *angiotensin converting enzyme* very specifically defines a protein, the tendency of the scientific community is to use the symbol *ACE*. The reality, however, is that there is not one, but almost 20 terms that also map to ACE (*affinity capillary electrophoresis*, *acetylcholinesterase*, etc.). Anyone who has searched for an abbreviation or acronym[1] on PubMed or any similar system has probably discovered this on his or her own. While for a human it is not difficult, although sometimes time consuming, to determine which "meaning" was intended by scanning through the abstract, it is a far more difficult problem for an automated program to solve.

Due to this limitation, applications that automatically analyze textual collections frequently experience noise introduced by an inability to disambiguate between different abbreviations. For example, in an application recently built at our lab to correlate genes with various diseases [Ada02] it was critical to determine if a term such as *ER* meant *Estrogen Receptor* (an alias of the gene *ESR1*) or meant *Endoplasmic Reticulum.* The need to solve this problem led to the development of the system described here.

The Simple and Robust Abbreviation Dictionary (S-RAD) system provides a mechanism for building a dictionary of possible definitions for abbreviations, the clustering of those definitions, and the generation of a classifier for the disambiguation of new definitions. The resulting output can be used both through user-interface (see Section 4), or utilized by automated programs.

One of the main design principles for S-RAD was to make each component as easy to implement as possible while still providing highly accurate results. While substantial work exists to address the sub-steps of building dictionaries and classifiers, we feel that the system presented here uniquely provides both a simple and robust solution to the larger problem. In addition, we believe that each of the sub-modules of the system are also novel on their own and perform as well or beyond the level of existing solutions while reducing the complexity of implementation.

2. Related Work

With very few exceptions, work in abbreviation definition has primarily been centered with the information-extraction aspect of the problem. Finding definitions for abbreviations within the text is only the first part of dictionary building. Merging definitions, cross-linking, and providing disambiguation information are also necessary for generating a high quality dictionary.

Existing extraction systems fall into three broad categories: *natural-language based, rule based*, and *generalized string matching*.

[1] We will in general use the term abbreviation (a shortened form of a word or phrase) since acronyms (words formed from the initial letters of a phrase) are a subclass of abbreviations.

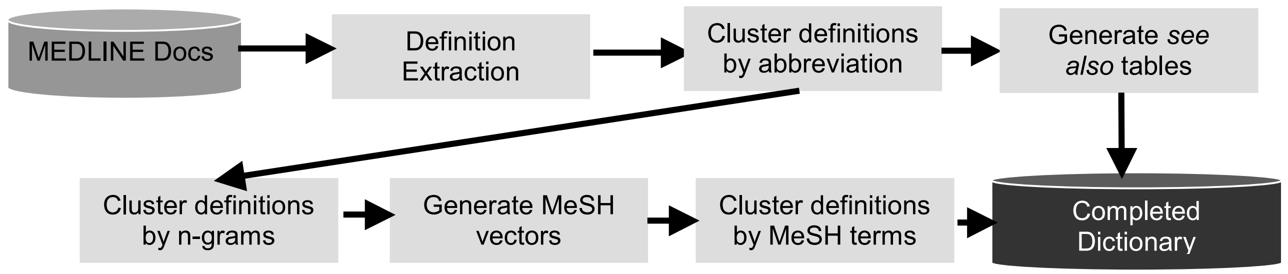

MEDLINE Docs
Definition Extraction
Cluster definitions by abbreviation
Generate *see also* tables
Cluster definitions by n-grams
Generate MeSH vectors
Cluster definitions by MeSH terms
Completed Dictionary

Figure 1: The flow of the S-RAD system. MEDLINE documents are used to generate an initial set of abbreviation definition pairs. These are clustered by abbreviation. The result is used to generate cross links between definitions as well as providing data for additional clustering based on n-grams and MeSH terms. The MeSH vectors and n-grams are retained for disambiguation purposes.

Natural-language based approaches attempt to use parts-of-speech tagging to find noun phrases around abbreviation symbols. Solutions such as AcroMed [Pus01] utilize this technique with great success (98% accuracy on an internal test bed). However, this approach is difficult to implement (requiring complex pattern matching algorithms) and generally depends on large, but generally limited, lexicons. For these reasons it does not meet our design objective of simplicity.

The second approach is one based on general pattern matching rules. Acrophile [Lar00], for example, uses patterns such as "uppercase letter, followed by lower case letter, followed by uppercase letter." Abbreviations matching such patterns are "validated" by a search on the WWW for other examples of the match. Unfortunately, the work necessary to define these rules result in lower accuracy due to the complexity of abbreviations. Other systems in this domain also include [Par01] and [Yu02].

The most popular algorithmic choice is generalized string matching. The work by [Tag95] is an early example in this category. The system described by [Tag95] for finding abbreviation definition in scanned documents utilized a Longest-Common-Subsequence (LCS) algorithm[2] to find all possible alignments of the abbreviation to the text followed by a very simple scoring rule based on matches. In [Cha02b], which also generates possibilities using LCS, matches are scored with a logistic regression on a feature vector. Every dimension in this vector represents an independent score for a feature determined to be relevant to abbreviations (whether definition starts at the beginning of a word, where the word boundaries are, etc.).

[2] A good source of information on LCS is [Gus97]

The abbreviation definition technique in [Bow00] (in the context of the Knowledge Extraction Program [Bow99]) also utilizes an LCS variant (the Ratcliff-Obershelp algorithm). Finally, the PNAD-CSS system [Yos00] provides a mechanism for building a protein name abbreviation dictionary again using an LCS like algorithm.

Non-LCS techniques include a very simple finite-state-automaton described in [Yea99] which is easy to implement but results in poor performance. The follow-up described in [Yea00] is interesting in the use of a compression algorithm to find abbreviations with better results but has an increase in implementation difficulty.

The S-RAD system described in this paper is most similar to the techniques of [Cha02b] and [Yos00]. However, while these systems generate an abbreviation to definition mapping we carry this further to construct a dictionary with clustered entries and the generation of classification rules for disambiguating definitions. Furthermore, while similar to LCS-based approaches, our technique is amenable to modifications that cut the search space and optimize the dictionary building process.

3. The Algorithms

Figure 1 illustrates the general architecture of the system. A collection of documents or abstracts is passed through the definition extraction module that finds abbreviations and generates potential definitions. A second module takes all definitions and groups them by abbreviation. The results of this step are used both for cross-linking abbreviations as well as the clustering of related

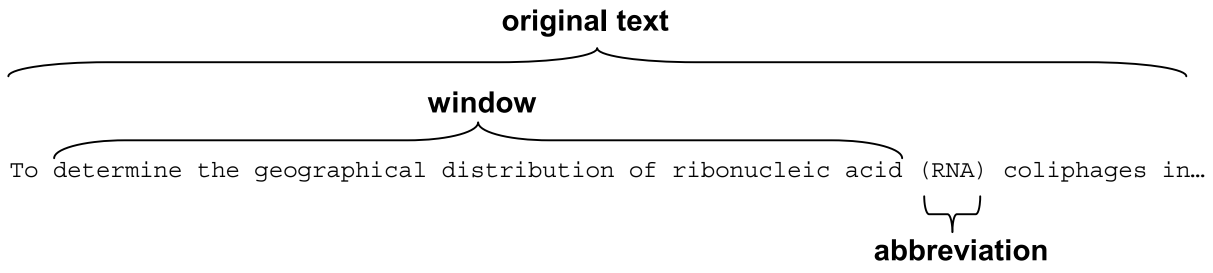

original text
window
To determine the geographical distribution of ribonucleic acid (RNA) coliphages in…
abbreviation

Figure 2: Finding a window in text

definitions. A secondary clustering based on the analysis of Medical Subject Headings (MeSH) of the source MEDLINE documents completes the dictionary. The following sections explore these algorithms in much greater detail.

3.1 Definition Extraction

The initial step in building the dictionary is one of information extraction through a three step process:

- Finding abbreviations in the source text as well as a *window* in which the definition can possibly be found.
- Generating *paths* through the text that may define the abbreviation.
- Scoring the paths to find the most likely definition.

3.1.1 Finding Definition Windows

There are many possible ways in which an abbreviation can be defined within text. The sentence, "Efferent feedback is provided by the release of acetylcholine (ACh) from olivocochlear …" [Fu02], appears to represent the most frequent pattern: <some text> definition (abbreviation) <some text>. However, we can also imagine the sentence, "Efferent feedback is provided by the release of ACh (acetylcholine) from olivocochlear …" and other variations. While multiple forms may be theoretically defined, the reality is that there are almost 5000 documents in the MEDLINE collection that define *ACh* in the first form, and none that define it in the second. It appears to be safe to assume that the first pattern is by far the most common, and that even if other forms exist, with enough documents we will inevitably run across an abstract that matches this pattern.

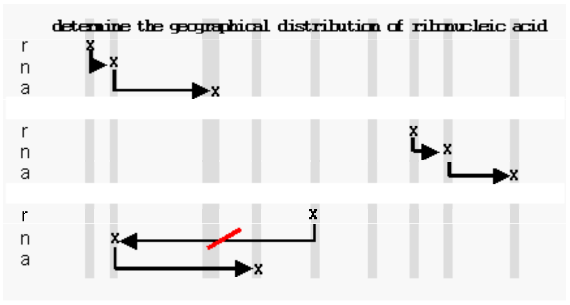


Figure 3: Finding 3 paths within the text. The first two are valid whereas the last is not.

Based on our observations of the MEDLINE data set, the extension module was designed to locate an abbreviation within a parenthesis and a window of text before this. A visual representation of this is in Figure 2. An abbreviation is defined as a single word (or hyphenated words) where there is at least one capital character. To determine the definition window we pick at maximum a number of words equal to *n* + buffer words before the parenthesis where *n* is the number of characters in the abbreviation and the buffer is four[3]. The buffer is used to account for words that are sometimes not included in the abbreviation. This is important in cases such as *AABB* where the definition *American Association of Blood Banks* includes "of." The algorithm crawls back through the text collecting until either the maximum number of words have been collected or a separator character is reached (such as a period or another parenthesis). In the first example sentence, the abbreviation to be defined is *ACh* and the window is "is provided by the release of acetylcholine."

All window text is "normalized" to lower case characters and non-alphanumeric characters are removed.

[3] There are alternate methods selecting this window (select *n* words where *n* = 2 * number of abbreviation characters, for example), but our solution appears to function well in our experience.

3.1.2 Generating Potential Paths

Once a window has been defined it is necessary to find potential definitions for the abbreviation within this window. Briefly, this is achieved by searching forward through the text window and matching characters from the abbreviation *in order*. This process allows us to build a path which pinpoints the location of the abbreviation characters within the text. A path can be converted into a definition by taking the span of text starting at the word with the first letter of the abbreviation and ending after the word where the last character of the abbreviation is found.

Figure 3 is an example of this approach for 3 possible paths which attempt to define RNA within the window "determine the geographical location of ribonucleic acid."

The first path locates the *r* in "determine." The path is *extended* and the first *n* is also found in "determine," and the first *a* is in "geographical." This means that the first proposed definition is "determine the geographical." The second path (the correct one) is similarly generated. The final path is considered invalid as paths are only built in a forward direction (inversions are not allowed).

There are some exceptions to the rule of only allowing forward-looking matches. For example, *PhD* is generally defined as *Doctor of Philosophy*, a definition which would never be found using our process. Nevertheless, from a few small experiments it appears that allowing for inversions does more harm than good in terms of accuracy.

Appendix A gives a pseudo-code implementation of this path building algorithm as well as a number of potential optimizations which are also discussed in the performance section.

3.1.3 Scoring Rules

Once the paths are found it is possible to quickly score them to find the most likely definition (or that one doesn't exist). Our scoring rules were chosen because they were extremely simple but were nonetheless able to accurately identify the correct abbreviation. There are many other potential rules one could apply. The specific rules we chose were:

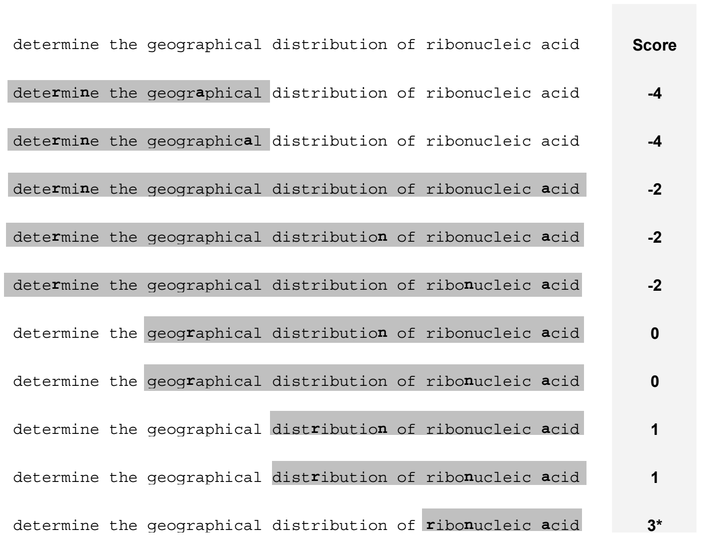
- For every abbreviation character that is at the start of a definition word add 1 to the score.
- For every extra word between the definition and the parenthesis where the abbreviation was found subtract 1. This rule rewarded definitions near the abbreviation in the window.
- A bonus point is awarded for definitions that are immediately adjacent to the parenthesis.
- The number of definition words should be less than or equal to the number abbreviation characters. For every extra word subtract 1.

The results of applying these scoring rules to the sample window are in Figure 4. As is evident from this small example, the score of the correct definition is higher than the incorrect definitions. We have also refined the scoring process so that 0 is the breakeven for likely definitions. While it is clear from the figure that not all positive scores are correct, negative scores were judged to be highly unlikely.

There are many alternative scoring methods that can be applied here. For example, depending on the characteristics of the data one may check a non-normalized window for capitalizations and attempt to match those to the abbreviation. Alternatively, it may be sufficient to simply look at the first character of the definition instead of every definition word. Other scoring rules can explicitly ignore *stop words* such as "to" or "of."

3.2 Clustering

Once the first step of extracting likely definitions has been completed it is necessary to group definitions that are related into clusters. Related definitions are frequently simple plurals of the basic abbreviation. For example, *Estrogen Receptor* and *Estrogen Receptors* are both valid



	Score
determine the geographical distribution of ribonucleic acid	
dete**r**mi**n**e the geogr**a**phical distribution of ribonucleic acid	-4
dete**r**mi**n**e the geographic**a**l distribution of ribonucleic acid	-4
dete**r**mi**n**e the geographical distribution of ribonucleic **a**cid	-2
dete**r**mine the geographical distributio**n** of ribonucleic **a**cid	-2
dete**r**mine the geographical distribution of ribo**n**ucleic **a**cid	-2
determine the geog**r**aphical distributio**n** of ribonucleic **a**cid	0
determine the geog**r**aphical distribution of ribo**n**ucleic **a**cid	0
determine the geographical dist**r**ibutio**n** of ribonucleic **a**cid	1
determine the geographical dist**r**ibution of ribo**n**ucleic **a**cid	1
determine the geographical distribution of **r**ibo**n**ucleic **a**cid	3*

Figure 4: Various paths and scores possible for the abbreviation RNA within the window. The last path (score = 3) was the top scoring path and also the correct definition.

definitions for *ER* (there are actually 86 such unique definitions). However, there are certain situations where definitions contain different words represent the same concept. An example of this is the abbreviation *EREs* that is defined as both *Estrogen Response Elements* and *Estrogen Responsive Elements*.

A possible approach to this problem is to reduce all terms in the definition to stems using a lexicon or an algorithm such as the Porter stemmer [Por80]. Given the enormity of biomedical vocabulary and all possible hyphenation possibilities for chemical names this is not the ideal approach. Instead we utilize two clustering techniques. The first, an n-gram based, approach attempts to find definitions with related stems. The second utilizes MeSH headings to further cluster definitions.

3.2.1 N-gram based

We use an n-gram based approach due to its versatility in comparing words stems. It is clear for a human seeing a set of definitions such as *Computer Tomography, Computed Tomographs, Computerised Tomographic, and Computetomographic* (some of the many definitions of *CT*) that they are all related. We would like to find an algorithm that is easy to implement and does not require any training to make the same determination.

The n-gram technique we utilize (or more specifically the tri-gram) breaks each definition into a set of three letter chunks that are then compared to each other. For example the definition ABCDE contains the tri-gram set: (ABC,BCD,CDE). The similarity between two tri-gram sets is then:

$$similarity_{NGRAM}(D_1, D_2) = \frac{\| D_1 \cap D_2 \|}{\sqrt{\| D_1 \|} * \sqrt{\| D_2 \|}}$$

The numerator represents the number of intersecting tri-grams between the definition D_1 and definition D_2. The denominator is a normalization factor based on the number of tri-grams in both definitions. An alternative to the similarity metric used is strategy like [Kra00] which utilizes BLAST to determine similarity of text.

For efficiency reasons we would prefer to not compare every definition to every other definition. Instead, we apply the following algorithm:

- Sort all definitions for an abbreviation by the number of instances of that definition.
- The most popular definition is the *primary* definition of the first cluster for that abbreviation.
- In order, starting from the second most popular definition look at all definitions.
  - Compare the definition to all existing clusters using the similarity metric applied to the primary definition in that cluster and the definition being considered.
  - Assign the new definition to the cluster with the maximum similarity score above the threshold (in our case .6).
  - If no similarity score is above threshold, the definition becomes the primary definition of a new cluster.

This repeated folding operation attempts to find the most common definition for each cluster and folds in alternative definitions. We will return to an evaluation of this technique and a description of how definition clusters are represented to the user later on.

3.2.2 MeSH Based

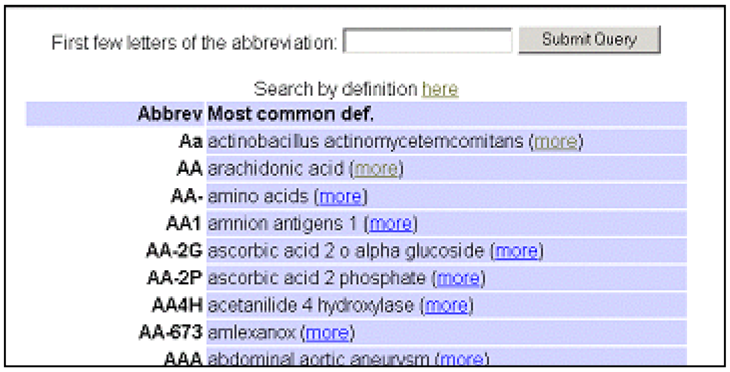
After completing the initial n-gram implementation it became clear that certain related definitions were being placed in separate clusters. This appeared to be the result of borderline scores that did not exceed the defined threshold. However, we determined that lowering the threshold score resulted in too many false clusters.

Instead, we opted to use the MeSH terms available in the data set to augment the initial results. MeSH terms are manually assigned annotations that are applied to each individual MEDLINE document. The headings represent which general concept(s) a document is related to (diseases, organism, etc.). While this feature is specific to MEDLINE, other similar annotations may be used in the same way in other collections.

To perform MeSH based clustering it is necessary to generate a representation of the MeSH concepts representing each initial cluster. This process includes taking each cluster, finding the initial documents from which the definitions were extracted, extracting the MeSH headings from those documents, and finally generating a vector representing the MeSH terms. For vector generation we utilized a standard Information Retrieval technique known as Term Frequency, Inverse Document Frequency (TFIDF) [Bae99] to compare each cluster. Each dimension (each MeSH term is represented by a dimension) in the vector is calculated by:

$$w_i = \frac{freq_i}{freq_{\max}} * \log \frac{N}{n_i}$$

5a

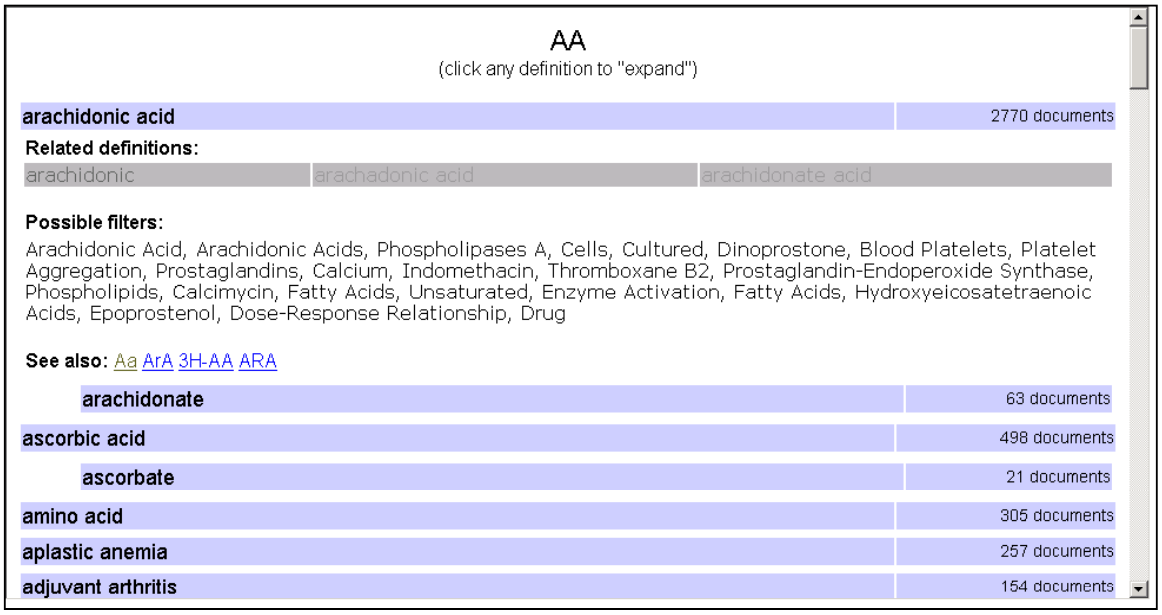


First few letters of the abbreviation: Submit Query

Search by definition here

Abbrev	Most common def.
Aa	actinobacillus actinomycetemcomitans (more)
AA	arachidonic acid (more)
AA-	amino acids (more)
AA1	amnion antigens 1 (more)
AA-2G	ascorbic acid 2 o alpha glucoside (more)
AA-2P	ascorbic acid 2 phosphate (more)
AA4H	acetanilide 4 hydroxylase (more)
AA-673	amlexanox (more)
AAA	abdominal aortic aneurysm (more)

5b



AA

(click any definition to "expand")

arachidonic acid	2770 documents

Related definitions:

arachidonic | arachadonic acid | arachidonate acid

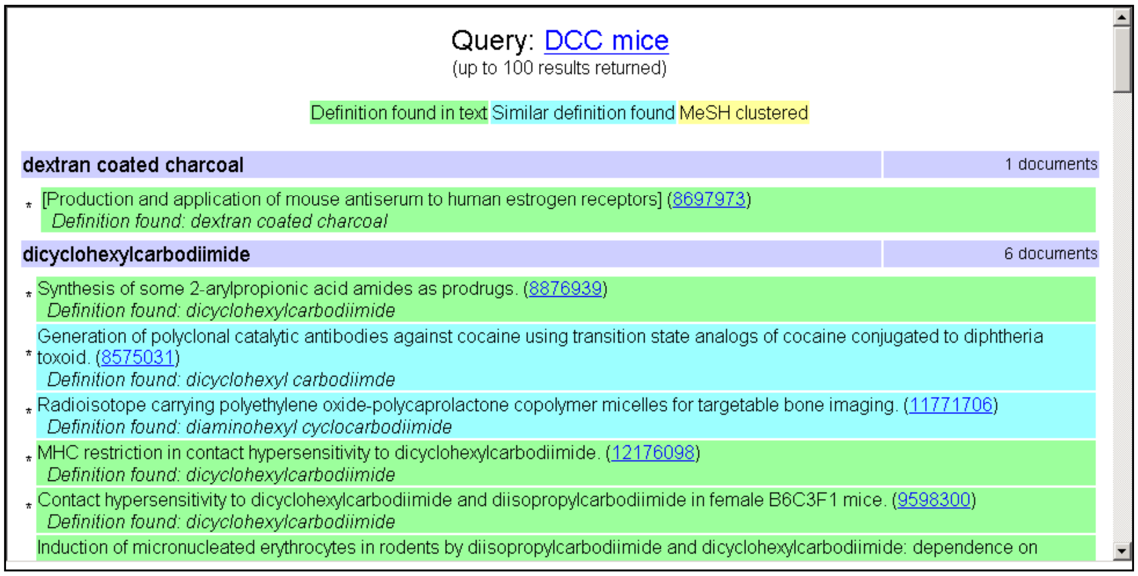
Possible filters:

Arachidonic Acid, Arachidonic Acids, Phospholipases A, Cells, Cultured, Dinoprostone, Blood Platelets, Platelet Aggregation, Prostaglandins, Calcium, Indomethacin, Thromboxane B2, Prostaglandin-Endoperoxide Synthase, Phospholipids, Calcimycin, Fatty Acids, Unsaturated, Enzyme Activation, Fatty Acids, Hydroxyeicosatetraenoic Acids, Epoprostenol, Dose-Response Relationship, Drug

See also: Aa ArA 3H-AA ARA

arachidonate	63 documents
ascorbic acid	498 documents
ascorbate	21 documents
amino acid	305 documents
aplastic anemia	257 documents
adjuvant arthritis	154 documents

5c



Query: DCC mice

(up to 100 results returned)

Definition found in text Similar definition found MeSH clustered

dextran coated charcoal	1 documents

* [Production and application of mouse antiserum to human estrogen receptors] (8697973)
  Definition found: dextran coated charcoal

dicyclohexylcarbodiimide	6 documents

* Synthesis of some 2-arylpropionic acid amides as prodrugs. (8876939)
  Definition found: dicyclohexylcarbodiimide
* Generation of polyclonal catalytic antibodies against cocaine using transition state analogs of cocaine conjugated to diphtheria toxoid. (8575031)
  Definition found: dicyclohexyl carbodiimde
* Radioisotope carrying polyethylene oxide-polycaprolactone copolymer micelles for targetable bone imaging. (11771706)
  Definition found: diaminohexyl cyclocarbodiimide
* MHC restriction in contact hypersensitivity to dicyclohexylcarbodiimide. (12176098)
  Definition found: dicyclohexylcarbodiimide
* Contact hypersensitivity to dicyclohexylcarbodiimide and diisopropylcarbodiimide in female B6C3F1 mice. (9598300)
  Definition found: dicyclohexylcarbodiimide

Induction of micronucleated erythrocytes in rodents by diisopropylcarbodiimide and dicyclohexylcarbodiimide: dependence on

Figure 5(a-c): Various captured screens from the web-based user interface. Figure 5a displays the results of a search for abbreviations beginning with “AA.” Figure 5b is the details screen for the abbreviation *AA*. Finally, Figure 5c is clustered PubMed search. The results of *DCC mice* has been clustered by different definitions of *DCC*.

Where the weight for the MeSH term i is calculated by the frequency of that term in the cluster normalized by the count of the most frequently occurring term. This is multiplied by $\log(N/n_i)$ where N is the total number of documents and n_i is the number of documents in which the term i appears (for this step N is actually the number of documents mentioning the abbreviation we want clustered and n_i is a subset of that group). This reduces the weight of MeSH terms that frequently occur in all documents and therefore do not hold much information.

The similarity metric that is applied is based on the angle between two vectors (representing two unique definition clusters). Specifically:

$$similarity_{COS}(\vec{D}_1, \vec{D}_2) = \frac{\sum D_{1,i} * D_{2,i}}{\| \vec{D}_1 \| * \| \vec{D}_2 \|}$$

The same algorithm used in the n-gram clustering is applied here utilizing the new cluster representations and new similarity metric (and a threshold of .3).

We found that MeSH based clustering was particularly useful for finding nested abbreviations. For example, *RGH* is defined as both *Rat GH* and *Rat Growth Hormone*. Because of the threshold of the n-gram clustering, definition folds of this type were missed.

An alternative to consider where MeSH or other equivalent annotations are not available is to simply use the text in the titles and abstracts of the articles to generate term vectors that can be compared using the similarity metric above. This is something we reserve for future work.

3.2.3 See Also

A final piece of analysis we apply to creating a dictionary is cross linking abbreviations with related definitions. This is mainly useful for abbreviations with capitalization variations or numbered. For example, *ACH, AcH, ACh*, and *Ach* are all used as abbreviations for *acetylcholine.* Similarly, *CCK*, *CCK-33* and *CCK-39* are three (of many) abbreviations for forms of *cholecystokinin*.

To cross-link definitions and generate a "see also list" we find all equivalent definitions in different abbreviations in one sweep across the formed dictionary. This additional information is made available to the user through the user interface.

3.3 Symbol Disambiguation

For an individual wishing to make use of the constructed dictionary, one of the most challenging problems is classifying new abbreviations when no definition can be extracted from the text. Users may also wish to have MEDLINE results clustered or filtered quickly based on the context of various definitions senses.

Luckily, we can reuse the MeSH vectors generated for clustering to correctly classify new abbreviations into the appropriate definition cluster. A new document can then be compared to previous MeSH vectors and bucketed correctly. This approach provides 80% classification accuracy (much higher for certain definitions) and will be discussed again in the results section.

The results of [Ada02] demonstrate an application of the n-gram similarity metric. Here, the system determines if expansions of common gene symbols found in a subset of MEDLINE documents are in fact the gene name. This technique provides a mechanism for determining which actual genes (rather than abbreviations that sometimes coincide with genes) are mentioned frequently

4. Interface

The dictionary output of the S-RAD system is maintained in a database and is processed into a graphical interface through a series of web pages. The interface we have created[4] gives our users access to the various annotations generated by the dictionary building modules.

Figure 5a-c illustrates various screens from this interface. Users are able to search for both abbreviations and definitions. A search for abbreviations beginning with "AA" are returned in a window such as that in Figure 5a. Next to each matches, the most common definition is shown. Clicking on the "more" link opens up a summary screen. Figure 5b is a summary screen for the abbreviation *AA*.

The page initially displays all the main definitions of each cluster for the abbreviation. Clicking on each of these bars (as was done for *arachiodonic acid*) causes the interface to dynamically expand the screen to include other definition features. Related definitions are those folded by the n-gram clustering technique. The different colors of these definitions represent how similar (based on the similarity metric) they are to the main definition. For example, *arachiodonic* is considered more similar to

[4] Available at: www.hpl.hp.com/shl/projects/abbrev.html

arachiodonic acid then is *arachiodate acid* (although both are very similar).

The possible filters are the various concepts that are unique to a particular definition. For the MEDLINE data set these are simply the MeSH terms that were produced by vector definition in the MeSH clustering technique. A user wishing to find only documents mentioning an abbreviation within a given context may make use of these filter terms to narrow down the MEDLINE search.

Definitions with cross-links will display these links under a "see also" heading. The user can easily navigate between summary screens in order to find alternate abbreviations for the definitions they are searching for. Those definitions that were determined to be relevant by the MeSH clustering are listed underneath the main definition with a slight indentation. For example, *ascorbate* is displayed in this way under *ascorbic acid*.

Finally, Figure 5c illustrates a sample application built on the disambiguation capabilities of the S-RAD system. Users are able to issue queries against the MEDLINE (in the standard PubMed syntax). They are also able to determine which abbreviation should be used to cluster the results. In the figure, the user has issued a query for documents containing *DCC* and *mice*, and would like the results clustered into the various meanings of *DCC*. Each document is placed under the definition to which it relates (either through exact matches, n-gram similarity, or MeSH similarity). In this way users can quickly narrow down their results to a specific definition.

5. Results

To evaluate the system we have run all the algorithms presented here against MEDLINE data containing all documents up to January 2002. This dataset represents over 11,253,125 documents from which 5,186,441 potential abbreviations were found (in document titles and abstracts only). All abbreviation/definition pairs scored below 0 were excluded resulting in 3,960,168 usable abbreviation/definition pairs. The count of unique pairs where there were at least two instances of a given definition was 193,103. Of these, 136,082 were "main" definitions, whereas the remainder were folded using one of the clustering techniques.

5.1 System Accuracy

Of the 136,082 main definitions 644 (randomly selected) definitions were manually tested for correctness. This evaluation indicated a 96% accuracy rate for those abbreviations selected. Of the 54,399 definitions that were folded in by n-gram clustering 555 were tested. Of those, 550 were considered legitimate (the two clusters were related) resulting in 99% accuracy rate. Finally, of the 2,622 MeSH folded definitions, 500 were tested. With this subset we found 76% accuracy.

The low score for the MeSH evaluation is partially a function of the evaluation strategy. A decision was only counted as correct if the two definitions were clearly synonymous. Unfortunately there were various instances in which definitions were related but not synonymous. For example, *energy expenditure* and *endurance exercise*. In addition, we have generally erred on the conservative side when it was not clear if the definitions were related during our manual evaluation. Despite this lower score we feel it is important to display these folded definitions to the user since they provide hints as to how searches can be modified.

While it is simpler to evaluate accuracy for the system it is a much harder problem to determine recall since there is no established dictionary to compare to. To get some sense of the overlap of the S-RAD algorithms to existing dictionaries we have used the abbreviations available in the Unified Medical Language System (UMLS) as a basis of comparison. The UMLS data set includes 10,161 definition pairs. Of those, 547 were used in the overlap experiment.

Of the 547, our dictionary included 197 definitions as main definition matches, 23 were included in the n-gram folded definitions, and 3 were found in the MeSH folded set. An additional 23 were matched by applying the n-gram similarity of the UMLS definition to the main definitions. Finally, 93 of the 547 abbreviations were found only once in the MEDLINE data set and were discarded from the final dictionary. The total of 339 definitions that were found in some way in our data set indicates an overlap of 62%. This score may be the result of the construction of the UMLS dictionary. A simple test shows that UMLS abbreviations are on the average 5.6 characters in length whereas our average is 4.5 characters (a difference of 25%) this indicates some differences in the terms defined in UMLS and those actually used in the literature. Additionally, many definitions from the test set were only defined once in 11 million documents. This leads us to believe that other UMLS definitions cannot be found at all in the MEDLINE data set[5]. It is difficult to test this hypothesis as the PubMed interface to MEDLINE transforms terms that produce no hits (through a UMLS

[5] A very extensive analysis of the differences between abbreviations in UMLS and MEDLINE is available in [Liu02b].

lexicon) into alternate symbols. Thus, significant manual work is necessary to confirm this.

One final data point can be obtained by applying our algorithms to the AcroMed Gold Standard data set [Med02]. It is important to note, however, that such a metric is not a useful measure of performance for a dictionary building task. The data set contains 201 documents similar to those available in MEDLINE. However, the abbreviations are never repeated nor are definition variants. Systems such as ours, that build the dictionary based on repeated evidence for added accuracy are not applicable.

Regardless, others have used this set so we provide our results for comparison. Running our algorithm on the AcroMed data finds 144 abbreviations. With the threshold at 0, a manual review for correctness reveals an accuracy of 86% and the recall of 88%. It should be noted that of the 17 “incorrect” definitions, 14 are actually present as main definitions in the complete dictionary. Generated from the MEDLINE data. This suggests that given enough data 98% of the abbreviations in the AcroMed data set would have been correctly defined.

5.2 Disambiguation Performance

Performance for automated disambiguation based on MeSH similarity was measured by generating a training/test subsets on the dictionary data. Specifically, abbreviations that had 2 or more definitions with 50 or more documents in which each definition was found were considered valid. This resulted in 1,183 abbreviations to be used for this analysis.

For each abbreviation 70% of the documents were used to generate the MeSH vectors. Documents were grouped by definition, MeSH terms from those documents were extracted, and the vector weighting equation described above was applied. Each of the remaining documents (the 30%) was similarly transformed into a MeSH vector. The cosine similarity metric was the applied to this vector and all definition vectors and the document was classified into the highest scoring category. Of the 391,326 documents in the test set, 286,203, or 73%, were classified correctly.

In the initial experiment we used only data that was not MeSH clustered. That is, abbreviation definitions with similar MeSH vectors were not merged. Once merging was enabled performance increased. Despite the moderate performance of MeSH clustering in terms of accuracy, we found that the 170 suggested merges where largely correct. This is most likely due to the fact that only definitions with many documents (and therefore many MeSH headings) were used allowing for higher quality vector. With this adjustment, 314,340 test documents were classified correctly (or 80%).

Figure 6 depicts a histogram that buckets abbreviations with similar accuracy together for both the initial test and MeSH folded test. We see a substantial spike after the .8 (80% mark). In fact, the median accuracy score for the MeSH folded test is 88%. Using individual accuracy marks it is possible to label each classifier in terms of correctness. A user or programmer wishing to make use of the training data may then use these scores to determine if a classifier is high enough quality for specific applications.

5.3 Algorithmic Performance

The most computationally intensive S-RAD module is the initial path-building phase. While it has a high potential computational cost given unbounded abbreviation and window sizes ($\Theta(mn)$ for the optimal LCS algorithm), the reality is that the properties of the data limit this worst

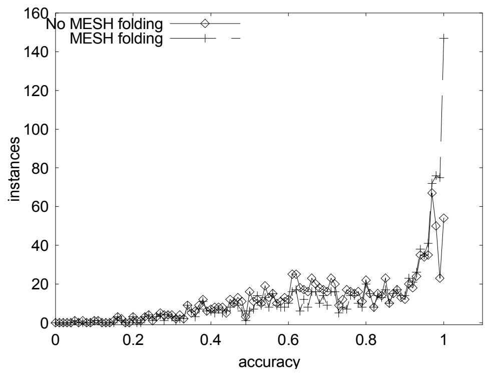


Figure 6: Accuracy of classification with and without MeSH folding.

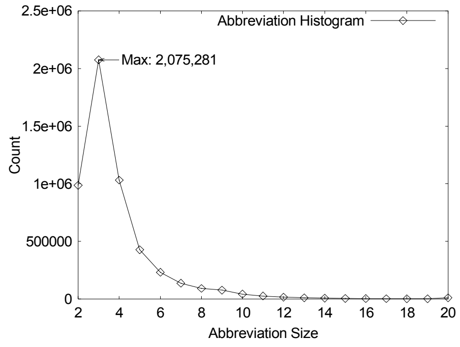


Figure 7: The distribution of abbreviation sizes in all analyzed documents.

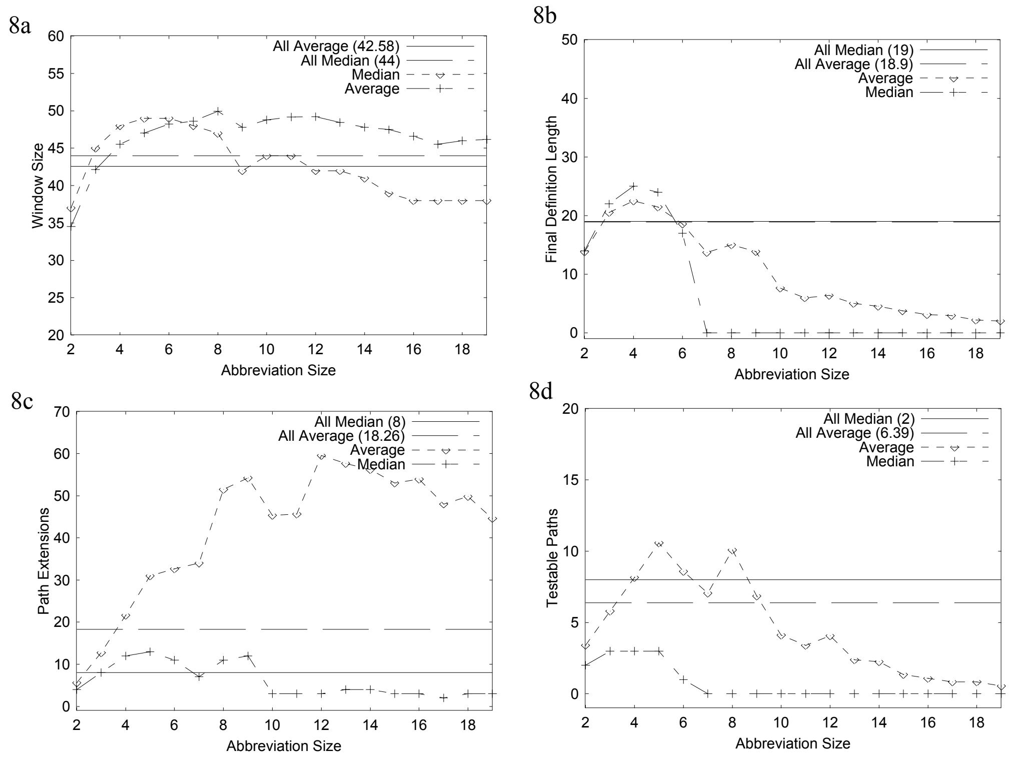


Figure 8a-d: The distributions of various performance metrics (a: window size, b: definition length, c: path extensions, and d: testable paths) as a function of the length (in characters) of the abbreviation.

case and various optimizations can be applied for better performance.

Figure 7 illustrates a histogram of abbreviation sizes found in the document set. The average for this distribution is 3.8 characters with a median of 3.

Figure 8a-d compares various metrics as a function of abbreviation size. Window size (8a) very quickly stabilizes. This is due largely to our restriction that a window extend only as far back as a sentence delimiting character. Definition length (8b) grows with abbreviation size for a short while but subsequently drops. This is most likely due to the decreasing likelihood that long abbreviation (an oxymoron) candidates are actually abbreviations. Figures 8c and 8d are representation of the number of path extensions and testable paths respectively. Recall that each path is extended for each abbreviation character. Each time a path can be added to we increase the path extension count. As discussed earlier we see that the average number of path extensions increases with abbreviation size. However, the median remains relatively stable and is additionally contained by the short average abbreviations. Similarly, the number of testable paths is fairly well contained.

These two facts lead to one possible optimization. Figure 9 illustrates the accuracy of definition extraction depending on the number of path extensions and paths necessary to find the definition in a window. This performance curve allows us to stop extending and building paths early since most definitions are found without much work. Of the 591 tested abbreviation/definition pairs, 90% accuracy was achieved with 47 path extensions or 18 paths. At 50 paths or 142 extensions the system accurately found 98% of all definitions. Since the actual window to compare varies from document to document it likely that the 2% would have been found elsewhere in a window more conducive to fast path-building. We can speed up the algorithm greatly, by eliminating worst-case extractions through a threshold on the number of extensions to attempt before giving up.

The S-RAD algorithms presented were entirely implemented in Perl and executed on a single-processor 900MHz Pentium III Linux server. The most time consuming task, extracting the definitions, took under 24 hours. Folding was under 4 hours.

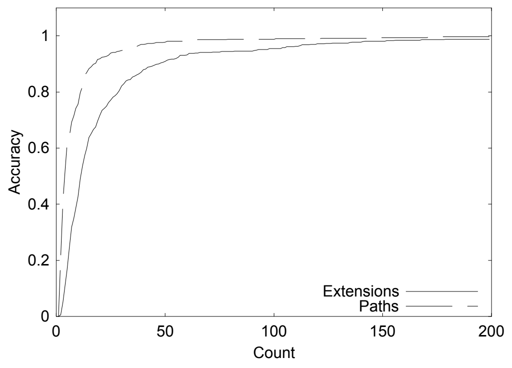


Figure 9: Accuracy of results as a function of paths tested and number of extensions.

6. Conclusions

The S-RAD system represents a simple, easy to implement, and accurate toolset for building biomedical[6] databases. The byproducts of the dictionary building effort are a useful set of classification tools that classify new documents with undefined abbreviations in a simple manner.

We believe that the approach we describe will allow those wishing to analyze biomedical text collections with a simple effective way to solve common problems.

We have placed the demonstration version of the system at: http://www.hpl.hp.com/shl/projects/abbrev.html for users to try.

7. Acknowledgements

Thanks to Lada Adamic, Sara Dubowsky, Rajan Lukose Dennis Wilkinson and Bernardo Huberman for their comments, advice, support, and use of this system.

Appendix A: Path Generation

Here we describe a simplified version of the path finding algorithm. Optimizations have been removed for clarity, but are discussed briefly where appropriate. The general function of the algorithm is to build a list of possible paths that "explains" the abbreviation *A* in the window W. A path is a unique collection of ordered indexes placing the abbreviation characters within the window. If the window was "AB CD" and the abbreviation was "AC" one path (in this case the only path) would simply be [0,3].

[6] Although we have applied the algorithms (obviously, without the MeSH clustering) to the entire HP website to generate an abbreviation server for internal purposes.

The first portion of the algorithm (Figure 10), lines 1-3, constructs an index *I*, holding the indexes of all appearances of each character. Clearly, this can be optimized to only hold the map for those characters that are actually in the abbreviation. The list of occurrences for character *k* is then *I*[*k*], and the *n*th occurrence of character k is accessed by *I*[*k,n*].

The next step of the algorithm is to loop over all characters in the abbreviation, gradually building the set of paths through iteration. The empty path set *P* is seeded with all the occurrences of the first character (lines 6-10). If this set is still empty null is returned. Once the path set is seeded we can extend the path. The algorithm loops through every path currently in the path set (line 13) and finds all instances of the next character in the abbreviation. Where the algorithm described here and the one actually implemented differ is the sensitivity to missing characters. In the simplified version above, if any character is missing (the path can not be extended), null is returned. Clearly, this may be undesirable for certain situations and it is here that modifications can be made to support "fuzzy" matching. Our implemented system, for example, accepts missing numeric characters.

For all occurrences of the next character in the text a decision is made whether the index is after the last index

```
BUILD-PATHS(A,W)
 1  for i ← 0 to length[W]
 2      c ← W[i]
 3      I[c][length[I[c]]] ← i
 4  for j ← 0 to length[A]
 5      c ← A[j]
 6      if length[P] = 0
 7          for i ← 0 to length[I[c]]
 8              P[length[P],0] ← I[c,i]
 9          if length[P] = 0
10              return null
11      else
12              N ← null
13              for i ← 0 to length[P]
14                  P_i ← P[i]
15                  if length[I[c]] = 0
16                      return null
17                  for k ← 0 to length[I[c]]
18                      l_i ← P_i [length[P_i] - 1]
19                      if I[c,k] > l_i
20                          P_new ← copy[P_i]
21                          P_new [length[P_new]] ← [c,k]
22                          N[length[N]] ← P_new
23              P ← N
24  return P
```

Figure 10: the BUILD-PATHS algorithm

in the path (lines 18-19). Instances of the abbreviation character prior to the last index are ignored as they represent an invalid match. If the condition is satisfied a copy of the path is extended and stored. The loop repeats until all characters are accounted for. In this way the set *P* always contains valid paths, and the result includes only full paths.

Once all paths have been built, each is scored based on the scoring metrics described in the paper, and the maximum matching path is returned as the most likely definition candidate. Additional optimizations to this algorithm include the scoring of paths as they are being built. Doing this allows us to ignore paths early on when we know they will not perform better than the current maximum scoring definition.

References

[Ada02] Adamic, L.A., et. al., "A Literature Based Method for Identifying Gene-Disease Connections," IEEE Computer Society Bioinformatics Conference, Stanford, CA, August 14-16, 2002.

[Bae99] Baeza-Yates, R. and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, Harlow, England, 1999.

[Bla99] Blaschke C., et. al., "Automatic extraction of biological information from scientific text: protein-protein interactions," *7th International Conference on Intelligent Systems for Molecular Biology,* Heidelberg, Germany, August 6-10, 1999.

[Bow99]Bowden, P.R., "Automatic Glossary Construction for Technical Papers," Nottingham Trent University, Department Working Paper, December 1999.

[Bow00]Bowden, P.R., et. al, "Automatic Acronym Acquisitions in a Knowledge Extraction Program," Unpublished manuscript.

[Bow96]Bowden, P.R., et. al., "Dictionaryless English Plural Noun Singularisation Using a Corpus-Based List of Irregular Forms," *17th International Conference on English Language Research on Computerized Corpora*, Stockholm, May 15-19, 1996.

[Cha01] Chang, J.T., et. al., "Including biological literature improves homology search," *Proceedings of the Pacific Symposium on Biocomputing* 2001, 5:274-383.

[Cha02] Chang, J.T., "NLP in Bioinformatics," http://smi-web.stanford.edu/people/jchang/bionlp.html

[Cha02b] Chang., J.T., et. al., "Creating an Online Dictionary of Abbreviations from MEDLINE," *Journal of the American Medical Informatics Association*, preprint.

[Fu02] Fuchs, P., "The synaptic physiology of cochlear hair cells," *Audiology and Neuro-Otology*, 7(1):40-4;

[Gus97] Gusfield, D., *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, New York, 1997, pp. 227-35.

[Jen01] Jenssen., T.K., et. al., "A Literature Network of human genes for high-throughput analysis of gene expression," *Nature Genetics,* 28:21-28.

[Kra00] Krauthammer, M., et. al., "Using BLAST for identifying gene and protein names in journal articles," *Gene*, 259:245-52.

[Lar00] Larkey., L.S., et. al., "Acrophile: An Automated Acronym Extractor and Server," *5th ACM Conference on Digital Libraries*, San Antonio, TX, June 2-7, 2000.

[Liu02] Liu, H., et. al., "Disambiguating Ambiguous Biomedical Terms in Biomedical Narrative Text: An Unsupervised Method," *Journal of Biomedical Informatics*, 34:249-261.

[Liu02b] Liu, H. et. al, "A Study of Abbreviations in MEDLINE Abstracts," *American Medical Informatics Association Symposium*, San Antonio, TX, November 9-13, 2002.

[Med02] http://www.medstract.org

[Par01] Park, Y., and R.J. Byrd, "Hybrid text mining for finding abbreviations and their definitions," *Empirical Methods in Natural Language Processing*, Pittsburgh, PA, June 3-4, 2001.

[Por80] Porter, M.F., 1980, An algorithm for suffix stripping, *Program*, 14(3) :130-137.

[Pus01] Pustejovsky, J., et. al., "Extraction and Disambiguation of Acronym-Meaning Pairs in MEDLINE," *10th World Congress on Health and Medical Informatics*, London, September 2-5, 2001.

[Tag95] Taghva, K., and J. Gilbreth, "Recognizing Acronyms and their Definitions," Information Science Research Institute, University of Nevada, TR 95-03, June 1995.

[Yan02] Yandell, M. D., and W. J. Marjoros, "Genomics and Natural Language Processing," *Nature Reviews Genetics*, 3:601-609.

[Yea99] Yeates, S., "Automatic Extraction of Acronyms from Text," *New Zealand Computer Science Research Students' Conference*, pp. 117-124, 1999.

[Yea00] Yeates, S., et. al., "Using compression to identify acronyms in text," *Data Compression Conference*, Snowbird, UT, March 28-30, 2000.

[Yos00] Yoshida, M., et. al., "PNAD_CSS: a workbench for constructing a protein name abbreviation dictionary," Bioinformatics, 16(2):169-175.

[Yu02] Yu, H., et. al, "Mapping Abbreviations to Full Forms in Biomedical Articles," *Journal of the American Medical Informatics Association*, 9:262-272.