Bioinformatics Advance Access published January 22, 2004

SaRAD: A Simple and Robust Abbreviation Dictionary

Eytan Adar[1]

[1]*HP Laboratories, 1501 Page Mill Rd., Palo Alto, CA 94304, USA*

ABSTRACT

Motivation: Due to recent interest in the use of textual material to augment traditional experiments it has become necessary to automatically cluster, classify, and filter natural language information.

Results: The Simple and Robust Abbreviation Dictionary (SaRAD) provides an easy to implement, high performance tool for the construction of a biomedical symbol dictionary. The algorithms, applied to the MEDLINE document set, result in a high quality dictionary and toolset to disambiguate abbreviation symbols automatically.

Availability: The SaRAD dictionary is available as a web based demonstration, and in pseudo-code form.

Contact: eytan@hpl.hp.com

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 (Jenssen et al., 2001; Adamic et al., 2002) and protein-protein interactions (Blaschke et al., 1999), improving homology searches (Chang et al., 2001) as well as dozens of other applications (Yandell and Majoros, 2002; Chang et al., 2002).

One of the main challenges in the analysis of text is the disambiguation of 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*. Unfortunately, there is not one but almost 20 expansions for ACE (*affinity capillary electrophoresis*, *acetylcholinesterase*, etc.). While for a human it is not difficult, although sometimes time consuming, to determine which "meaning" was intended by scanning through a PubMed abstract or similar search result, it is far more difficult problem for an automated program.

Due to this limitation, applications that automatically analyze textual collections frequently experience noise introduced by an inability to disambiguate abbreviations. For example, in an application recently built at our lab to correlate genes with various diseases (Adamic et al., 2002) it was critical to determine if symbols were gene symbols.

The Simple and Robust Abbreviation Dictionary (SaRAD) 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.

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 been centered on 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 *multiple alignment of text*.

Natural-language based approaches attempt to use parts-of-speech tagging to find noun phrases around abbreviation symbols. Solutions such as AcroMed (Pustejovsky et al., 2001) 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 limited, lexicons.

The second approach is one based on general pattern matching rules. Acrophile (Larkey et al., 2000), 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 results in lower accuracy due to the complexity of abbreviations. Other systems in this domain also include (Park and Byrd, 2001) and (Yu et al., 2002).

The most popular algorithmic choice is multiple alignment of text. The work by (Taghva and Gilbreth, 1995; Chang et al., 2002; Bowden, 1999; Bowden et al., 2000; Yoshida et al., 2000) and most recently (Schwartz and Hearst, 2003) are examples in this category. The system described by (Taghva and Gilbreth, 1995) for finding abbreviation definition in scanned documents utilized a Longest-Common-Subsequence (LCS) algorithm (Gusfield, 1997) to find all possible alignments of the abbreviation to the text followed by a very simple scoring rule based on matches.

Non-LCS techniques include a very simple finite-state-automaton described in (Yeates, 1999) which is easy to implement but results in poor performance. The follow-up described in (Yeates et al., 2000) is interesting in the use of a compression algorithm to find abbreviations with better results but has an increase in implementation difficulty.

The SaRAD system described in this paper is most similar to the techniques of (Chang et al., 2002; Yoshida et al., 2000). 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

The SaRAD system contains a number of core modules a) definition extraction which find likely abbreviations, b) n-gram based clustering is then used to group related definition, c) and finally MeSH based clustering further refines these groups. As a side effect of this process the information generated (document and MeSH vectors) is used for disambiguation.

3.1 Definition Extraction

The initial step in building the dictionary is one of information extraction through a three-step process. These are a) Finding abbreviations in the source text as well as a *window* in which the definition can possibly be found, b) Generating *paths* through the text that may define the abbreviation, and c) finding 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 …" (Fuchs, 2002), 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 in theory, multiple forms may be 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.

Because the first pattern is by far the most common the extension module was designed to locate an abbreviation within a parenthesis and a window of text before this. 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. Alternate methods exists for selecting this window (e.g. select n words where n = 2 * number of abbreviation characters), but our solution appears to function well in our experience. The buffer is used to account for words that are sometimes not included in the abbreviation (such as "of" in the definition for *AABB - American Association of Blood Banks*). The algorithm searches back through the text collecting words 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.

3.1.2 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

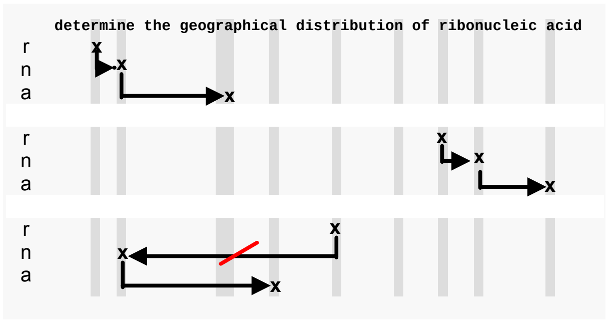


Figure 1: Three paths locating *RNA* within the window. The first path locates the *r* in "determine." The path is *extended* and the first *n* is also found in "determine," and the first *a* is in "geographical" making the first proposed definition is "determine the geographical." The second path (the correct one) is similarly generated. The final path is considered invalid as paths are only built in a forward direction (inversions are not allowed).

the abbreviation *in order*. This model allows us to pinpoints the location of the abbreviation characters within the text, and as we will later show, facilitates various optimizations. 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 1 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.”

Paths are only constructed in a forward direction. Exceptions such as *Doctor of Philosophy (PhD)* appear rare enough that allowing for inversions does more harm than good in terms of accuracy.

The Web supplement gives a pseudo-code implementation of the path building algorithm.

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. 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.

For the RNA example above the 9 incorrect paths range in score from –4 to 1 (7 have a score of 0 or below), whereas the correct path has a score of 3. We have refined the scoring process so that 0 is a strong threshold. While not all definitions with positive scores are correct, those scoring less than 0 are highly unlikely and are ignored in our implementation.

The selected rules were evolved through experimentation and appear to share features in common with the work of others (Chang et al, 2002; Schwartz and Hearst, 2003) who offer additional alternatives.

3.2 Clustering

We believe that it is a necessary function of an abbreviation dictionary 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 definitions for *ER* (there are actually 86 such unique definitions). However, there are certain situations where definitions containing 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 (Porter, 1980). Given the enormity of biomedical vocabulary and all possible hyphenation possibilities for chemical names this is not the ideal approach. Instead we utilize various clustering techniques.

3.2.1 N-gram based clustering

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. A suitable approach that is easy to implement and does not require training is n-gram clustering.

The n-gram technique we utilize (or more specifically, tri-gram) breaks each definition into a set of three letter groups that are then compared to each other. For example the definition ABCDE contains the tri-gram set: {ABC,BCD,CDE}. The set is order independent and only contains one instance of each tri-gram (although weighting is possible). Utilizing the cosine measure, 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 a strategy like (Krauthammer et al., 2000) which utilizes BLAST to determine similarity of text.

For efficiency we do not compare every definition to each other, applying a variant of the k-means clustering. The most frequent definition for an abbreviation becomes the initial cluster. From most to least popular all other definitions are clustered by applying the similarity metric above. If no cluster is found exceeding a threshold similarity – we have determined a .6 threshold – the definition becomes a new cluster. In this way the most common definition for each cluster is found and alternative definitions are grouped. The threshold was selected by creating a random sampling of clustered pairs and plotting the precision/recall curve at different threshold cutoffs. We chose a threshold that would maximize both precision and recall. It is notable, however, that other applications may benefit from different thresholds.

The results of (Adamic et al., 2002) demonstrate an application of the n-gram similarity metric. In which it was possible to determine which actual genes (rather than abbreviations that sometimes coincide with genes) are mentioned frequently in specific contexts.

3.2.2 MeSH Based

While the bulk of definitions were clustered correctly, due to borderline similarity scores, others were misplaced. For example, nested abbreviations, such as *Rat GH* and *Rat Growth Hormone*, for *RGH.* Instead of lowering the threshold score and precision with it, we opted to use the available Medical Subject Headings (MeSH) to augment the initial results. MeSH terms are manually assigned annotations for each MEDLINE document. The headings represent which general concept(s) a document is related to (diseases, organism, etc.).

MeSH clustering is achieved by processing the MeSH concepts representing each initial n-gram cluster. This process includes taking each cluster, finding the original 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) (Baeza-Yates and Ribeiro-Neto, 1999) to compare each cluster. Each dimension (each MeSH term is represented by a dimension) in the vector is weighted by the frequency of the MeSH term in the local (cluster) and global (all documents) contexts. 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 algorithm used in the n-gram clustering is applied here utilizing the new cluster representations and new similarity metric. Here we find that a threshold of around .3 works (determined similarly to the n-gram threshold).

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 applied to creating a dictionary is cross linking abbreviations with related definitions. This is mainly useful for abbreviations with capitalization variations or numbers. 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 (see Section 4).

3.2.4 Symbol Disambiguation

One of the most challenging problems in handling biological abbreviations is classification when no definition can be extracted from the text. Through the SaRAD system we are able to use the MeSH and n-gram vectors generated for clustering to correctly classify new abbreviations into the appropriate definition cluster. A document containing an abbreviation without a previously catalogued definition can then be compared to 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.

4. Interface

The dictionary output of the SaRAD system is maintained in a database and is rendered into a series of web pages. The interface we have created (see: www.hpl.hp.com/shl/projects/abbrev.html) gives users access to the various annotations generated by the dictionary building modules.

While this demonstration and the web supplement and more completely illustrate the look and feel of the user interface, we briefly describe the available tools.

Abbreviation Search: Users are able to search for both abbreviations and definitions. Matches are listed with their most common definition and a link to more details.

Abbreviation Details: For each abbreviation its detail page initially displays all the main definitions. Clicking on each definition causes the interface to dynamically display cluster details and related definitions. Related definitions are those clustered by the n-gram clustering technique. Different colors represent how similar (based on the similarity metric) they are to the main definition. 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.

"See also" links (as described in Section 3.2.3) are also made available here.

Contextualized Document Search: To demonstrate the disambiguation tool we have built a sample application. 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. For example, a user may query for "*DCC* and *mice*", and see 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, or MeSH similarity), allowing users to 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 clustered using one of the techniques described above.

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 the extraction step. Of 54,399 n-gram clustered definitions, 555 were tested. Of those, 550 were considered legitimate (the two definitions were related) yielding a 99% accuracy rate. Of the 2,622 MeSH folded definitions, 500 were tested, with 76% accuracy.

The lower 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

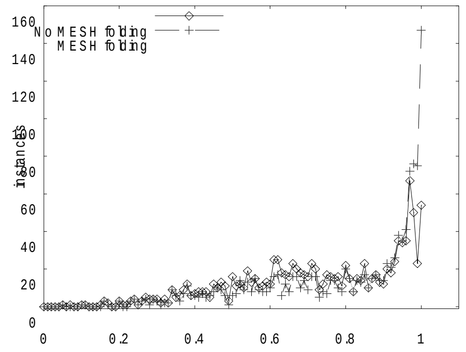


Figure 2: Accuracy of classification with and without MeSH clustering (folding).

in which definitions were related but not synonymous. For example, *energy expenditure* and *endurance exercise.* Despite this lower score we feel it is important to display these clustered definitions to the user since they provide hints as to how searches can be modified.

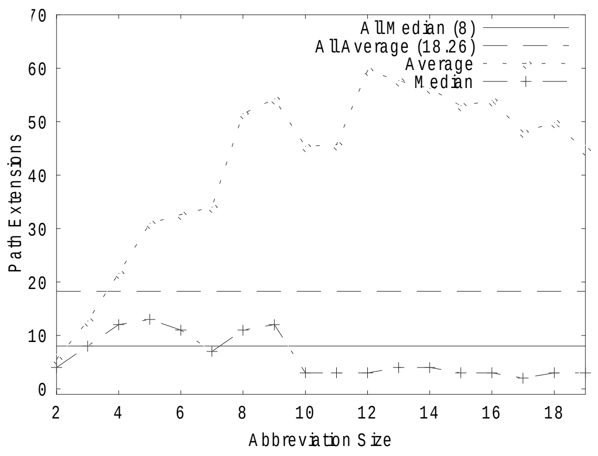
While there is no established dictionary to facilitate recall testing we have used the abbreviations available in the Unified Medical Language System (UMLS) as a basis of comparison. The UMLS data set includes 10,161 definition pairs. Of those, 547 were used in the overlap experiment.

Of the 547, our dictionary contained 197 definitions as main definition matches, 23 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 while not included in the final dictionary were correctly extracted from the text. A total of 339 definitions in our database 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 indicating a difference 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 (see Liu et al., 2002, for an extensive discussion of the differences in UMLS and MEDLINE). It is difficult to test this hypothesis as the PubMed interface to MEDLINE transforms terms that produce no hits (through a UMLS lexicon) into alternate symbols.

One final data point can be obtained by applying our algorithms to the AcroMed Gold Standard data set (Medstract, 2002). It is important to note, however, that such a metric is not a useful measure of performance for a dictionary building task, and that the AcroMed data contains errors and mislabeled data. Regardless, others have used this set so we provide our results for comparison. Running our algorithm on a manually cleaned AcroMed data set, and with the threshold at 0, a manual review for correctness reveals an accuracy of 95% and the recall of 85%. These results appear on par with the best published abbreviation extraction algorithms that score in the range of 95-99% for accuracy, and 72-84% in recall (see Schwartz and Hearst, 2003, for a full comparison).

5.2 Disambiguation Performance

Performance for automated disambiguation based on MeSH similarity was measured by generating 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.



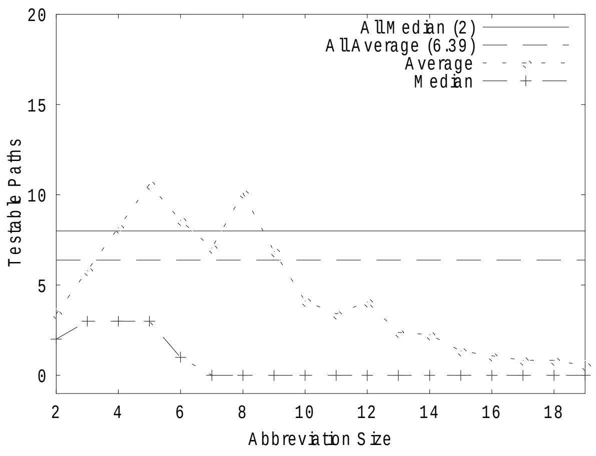


Figure 3a-b: The distributions of various performance metrics (a: path extensions, and b: testable paths) as a function of the length (in characters) of the abbreviation.

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. This number was chosen as we felt that 35 "training" documents was conservative given that the average number of documents associated with each definition of multiple definition abbreviations was 40. 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 then 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 2 depicts a histogram that buckets abbreviation classifiers 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%. A user or programmer wishing to make use of training data may then use these scores to determine if a specific abbreviation classifier is high enough quality.

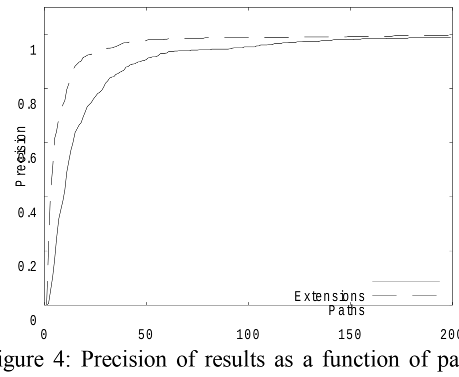


Figure 4: Precision of results as a function of paths tested and number of extensions.

5.3 Algorithmic Performance

The most computationally intensive SaRAD 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 case and various optimizations can be applied for better performance. The average abbreviation size for our full data set is 3.8 characters with a median of 3.

Figure 3a-b compares various metrics as a function of abbreviation size. Despite the growth in the average number of path extensions (3a), the average and median remain relatively low. The number of testable paths (3b) grows and then decays.

The features of the real world data allow us to develop an optimization to give us constant time performance. Figure 4 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 SaRAD 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.

6. Conclusions

The SaRAD system represents a simple, easy to implement, and accurate toolset for building biomedical databases (although, we have applied the algorithms to the entire HP website to generate an abbreviation server for internal purposes). 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, Bernardo Huberman, Ariel Schwartz, Jeff Chang, Marti Hearst, Russ Altman, and Paul Bowden for their comments, advice, support, and use of this system.

References

Adamic,L.A., Wilkinson,D., Huberman,B.A., and Adar, E. A Literature Based Method for Identifying Gene-Disease Connections, in: Markstein, V., and Markstein P. eds., *Proceedings of the First IEEE Computer Society Bioinformatics Conference*, IEEE Press, New York, NY, pp. 109-117.

Baeza-Yates,R. and Ribeiro-Neto,B. (1999) *Modern Information Retrieval*, Addison-Wesley, Harlow, England.

Blaschke,C., Andrade,M.A., Ouzounis,C., and Valencia,A. Automatic extraction of biological information from scientific text: protein-protein interactions, in: Lengauer,T., Schneider,R., Bork,P., Brutlag,D., Glasgow,J., Mewes,H.W. and Zimmer,R. eds., *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology,* AAAI Press, Menlo Park, CA, pp. 60-67.

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

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

Bowden,P.R., Halstead,P., and Rose,T.G. Dictionaryless English Plural Noun Singularisation Using a Corpus-Based List of Irregular Forms, in: Ljung M., ed. *Papers from the Seventeenth International Conference on English Language Research on Computerized Corpora*, Rodopi, Amersterdam, The Netherlands.

Chang, J.T., Raychaudhuri S., and Altman, R.B. Including biological literature improves homology search, in: Altman,R.B., Dunker A.K., Hunter,L., Lauderdale,K., and Klein,T.E., eds., *Proceedings of the 2001 Pacific Symposium on Biocomputing*, World Scientific Press, Singapore, pp. 274-383.

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

Chang., J.T., Schutze H., and Altman R.B. (2002) Creating an Online Dictionary of Abbreviations from MEDLINE, *J. Am. Med. Inform. Assoc*., **9**, pp. 612-620.

Fuchs, P., (2002) The synaptic physiology of cochlear hair cells, *Audiol. Neurootol*, **7**, pp. 40-44.

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

Jenssen.,T.K., Laegreid, A., Komorowski, J., and Hovig, E. (2001) A Literature Network of human genes for high-throughput analysis of gene expression, *Nature Genetics,* **28**, pp. 21-28.

Krauthammer,M., Rzhetsky,A., Morozov,P., and Friedman,C. (2000) Using BLAST for identifying gene and protein names in journal articles, *Gene*, **259**, pp. 245-52.

Larkey.,L.S., Ogilvie,P., Price,M.A., and Tamillo, P. Acrophile: An Automated Acronym Extractor and Server, in: *Proceedings of the Fifth ACM Conference on Digital Libraries*, ACM Press, New York, NY, pp. 205-214.

Liu,H., Lussier,Y.A., and Friedman,C. (2001) Disambiguating Ambiguous Biomedical Terms in Biomedical Narrative Text: An Unsupervised Method, *J. Biomed. Inform.* **34**, pp. 249-261.

Liu,H., Aronson, A.R., and Friedman,C. A Study of Abbreviations in MEDLINE Abstracts, in: *Proceedings of the American Medical Informatics Association Symposium 2002*, Hanley & Belfus, Philadelphia, PA.

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

Park, Y., and Byrd, R.J. Hybrid text mining for finding abbreviations and their definitions, in: Lee,L., and Harman,D., eds., *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Somerset, NJ.

Porter,M.F. (1980) An algorithm for suffix stripping, *Program*, **14**, pp. 130-137.

Pustejovsky,J., Castano,J., Cochran,B., Kotecki,M., and Morrell,M. (2001) Extraction and Disambiguation of Acronym-Meaning Pairs in MEDLINE, Medinfo., **10**, pp. 371-375.

Schwartz,A.S., and Hearst,M.A. A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text, in: Altman,R.B., Dunker A.K., Hunter, L., Jung,T.A., and

Klein,T.E., eds., *Proceedings of the 2003 Pacific Symposium on Biocomputing*, World Scientific Press, Singapore.

Taghva,K., and Gilbreth,J. (1995) Recognizing Acronyms and their Definitions, Information Science Research Institute, University of Nevada, Technical Report TR 95-03.

Yandell,M.D., and Majoros, W.J. (2002) Genomics and Natural Language Processing, *Nat. Rev. Gen.*, **3**, pp. 601-609.

Yeates,S., (1999) Automatic Extraction of Acronyms from Text, in: *Proceedings of the Fourth New Zealand Computer Science Research Students' Conference*, pp. 117-124.

Yeates,S., Bainbridge,D., and Witten,I.H. (2000) Using compression to identify acronyms in text, in: *Data Compression Conference*, IEEE Press, New York, NY, p. 582.

Yoshida,M., Fukuda,K., and Takagi,T. (200) PNAD-CSS: a workbench for constructing a protein name abbreviation dictionary, *Bioinformatics*, **16**, pp. 169-175.

Yu,H., Hripcsak,G., and Friedman,C. (2002) Mapping Abbreviations to Full Forms in Biomedical Articles, *J. Am. Med. Inform. Assoc.* **9** pp. 262-272.