

## Web Supplement

# “SaRAD: A Simple and Robust Abbreviation Dictionary,” Eytan Adar

### Additional Nodes

#### MeSH Clustering

The vectors representing MeSH information for clusters are weighted by a basic TFIDF scheme. Each dimension (each MeSH term is represented by a dimension) in the vector is calculated by:

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

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

#### Threshold Selection and Alternative Similarity Metrics

Threshold selection can be easily achieved by applying the similarity metric to a small subset of random clusters. These pairs are then manually labeled as correct or incorrect. The construction of a precision/recall curve at different thresholds should indicate an optimal cutoff. In such an experiment on n-gram clustering we obtained the curve in Figure 1. We have also determined that different similarity metrics (Jaccard and Dice specifically) do not perform better than the simple cosine metric and are not necessarily any more useful in classification.

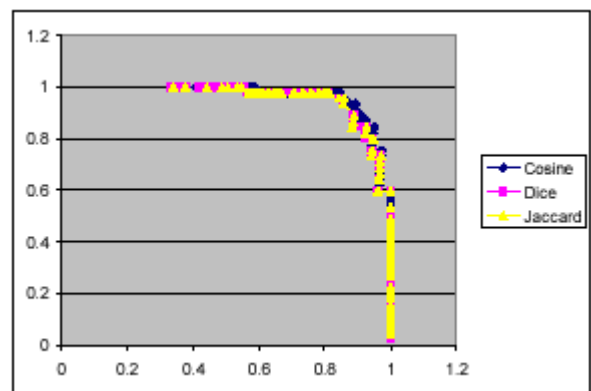


Figure 1: precision/recall at different thresholds for different similarity metrics (applied to n-gram clustering)

## 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].

The first portion of the algorithm (Figure 2), 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  $n^{\text{th}}$  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 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.

### Alternative rules

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.” Our rules were evolved through experimentation and our final choices appear to be similar to others

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

Figure 2: the BUILD-PATHS algorithm

## Appendix B: User Interface

3a

First few letters of the abbreviation:

Search by definition [here](#)

Abbrev	Most common def.
Aa	actinobacillus actinomycetemcomitans ( <a href="#">more</a> )
AA	arachidonic acid ( <a href="#">more</a> )
AA-	amino acids ( <a href="#">more</a> )
AA1	amniotic antigens 1 ( <a href="#">more</a> )
AA-2G	ascorbic acid 2 o alpha glucoside ( <a href="#">more</a> )
AA-2P	ascorbic acid 2 phosphate ( <a href="#">more</a> )
AA4H	acetanilide 4 hydroxylase ( <a href="#">more</a> )
AA-673	amlexanox ( <a href="#">more</a> )
AAA	abdominal aortic aneurysm ( <a href="#">more</a> )

3b

**AA**  
(click any definition to "expand")

<b>arachidonic acid</b>	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](#)

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

3c

Query: **DCC mice**  
(up to 100 results returned)

Definition found in text Similar definition found MeSH clustered

<b>dextran coated charcoal</b>	1 documents
* [Production and application of mouse antiserum to human estrogen receptors] ( <a href="#">8697973</a> ) Definition found: dextran coated charcoal	
<b>dicyclohexylcarbodiimide</b>	6 documents
* Synthesis of some 2 arylpropionic acid amides as prodrugs. ( <a href="#">8876939</a> ) Definition found: dicyclohexylcarbodiimide	
* Generation of polyclonal catalytic antibodies against cocaine using transition state analogs of cocaine conjugated to diphtheria toxoid. ( <a href="#">8575031</a> ) Definition found: dicyclohexyl carbodiimide	
* Radioisotope carrying polyethylene oxide-polycaprolactone copolymer micelles for targetable bone imaging. ( <a href="#">11771706</a> ) Definition found: diaminoethyl cyclocarbodiimide	
* MHC restriction in contact hypersensitivity to dicyclohexylcarbodiimide. ( <a href="#">12176098</a> ) Definition found: dicyclohexylcarbodiimide	
* Contact hypersensitivity to dicyclohexylcarbodiimide and diisopropylcarbodiimide in female B6C3F1 mice. ( <a href="#">9598300</a> ) Definition found: dicyclohexylcarbodiimide	
* Induction of micronucleated erythrocytes in rodents by diisopropylcarbodiimide and dicyclohexylcarbodiimide: dependence on	

Figure 3(a-c): Various captured screens from the web-based user interface. Figure 3a displays the results of a search for abbreviations beginning with "AA." Figure 3b is the details screen for the abbreviation AA. Finally, Figure 3c is clustered PubMed search. The results of DCC mice has been clustered by different definitions of DCC.