EMU: A Web Portal Generator

Aaron R. Bradley
arbrad@stanford.edu

Andrew M. Bradley
ambrad@stanford.edu

June 10, 2001

1 Introduction

A web portal is a set of HTML documents that comprise a tree rooted at the portal entry site; the leaves are links to other websites. Each node within the tree represents a cluster of documents, decreasing in generality as one descends down the tree. Portals are useful for organizing and enabling users to search large repositories of documents, either centralized or distributed. Whereas search engines require the user to specify search criteria, typically as key words, to search an opaque document bank, a portal visualizes the document bank and can be particularly useful if the query is not amenable to representation by key words.

Popular portals, such as those hosted by AltaVista, Google, Yahoo, and dmoz.org, are created by hand. For example, the `Open Directory Project` at `dmoz.org` enlists volunteers to manages subtopics. In contrast, EMU is an automated system that uses statistical NLP and unsupervised machine learning techniques to generate a web portal. While the quality of the portal must certainly suffer, less frequented document repositories and Usenets may benefit from an automated system. Additionally, EMU is practical for generating daily navigators of newspapers, internal documents of a company, and other dynamic document banks. Many companies, for example, provide employees with daily or weekly summaries of domain-specific periodicals relating to the company's business; EMU would provide a practical alternative. Finally, hierarchical clustering can serve as a preprocessing step for fast online clustering used in document retrieval and user queries.

Previous work has used hierarchical clustering for navigational purposes. Scatter/Gather [3] uses a clustering approach to summarize a set of documents; however, they adapt an interactive approach with the user. A typical session consists of (flat) clustering a set of documents and labeling the clusters, allowing the chooser to select a cluster to explore, and repeating the process on the new cluster. This corresponds directly to the EMU interface of selecting superclusters to explore in more depth, although the interactive approach allows the system to take advantage of fast non-hierarchical clustering algorithms. EMU, however, runs only once to generate a summary; hence, it is appropriate for generating summaries that many people will read via the web.

This report is organized as follows. In Section 2, we present an overview of the EMU system, while in Section 3, we consider EMU's document representation and learning algorithms. Section 4 presents our testing techniques; we also evaluate our results on a test set. Finally, Section 5 concludes with ideas for applications and future work.

2 Overview of the System

The primary goal of EMU is to extract from a set of documents a portal that is clustered and labeled well enough to allow the user to browse the documents efficiently. We implemented the portal generator as a series of processing steps parameterized by criterion and metric functions, and tested permutations of the functions (see §4). The stages roughly correspond to gathering documents and translating the documents into a word-by-document matrix (§3.1), analyzing the matrix to produce a binary dendrogram (§3.2, §3.3, §3.4), and converting the dendrogram to a portal (§3.5).

The preprocessing stage relies on several assumptions about language:

- documents discuss one subject;
- a document's use of a word is consistent (*one sense per discourse* constraint [13]); we rely on a stronger assumption: only one sense of a stem that appears in several words is used in a document;
- loss of capital letters and removal of symbols, including numbers, does not obscure meaning in a bag-of-words technique.

The analysis stage assumes that topics have an associated set of words often used in discussions. In tests without filtering, symbols and numbers often appeared only once; words that appear only once or twice were already being removed for computational reasons. The stem assumption may be too strong in some domains; however, in our model, we look for words that discriminate between meanings, which seem to satisfy this assumption. On a larger scale, conflating dissimilar words may adversely affect the analysis stage, although the benefit of conflating similar words seems to more than make up for this error [5]. Finally, the existence of sets of topical words is suggested by our results.

2.1 Summary of the Code

The following table summarizes the program files corresponding to each phase of portal generation. This chart is meant to be a reference while reading §3.

Preprocessing

`silly_bot.py`	Retrieves websites starting at specified domain, filters HTML tags and scripts, and writes to a document file. Optionally treats each page or tree of pages as a document.
`doc_stats.py`	Reads documents retrieved by `silly_bot.py`, filters words, and produces word-by-document matrix.
`ace_stats.py`	Subclass of `doc_stats.py`. Reads ACE documents.
`porter.py`	Vivake Gupta's implementation of Porter's algorithm.
`doc_to_mat.m`	Translates *.docspc file containing word-by-document matrix into a Matlab matrix.
`tf_idf.m`	Accepts (Matlab) word-by-document matrix and TF.IDF specifications. Produces weighted matrix.

Clustering

`lsi.m`	Reduces dimensionality of term-by-document space while emphasizing co-occurrence.
`mds.m`, `mdsmex.c`, and `mds_mat.m`	Metric multidimensional scaling implementation for dimensionality reduction.
`buhc.m`	Bottom-up (agglomerative) hierarchical clustering. The available clustering criteria are '`ncos`', '`avg`', '`min`', and '`max`', where the latter two correspond to single- and complete-linkage, respectively. '`dice`', '`jaccard`', normalized cosine, and Euclidean distance (the latter two are defaults) are the available distance metrics.
`buhc_dist.m`	Computes the distance matrices for the buhc* procedures.
`buhc_vis.m`	Produces a dendrogram from the `buhc.m` clustering.
`buhc_write.m`	Writes the clustering to file for future processing.

Portal

`buhc_process.py`	Reads file produced by clustering algorithm and writes a portal into a given directory. The root is at <dir>/portal.html.

Evaluation

`buhc_valid.m`	Determines whether the data and corresponding feature space representation are amenable to clustering.
`buhc_coph.m`	Computes the cophenetic correlation coefficient for a clustering.

The following is an example generation of a portal:

```
    >  python silly_bot.py -url <root> -tree 2
    >  python doc_stats.py -in emu.docs -out wbd
Matlab:  [X,docs,words] = doc_to_mat('wbd.docspc')
Matlab:  Optionally, apply LSI (see §3.2).
Matlab:  C = buhc(X,'avg',0,0)
Matlab:  buhc_write('c.out',C,docs)
    >  python buhc_process.py c.out wbd.docspc wbd.trnslt  /WWW/portal
```

3 Algorithms

3.1 Preprocessing

3.1.1 Building a Word-by-Document Matrix

The preprocessing stage accepts a set of documents and produces a word-by-document matrix, as described in §**8.5.1** of [8]. Element e_{ij} indicates the number of times that (filtered) word t_j appears in document d_i; this representation is useful for further weighting (see §3.1.2).

The first step is the parsing of the documents. In the case of a prepared corpus, the corpus is neatly split into documents using corpus-specific tags (*e.g.*, the `<subsection>` tag in ACE); however, parsing webpages is considerably more difficult. Two complications arise: first, the definition of a document is unclear; second, the relevant text on an HTML page is obscured by the HTML itself. While the latter problem is satisfactorily solved with regular expressions, the first requires a more complex approach. One possibility is the TextTiling algorithm, which searches for topic shifts in a text; these shifts define the boundaries of extracted documents [8]. First, the text is segmented into sequences of terms (perhaps multiple sentences); then the *cohesion score* measures the cohesion between two sequences (at a *gap*), while the *depth score* measures the cohesion relative to surrounding cohesion scores. A *boundary selector* uses the depth score to determine segments. However, because the ACE corpus that we used provided short articles that concentrated on one subject, while (web) portals usually provide links to home pages, we chose a different description of a (web) document. Specifically, a document is a root page and all referenced pages contained in the same directory or subdirectory. References to other pages provide new roots of documents.

After determining the document boundaries, each document is parsed. Words are filtered as follows:

- all tokens are converted to lowercase;
- symbols within tokens are converted to whitespace, splitting the token;
- numbers are removed;
- tokens are stemmed with the Porter algorithm, producing the final word.

This set of filters is common (see [5] for an example). The Porter algorithm is an automatic stemmer widely used for information retrieval [10]. Stemmers map words with suffixes to their stems (or some kernel of the word); for example, *generalizations* is converted to *gener* through stages *generalization*, *generalize*, and *general*. Clearly, the result does not need to be a word, as long as all words with the same stems are conflated to the same kernel. The algorithm consists of five stages of rules of the form

$$(condition)\ S1 \Rightarrow S2,$$

where suffix $S1$ is replaced with suffix $S2$ when condition *condition* holds. Conditions include constraints on stem length and inclusion of vowels and consonants in the stem. A mapping of stems to particular instances is maintained for later use in constructing cluster headings.

The filtered words are inserted into an array of dictionaries corresponding to documents, as well as a global dictionary. Both levels of dictionaries store word counts; these counts enable further filtering after the documents have been parsed:

- words that appear in some (large) proportion of documents are removed; these words cannot aid in discriminating meanings;
- words that appear in less than some number of documents are removed; many words appear only once or twice (Zipf's Law);
- words occurring in a *stoplist* are removed.

A *stoplist* is a set of words that are either used frequently or provide no useful information. In addition to decreasing computation time, the first and second filtering steps give a very rough approximation to non-monotonic weighting, giving middle-frequency words, which may have the greatest "resolving power," the greatest weight [5]. Finally, the word-by-document matrix is constructed over the remaining words.

3.1.2 Weighting the Matrix

The final (optional) stage of preprocessing consists of substituting non-linear (but still monotonic) weights in the word-by-document matrix. The TF.IDF weighting scheme provides a useful set of formulae ([5], [8]):

Term occurrence		Document frequency		Normalization	
n	tf_{td}	n	df_t	n	(no normalization)
l	$1 + \log(tf_{td})$	t	$log(\frac{N}{df_t})$	c	$\left(\sum_{i=1}^{n} w_i^2\right)^{\frac{1}{2}}$
a	$0.5 + \frac{0.5 \times tf_{td}}{\max_t(tf_{td})}$				

where

$$tf_{td} = e_{ij} \text{ and } df_t = \sum_j e_{ij}$$

in the word-by-document matrix. Constructing a weight simply requires choosing a combination of options; *lnn*, for example, describes

$$(1 + \log(tf_{td})) \times df_t,$$

where we add the extra condition that if $tf_{td} = 0$, the result is 0. See §4 for an empirical comparison of some possible weights.

3.2 Latent Semantic Indexing

Latent Semantic Indexing (LSI) applies Singular Value Decomposition (SVD) to a word-by-document matrix (possibly weighted by a TF.IDF formula), thus finding its optimal projection onto a lower-dimension space [8]. For the decomposition of the word-by-document matrix A_{td} (for $n = \min(t, d)$):

$$A_{td} = T_{tn} S_{nn} D_{dn}^T,$$

the LSI module returns the matrix

$$S_{kn} D_{nk}^T$$

for some $k < n$. Choosing k so that $s_k >> s_{k+1}$ produces a matrix that explains most of the data and whose rank, consequently is approximately the rank of the original space. However, the transformed feature vectors may contain information implying co-occurrence relationships that were not evident in the original feature space. Hence, the dimensionality reduction has a linguistic foundation.

3.3 Multidimensional Scaling

Multidimensional Scaling (MDS) provides another means of reducing the dimensionality of the data while preserving clusters [4]. It maps a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Re^n$ with distance δ_{ij} between points $\mathbf{x}_i$ and $\mathbf{x}_j$, to $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \Re^k$ with distance d_{ij} between points $\mathbf{y}_i$ and $\mathbf{y}_j$, for $k < n$. MDS attempts to make the points

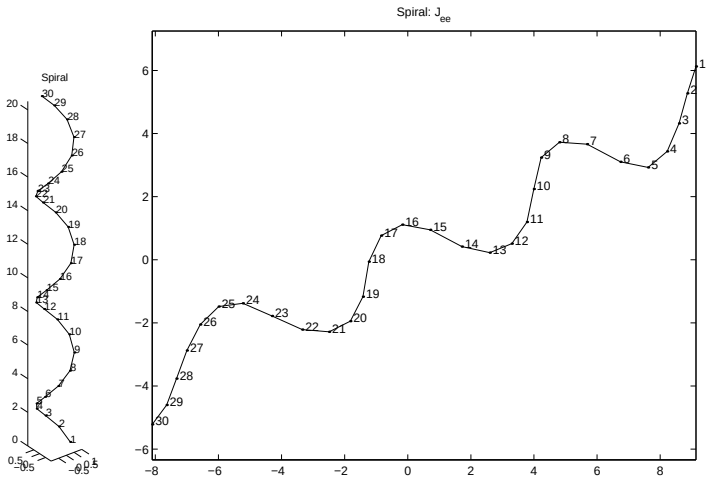

Figure 1: Example of metric multidimensional scaling. **(a)** Original, 3-dimensional spiral. **(b)** Two-dimensional mapping resulting from MDS.

of the image space $\mathbf{y}$ have distances close to their corresponding distances in $\mathbf{x}$, using these sum-of-squared error criterion functions:

	Criterion Function	Gradient ($\nabla_{y_k} J$)
J_{ee}	$\frac{\sum_{i<j}(d_{ij}-\delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$	$\frac{2}{\sum_{i<j}\delta_{ij}^2}\sum_{j\neq k}(d_{kj}-\delta_{kj})\frac{\mathbf{y}_k-\mathbf{y}_j}{d_{kj}}$
J_{ff}	$\sum_{i<j}\left(\frac{d_{ij}-\delta_{ij}}{\delta_{ij}}\right)^2$	$2\sum_{j\neq k}\frac{d_{kj}-\delta_{kj}}{\delta_{kj}^2}\cdot\frac{\mathbf{y}_k-\mathbf{y}_j}{d_{kj}}$
J_{ef}	$\sum_{i<j}\frac{(d_{ij}-\delta_{ij})^2}{\delta_{ij}\sum_{i<j}\delta_{ij}}$	$\frac{2}{\sum_{i<j}\delta_{ij}}\sum_{j\neq k}\frac{d_{kj}-\delta_{kj}}{\delta_{kj}}\cdot\frac{\mathbf{y}_k-\mathbf{y}_j}{d_{kj}}$

J_{ee} attempts to preserve *relative* distances, so it is insensitive to large differences in absolute distances between points in x and points in y. J_{ff}, however, is acutely sensitive to absolute differences, while J_{ef}, a hybrid of the two approaches, is not overly sensitive. Since our application attempts to find clusters in the data, which does not require the preservation of absolute distances, J_{ee} and J_{ef} provide the most robust performance. Given an initial setting—perhaps random or constructed from coordinates along the axes with the greatest variance—MDS executes gradient descent using the corresponding gradient functions.

Fig. 1 provides an example of metric MDS's behavior, derived from [4]. In practice, our implementation has proven to be numerically unstable for large matrices; given the time constraints, we were unable to solve this problem sufficiently such that the routines could yield adequate results.

3.4 Agglomerative Hierarchical Clustering

The general problem of document clustering and querying is addressed in the literature of the Information Retrieval community. The text [11] provides a foundation for the problem, defining the *Cluster Hypothesis* in query retrieval terms, namely that "closely associated documents tend to be relevant to the same requests." That is, given a query and a document that is relevant to the query, it is likely that similar documents are also relevant. The measure of fitness of this hypothesis to a document collection can be determined by computing the similarities between all pairs of relevant documents and all pairs such that one document is relevant but the other is not. Over many queries, it is hypothesized that the relative frequency of large relevant-relevant similarities exceeds that of the relevant-nonrelevant similarities. Implicit in this argument is the quality of the *document description*, or in machine learning terms, the feature vector.

BUHC(X,*dist*,*criterion*)

Input: X is the feature matrix.
dist is a distance metric.
criterion is a clustering criterion.

Output: A hierarchical clustering C.

Compute the distance matrix D according to *dist*.

for *iteration* := 1 **to** *length*(X)−1

Choose the nearest clusters according to *criterion*, say C_i and C_j.

Merge C_i and C_j into C_i.

Recompute row and column i of D.

Excise row and column j from D.

return C

Figure 2: General agglomerative hierarchical clustering algorithm.

Clustering as a statistical technique is widely treated in the literature of several disciplines, ranging from taxonomy applications in biology to machine learning in AI [12]. Determining optimal clusters is an NP-hard problem, for the optimal solution under a criterion function is an Integer Programming problem [4]. Thus, heuristic techniques are necessary. Several clustering algorithms, both hierarchical and flat, are natural in the domain of information retrieval. Willett [12] provides a review of hierarchical techniques, as well as evaluation methods. First, *agglomerative* is preferred to *divisive* clustering; whereas the former produces *polythetic* classifications, where each document in a cluster has one or more terms in common with each of the other documents in the cluster, but no one term is necessary for membership, the latter approach often produces *monothetic* classifications, in which all the members of a cluster must possess a certain set of terms.

Next, several clustering criteria exist. It is noteworthy that many of them, in concert with agglomerative algorithms, do not explicitly attempt to extremize a criterion function, although iterative refinement is possible [4]. Let C_i and C_j be two clusters. Then the following criteria are used [4]:

$$\begin{aligned} d_{min}(C_i, C_j) &= \min_{\mathbf{x}\in C_i, \mathbf{x}'\in C_j} ||\mathbf{x} - \mathbf{x}'|| \\ d_{max}(C_i, C_j) &= \max_{\mathbf{x}\in C_i, \mathbf{x}'\in C_j} ||\mathbf{x} - \mathbf{x}'|| \\ d_{avg}(C_i, C_j) &= \frac{1}{|C_i| \times |C_j|} \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{x}'\in C_j} ||\mathbf{x} - \mathbf{x}'||. \end{aligned}$$

The first two criteria produce *single-linkage* and *complete-linkage* algorithms, respectively, and have many graph-theoretic properties [4], [6]. In addition, the normalized cosine metric is useful and provides an $O(n^2)$ algorithm [8], rather than the typical $O(n^3)$. In this case, the feature vectors must be normalized. Finally, for single- and complete-linkage, the *Dice* and *Jaccard* coefficients [8], [12] are often used:

$$\begin{aligned} dice &= \frac{2|X \cap Y|}{|X| + |Y|} \\ jaccard &= \frac{|X \cap Y|}{|X \cup Y|}, \end{aligned}$$

where X and Y are feature vectors for two documents.

Our implementation of hierarchical clustering offers the foregoing cluster and distance metrics. All similarity metrics are converted to distances for continuity of implementation. The algorithm is summarized in Fig. 2.

The resulting clusters are displayed in *dendrograms*. Fig. 3 displays several dendrograms produced by `buhc_vis.m` for random data.

3.5 Extracting a Portal

The output of the clustering algorithm is not the end of the story. Two more steps are necessary to produce a portal: first, the clusters must be determined since hierarchical clustering produces a full dendrogram;

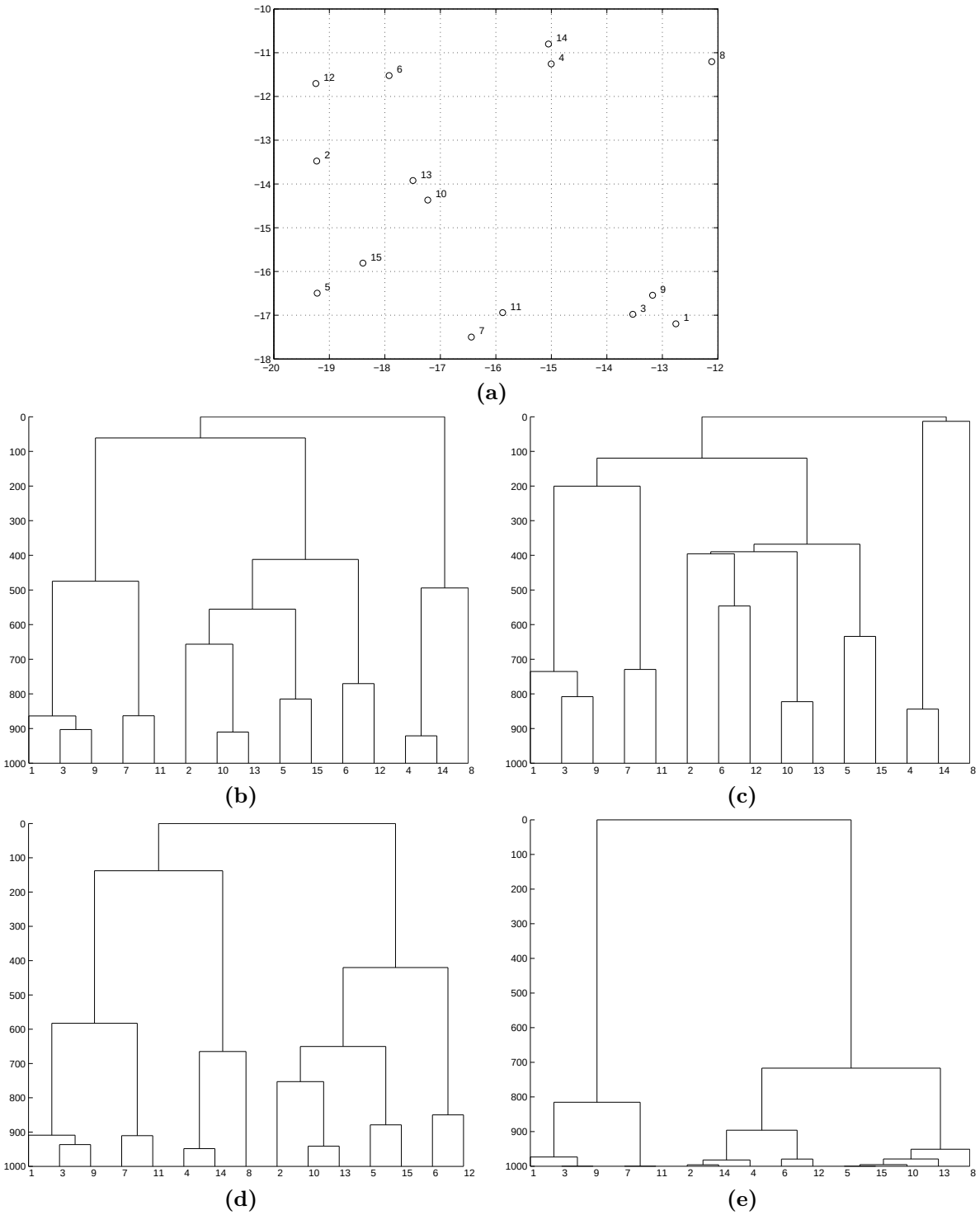

Figure 3: Example of hierarchical clustering and the resulting dendrograms. The data are 15 uniformly random points in 2-dimensional feature space. The dendrograms are formed using various criteria and distance metrics. **(a)** Original data. **(b)** Cluster average. **(c)** Single linkage. **(d)** Complete linkage. **(e)** Normalized cosine (the data were normalied).

second, the clusters must be given labels.

Determining a reasonable set of clusters can be accomplished using several approaches. Stork (from [4]) suggested in lecture to include all subnodes of a node for which the difference in similarity between itself and its parent is greatest along the path from the root. If the difference is significantly greater than other differences, then this suggests a "natural" cluster. A simpler method is to stop clustering when the similarity is greater than some prechosen value. Since our goal is twofold (choose both the number of clusters and merge supernodes to create non-binary supergroups), we chose a hybrid of the two: a binary node is merged with its parent if the difference between its similarity measure and its parent's is less than some value that may depend on depth. If the small difference is significant, then it may suggest a "natural" cluster (or super-cluster, if the merge is high in the dendrogram).

Once the clusters are determined, compact and informative labels are assigned. [3] suggests using *topical terms* to describe the clusters as part of their *cluster digest*. Essentially, the terms with the highest weights are chosen. Since we describe not only clusters of documents, but clusters of clusters, weights that emphasize term frequency or document frequency lead to labels that are influenced by sub-cluster size. Hence, we developed the following weighting schemes. First, we can simply let the weight of a term at a cluster C, $weight(t, C)$, equal the number of sub-clusters of C that contain documents containing the term t. Then words that appear in all sub-clusters receive the highest weight; words that appear with high frequency but in only some sub-clusters receive less weight. Additionally, terms that appear mostly in the documents contained in C should be preferred to terms that appear both in documents in C and elsewhere; therefore, we can augment the weight:

$$weight'(t, C) \;=\; \frac{weight(t, C)}{\sqrt{1 + |docs(t) - docs_C(t)|}},$$

where $docs(t)$ is the number of documents in which t appears, and $docs_C(t)$ is the number of documents in C in which t appears. These weights seem to lead to descriptions that capture all documents (or sub-clusters) of a cluster.

4 Results

4.1 Review of Methods

Cormack polemicizes the widespread practice of "often irrelevantly and unjustifiably" clustering data; moreover, "the growing tendency to regard numerical taxonomy as a satisfactory alternative to clear thinking is condemned" [1]. Yet Van Rijsbergen observes that evaluating the validity of a clustering procedure or its output is often a difficult and *ad hoc* craft [11]. This motivates an attempt to evaluate quantitatively the validity of our results, grounded on a firm theoretical foundation. In our analysis of our results, we employ a two-tier system of validating our hierarchies. First, we use the statistics based on the theory of random graphs [7] to determine whether the initial data lend themselves to clustering at a statistically significant level. Second, we determine the *distortion* of the clustered output relative to the input. Let us review the techniques.

4.1.1 Testing the Clustering Hypothesis

Ling *et al.* [7] provide statistics $P_{n,v}$ based on the theory of random graphs. Let $G_{n,v}$ be the set of all undirected graphs with n nodes and v edges. Then a random graph $G \in G_{n,v}$ has probability

$$\left(\begin{pmatrix} n \\ 2 \\ n \end{pmatrix} \right)^{-1},$$

since there are

$$\begin{pmatrix} n \\ 2 \end{pmatrix}$$

ways to choose two nodes, and then from this result, one chooses v edges. Now, to evaluate whether n elements to be clustered contain natural clusters, we can determine the first v for which the graph representing the elements becomes connected. v is obtained as follows. Compute the $n \times n$ similarity matrix S over the elements. Determine the single-linkage clusters for a threshold t; with increasing t, fewer clusters emerge. The minimum t for which the graph is connected corresponds to the desired v. Having obtained this v, we wish to determine whether it is greater than what one expects on a random graph at a statistically significant level. Intuitively, if a set of elements contains natural clusters, then we expect v to be quite high, since a connected graph corresponds to a single (and hence, unnatural) cluster. Thus, we require the probability that a random graph becomes connected at $V \leq v$, denoted $P_{n,v} = P(V \leq v|n)$. Ling *et al.* provide the following exact solution to $P_{n,v}$:

$$P_{n,v} = P(V \leq v) = \frac{C_{n,v}}{\left(\begin{pmatrix} n \\ 2 \end{pmatrix} \right)},$$

for $v = n-1, \ldots, \begin{pmatrix} n \\ 2 \end{pmatrix}$, and

$$C_{n,v} = \left(\begin{pmatrix} n \\ 2 \end{pmatrix} \right) - \sum_{i=2}^{n} \frac{(-1)^i}{i} \Sigma' \frac{n!}{\prod_{j=1}^{i} n_j!} \left(\begin{pmatrix} \sum_{k=1}^{i} n_k \\ 2 \end{pmatrix} \right),$$

where Σ' sums over all positive integral solutions to $\sum_{k=1}^{i} n_k = n$.

Ling *et al.* computed $P_{n,v}$ for $n \leq 100$, providing a table summarizing their results. To use the table, one determines v for input data, compares it with the table's value for random graphs, and accepts the clustering hypothesis at the p level of significance, where p is the indicated table value. In summary, Ling *et al.*'s technique allows one to determine whether the input data contain natural clusters, and hence whether clustering is a valid analysis technique for the data.

4.1.2 Testing the Validity of the Clustering

The second part of the evaluation process evaluates the degree to which the obtained clustering respects the similarities in the original data. A member of the class of such techniques is called a *distortion measure* [12]. We adopt the commonly used *cophenetic correlation coefficient*, which is computed as follows [1]. Adopting Cormack's notation, let s_{ij} s^*_{ij} be the similarity between elements i and j in the original data and after clustering, respectively, where s^*_{ij} is similarity at the first node of the hierarchy at which elements i and j become members of the same cluster. Let $\bar{s}_{ij}$ and $\bar{s}^*_{ij}$ be the means of the respective similarity measures. Then the cophenetic correlation coefficient c_{coph} is defined as

$$c_{coph} = \frac{\sum_{i,j}(s_{ij} - \bar{s}_{ij})(s^*_{ij} - \bar{s}^*_{ij})}{\sqrt{\sum_{i,j}(s_{ij} - \bar{s}_{ij})^2 \sum_{i,j}(s^*_{ij} - \bar{s}^*_{ij})^2}} \in [0, 1].$$

A value close to 1.0 indicates a valid clustering.

4.2 Analysis of Results

We use a subset of the the Ace corpus (ICAME/ace) to test our method. While we initially intended to collect a corpus of web documents via a "bot," we discovered that the quality of the corpus suffers considerably.

First, we apply the statistics of Ling *et al.* to the data and the corresponding feature vector representation to validate the cluster hypothesis. The algorithm is implemented in `buhc_valid.m` and works as follows. Sort the computed distance matrix for the data. Step through the sorted distances, setting the threshold t, until the induced graph is connected. We use a disjoint set structure, which is easily queried for graph connectedness [2].

	Distance Metric	Clustering Criterion	LSI Dimensions	TF.IDF	Cophenetic Correlation Coefficient
1	Euc	Avg	-	-	0.9761
2	Euc	Min	-	-	0.9731
3	Euc	Max	-	-	0.9654
4	Euc	Avg	150	-	0.9761
5	Euc	Avg	140	-	0.9748
6	Euc	Avg	130	-	0.9740
7	Euc	Avg	120	-	0.9734
8	Euc	Avg	110	-	0.9734
9	Euc	Avg	100	-	0.9743
10	Ncos	Ncos	-	ltc	0.9567
11	Ncos	Ncos	100	ltc	0.8222
12	Euc	Avg	-	ltn	0.9571
13	Euc	Avg	-	atn	0.9543
14	Dice	Min	-	-	0.9123
15	Dice	Max	-	-	0.9013
16	Jac	Min	-	-	0.9714
17	Jac	Max	-	-	0.9685

Figure 4:

Since Ling *et al.* provide statistics for graphs containing at most one hundred nodes, we restrict the validation test to one hundred documents. Our representation is Euclidean distance over the term-by-document matrix. For the 99-document set we use, we obtain 4743 edges (note that $\frac{1}{2}98(98-1) = 4753$ is the greatest number of edges a 99-node graph can have without being connected), which is significantly greater than the 385 reported at the 0.99 level of significance. Ling *et al.*'s method is highly sensitive to outliers, that is, points in the data that are at a great distance from the main body of the data. In this case, the graph necessarily becomes almost *fully* connected before actually being connected. Thus, we perform the *ad hoc* test of lowering the threshold of connectedness to, say, 90% connected. This yields 3184 edges, which is still significantly greater than the reported statistic, but at least somewhat further from the maximum possible ($\frac{1}{2}88(88-1) = 3916$). We conclude that the Ace test data and their corresponding representation is amenable to cluster analysis and supports the cluster hypothesis.

Next, we perform a series of clusterings, using different similarity metrics and clustering criteria. For this, we used a corpus of 150 documents, again from the Ace repository. Table 4 presents the various permutations of parameters tested, as well as the cophenetic correlation coefficients for each clustering relative to the corresponding representation of the documents. Fig. 5 displays several of the resulting dendrograms.

Over all, the cophenetic correlation coefficients are quite high, indicating that the clustering method respects the original distance matrix. LSI did not significantly boost the score. Interestingly, the lowest-scoring parameterization yields the best qualitative clustering (§ 4.3). However, this has support in the literature; Willett states that "distortion of the similarity matrix is not necessarily to be avoided...a clustering method should attempt to identify groupings that are more intense than those present in the similarity matrix" [12]. Thus, while the coefficients give us some confidence about the validity of the clustering, they do not provide as much information as we would like.

The dendrograms support the assertion that the lowest-scoring parameterization yields the best qualitative clustering. Whereas parts (a)–(e) display dendrograms that provided perceptually insignificant clusters, (f), which corresponds to row 11 of the table, displays aggressive clustering.

Other quantitative measures of goodness of fit exist. Duda *et al.* and Cormack suggest a hypothesis test whereby the significance of, say $c+1$ clusters is tested against the null hypothesis of c clusters. While an exhaustive test of the clusters induced by a hierarchical cluster is perhaps impractical, tests of significance of the best clusters derived from the dendrogram may be helpful.

Quantitatively comparing clusterings is possible by comparing the cophenetic correlation coefficients, as

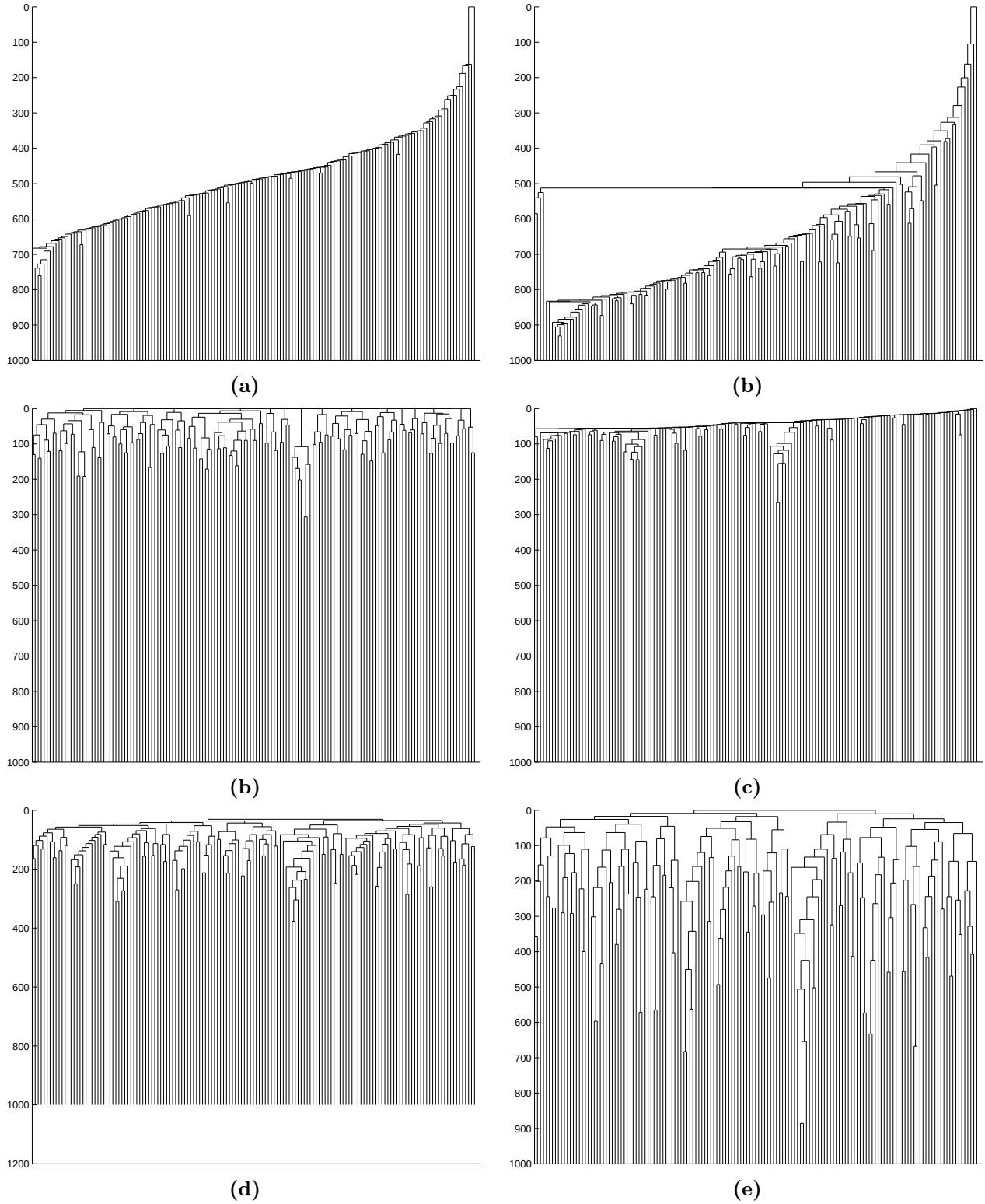

Figure 5: Selected dendrograms corresponding to rows in Table 4. **(a)** 13 **(b)** 3 **(c)** 17 **(d)** 16 **(e)** 10 **(f)** 11

we have done. However, a more interesting and subtle question is whether a clustering adequately matches the linguistic data; that is, whether a clustering matches what we perceive to be the semantic clusters in the document database. While we did not have the time to prepare the scripts necessary to test this, the following procedure may be useful for future consideration. Extract the dendrogram implied by a web portal such as hosted by AltaVista, Yahoo, or Google. Determine the induced distance matrix, as done in `buhc.m`. Compute a clustering from the represented web documents by one of the foregoing techniques and determine the induced distance matrix. Since the induced distance matrix of the web portal is a metric [6], we can treat it as the distance matrix of the underlying data that produced our clustering. Consequently, the cophenetic correlation coefficient produced by the two distance matrices is meaningful, and its value indicates the degree to which our clustering matches the portal's implied clustering. Insofar as the web portal is the "desired" clustering, this measures the success of the automated clustering technique.

4.3 The Portal

The portal at `http://www.stanford.edu/ arbrad/portal/portal.html` navigates the documents in `/ICAME/ace/ace_a.txt` which is a collection of newspaper articles on a broad range of subjects. It was generated using the `ltc` TF.IDF weighting scheme, projecting the resulting matrix onto 100 dimensions with LSI, and then applying hierarchical clustering with the *ncos* metric. Some of the resulting clusters are intuitive; for example,

- "crashed ride injured seats pacific quickly treated doctors above badly" labels a cluster of articles about strange accidents ("ferry mishap," injury on "fun park ride," a "plunge" off a "highway");
- "fallen passengers flights airline pilot technical aviation fully apple fare" heads a cluster relating to airlines;
- "fighter fought neck gloves heavyweight amateur ring hell sparring recalled" labels a cluster of boxing articles;
- "window boy rivers metres disappearance died m fell medication friend" describes a large set of articles about tragedies mainly involving children.

Unfortunately, the portal also contains several large clusters of random articles that do not closely match any other article. These clusters appeared with every parameterization that we tried. Nevertheless, we feel that several large clusters of unrelated documents is a better result than many small clusters of related documents with excessive noise, which a more sensitive node-merging phase produces.

5 Conclusion

We have demonstrated EMU, a system for hierarchically clustering a document bank without supervision. We have asserted that the Clustering Hypothesis holds for natural language documents represented by term-by-document matrices and their transformations by LSI and TF.IDF, which is supported by the first of our two-tier evaluation system. Moreover, the computed clusterings are valid according to the often-used distortion measure, the cophenetic correlation coefficient. Finally, we have observed qualitatively valid clusterings in the generated portals.

Several challenges motivate future research. First, the representation of documents deserves greater attention. While the term-by-document representation (and its transformations) yields valid results, there may be other, more sophisticated features, such as hyperlinks, that yields more relevant clusters. Second, evaluation of the hierarchical clusters is an open problem that is not adequately addressed in the literature. Third, representing clusters, for example by most frequent words in the clustered documents, deserves considerable attention, for this directly affects the opacity (or more optimistically, the transparency) of the document bank. Finally, supervised methods may apply in certain instances. For example, given a hand-built portal, perhaps there are techniques for identifying the underlying criteria and thereby automatically elaborating the portal.

References

[1] R. M. Cormack, "A Review of Classification", *Journal of the Royal Statistical Society*, 134:321–377, 1971.

[2] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, pp. 440-445, MIT Press, Cambridge, Massachusetts, 1990.

[3] Douglass Cutting, David Karger, Jan Pedersen, and John W. Tukey, "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections," *Proceedings of the 15th Annual International ACM/SIGIR Conference*, Copenhagen, 1992.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*, pp. 99–99, John Wiley & Sons, Inc., USA, 2nd edition, 2001.

[5] Taher H. Haveliwala, Aristides Gionis, and Dan Klein, "Similarity Search on the Web: Evaluation and Scalability Considerations," *Stanford Technical Report*, 2001.

[6] Nicholas Jardine and Robin Sibson, *Mathematical Taxonomy*, John Wiley & Sons, Inc., New York, 1971.

[7] Robert F. Ling and George G. Killough, "Probability Tables for Cluster Analysis Based on a Theory of Random Graphs", *Journal of the American Statistical Society*, 71:293–300, 1976.

[8] Christopher D. Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, Massachusetts, 1999.

[9] J. Pedersen and C. Silverstein, "Almost-Constant-Time Clustering of Arbitrary Corpus Subsets", *Proceedings of the Twentieth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 60-66, July 1997.

[10] M. F. Porter, "An Algorithm for Suffix Stripping", *Program*, 14,No.3:130-137, July 1980.

[11] C. J. Van Rijsbergen, *Information Retrieval*, Butterworth, London, 2nd edition, 1979.

[12] Peter Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review", *Information Processing & Management*, 24:577-597.

[13] David Yarowsky, "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods," *ACL*, 33:189-196.