A SURVEY OF TEXT CLUSTERING TECHNIQUES USED FOR WEB MINING

Dan MUNTEANU, Severin BUMBARU

"Dunărea de Jos" University of Galatz
Faculty of Computer Science
Department of Computers and Applied Informatics
111 Domnească Street, 800201-Galatz, Romania
Phone/Fax: (+40) 236 460182; (+40) 236 461353
E-mail: dan.munteanu@ugal.ro severin.bumbaru@ugal.ro

Abstract: this paper contains an overview of basic formulations and approaches to clustering. Then it presents two important clustering paradigms: a bottom-up agglomerative technique, which collects similar documents into larger and larger groups, and a top-down partitioning technique, which divides a corpus into topic-oriented partitions.

Keywords: information retrieval, algorithms, machine learning, web mining, text clustering

1. INTRODUCTION

The Web has become a vast storehouse of knowledge, built in a decentralized yet collaborative manner. It is a living, growing, populist, and participatory medium of expression with no central editorship. This has positive and negative implications. On the positive side, there is widespread participation in authoring content. Compared to print or broadcast media, the ratio of content creators to the audience is more equitable. On the negative side, the heterogeneity and lack of structure makes it hard to frame queries and satisfy information needs.

Statistical dependencies between terms, Web pages, and hyperlinks are also called patterns; the act of searching for such patterns is called machine learning, or data mining.

The data in web mining consists of text, hypertext markup, hyperlinks, sites, and topic directories. This distinguishes the area of Web mining as a new field, although it also borrows liberally from traditional data analysis.

The World Wide Web is the largest and most widely known repository of hypertext. Hypertext documents contain text and generally embed hyperlinks to other documents distributed across the Web. Today, the Web comprises billions of documents, authored by millions of diverse people, edited by no one in particular, and distributed over millions of computers that are connected. Citation, a form of hyperlinking, is as old as written language itself. Dictionaries and encyclopedias can be viewed as a self-contained network of textual nodes joined by referential links. Words and concepts are described by appealing to other words and concepts.

The richness of Web content has also made it progressively more difficult to leverage the value of information. The new medium has no inherent requirements of editorship and approval from authority.

The Web is a set of documents, where each document is a multiset (bag) of terms. Hypertext data is semistructured or unstructured, because they do not have a compact or precise description of data items. Such a description is called a schema, which

This paper was recommended for publication by Cornelia NOVAC

is mandatory for relational databases. The second major difference is that unstructured and semistructured hypertext has a very large number of attributes, if each lexical unit (word or token) is considered as a potential attribute.

Organizing knowledge into ontologies is an ancient art, descended from philosophy and epistemology. An ontology defines a vocabulary, the entities referred to by elements in the vocabulary, and relations between the entities. The entities may be fine-grained, as in WordNet, a lexical network for English, or they may be relatively coarse-grained topics, as in the Yahoo! topic directory.

Topic directories offer value in two forms. The obvious contribution is the cataloging of Web content, which makes it easier to search. The second contribution is in the form of quality control and tend to reflect the more authoritative and popular sections of the Web.

Topic directories built with human effort (e.g., Yahoo! or the Open Directory) lead to a question: Can they be constructed automatically out of a corpus of Web pages, such as collected by a crawler?

The practice of classifying objects according to perceived similarities is the basis for much of science (Jain and Dubes 1988). Organizing data into sensible groupings is one of the most fundamental modes of understanding and learning. Cluster analysis is the formal study of algorithms and methods for grouping, or classifying, objects. An object is described either by a set of measurements or by relationships between the object and other objects. Cluster analysis does not use category labels that tag objects with prior identifiers. The absence of category labels distinguishes cluster analysis from discriminant analysis (and pattern recognition and decision analysis). The objective of cluster analysis is simply to find a convenient and valid organization of the data, not to establish rules for separating future data into categories. Clustering algorithms are geared toward finding structure in the data.

A cluster is comprised of a number of similar objects collected or grouped together. Everitt documents some of the following definitions of a cluster (Everitt, 1974):

1. “A cluster is a set of entities which are alike, and entities from different clusters are not alike.”

2. “A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.”

3. “Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.”

Making sense of data is an ongoing task of researchers and professionals in almost every practical endeavor (Pedrycz, 2005). The age of information technology, characterized by a vast array of data, has enormously amplified this quest and made it even more challenging. Data collection anytime and everywhere has become the reality of our lives. Understanding the data, revealing underlying phenomena, and visualizing major tendencies are major undertakings pursued in intelligent data analysis, data mining, and system modeling.

A clustering algorithm discovers groups in the set of documents such that documents within a group are more similar than documents across groups.

Clustering is a classic area of machine learning and pattern recognition. Clustering and classification are at two opposite extremes with regard to the extent of human supervision they need. Real-life applications are somewhere in between, because unlabeled data is easy to collect but labeling data is onerous.

Cluster analysis is a statistical technique used to generate a category structure which fits a set of observations (Frakes and Baeza-Yates, 1992). The groups which are formed should have a high degree of association between members of the same group and a low degree between members of different groups. While cluster analysis is sometimes referred to as automatic classification, this is not strictly accurate since the classes formed are not known prior to processing, as classification implies, but are defined by the items assigned to them.

Clustering is useful for taxonomy design and similarity search. Topic taxonomies such as Yahoo! and the Open Directory (dmoz.org) are constructed manually, but this process can be greatly facilitated by a preliminary clustering of large samples of Web documents. Clustering can also assist fast similarity search. Similarity, in a rather general way, is fundamental to many search and mining operations on hypertext and is central to most of this book. The utility of clustering for text and hypertext information retrieval lies in the so-called cluster hypothesis: given a “suitable” clustering of a collection, if the user is interested in document d, he is likely to be interested in other members of the cluster to which d belongs. The cluster hypothesis (Rijsbergen, 1979) is not limited to documents alone.

This article contains an overview of basic formulations and approaches to clustering. Also it presents two important clustering paradigms: a bottom-up agglomerative technique, which collects similar documents into larger and larger groups, and a top-down partitioning technique, which divides a corpus into topic-oriented partitions. These are followed by a slew of clustering techniques that can be broadly classified as embeddings of the corpus in a low-dimensional space so as to bring out the clustering present in the data.

2. PROBLEM FORMULATION

It is given a collection D of documents (in general, entities to be clustered). Entities either may be characterized by some internal property, such as the vector-space model for documents, or they may be characterized only externally, via a measure of distance (dissimilarity) $\delta(d_1, d_2)$ or resemblance (similarity) $\rho(d_1, d_2)$ specified between any two pairs of documents. For example the Euclidean distance between length-normalized document vectors for δ can be used and cosine similarity for ρ (Chakrabarti, 2003).

One possible goal that can be set up for a clustering algorithm is to partition the document collection into k subsets or clusters $D_1, D_2, \ldots, D_k$ so as to minimize the intracluster distance $\sum_i \sum_{d_1, d_2 \in D_i} \delta(d_1, d_2)$ or maximize the intracluster resemblance $\sum_i \sum_{d_1, d_2 \in D_i} \rho(d_1, d_2)$. If an internal representation of documents is available, then it is also usual to specify a representation of clusters with regard to that same model. For example, if documents are represented using the vector space model, a cluster of documents may be represented by the centroid (average) of the document vectors. When a cluster representation is available, a modified goal could be to partition D into $D_1, D_2, \ldots, D_k$ so as to minimize $\sum_i \sum_{d \in D_i} \delta(d, \vec{D}_i)$ or maximize $\sum_i \sum_{d \in D_i} \rho(d, \vec{D}_i)$, where $\vec{D}_i$ is the vector-space representation of cluster i.

One could think of assigning document d to cluster i as setting a Boolean variable $z_{d,i}$ to 1. This can be generalized to fuzzy or soft clustering where $z_{d,i}$ is a real number between zero and one. In such a scenario, one may wish to find $z_{d,i}$ so as to minimize $\sum_i \sum_{d \in D} z_{d,i} \delta(d, \vec{D}_i)$ or maximize $\sum_i \sum_{d \in D} z_{d,i} \rho(d, \vec{D}_i)$.

3. BOTTOM-UP AND TOP-DOWN PARTITIONING TECHNIQUES

The heuristic is to start with all the documents and successively combine them into groups within which interdocument similarity is high, collapsing down to as many groups as desired. This style is called bottom-up, agglomerative, or hierarchical agglomerative clustering (HAC) and is characterized by the pseudocode:

1. let each document d be in a singleton group $\{d\}$
2. let G be the set of groups
3. **while** $|G| > 1$ **do**
4. choose $\Gamma, \Delta \in G$ according to some measure of similarity $s(\Gamma, \Delta)$
5. remove Γ and Δ from G
6. let $\Phi = \Gamma \bigcup \Delta$
7. insert Φ into G
8. **end while**

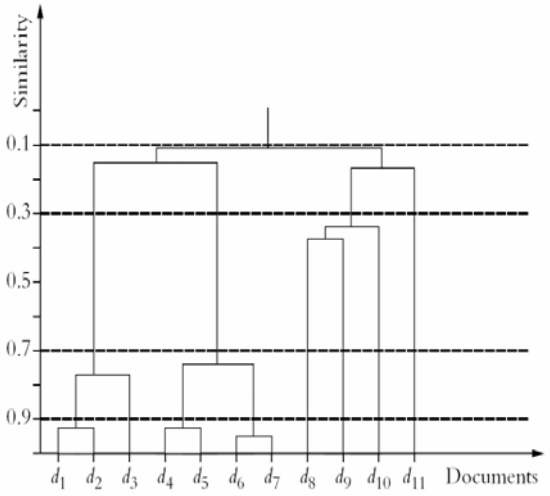

Fig. 1. A dendrogram presents the progressive, hierarchy-forming merging process pictorially.

Typically, the earlier mergers happen between groups with a large similarity $s(\Gamma \bigcup \Delta)$. This value becomes lower and lower for later merges.

Algorithms differ as to how they compute the figure of merit for merging Γ and Δ. One commonly used measure is the self-similarity of $\Gamma \cup \Delta$. The selfsimilarity of a group of documents Φ is defined as the average pairwise similarity between documents in Φ

$$s(\Phi) = \frac{1}{\binom{|\Phi|}{2}} \sum_{d_1,d_2 \in \Phi} s(d_1,d_2) = \frac{2}{|\Phi|(|\Phi|-1)} \sum_{d_1,d_2 \in \Phi} s(d_1,d_2)$$

where the TF-IDF cosine measure is commonly used for interdocument similarity $s(d_1,d_2)$. Other merger criteria exist. One may choose to merge that pair of clusters (Γ,Δ), which maximizes $\min_{d_1 \in \Gamma, d_2 \in \Delta} s(d_1,d_2)$, $\max_{d_1 \in \Gamma, d_2 \in \Delta} s(d_1,d_2)$, or $\sum_{d_1 \in \Gamma, d_2 \in \Delta} s(d_1,d_2)/(|\Gamma||\Delta|)$.

3.1. The k-Means Algorithm

Bottom-up clustering, used directly, takes quadratic time and space and is not practical for large document collections. If the user can preset a (small) number *k* of desired clusters, a more efficient top-down partitioning strategy may be used.

The best-known member of this family of algorithms is the k-means algorithm. Further, will be discusses two forms of the k-means algorithm here. One makes "hard" (0/1) assignments of documents to clusters. The other makes "soft" assignments, meaning documents belong to clusters with a fractional score between 0 and 1.

3.2. k-means with "hard" assignment

In its common form, k-means uses internal representations for both the objects being clustered and the clusters themselves. For documents, the vector-space representation is used, and the cluster is represented as the centroid of the documents belonging to that cluster.

The initial configuration is arbitrary (or chosen by a heuristic external to the k-means algorithm), consisting of a grouping of the documents into k groups, and k corresponding vector-space centroids computed accordingly. Thereafter, the algorithm proceeds in alternating half-steps, as shown below

1. initialize cluster centroids to arbitrary vectors
2. **while** further improvement is possible *do*
3. **for** each document d **do**
4. find the cluster c whose centroid is most similar to d
5. assign d to this cluster c
6. **end for**
7. **for** each cluster c **do**
8. recompute the centroid of cluster c based on documents assigned to it
9. **end for**
10. **end while**

The basic step in k-means is also called move-to-nearest, for obvious reasons. A variety of criteria may be used for terminating the loop. One may exit when the assignment of documents to clusters ceases to change (much), or when cluster centroids move by negligible distances in successive iterations.

3.3. k-means with "soft' assignment

Rather than make any specific assignment of documents to clusters, the "soft" variant of k-means represents each cluster *c* using a vector μ_c in term space. Since there is no explicit assignment of documents to clusters, μ_c is not directly related to documents — for example, it is not necessarily the centroid of some documents.

The goal of "soft" k-means is to find a μ_c for each *c* so as to minimize the *quantization error* $\mu_c \sum_d \min_c |d - \mu_c|^2$.

A simple strategy to iteratively reduce the error is to bring the mean vectors closer to the documents that they are closest to. The documents are scan repeatedly, and for each document *d*, a "correction" $\Delta\mu_c$ c is accumulated for that μ_c that is closest to *d*:

$$\Delta\mu_c = \sum_d \begin{cases} \eta(d-\mu_c), & \text{if } \mu_c \text{ is closest to d} \\ 0 & \text{otherwise} \end{cases}$$

After scanning once through all documents, all the $\mu_c s$ are updated in a batch by setting all $\mu_c \leftarrow \mu_c + \Delta\mu_c \cdot \eta$ is called the *learning rate*. It maintains some memory of the past and stabilizes the system. Note that each d moves only one μ_c in each batch.

The contribution from *d* need not be limited to only that μ_c that is closest to it. The contribution can be shared among many clusters, the portion for cluster *c* being directly related to the current similarity between μ_c and *d*. For example, it can be soften to

$$\Delta\mu_c = \eta \frac{1/|d-\mu_c|^2}{\sum_\gamma 1/|d-\mu_\gamma|^2}(d-\mu_c)$$

or

$$\Delta\mu_c = \eta \frac{\exp(-|d-\mu_c|^2)}{\sum_\gamma \exp(-|d-\mu_\gamma|^2)}(d-\mu_c)$$

Many other update rules, similar in spirit, are possible.

4. LATENT SEMANTIC INDEXING (LSI)

Let the term-document matrix be *A* where the entry *A*[*t*, *d*] may be a 0/1 value denoting the occurrence or otherwise of term *t* in document *d*. More commonly, documents are transformed into TFIDF vectors and each column of A is a document vector.

In the vector-space model, it is allocated a distinct orthogonal direction for each token. The obvious intuition is that there is no need for so many (tens of thousands) of orthogonal directions because there are all sorts of latent relationships between the corresponding tokens. *Car* and *automobile* are likely to occur in similar documents, as are *cows* and *sheep*. Thus, documents as points in this space are not likely to nearly "use up" all possible regions, but are likely to occupy semantically meaningful subspaces of it. Another way of saying this is that *A* has a much lower rank than $\min\{|D|,|T|\}$. One way to reveal the rank of A is to compute its singular value decomposition (SVD). Without going into the details of how the SVD is computed, which is standard, the decomposed form of *A* is

$$A_{|T|\times|D|} = U_{|T|\times r}\begin{pmatrix}\sigma_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \sigma_r\end{pmatrix}V^T_{r\times|D|}$$

where r is the rank of *A*, *U* and *V* are column-orthonormal ($U^TU = V^TV = I$, the identity matrix), and the diagonal matrix Σ in the middle can be organized (by modifying U and V) such that $\sigma_1 \geq \ldots \geq \sigma_r > 0$.

The standard cosine measure of similarity between documents can be applied to the A matrix: the entries of $(A^TA)_{|D|\times|D|}$ may be interpreted as the pairwise document similarities in vector space. The situation is completely symmetric with regard to terms, and can be regarded the entries of $(AA^T)_{|T|\times|T|}$ as the pairwise term, similarity based on their co-occurrence in documents.

The *t*th row of *A* may therefore be regarded as a $|D|$-dimensional representation of term *t*, just as the *d*th column of *A* is the $|T|$-dimensional vector-space representation of document *d*. Because A has redundancy revealed by the SVD operation, a "better" way to compute document-to-document similarities can be used as $(V\Sigma^2V^T)_{|D|\times|D|}$ and term-to-term similarities as $(U\Sigma^2U^T)_{|T|\times|T|}$. In other words, the *t*th row of U is a refined representation of term *t*, and the *d*th row of V is a refined representation of document *d*. Both representations are vectors in an *r*-dimensional subspace, and it can be talked about the similarity of a term with a document in this subspace. In latent semantic indexing (LSI), the corpus is first used to precompute the matrices *U*, Σ, and *V*. A query is regarded as a document. When a query "q" is submitted, it is first projected to the *r*-dimensional "LSI space" using the transformation

$$\hat{q} = \sum\nolimits^{-1}_{r\times r} U^T_{r\times|T|} q_{|T|}$$

At this point $\hat{q}$ becomes comparable with the *r*-dimensional document representations in LSI space. Now one can look for document vectors close to the transformed query vector. In LSI implementations, not all *r* singular values are retained. A smaller number *k*, roughly 200 to 300, of the top singular values are retained—that is, A is approximated as

$$A_k = \sum_{1\leq i\leq k} \vec{u}_i \cdot \sigma_i \cdot \vec{v}_i^T$$

where $\vec{u}_i$ and $\vec{v}_i$ are the *i*th columns of *U* and *V*. How good an approximation is A_k? The Frobenius norm of *A* is given by

$$|A|_F = \sqrt{\sum_{t,d} A[t,d]^2}$$

It can be shown that

$$|A|_F^2 = \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_r^2,$$

and

$$\min_{rank(B)=k} |A-B|_F^2 = |A-A_k|_F^2 = \sigma_1^2 + \sigma_{k+1}^2 + \ldots + \sigma_r^2$$

That is, A_k is the best rank-*k* approximation to A under the Frobenius norm.

The above results may explain why retrieval based on LSI may be close to vector-space quality, despite reduced space and perhaps query time requirements (although the preprocessing involved is quite time-consuming). Interestingly, in practice, LSI does better, in terms of recall/precision, than TFIDF retrieval. Heuristic explanations may be sought in signal-processing practice, where SVD has been used for decades, with the experience that the dominating singular values capture the "signal" in A, leaving the smaller singular values to account for the "noise." In IR terms, LSI maps synonymous and related words to similar vectors, potentially bridging the "syntax gap" in traditional IR and thus improving recall.

LSI may also be able to exploit correlations between terms to resolve polysemy in some situations, improving precision as well.

LSI/SVD was view as a device for dimensionality reduction, noise filtering, and ad hoc retrieval. It can also be used for visualization (choose k = 2 or 3) or clustering, by using any of the other algorithms after applying SVD.

5. CONCLUSIONS

This paper has presented an overview of basic formulations and approaches to clustering. Then it presented two important clustering paradigms: a bottom-up agglomerative technique, which collects similar documents into larger and larger groups, and a top-down partitioning technique, which divides a corpus into topic-oriented partitions. After that a slew of clustering techniques were presented that can be broadly classified as embeddings of the corpus in a low-dimensional space so as to bring out the clustering present in the data.

6. REFERENCES

Chakrabarti, S. (2003). *Mining the Web: Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers, San Francisco, USA.

Everitt, B. S. (1978). *Graphical Techniques for Multivariate Data*. ElsevierNorth-Holland Inc., New York, USA.

Frakes, W. B. and R. Baeza-Yates (1992) Information Retrieval: Data Structures & Algorithms. Prentice Hall PTR, New York, USA

Jain, K.A. and R.C. Dubes (1988). *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey.

Pedrycz, W. (2005). *Knowledge-Based Clustering. From Data to Information Granules*. John Wiley & Sons, Inc., Hoboken, New Jersey.

Rijsbergen, C. J. (1979) *Information Retrieval*, Butterworths, London.