

4. Research in Information Organization

Knowledge organization, the process of organizing concepts into an ordered group of categories (i.e. taxonomy), is the way we humans understand the world. When we encounter a phenomenon for the first time, we try to understand it by comparing it to what we already know and identifying patterns with which to categorize it into our existing frame of reference, thus “transforming isolated and incoherent sense impressions into recognizable objects and recurring patterns” (Langridge, 1992). We learn about the world by looking for differences and similarities in things to find patterns of concepts and relationships. The process of grouping together similar things according to some common quality or characteristics is an integral part of daily activities throughout our lives. As infants, we begin by partitioning the world into those that give us comfort and those that cause us discomfort. As we grow and accumulate more information, we refine our understanding of the world by updating our system of knowledge organization with more complex categories and relationships between them.

John Dewey, an American philosopher, said, “knowledge is classification.” Langridge (1992) states that without classification there could be no human thought, action and organization. Classification, in a broad sense, is a mechanism for both organizing and utilizing knowledge. At a fundamental level, we make sense of the world by organizing its sea of information. In other words, information organization (IO) is essential to learning and ultimately the acquisition of knowledge by humans. Not surprisingly, researchers in the field of Artificial Intelligence, Machine Learning in particular, have been studying IO as a way to emulate human intelligence. IO has also been explored in Information and Library Science as a mechanism with which to bridge the user’s information need and the information collection. Librarians for centuries have been organizing library collections in various ways (e.g. LCC, DDC) to help library patrons find information. Web directories, though more ad-hoc and less standardized than traditional library systems, are another application of IO that guides users through an information discovery process.

IO approaches in IR reflect ideas from both Machine Learning and Library Science. Based on the notion that IO facilitates learning and is one of the basic building blocks of knowledge, some researchers have experimented with IO as a tool to develop intelligent IR systems that go beyond term matching, while others have investigated it as a way to present information in a more sensible way so that users can better grasp the content of information collection. Though system- and user-

oriented IO approaches differ in their focus, both hope to harness intelligence or knowledge to enhance retrieval by first organizing information in some manner. Naturally, a considerable amount of IR research on IO deals with the means of organizing information.

There are two types of information organization mechanisms studied in IR: namely classification²⁹ and clustering. Classification organizes entities by pigeonholing them into predefined categories, whereas clustering organizes information by grouping similar or related entities together. In classification, categories are first determined and objects are assigned to them according to characteristics of interest. In clustering, on the other hand, categories are revealed as a result of grouping objects based on some common characteristics. In the following sections, we will review research that investigated various clustering and classification approaches for IR, after which we will consider their implications to Web IR.

4.1 Clustering

Clustering has been explored in various ways in IR. Hypertext researchers have experimented with clustering to reduce the complexity of document networks (Botafogo & Shneiderman, 1991; Botafogo, 1993), while researchers in visualization have sought to generate a sensible overview of retrieval results by clustering documents on the fly (Cutting et al., 1992; Hearst & Pedersen, 1996). Researchers in Web IR have investigated clustering as a means to identify topical communities (Kumar et al., 1999; Mukherjea, 2000) and to extract useful structures from the Web (Larson, 1996; Pirolli et al., 1996). Clustering has also been used to find duplicate documents (Broder et al. 1997) and as a query expansion mechanism via relevance feedback (Chang & Hsu, 1998).

One of the earliest clustering IR approaches was based on grouping related terms, which extended the idea of vocabulary control to enhance retrieval from the manual keyword indexing days. Term clustering, however, did not prove overly successful and was mostly abandoned in favor of document clustering approaches in traditional IR research, where clustering was applied to the whole document collection based on their textual similarity. More recent research in hypertext and Web IR have explored both link-based clustering algorithms and dynamic clustering of retrieval results.

²⁹ In this paper, “classification” is used in a narrow sense of the term, mostly in the context of text categorization.

4.1.1 Term Clustering

Early researchers in IR investigated term clustering, grouping related terms on the basis of the documents in which they co-occur, as a mechanism to boost recall, either by indexing documents with the controlled vocabulary of an automatically generated thesaurus that maps terms to their cluster identifiers, or automatically expanding the query with related terms. Despite the report of early success by Sparck Jones (1971), subsequent term clustering experiments have met with mixed results and the effectiveness of term clustering in IR is generally in doubt (Salton, 1972; Minker et al., 1972; Salton 1986; Peat & Willett 1991).

In a more recent study, Voorhees (1993) experimented with using WordNet, an online thesaurus of sorts, in an attempt to exploit semantic relationships of terms embodied in an existing lexical knowledge base. Voorhees indexed five test collections (CACM, CISI, CRAN, MED, TIME) with the word sense of terms (e.g. reduce synonyms to a single feature) provided by WordNet and compared it to the traditional term-based representations. The experiment results showed inferior retrieval performances of word sense representations, which was attributed to the difficulty of disambiguating word senses in short queries.

4.1.2 Document Clustering: Conventional Text-based approaches

The idea of document clustering originates from information organization by abstraction (Smith & Smith, 1977), where related objects are “aggregated” to form a higher level object (e.g. arm, leg, head, and trunk are aggregated into a body) or similar objects are grouped together to form a generalized object (e.g. hawk, eagle, sparrow are generalized into a bird). Document clustering methods, depending on the context and the motivation for information organization, can be based on textual similarity, citations, or link structure. Text-based document clustering, otherwise known as cluster analysis in IR, usually groups together similar documents based on their textual similarity, while citation and link-based clustering forms clusters of related documents based on their citation or link topologies.

Text-based document clustering usually involves preprocessing (i.e. stopping and stemming of documents), term weighting, and determination of pairwise similarities between documents or clusters. Though the choice of term weighting scheme is considered critical in conventional term-based retrieval (Robertson & Sparck Jones, 1976; Salton, 1971; Salton & Buckley, 1988; van

Rijsbergen, 1979), it has been found to have little impact on clustering (Rasmussen, 1992). The choice of similarity function has also been shown to make little difference in clustering as long as it includes normalization to account for different document lengths (van Rijsbergen, 1979).

There are two types of document clustering: nonhierarchical and hierarchical. Nonhierarchical clustering simply partitions documents into a given number of clusters, whereas hierarchical clustering groups documents into a hierarchical tree of clusters. Given a target number of M clusters, nonhierarchical clustering can be achieved by generating and evaluating all possible partitions to determine M optimal clusters. Since such complete enumeration is infeasible in practice, however, nonhierarchical clustering usually involves heuristics to generate suboptimal partitions in an iterative fashion. One of the more common approaches is to successively reduce the total within-cluster document similarities to cluster centroids by iterative reassignment of documents to clusters until the clusters stabilize or some threshold similarity level has been reached (Willet, 1988). Determination of final clusters by nonhierarchical clustering methods is arbitrary in a sense that they can depend on parameters such as processing order of documents, number of clusters, or threshold similarity level. Consequently, nonhierarchical clustering is not often used in IR despite its superior efficiency over hierarchical clustering methods that require computationally demanding calculation of similarities between all possible pairs of documents and clusters.

Hierarchical clustering methods can be divisive or agglomerative. Divisive methods build cluster hierarchies from top down by successively dividing more general clusters into finer clusters, while agglomerative methods build them from bottom up by successively agglomerating finer clusters to more general clusters. Divisive clusters are typically monothetic, meaning that cluster members must contain certain terms, whereas agglomerative clusters are polythetic, meaning that documents in a cluster have some terms in common but none are required for membership. Hierarchical agglomerative clustering (HAC), being the most popular type of clustering procedure, dominates clustering research in IR literature and is sometimes referred to as simply hierarchical clustering (El-Hamdouchi & Willett, 1989; Griffiths, Lackhurst & Willett, 1984; Griffiths, Robinson & Willett, 1984; Jardin & van Rijsbergen, 1971; Voorhees, 1986; Willet, 1988).

HAC begins by building a similarity matrix whose elements contain similarity scores of all document pairs to be clustered. Pairs with the highest similarity scores are then fused and the similarity matrix is updated by replacing the rows and columns of fused items with those corresponding to the clusters. The process of clustering and recomputing similarities is repeated

until only the root cluster remains. The final result of HAC is a binary tree-like hierarchy of clusters whose inter-cluster similarity levels increase with the depth of the tree. There are several HAC methods, all of which essentially follow the procedure outlined above but differ in their approaches to how cluster similarities are computed.

The single linkage method uses the most similar pair of documents (i.e. nearest neighbor) between clusters to determine the cluster similarity, thus creating clusters where any cluster member is more similar to at least one member of that cluster than any member of any other cluster. This method tends to form large, loosely bound clusters with little internal cohesion, but is the most popular of HAC methods due to its computational efficiency. The complete linkage method is the converse of single linkage and uses the least similar pair of documents (i.e. furthest neighbor) between clusters to determine their similarity, creating clusters where any cluster member is more similar to the least similar member of that cluster than to the least similar member of any other cluster. The definition of similarity in the complete linkage method is much stricter than that of the single linkage method, and thus affects the creation of many small tightly bound clusters. The group average method uses the average of inter-cluster document pair similarities as the cluster similarity, in an attempt to achieve a compromise between the single and complete linkage method. The group average method has been criticized for its tendency to create small, outlier clusters (Willet, 1988). The Ward method, otherwise known as the minimum variance method, joins clusters whose fusion results in the least increase in the sum of distances from each document to the centroid of its cluster, and tends to produce homogeneous, spherical clusters with symmetric hierarchy that may or may not reflect the true cluster shape of the data.

Given the numerous approaches to document clustering, how do we determine the best clustering method? Conceptually, a good clustering method should produce clusters that reflect the state of data. One way to test for the validity of clusters is to determine whether the clusters are significantly different from random. Another way is to apply clustering methods to documents with known cluster structures and assess the recovery of the original cluster structure. One of the problems with cluster method evaluations stem from the fact that different document collections react differently to clustering, which means the evaluation results from one test collection may not necessarily hold true in other document collections. Consequently, several methods of testing for document collection's tendency to cluster have been proposed. The cluster hypothesis test (Van Rijsbergen, 1979; Jadin & Van Rijsbergen, 1971), for example, is based on the assumption of the

cluster hypothesis that says similar documents are likely to be relevant to the same query, and tests to see if the average similarity between relevant document pairs is greater than that between relevant and nonrelevant document pairs for a given query. Other examples of cluster tendency tests are the overlap test (El-Hamdouchi & Willet, 1989), the nearest neighbor test (Voorhees, 1985), and the term density test (El-Hamdouchi & Willet, 1989). These tests are usually performed prior to clustering to determine if the document collection at hand possesses enough cluster tendency that makes it worthwhile to cluster.

The question of which clustering method is “best” is evaluated in terms of retrieval effectiveness. Retrieval using clusters, known as cluster searching, computes similarities between a query and the cluster representatives, which consist of the top N most frequent terms in clusters (Jardin & Van Rijsbergen, 1971; Van Rijsbergen & Croft, 1975). Since cluster representatives are inappropriate for large clusters, small clusters are considered best for retrieval purposes (Griffiths et al., 1986). With the cluster hierarchy produced by HAC, cluster searching can proceed in a selective fashion without computing all query-document similarities. Top-down search traverses to the child node with the highest query-cluster similarity until the cluster size falls below the minimum number of documents to retrieve or query-cluster similarity falls below a threshold. Since the first few traversal choices at the top of the cluster hierarchy is almost arbitrary due to the large size of clusters, top-down search is often modified to start at a level with some threshold size clusters. Top-down search has been shown to work well with the complete linkage method, which produces a large number of very small clusters even at the topmost levels (Voorhees, 1986). Unfortunately, the combination of top-down search and complete linkage clustering requires the greatest computational resources of all cluster searching methods.

The bottom-up search proceeds in the reverse order of top-down search by starting at the bottom of the tree and traversing upwards until the cutoff in number of documents or similarity threshold has been reached. The starting point of the bottom-up search is usually the bottom level cluster with relevant documents, which are determined by conventional query-document search if not already known (Croft, 1980; Van Rijsbergen & Croft, 1975). Van Rijsbergen and Croft (1975) conducted a comparative study of top-down and bottom-up searches on the Cranfield-1400 collection and found the bottom-up search to be superior to top-down search. In another comparative study of cluster searching strategies, Croft (1980) proposed a modified cluster searching strategy of scanning just the bottom level clusters and showed that simply returning the

most similar bottom-level clusters can achieve good retrieval performance, which was later confirmed by El-Hamdouchi and Willet (1989). Another modified cluster searching strategy was proposed by Voorhees (1985), who suggested that ranking the documents in the top ranking clusters by the query-document similarity scores would result in good retrieval performance.

The effectiveness of cluster searching depends on the clustering method to some extent. Voorhees (1986), for example, compared the retrieval effectiveness of single linkage, complete linkage, and group average methods and found the complete linkage using top-down search to be most effective, while El-Hamdouchi and Willet (1989) compared all four HAC methods using bottom up searches and found the group average method to be most suitable. The comparison of cluster searching to noncluster searching in most experiments showed that clustering did not improve retrieval performance in general and could be even inferior to conventional noncluster searching sometimes (Griffiths et al., 1986; Jardin & van Rijsbergen, 1971; Salton, 1971; van Rijsbergen & Croft, 1975; Willet, 1988).

In addition to being ineffectual for searching, conventional document clustering approaches are generally considered to be too slow and resource intensive for dealing with a large document collection (Rasmussen, 1992; Willet, 1988). Faced with these critical shortcomings, researchers began to explore alternative approaches that could exploit clustering's ability to automatically organize unstructured data while minimizing its weaknesses. The disappointing performance of cluster searching prompted the investigation of clustering as a browsing tool, where users rather than the system were allowed to navigate through the cluster hierarchy to find relevant documents.

An example of such approach is a research by Crouch, Crouch and Andreas (1989), who studied the effectiveness of cluster hierarchy as a visual search aid in an interactive searching experiment where searchers viewed suitable portions of the cluster hierarchy (i.e. local and global views, cluster summaries) and traversed the cluster hierarchy to find relevant documents instead of the system performing the top-down cluster search. The result of this study showed significant improvement by user-guided cluster searching over traditional cluster searching, which suggested that incorporation of human judgment is not only desirable but a viable alternative to conventional system-oriented approaches to cluster searching.

4.1.3 Document Clustering: Scatter/Gather

The idea of clustering as a browsing tool that incorporates human judgment to improve the cluster searching process is extended in an interactive clustering method called “Scatter/Gather” (SG), where users can select a set of clusters from which to dynamically generate a new set of clusters (Cutting et al., 1992; Hearst & Pedersen, 1996). One of the fundamental differences between Scatter/Gather (SG) and conventional clustering methods lies in their assumptions about clustering³⁰. SG revises the conventional assumption of the cluster hypothesis, which states that relevant documents tend to cluster together with other relevant documents, by contending that clustering of relevant documents is query-dependent. In other words, SG assumes that clustered documents relevant to one query may not all be relevant to another query, which implies clustering applied to the retrieved set of documents would find more relevant documents for a given query than a static clustering of the whole collection.

Scatter/Gather (SG) combines the notions of browsing, relevance feedback, and dynamic clustering by presenting a small number of query-dependent clusters (“scatter”) and allowing users to select relevant ones (“gather”), which are then reclustered for further cycles of “scatter” and “gather”. All three aspects of SG are integrated in such a way to actively involve the user in the search process with minimum effort and maximum reward. SG lessens the cognitive load required to get the gist of documents by presenting short summaries of clusters, while facilitating interactive learning and information discovery by allowing users to dynamically manipulate the cluster structure presentations to aid understanding about both the document corpus and the information need.

The advantages of SG over both conventional clustering and searching are manifold. By dynamically clustering retrieved documents or user-selected sets of clusters, SG efficiently creates query-dependent clusters that cater to the user’s information need from successively smaller and refined sets of documents. SG also helps the user to get a quick overview of retrieval results by presenting documents in the form of clusters instead of a ranked list, and facilitates the query refinement process by allowing the user to simply select relevant clusters rather than requiring him to formulate a precise query with “correct” words. The combined strength of SG makes it an

³⁰ Though SG can be applied to the whole document collection (Pirolli et al., 1996), it is typically applied to the set of retrieved documents to alleviate the computational cost of clustering as well as to improve retrieval performance. Consequently, SG in this paper will generally refer to the query-driven SG.

effective mechanism with which to deal with short, broad, ill-formed, or partially defined expressions of information need, which are typical of Web queries.

Crucial to the success of SG are good partitioning of documents into valid clusters and generation of coherent cluster summaries that capture the essences of cluster members. Incorrect or incomplete cluster summaries as well as inappropriate grouping of documents can result in inadvertent filtering out of relevant documents. The effectiveness of SG methods alone does not guarantee the success of SG, however. The SG process is iterative, which means the search by SG could be more time consuming and demanding than the conventional one-shot search approach. Therefore, SG methods need to be efficient as well as effective so that good clusters and coherent cluster summaries can be generated quickly on the fly to keep the user engaged until his information need has been satisfied.

Cutting et al. (1992) outline two clustering algorithms tailor-made for SG. *Buckshot*, the faster of the two, and *Fractionation*, the more accurate of the two, both retain the clustering effectiveness while speeding up the clustering process by working in conjunction with an effective but slow conventional clustering method (group average agglomerative clustering in their case). SG clustering is achieved by first finding a set number of cluster centers³¹ using either Buckshot or Fractionation to which documents are assigned³², and then applying iterations of cluster refinement steps³³ until some stopping condition has been reached. The *Buckshot* algorithm chooses a small random sample of documents and applies the group average method to produce k clusters. Since documents are sampled randomly each time, this algorithm applied to the same corpus at different times may not produce the same k cluster centers. The *Fractionation* algorithm partitions documents into fixed size buckets, each of which is agglomerated by the group average method. The resulting clusters are partitioned into buckets to be agglomerated and the process repeats until only k clusters remain.

The SG clustering process can be thought of as a nonhierarchical clustering infused with the strength of an agglomerative clustering method, which produces a good set of initial clusters that

³¹ Cutting et al. (1996) define the cluster center as the length normalized centroid vector of m most “central documents” of the cluster, which are the m documents closest to the cluster centroid.

³² Assignment of a document to the “nearest” cluster center is achieved using the *Assign-to-Nearest* algorithm, which finds the cluster center with the highest similarity to the document.

³³ The cluster refinement step consists of recomputation of cluster centers and reassignment of documents to clusters based on a combination of *Assign-to-Nearest*, *Split*, and *Join* algorithms. The *Split* algorithm separates poorly defined clusters into two better-defined clusters, whereas *Join* merges highly similar clusters.

can be refined in a few iterations to quickly produce high-quality clusters. Cutting et al. also define a “cluster digest” as a combination of the most “central” words and documents of the cluster (e.g. n highest weighted terms of the cluster center plus the titles of m documents most similar to the cluster centroid) and suggest that such cluster digests can produce short but revealing descriptions of clusters.

Integration of SG with the conventional search strategy (i.e. application of SG to organize the retrieved documents) was compared with conventional keyword searching without SG in a TREC-4 interactive track experiment (Hearst et al., 1996). The results showed significant improvement by SG search over conventional search³⁴, thus accrediting the assertions from an earlier work (Hearst, Karger & Pederson, 1995) that suggested SG could facilitate interactive searching by identifying and segregating themes specific to the retrieved documents. The capacity to form topic-coherent clusters that are customized to a given set of documents is one of the key strengths of SG over static clustering methods. SG clusters are more closely tailored to the characteristics of a query than static clusters since SG works with local features of documents that are focused on the query rather than the global features that are independent of the query.

Hearst and Pederson (1996) illustrate this with examples, where SG produced clusters of “astronomy” and “film stars” given a query of “star”, and clusters of “astronomy”, “automobile”, and “game” with a “Saturn” query. Hearst and Pederson point out that conventional text-based search can retrieve documents that are textually similar to the query but not central to the intention of the query, and suggests that SG can help the user to filter out such irrelevant document subsets by grouping documents according to their topical contexts.

4.1.4 Document Clustering: Link-based Approaches

Text-based clustering groups together similar documents based on their textual content, but link-based approaches to clustering leverage the information embedded in the hyperlinks to group related documents. The basic assumption underlying the link-based approaches is that hyperlinks indicate semantic relationships between documents, where the relationship is transitive but diminished with path length and more links signify a stronger relationship.

³⁴ The experiment also reported that SG performance is strongly related to cluster size (optimal size of 20 to 50 documents) and more than 5 clusters is not good idea.

Link-based clustering methods were first investigated in hypertext research using graph-based algorithms. Botafogo and Shneiderman (1991) used a graph-based algorithm that found and clustered sets of “strongly connected” documents based on the compactness measure, which was defined as a function of the average link distance between hypertext nodes. The resulting clusters were not very discriminative, however, and highly linked documents tended to end up in a same cluster. Based on the idea that the relationship between hypertext documents is proportional to the number of independent paths between them, Botafogo (1993) extended his earlier work by using the number of independent paths between nodes to identify clusters in hypertext. Botafogo’s modified algorithm was shown to separate hypertext into reasonable size clusters, where nodes in a cluster were highly related.

Another approach to link-based clustering originates from citation analysis (White & McCain, 1998). In an exploratory study examining the growth and the bibliometrics of the Web, Larson (1996) applied cocitation analysis with a dimension-reduction technique called multidimensional scaling (MDS) to cluster a set of Web pages about Earth Science. Larson’s clustering method begins with the construction of a cocitation matrix, whose $(i,j)^{\text{th}}$ entry contains the number of documents linking to both document i and j . Larson then converts the cocitation matrix to a correlation matrix, thereby changing raw co-occurrence information to values that can be treated as a form of topical proximity between Web documents. The final step involves processing the correlation matrix using the SAS MDS procedure to map the multidimensional set of proximities onto a two-dimensional graph. The resulting MDS map of Web cocitations (i.e. hyperlinks) revealed several document groupings that seemed to cluster around various topics of Earth Science.

Kumar et al. (1999) describe a novel link-based clustering process called “trawling”, which combines the idea of cocitation and graph analysis to identify clusters of related Web pages. The main objective of trawling is to automatically identify “emerging” Web communities, which are clusters of related Web pages whose presence is too new or whose topic is too fine-grained to attract the attention of large Web directory services such as Yahoo. Since such emerging Web communities tend to have relatively rare structures, they are likely not be captured by methods that look for major patterns in data (e.g. MDS). Theoretically, non-principal eigenvectors of an adjacency matrix (Gibson, Kleinberg, & Raghavan, 1998) could reveal such Web communities, but

it is not only computationally expensive on the scale of Web³⁵ but could also miss some interesting yet sparsely linked communities while identifying false communities.

To circumvent these problems, trawling employs graph-based algorithms that build on assumptions of cocitation and HITS. The fundamental assumption in trawling is that related pages are frequently referenced together, even in the case of newly emerging communities. Trawling also subscribes to the idea of Web communities consisting of mutually reinforcing hubs and authorities (Kleinberg, 1997), and further assumes that such communities of dense directed bipartite subgraphs have at least one core, where core is a complete directed bipartite subgraph³⁶ with a certain number of nodes. The basic approach of trawling is to find such cores first and then use them to find the rest of the communities using graph-based algorithms.

The trawling process starts by identifying potential “fans,” which are defined as pages with links to multiple Websites³⁷. 24 million potential fans were extracted from 200 million pages from Alexa by finding pages with links to at least 6 different Websites. After potential fans have been identified, trawling employs an aggressive duplicate elimination strategy to exclude duplicate and near-duplicate Web pages that can produce large spurious cores. Trawling’s duplicate elimination strategy applies the shingling method proposed by Broder et al. (1997)³⁸ to the sequence of links to construct a number of local hashes of the Web page and compares the smallest few hash values (i.e. shingles) to detect duplication. The combination of aggressive duplicate elimination and large numbers of duplicates due to mirroring and URL variations resulted in a 60% reduction of 24 million potential fans.

The resulting set of “unique” potential fans and centers (i.e. pages pointed to by potential fans) are trimmed based on their indegree distributions. Since the focus of trawling is on identifying emerging rather than established Web communities, the set of potential cores are pruned by excluding highly referenced pages (i.e. pages with indegree beyond a threshold k , which was set

³⁵ Kumar et al. identified over 100,000 Web communities by trawling 200 million Web pages, the computational cost of which would have been prohibitive if eigen-analysis was applied directly.

³⁶ A complete bipartite graph on node-sets F , C contains all possible edges between a vertex of F and a vertex of C . Note that a bipartite graph is a graph whose node set can be partitioned into two sets.

³⁷ A “fan” and a “center” are special cases of a hub and an authority that make up a Web community core. The definition of a “potential fan” was arrived at by examining 500 good Clever search results to find the characteristics of good hubs, which revealed that good hubs tended to link to good resources on multiple Websites. Kumar et al. defines a “Website” as the first field of a URL (e.g. www.unc.edu).

³⁸ Broder et al. (1997) defines a “shingle” as a contiguous subsequence of words and “ n -shingling” as the set of all unique shingles of size n . For example, the 2-shingling of a document consisting of word sequence $(w_1, w_2, w_3, w_1, w_2, w_4)$ is the set $\{(w_1, w_2), (w_2, w_3), (w_3, w_1), (w_2, w_4)\}$.

to 50 by Kumar et al.) so that generally popular pages won't dominate the rest of the trawling computations. Pruning by indegree still left 2 million potential fans with 60 million links to over 20 million potential centers, which was still too large a dataset from which to find true cores.

Consequently, trawling applies a series of further pruning algorithms to reduce the data to a manageable size. The first algorithm, called iterative pruning, iteratively prunes nodes and edges of potential fans and centers with in- or outdegree smaller than threshold values defined by the target core size, which is followed by the inclusion-exclusion pruning algorithm that applies the conditions of core membership to prune and identify cores. The definition of a core as a complete bipartite graph requires that a (i,j) core should consist of i centers all of which point to j fans and j fans all of which point to i centers. Thus, the inclusion-exclusion pruning algorithm can check for each fan the existence of $1-j$ fans that point to i centers and do similarly for each center to either eliminate a page or discover an (i,j) core. The final stage of trawling consists of the core filtering step, which filters out cores with fans residing on the same Websites to eliminate artificially constructed communities serving the interest of a single entity, and the core building step, which identifies cores by the iterative enumeration algorithm of Agrawal and Srikant (1994). Kumar et al. manually inspected 400 randomly selected cases out of over 100,000 cores discovered by trawling, and found that 96% were thematically cogent and 56% were not categorized by Yahoo.

4.1.5 Document Clustering: Hybrid Approaches

The link-based clustering approaches reviewed so far rely solely on the information provided by links, just as text-based clustering approaches rely solely on the textual contents to identify clusters of related documents. In text-based clustering, the relation between documents that form clusters is that of similarity, whereas in link-based clustering, the relation is that of citation. Although similarity could be one of the evidences given by citations, citations are created by authors for reasons other than similarity. As a result, link-based clustering can suffer from the very nature of the links that it attempts to exploit, unless some compensatory measures are taken to augment its strengths while diminishing its weaknesses. The same could be said for most any singular approach to IR, where hybrid or fusion approaches have shown to almost always outperform their singular component approaches.

Clustering is no exception. As we have seen with Web searching, there have been various Web clustering researches that combine text- and link-based methods. One of the pioneers of such

hybrid clustering approaches are Pirolli et al. (1996), who clustered and categorized Web pages by representing them as vectors of hybrid feature combinations, and Weiss et al. (1996), who proposed an agglomerative clustering method that used a hybrid similarity measure to cluster Web pages. Pirolli et al. represent each Web page as a single vector of features, which consists of values derived from link topology, textual similarity, usage data, and page meta-information (e.g. size, title), and use spreading activation techniques on various Web graphs to cluster related pages³⁹. Though Pirolli et al. represent documents with a hybrid combination of features, they do not propose an explicit method of “fusing” these information values for the purpose of clustering. Instead, they apply spreading activation separately to each type of feature graph, such as the link topology graph with arcs denoting hyperlinks, the text-overlap graph with arcs denoting text-similarity scores, or the usage pattern graph with arcs denoting number of clicked links, thus creating different types of clusters.

In contrast, Weiss et al. propose the “content-link” clustering method, which uses complete linkage clustering based on a hybrid similarity measure derived from both content and link information. As described earlier, the complete linkage clustering is a form of the agglomerative clustering method that produces a hierarchy of clusters by iteratively merging the two most similar clusters, where similarity is defined as the minimum similarity between inter-cluster document pairs. Instead of using the textual similarity measures to compute the document pair similarity, as is the case in the conventional agglomerative clustering, content-link clustering uses a hybrid similarity measure, which is defined as the larger of text- and link-based similarity scores. The link-based similarity is computed by a linear combination of Direct Path (DP), Common Ancestors (CA), and Common Descendants (CD), where DP varies inversely with the length of the shortest link path between documents, CA is proportional to the number of ancestors (i.e. inlinks) that two documents have in common, and CD is proportional to the number of descendents (i.e. outlinks) that two documents have in common. The text-based similarity is computed by the normalized dot product of document term vectors, which is similar to the similarity function by Salton and Buckley (1988) except for the omission of the collection frequency component (i.e. *idf*). Weiss et al. tested

³⁹ Spreading activation works by “activating” a node (i.e. initializing a nodes with some value) at some starting point in a graph network and “spreading” the activation throughout the network by traversing the arcs. The spreading of activation is modulated by arc strength (i.e. the activation value transferred between two nodes are amplified or diminished by the value associated with the arc connecting them). The topmost “active” result nodes (i.e. nodes with highest activation values) can be considered to be related to the starting node.

their hybrid clustering method against term- and link-based clustering methods by clustering pages on the CNN Website (www.cnn.com) with each method and comparing the validity of resulting clusters using the manual clustering of categories provided on CNN's homepage. The result showed that the hybrid method produced clusters that most closely resemble the manually constructed clusters of CNN.

Content-link clustering is only one aspect of HyPursuit, an experimental hierarchical network search engine that incorporates a multitude of features designed to help users in their search tasks (Weiss et al., 1996). Such features include an interface that allows users to browse the information space by traversing the cluster hierarchy, an abstraction function to generate cluster summaries, a query refinement component that suggests recall- and precision-enhancing terms, and a result set expansion option by clusters. The abstraction function generates cluster summaries called "content labels" by selecting the most heavily weighted terms of cluster members, and the query refinement component suggests "collocated" terms, which are the highest weighted terms from content labels that match the query, as well as broader and narrower terms from an automatically generated thesaurus. The result set expansion option, when exercised, suggests documents that are in the same clusters with those in the result set but are not found by the search. Another novel feature of HyPursuit is its support of multiple cluster hierarchies, which are constructed by different clustering methods based on the context and the motivations for the information organization. For example, HyPursuit users can browse and search document clusters based on author, topic, or source.

More recent studies on hybrid clustering approaches explore closer integration of text- and link-based methods. Both approaches by Modha and Spangler (2000) and Mukherjea (2000a; 2000b) start with a text-based search to identify a set of seed pages, which are expanded using link traversal to construct a collection of potentially related documents that can be organized by clustering methods. Such a strategy of link-based expansion of text-based search results strengthens the modified cluster hypothesis of Scatter/Gather by defining a local neighborhood of documents that are related to the query by both content and citation. Mukherjea boosts the quality of the link-based expansion by setting a text similarity threshold as was done in a hybrid search strategy (Bharat and Henzinger, 1998). Once the subcollection on a particular topic has been determined, Modha and Spangler identify document clusters based on a hybrid similarity measure that combines

text and link information to define the cluster similarity, while Mukherjea hierarchically agglomerates documents based on successive levels of abstraction.

Modha and Spangler's hybrid similarity measure differs from that of content-link clustering by Weiss et al. (1996). Whereas the content-link clustering computes text- and link-based similarities separately and takes the larger of the two, Modha and Spangler uses a linear combination of text- and link-based similarities to arrive at the hybrid similarity measure. This hybrid similarity measure reflects the extended similarity assumption that considers two documents sharing words and hyperlinks to be semantically similar. Modha and Spangler build on the idea of hybrid feature vector by Pirolli et al. (1996) by representing each documents as a triplet of unit vectors $(\mathbf{D}, \mathbf{F}, \mathbf{B})$, where \mathbf{D} is a vector of terms with term frequency, \mathbf{F} is a vector of outlink with outlink frequency, and \mathbf{B} is a vector of inlinks with inlink frequency, all of which are length-normalized to be unit vectors. The hybrid similarity measure $S(\mathbf{x}, \mathbf{x}')$ between document $\mathbf{x}=(\mathbf{D}, \mathbf{F}, \mathbf{B})$ and $\mathbf{x}'=(\mathbf{D}', \mathbf{F}', \mathbf{B}')$ can then be expressed as a weighted sum of inner products as follows:

$$S(\mathbf{x}, \mathbf{x}') = a_1 * \mathbf{D}^T \mathbf{D}' + a_2 * \mathbf{F}^T \mathbf{F}' + a_3 * \mathbf{B}^T \mathbf{B}'. \quad (11)$$

Modha and Spangler also defines a cluster representative vector called "concept triplet" $\mathbf{c}_j=(\mathbf{D}_j, \mathbf{F}_j, \mathbf{B}_j)$, where each component vector is a length normalized sum of corresponding vectors in the cluster j . The sum of similarities between each document in a cluster and the concept triplet \mathbf{c}_j can be considered as a measure of "coherence" or "quality" of the cluster j , and the sum of sums of cluster qualities can be thought of as a measure of overall quality of the given partitioning. Modha and Spangler uses this measure to define an objective function, which is the basis of their clustering algorithm.

To construct a subcollection of related Web pages, Modha and Spangler expand 200 top-ranked AltaVista search results from a broad-topic query by collecting all outlinks and the top 20 inlinks of each seed page. The top 20 inlinks are determined arbitrarily by the order of "link:URL" query results returned by AltaVista. The subcollection is then partitioned into clusters using a nonhierarchical clustering approach. The clustering process starts with an arbitrary partitioning of documents, which are iteratively refined by computing new concept triplets and reassigning documents to clusters with the most similar concept triplets to maximize the partitioning quality

measured by the objective function. In practice, the iteration stops when some stopping condition is met, which is typically defined as a threshold of change in the objective function.

Modha and Spangler also propose a scheme for cluster annotation that leverages concept triplets to describe various aspects of clusters. Descriptive content of a cluster is given by “summary”, “breakthrough”, and “review”, which are documents whose component vectors are closest in cosine similarity to those of the concept triplet, and discriminative characteristics of a cluster is given by “keywords”, “citations”, and “references”, which are elements of each component vectors of the concept triplet with the highest weights among all concept triplets. “Summary”, the document with highest textual similarity to the concept triplet, represents the most typical words in the cluster, whereas “breakthrough” represents the most typical inlinks and “review” represents the most typical outlinks in the cluster. “Keywords”, which are words whose weights are higher in the given concept triplet than in any other concept triplet, represents the most discriminating words in a cluster, while “citations” represents a set of most typical links *entering* a given cluster and “references” represents a set of most typical links *exiting* a given cluster.

Another recent study by Mukherjea (2000a; 2000b) describes WTMS⁴⁰, a system for collecting and analyzing topic-specific Web information, which employs a type of hybrid clustering approach that combines link- and text-based methods. The aim of WTMS is to find Web pages related to a particular topic and organize them in ways that can help the user to understand the information space better, which is different from other link-based clustering approaches that focus on identifying clusters of documents on various topics without an explicit means of organizing them. Three main components of WTMS, namely the selective crawler that collects related pages, the organizer that groups together crawled pages at various level of abstraction, and the interface that allows the user to navigate and search through the organized information space, work in conjunction with one another to provide an integrated approach to information discovery.

The WTMS crawler starts with 100 seed pages which are the top 100 pages from a Google search, and expands them by collecting their in- and out-links whose textual similarities to the representative document vector (RDV)⁴¹ are above a threshold score. As with Clever and other advanced HITS-based methods, the seed set expansion is repeated twice (i.e. 2 link hops) using a stoplist of URLs. The result of such selective crawling that tempers link-based expansion with text

⁴⁰ WTMS is an acronym for Web Topic Management System.

⁴¹ RDV is defined as a vector of 25 most frequently occurring terms excluding stopwords in the seed page set.

similarity comparison is a collection of densely linked pages related to the topic of the original query (Bharat and Henzinger, 1998).

Once the pages have been collected, WTMS organizes those pages in a hierarchical fashion by agglomerating them in successive levels of abstraction. The pages are first agglomerated by their domains (e.g. www.yahoo.com), after which the domain level clusters are agglomerated by their logical Websites indicated by URL substrings common in domain names (e.g. www.yahoo.com and shopping.yahoo.com are grouped into yahoo.com). The next level of abstraction involves grouping together of logical Websites that form “strongly connected components”⁴² (SCC) in the site graph induced by Websites and links between them. The top level of the abstraction hierarchy is determined by the final abstraction step that groups together SCCs that form connected components⁴³ (CC) in the SCC graph. WTMS clustering of three topic collections showed the number of SCC clusters to be not much smaller than that of Websites in all collections, which confirmed the previous findings by Bray (1996) and Broder et al. (2000) who observed the pattern of sparse cyclical references between Websites. In contrast, the number of CCs was small (under 1% of Web pages) in all three collections, which Mukherjea interpreted as an indication of WTMS’ ability to find a small number of meaningful clusters from densely linked Web pages.

The interface of WTMS includes several types of views that display the organized collection in various ways to facilitate information discovery. The tabular view lists logical Websites and displays various information about each Website such as URL, number of pages, the title of the main page determined by the connectivity and the depth of the page in the site hierarchy (Mukherjea & Hara, 1997), and hub and authority scores computed by applying the HITS algorithm to the site graph rather than the Web graph. The abstraction hierarchy view uses glyphs to display the hierarchical structure of the collection, where different shapes signifying different levels of abstractions (e.g. circle for Web pages, rectangles for clusters) are presented in a horizontal hierarchy. In this view, size, brightness, color, shape, and order of glyphs convey various information about the object each feature represents. For example, the glyph size is proportional to

⁴² A strongly connected component is a maximal subgraph of a directed graph, whose every node pair (i,j) has a directed path from i to j and a directed path from j to i .

⁴³ A connected component is a maximal subgraph of an undirected graph, whose every node pair (i,j) has a path between i and j .

the cluster size, the brightness proportional to its freshness, the thickness of the arrow next to a circle or a rectangle proportional to the number of in- or outlinks as indicated by the direction of the arrow, and the order of glyphs signifies their relative importance or quality measured by the maximum authority scores of Websites. WTMS also includes a popup window that shows additional information about each glyph, as well as a scatterplot view that displays related pages of a given URL. The scatterplot is constructed by mapping Web pages' structural and semantic (i.e. textual) similarities to a given URL on x - and y -axes⁴⁴, thus presenting an overview of a cluster in terms of the textual and link-based relationships.

4.2 Classification

In some contexts, classification is used in a broad sense that subsumes the concept of clustering. Clustering in turn implies organization sometimes, which is a typical characteristic of classification. No matter how broadly or narrowly these two terms are used, clustering and classification differ in several essential aspects while sharing a set of basic principles. Clustering and classification both group together related things according to some common quality or characteristics, but they differ in how those common characteristics are determined. In classification, characteristics of predefined categories dictate which entity characteristics should be examined, whereas in clustering, the characteristics that form the groupings can be chosen dynamically to suit the purpose and the context of the organization.

The clustering in IR, however, typically restricts the binding characteristic to one of overall similarity or proximity between entities, which are derived from textual content, link topology, or a combination of the two. Interestingly enough, grouping by similarity often produces clusters with “cluster-centric” characteristics that are distinct from one another. Classification starts with a given set of characteristics that define each cluster (i.e. category), but clustering starts with only a similarity measure for the whole corpus, which discovers inherent clusters with defining characteristics. In some sense, IR clustering methods attempt to capture the *a priori* structure of a corpus by allowing the similarity to work like the magnetic force between documents. This is quite contrary to the general consensus on classification that says there is no unique, essential, natural or

⁴⁴ The structural similarity between pages A and B is based on the link structure and computed by counting the number of direct and indirect links between them. Indirect links, which consist of transitive links (e.g. B links to C and C links

a priori system of classification (Jevons, 1877; Langridge, 1992; Lesk, 1997; Norman, 1994; Soergel, 1985). If we view information organization as a strictly logical, precise, and deterministic tool for IR, it might be worthwhile to ponder the ramifications of seemingly conflicting assumptions about the nature of information organization in clustering and classification. As Norman (1994) points out, however, we should perhaps consider tools that match human capabilities, which are not precise, logical, or accurate.

Given that, it suffices to say that clustering reveals the inherent organization structure of the corpus, whereas classification imposes a predetermined organization scheme to the corpus. We should be aware that in classification there is seldom a unique location for a given item and exhaustive or universal classification as well as mutually exclusive classes are difficult if not impossible to establish. Having said that, we should also note that many classification approaches in IR overlook these difficulties of classification for the reasons of simplicity and practicality. In the following sections where we review various IR approaches to classification, we will sidestep these fundamental challenges of classification for the most part so that we may focus on the details of text categorization mechanisms in IR research.

4.2.1 Text Categorization: Overview

Text categorization (TC) is the task of classifying text documents. More specifically, TC assigns text documents to a pre-specified set of classes (i.e. categories), which are also called topics, themes, or even index terms in the context of automatic indexing in IR. The idea of TC is fundamental to manual indexing of documents, which has been used since the invention of writing to facilitate information access. For years librarians have been using controlled vocabularies such as LCSH and MeSH to index library collections for the purpose of organizing information and promoting knowledge management. As the body of information grew exponentially, however, means of automatically organizing information that can emulate or supplement human ontological effort became desirable. Consequently, researchers in the field of Artificial Intelligence, Machine Learning in particular, have been investigating various ways to automatically classify documents, the results of which carried over to IR research in the area of automatic text categorization⁴⁵.

to A), cocitation links (e.g. C links to both A and B), and bibliographic coupling link (e.g. both A and B point to C), are only counted as half.

⁴⁵ Henceforth, automatic TC will be simply referred to as TC.

Most TC approaches employ machine learning (ML) algorithms (Mitchell, 1998) that learn from a set of training examples to train a “classifier” and apply the trained classifier to a target set of documents to determine the best categories. Such ML algorithms are called “supervised” learning algorithms, as opposed to “unsupervised” learning algorithms that try to find useful relations between elements of the target set without the benefit of the training process (e.g. clustering). TC in IR, especially in the context of the Web, poses unique challenges due to the large number of attributes present in the data set, large number of training samples, attribute dependency, and multi-modality of categories. Despite these challenges, TC has been used in IR for organization, filtering, and retrieval of documents.

Though the complete enumeration of all TC techniques is beyond the scope of this dissertation, a list of major TC approaches, some of which will be briefly examined in the next section, is provided here for an overview. The major TC techniques are:

- regression models (Yang & Chute, 1992)
- relevance feedback
 - TFIDF (Salton, 1991; Rocchio, 1971)
 - Probabilistic TFIDF (Joachims, 1997)
- decision trees (Quinlan, 1986; Koller & Sahami, 1997)
- Bayesian probabilistic approaches
 - Naive Bayes (Lewis & Ringuette, 1994, McCallum et al. 1998)
- inductive rule learning (Apt, Damerau & Weiss, 1994; Cohen & Singer, 1996)
- nearest neighbor (Mitchell 1997; Yang & Pedersen, 1997; Yang, 1999)
- neural networks (Weiner, Pedersen & Weigend, 1995)
 - Self-Organizing Maps (Kohonen, 1989, Lin et al., 1991; Kaski et al., 1996)
- Support Vector Machines (Joachims, 1998; Dumais et al. 1998)

Before moving on to the details of TC techniques, a general review of neural networks, which constitute the underpinnings of the TC landscape along with the popular probabilistic and vector models, will be presented for the sake of completeness.

The neural network (NN)⁴⁶ model is inspired by the observation of biological organisms and their large interconnected Webs of neurons. NNs consist of a Web of “neurons” (i.e. units, nodes, processors), each of which receives one or more inputs and produces a single output, and “connections” (i.e. arcs, edges, links), which connect the neurons and function as unidirectional communication channels that carry data across the network. The output of a neuron becomes the input of other neurons connected to it, and in this fashion the set of inputs at some starting point propagate throughout the network via arc traversal, “activating” nodes along the way. Connections are usually assigned weights, which modulate the data that pass through them.

A “perceptron” is a type of NN that takes a vector of real-valued inputs (x_1, \dots, x_n) , and computes a linear combination of them to produce a single output. This process is described mathematically below:

$$O(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n w_i x_i + \theta > 0 \\ 0 & \text{otherwise} \end{cases}, \quad (12)$$

where each w_i is a weight that determines the contribution of the corresponding input value x_i , and θ is a threshold that the combination of inputs must surpass to set the output to 1. The learning process in a perceptron involves choosing the best values of w_i and θ based on the set of training examples. Geometrically, the above formula represents a line that separates data points into two partitions; thus, perceptrons can learn only linearly separable problems.

In contrast to the perceptron, which can be thought of as modeling a single layered NN, multi-layered networks have their neurons organized by layers, where the neurons in one layer are fully connected to the neurons in the next layer. Back-propagation networks are multi-layered NNs that learn the weights using a *Gradient Descent* algorithm (GDA). GDA finds the set of weights that best fit the training data that are not linearly separable by minimizing the squared error between network output values and the correct values. The GDA formula defines a parabolic error surface with a single global minimum, which assures convergence toward a best-fit approximation (Mitchell, 1998).

⁴⁶ In this paper, NN will refer to the artificial neural network rather than biological NN.

Since the idea of spreading activation is drawn from the cognitive processes of human associative memory, the NN model serves as a good theoretical foundation for modeling human information retrieval process. Consequently, NNs have been widely explored by various researchers in IR. Belew (1989) is one of the first IR researchers to adopt the NN model, which is otherwise known as the connectionist model in IR. He used a three-layer NN of authors, index terms, and documents and used relevance feedback to change its representation of authors, terms, and documents over time.

Kwok (1989) is another pioneer who defined a general connectionist model of IR as a three-layer NN of queries, index terms, and documents. In his model, document terms are mapped to nodes, which are connected to documents nodes in which they occur with term frequency weights attached to the links. Activation spreads from the query vector via the term layer to the document layer, at which point the activation value of a document node can be thought of as its relevance score. In the case of relevance feedback, activation from relevant documents spreads back to the term layer, reinforcing the matching query terms and adding activation to new terms from the relevant documents, after which activation spreads forward again to the document layer, where additional documents may be activated.

Other applications of NN for IR include the probabilistic IR model based on a Bayesian inference network that can combine multiple sources of evidence about documents and a query (Turtle & Croft, 1990), a self-organizing semantic map based on Kohonen's self-organizing feature map (SOM) that produces a two-dimensional grid representation of semantic relationships between documents (Lin, Soergel & Marchionini, 1991), and WEBSOM that uses a two-level architecture consisting of two hierarchically interrelated SOMs, which are the word category map describing word relationships and the document map reflecting document relations (Kaski et al., 1996). Though the NNs fit well with the vector space and probabilistic model, the size and complexity of document-term connections can become problematic for a very large corpus. In the context of TC, however, which usually involve the reduction of features, the NN approach becomes a viable alternative to the probabilistic or vector space approaches.

4.2.2 Text Categorization: Conventional Text-based Approaches

The TC process typically consists of preprocessing, indexing, dimensionality reduction, and classification steps. TC begins by preprocessing the training set of documents, which can involve

“stopping” that eliminates common English words, and “stemming” that conflates morphological variations of words. The indexing step represents documents as feature (i.e. word, term) vectors using “the bag-of-words representation”, which is a TC term that defines a document as a set of words without consideration to the order of words. The document feature vector can be Boolean (i.e. 0/1 weights) or weighted by term frequency information.

Application of the preprocessing and indexing steps produce a set of unique features of the training set, which often contains spurious information not germane to classification. Therefore, a dimensionality reduction (DR) step is applied next to reduce the number of features. Classification performance and training time are closely related to the quality of DR since the features that emerge from DR are ones that characterize a given category and bind appropriate documents to it. There are two approaches to DR in TC: namely, feature selection, which selects the “best” feature subset, and re-parameterization, which constructs new features as combinations or transformation of the original features. An example of the re-parameterization approach is Latent Semantic Indexing (LSI), which reduces the dimensionality by decomposing the term-document matrix into a set of k (typically 200 to 300) orthogonal factors using a technique called Singular Value Decomposition (Deerwester et al., 1990). The basic assumption underlying LSI is that there is some underlying or latent structure in the pattern of words, which can be captured in the reduced dimension of k factors.

There are two types of feature selection approaches. The wrapper approach attempts to identify the best feature subset for a given classification algorithm by an iterative error-correction process that consists of feature subset update and classification performance measurement. The filter approach, the most commonly used DR approach by far, attempts to assess the merits of the feature set from the data alone independent of a particular classification algorithm. There are several types of filter approaches, which differ mainly in their ranking criterion for features. The major filtering feature selection methods are Document Frequency Thresholding, Information Gain, χ^2 -statistic, Mutual Information, and Term Strength.

Document Frequency Thresholding (DF) simply removes infrequent words based on the assumption that rare terms are non-informative for category prediction (Joachims, 1997, 1998). DF is the simplest DR method with the lowest computational cost since it does not use the category information in its computations. Information Gain (IG), the term goodness criterion in machine learning (Quinlan, 1986), measures information gained for category prediction by knowing the

presence or absence of a word in a document using an entropy-based formula⁴⁷. Term Strength (TS) measures term importance based on how commonly a term is likely to appear in closely related documents by the conditional probability of a term occurring in a similar document in the same category given that it appears in a document. The χ^2 -statistic (CHI) measures the lack of independence between a word and a category by computing the chi-square statistic of the two-way contingency table of a term and a category. Mutual Information (MI), like CHI, uses the two-way contingency table of a term t and a category c to measure how well the occurrence of a word predicts the class membership of a document. $I(t,c)$, the MI criterion between t and c is defined as a log of probabilities and estimated using the contingency table as seen below:

$$I(t,c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)} \approx \log \frac{A \times N}{(A+C) \times (A+B)}, \quad (13)$$

where A is the number of times t and c co-occur (i.e. the number of documents with term t in category c), B is the number of times t occurs without c (i.e. the number of documents with term t in categories other than c), C is the number of times c occurs without t (i.e. the number of documents without term t in category c) and N is the total number of documents. The major difference between CHI and MI is that the chi statistic is a normalized value, which means CHI values are more comparable across terms for a given category. However, low cell values of the contingency table caused by infrequent terms diminish the validity of the chi statistic.

Once the feature set has been identified, the training of the classifier (i.e. learning) takes place by presenting each example represented by its set of features and letting the classification algorithm adjust its internal representation of the knowledge contained in the training set. After a pass of the whole training set, which is called an epoch, some algorithms such as neural networks check to see whether it has reached its learning goal. Algorithms such as Rocchio and Bayesian need only a single epoch. Some of the major classification algorithms are Rocchio's relevance

⁴⁷ Entropy is related to the amount of information it contains. Shannon's formula, $I(M) = -\log_2 P(M)$ gives the entropy $I(M)$ of a message M in terms of the probability $P(M)$ of the message occurrence (Shannon, 1948). The entropy of M , $I(M)$, can be thought of as the number of bits required to describe the probability of M , and is synonymous with the information content of M . The formula tells us that a message contains more bits of information (i.e. higher entropy) when the probability of its occurrence is lower. IG computes the change in entropy (i.e. entropy loss) of a document belonging to a category before and after a given term is introduced as a feature of the document in question. Entropy loss is synonymous with information gain.

feedback, Naïve Bayes, k -nearest neighbor, linear least square fit, decision trees, support vector machines, voted classification, and SOM.

Rocchio's relevance feedback algorithm (RF), otherwise known as the Find Similar classifier, uses the well-known relevance feedback formula (Rocchio, 1971) to build a prototype vector for each category. The similarities between a document and prototype vectors are computed to find the best category for a document. RF considers all training documents in a given category as relevant and constructs the category prototype vector by setting α and γ to zero in the relevance feedback formula⁴⁸, since there are no non-relevant documents and no previous query in the classification context. Therefore, the prototype vector is computed as the average vector (i.e. a vector with average term weights) over all documents in the category. RF is also called TFIDF classifier when $tf*idf$ term weights are used to represent documents. Joachims (1997) compared TFIDF, Native Bayes and prTFIDF, a probabilistic version of the TFIDF classifier, and suggested that probabilistic classifiers are preferable to the RF classifier. Han and Karypis (2000), however, who tested a variation of RF called a "centroid-based" classifier in an comparative classification experiment with several test collections, reported superior performance of the centroid-based classifier over other classifiers.

The centroid classifier is essentially the same as the RF classifier except that it represents documents as unit vectors (i.e. length normalized) and uses a cosine measure (i.e. centroid vector lengths normalized) to compute the document-category similarity. The centroid vector C given a set S of documents is determined as:

$$C = \frac{1}{|S|} \sum_{d \in S} d. \quad (14)$$

The cosine similarity measure is computed by:

$$^{48} Q_{new} = \alpha \cdot Q_{old} + \beta \cdot \frac{\sum_{i \in rel} D_i}{n_r} + \gamma \cdot \frac{\sum_{i \in non-rel} D_i}{N - n_r},$$

where Q is a query, D is a document, n_r is the number of relevant documents, and N is the total number of documents in the category.

$$\cos(d, C) = \frac{d \bullet C}{\|d\| * \|C\|} = \frac{d \bullet C}{\|C\|}, \quad (15)$$

where \bullet signifies dot product and $\| \ \|$ signifies the vector length. $\text{Sim}(\mathbf{x}, C)$, the similarity measure between document \mathbf{x} and category C , then becomes the dot product (\mathbf{x}, C) divided by the length of C . Thus, the numerator of the similarity coefficient represents the average similarity between \mathbf{x} and all other documents in the set, and the denominator represents the square root of average pairwise similarity between documents in the category. Based on these observations, Han and Karypis argue that the centroid classifier classifies a document based on how closely its behavior matches that of documents in different classes by dynamically adjusting to different density of classes and accounting for dependencies between terms in different classes.

The Naïve Bayes classifier (NB) uses the training data to estimate the probability of each category c_j given a new doc d as:

$$P(c_j | d) = \frac{P(c_j)P(d | c_j)}{P(d)}. \quad (16)$$

$P(d)$ in above formula can be left out since it does not differ between categories and the “naïve” assumption of term independence simplifies the formula to:

$$P(c_j | d) \approx P(c_j) \prod_{i=1}^m P(t_i | c_j), \quad (17)$$

where the document d consists of m terms t_1, t_2, \dots, t_m . $P(c_j)$ in the equation 17 can be estimated as the fraction of training documents assigned to class c_j as:

$$\hat{P}(c_j) = \frac{N_j}{N}, \quad (18)$$

where N is the number of documents in the training set and N_j is the number of documents in the class j . $P(t_i/c_j)$ in the equation 17 can be estimated by term occurrence ratio as:

$$\hat{P}(t_i | c_j) = \frac{1 + N_{ij}}{m + \sum_{k=1}^m N_{kj}}, \quad (19)$$

where N_{ij} is the number of times the term i occur in the class j . Equation 19 uses the Laplace estimator suggested by Vapnik (1982, pp. 54-55) that assumes the observation of each word is *a priori* equally likely.

NB estimates the probability of each category for a given document, based on the prior probability of a category occurring, and the conditional probabilities of particular words occurring in a document given that it belongs to a category, assuming that these probabilities are conditionally independent. Unfortunately, words within a document are not independent of each other in reality. Despite its unrealistic assumption of word independence, NB has been shown to be quite effective in classifying text documents (Yang and Pederson, 1997; Joachims, 1997).

K -nearest neighbor (KNN) is similar to RF, but uses a document's k nearest neighbors instead of all documents in the training set to compute the document-class similarity. To classify a document d , KNN finds k nearest neighbors of d in each category of the training set, whose similarity to d is used to rank the categories. Linear Least Square Fit (LLSF) is a mapping approach developed by Yang & Chute (1992) that maps the input document vector of weighted terms to the output binary vector of categories using a multivariate regression method. The decision tree approach matches the document vector d against a decision tree to determine whether a document is relevant to a category. CART, C4.5, and CHAID are examples of decision tree algorithms. Voted classification (VC) combines the prediction of multiple classifiers to produce a single classifier. Typically, VC trains a classifier multiple times on different versions of the training set, which are then combined to create the final classifier.

Support vector machines (SVM), a machine learning method by Vapnik (1995), learn from the training set of documents to find a decision surface in the vector space of documents that “best” separates the data points (i.e. documents) into two class (relevant and nonrelevant). Consequently, SVM is only applicable to binary classification that is linearly separable. SVM was shown to outperform a variety of other classification methods by Joachims (1998), who conjectured that SVM's ability to use all the words in the training set directly as features without employing feature reduction was advantageous to other classifiers that employ aggressive feature reduction that could result in a loss of information.

Neural network classifiers map input words to a category using the neural network model. In addition to the perceptron and multi-layered NN, Kohonen's SOM (1989) forms the basis for another type of classifier. SOM is different from all other classifiers reviewed so far in that it is an

unsupervised learning method (i.e. does not require a training set). The inspiration for SOM is based on the idea that some high level organization in the brain may be created during learning through self-organization and the representation of knowledge in categories may assume the form of a feature map that is geometrically organized over corresponding pieces of the brain. Accordingly, SOM maps a set of input objects, each represented as an N-dimensional vector, onto nodes of a two-dimensional map, which reveals the frequency and distribution of the underlying data through the spatial arrangement of nodes. For instance, the distance between documents, nearest neighbors of each word, and size of each area in a SOM map are well determined by, and therefore reflect, the internal structure of input data. SOM maps also preserve distance relationships between input data and map more frequent input patterns to larger regions of the map. SOM has been shown to perform relatively well in noise (Lippmann, 1987). The SOM algorithm is described in Appendix A.

4.2.3 Text Categorization: Hierarchical Approaches

Most research on TC approach classification as a binary categorization problem. Rather than determining the category of a document from a multitude of categories, typical TC algorithms classify a document as relevant or not relevant to a given category. However, most real world entities, such as Web documents, are composed of a variety of topics (i.e. multi-class) and can belong to more than one class (multi-label). The common approach of binary TC algorithms to multi-class categorization is to transform a multiple-category assignment problem into multiple binary decision problems by breaking the classification task into multiple disjoint binary tasks. In other words, a document is classified for each category, and the binary results are pooled to arrive at a single decision based on ranking of possible categories. The main problem with such an approach is that it ignores correlation between classes and trivializes multi-label categorization.

Furthermore, most TC approaches are ill equipped to deal with a hierarchical classification scheme like Yahoo, which is one of the most successful WWW paradigms for organizing diverse and complex information. In fact, the Web is rich with topic hierarchies and large training sets, but most classification research has ignored supervised learning that takes advantage of such hierarchical classes. It is possible to apply standard classification techniques to hierarchical classification by constructing a “flattened” class space, with one class for every leaf in the hierarchy, and treating it as a typical multi-class categorization. However, “flattening” is an

impractical solution for dealing with massive hierarchies such as those found in the Web⁴⁹, which consist of hundreds of classes and thousands of features, because of prohibitive computational cost of classification as well as the classifier's tendency to overfit the training data with too many parameters.

The Pachinko Machine by Koller and Sahami (1997) is one of the few true hierarchical classifiers that make full use of the hierarchy to classify documents. Koller and Sahami construct a hierarchical set of classifiers by training a separate classifier at each internal node of the tree (i.e. classification hierarchy). A document is then classified in a top-down fashion by starting at the root category and selecting the best sub-categories until the bottom level is reached. The divide and conquer approach of the Pachinko Machine (PM) uses the hierarchical structure to break up a large problem space into manageable size pieces by dividing the classification task into a set of smaller classification problems corresponding to the splits in the hierarchy. Consequently, PM only needs to distinguish between a few categories at a time instead of making judgment over all categories in the hierarchy. In other words, leveraging the hierarchical structure enables classifiers at each node to learn to differentiate among a specific set of similar documents by focusing only on a small set of features relevant to the classification subtask at hand, whereas classifiers with a flattened hierarchy need to learn to differentiate among the whole universe of documents using a much larger set of features.

For example, the task of classifying a document into the categories of animal husbandry, crop farming, computer software, and computer hardware becomes considerably simpler, if we first determine whether the given document is about agriculture or about computers, and then determine whether it is about hardware or software if about computers, or animal husbandry or crop farming if about agriculture. Of course, the accuracy of classifiers at each node of classification hierarchy must be very high, since one incorrect classification decision is likely to affect all subsequent decisions and thus invalidate the entire classification process.

⁴⁹ According to the Search Engine Watch (<http://searchenginewatch.com/>), Open Directory (<http://dmoz.org/>) had classified 2.2 million Web pages in 325,000 categories as of December 2000, LookSmart (<http://www.looksmart.com/>) had 2 million pages in 200,000 categories as of August 2000, and Yahoo (<http://www.yahoo.com>) had categorized 1.8 million pages as of August 2000. The Search Engine Watch does not report the number of Yahoo categories. Labrou and Finin (1999) estimated Yahoo categories to be over 150,000 in 1999.

PM uses probabilistic methods for both feature selection and classification: namely, a feature selector based on Mutual Information and a modified Bayesian classifier based on the KDB algorithm⁵⁰ of Sahami (1996). Before applying the MI feature selector, PM performs an initial feature selection based on Zipf’s Law (Zipf, 1949) by eliminate words appearing in less than 10 or more than 1000 documents. In an experiment using the Reuters-22173 test collection⁵¹, Koller and Sahami investigated the effects of the feature set size, KDB classifier, and hierarchical classification by comparing systems with varying number of features, different classifiers (Native Bayes or KDB), and flattened or hierarchical classification structure. The results showed significant improvement in classification accuracy as the number of features was reduced up to a point. The combination of the KDB classifier and hierarchical classification also showed significant classification accuracy gain over the performances of other systems, from which Koller and Sahami concluded that combination of hierarchical classification, aggressive feature selection of under 100 features, and the use of a richer dependency model in the KDB algorithm were the keys to the success of PM.

TAPER by Chakrabarti et al. (1997), the Taxonomy And Path Enhanced Retrieval system, is another system that classifies documents according to a hierarchical classification scheme. TAPER trains classifiers at each node with diverse feature sets, but chooses the leaf (i.e. category) with the least-cost path from root instead of greedily searching for the best leaf as in PM. The least-cost path is given by computing the “global” probability $P(c_i/d)$ of a document d belonging to a leaf class c_i as:

$$P(c_i | d) = P(c_{i-1} | d)P(c_i | c_{i-1}, d), \quad (20)$$

where $P(c_{i-1}/d)$ is the global probability of d belonging to the parent category of c_i (i.e. c_{i-1}), $P(c_i/c_{i-1}, d)$ is the conditional probability of d belonging to c_i given that it belongs to c_{i-1} . $P(c_i/c_{i-1}, d)$ is computed as the “local” probability of d belonging to c_i divided by the sum of “local” probabilities of d belonging to the sibling categories of c_i .

⁵⁰ KDB algorithm augments the Naïve Bayesian classifier’s term independent assumption by introducing a richer dependency model that allow limited interaction between features.

⁵¹ The Reuters test collection, used in many classification and filtering experiments, are Reuters newswires from 1987, consisting of a training set of 9,603 stories and a test set of 3,299 stories in 135 categories.

In addition to hierarchical classification, TAPER implements querying of the taxonomy to help users focus the query to the most relevant areas of the classification hierarchy. In response to a query, TAPER's taxonomy querying module returns a list of topic paths before retrieving any documents, so that users can refine their queries with related concept terms or restrict their search to selected topic paths. Analysis of TAPER's performance with the Reuters collection and the US patent database found that the use of hierarchy increased classification accuracy only moderately, but greatly increased the speed of classification.

McCallum et al. (1998) employed a statistical technique called "shrinkage" (James & Stein, 1961), which "shrinks" parameter estimates in data-sparse children towards the estimates of the data-rich ancestors, to leverage the topic hierarchy. The shrinkage technique offers an advantage in using large hierarchies whose classes are sparsely populated, where a small set of training documents can normally cause problems for a classifier. McCallum et al. describes a simple shrinkage technique of estimating $P(t_i/c_j)$ (probability of word i occurring in class j) by a linear combination of estimates found along the path from the leaf class j to the root of the class hierarchy. This shrinkage approach results in a large feature set for a single classifier at the bottom level, which is different from PM that makes a series of classification decisions based on small feature sets. In order to increase the computational efficiency, however, McCallum et al. suggest integrating PM's top-down tree traversal approach to prune the tree. By pruning all but the 3 branches with the highest probabilities at each level of the hierarchy, McCallum et al. avoid calculating shrinkage probabilities for a majority of leaf classes.

Results from experiments using the UseNet data with a two-level hierarchy of 20,000 articles in 75 categories (Joachims, 1997), company Web pages classified in a two-level hierarchy of 6440 industry sector Web pages in 71 categories, and the entire Yahoo Science hierarchy of 14831 pages in 264 categories, demonstrated the effectiveness of the shrinkage method with large class hierarchies. The experiments also showed that dynamically pruning the classification tree could exponentially reduce the computation time with minimal accuracy loss.

Ruiz and Srinivasan (1999) hypothesized that the hierarchical structure of a vocabulary system like MeSH, which reflects the conceptual structure, could be used to guide text classification. Building on this hypothesis, they proposed a hierarchical neural network (HNN) model of classification based on a divide-and-conquer principle, much like Koller and Sahami (1997). Their HNN, built by organizing MeSH terms hierarchically using associated relations, is

composed of internal nodes of “gating” networks, which represent high-level concepts and act as a gate to access lower levels, and the leaf nodes of “expert” networks, which specialize in recognizing documents corresponding to specific categories. Classification proceeds in a top-down fashion by spreading activation from the most general concept gates to the most specific concept gates. Comparison of the HNN classifier with the Rocchio classifier and flat neural network classifier showed that the HNN model performed significantly better than either of the other two classifiers.

An important distinction of Ruiz and Srinivasan’s classification approach lies in the method of training set selection. Typical TC approaches use training documents in a given category as positive examples (i.e. relevant) and all other documents in the training set as negative examples (i.e. non-relevant). With hierarchical categories, such methods result in a few positive examples and an overwhelming number of negatives examples, which can adversely affect the classifier. To avoid such an eventuality, Ruiz and Srinivasan employ the concept of a category zone similar to the concept of the query zone (Singhal, Mitra & Buckley, 1997). The training set in a given category zone is identified by using the centroid vector of all positive examples as a query to retrieve the top 10,000 documents, to which any unretrieved positive examples for the category are added. The resulting category zone will include many “nonrelevant” documents that are closely related to the target category, which provides a training set well suited for the task of the hierarchical classifier.

Though not directly related to hierarchical TC, the research by Sanderson and Croft (1999) is interesting in that they propose a method of automatically deriving a concept hierarchy without using training data or clustering methods, which can provide users with an overview of the topical structure of the retrieved documents. Their method of concept hierarchy creation builds on the ideas from previous research in discovering term relationships, which include organizing terms by relationship (Miller, 1995), locating synonyms by terms’ context similarity (Grefenstette, 1994), using key phrases as indicators of hyponym or hypernym (Hearst, 1995), and using principal component analysis of the term correlation matrix to cluster and visualize term relationships (Newby, 1998).

In contrast to most prior works on deriving term relationships from texts that do not explicitly create concept structures with an ordering from general to specific terms, Sanderson and Croft construct a term relation hierarchy by using a method based on co-occurrence known as “subsumption”. The subsumption method is based on the simple idea that the parent concept should subsume the child’s concept. Thus, term x is a parent of term y , if x subsumes y , which is defined

by the condition $P(x/y)=1$ and $P(y/x)<1$. In practice, however, some y do not always co-occur with x , so the subsumption condition is modified as $P(x/y)\geq 0.8$ and $P(y/x)<1$.

The subsumption method consists of the term selection step and term hierarchy creation step. The term selection step identifies the set of terms that best represent the concepts contained in the retrieved documents by first expanding the query terms via Local Context Analysis (LCA) that finds terms commonly co-occurring with each other across best matching passages of top ranked documents (Xu & Croft, 1996). The LCA-expanded query terms are then augmented by the set of terms in the best passage of each document, whose relative frequency of occurrence in the retrieval set to the whole corpus is above a threshold. Once the concept term set has been identified, the concept hierarchy is created by the process of pairwise term comparisons that test for the subsumption relationship between terms⁵².

4.2.4 Text Categorization: Link-based Approaches

Since text analysis lies at the heart of classification algorithms, it is difficult to classify documents based on link-based methods alone. As is the name implies, link-based IR approaches leverage links for retrieval purposes. Link-based searching ranks documents by a measure of importance or quality based on the endorsement implied by links, and link-based clustering groups together documents by a measure of similarity based on the relationship implied by links. Link-based classification, however, does not seem to be able to leverage links directly for pigeonholing documents into categories. Instead, the main idea of link-based TC is to go beyond the local content of a document and exploit non-local features introduced by links, which are then utilized by text-based classifiers.

IR research in document enrichment, however, found that naïve introduction of non-local features could be worse than using local text alone due to noise (Chakrabarti, Dom & Indyk, 1998; Salton & Zhang, 1986)⁵³. Based on the observation that it is the topics of documents, not their contents, that determine linking behavior, Chakrabarti, Dom and Indyk (1998) suggest that the topics, rather than the raw textual contents of linked documents, should be used to enrich the feature set of a category. Accordingly, they propose a technique called “relaxation labeling” that bootstraps

⁵² 200 subsumption relationships were identified out of a 2420 term set extracted from 500 retrieved documents.

off a text-only classifier and iteratively updates the resulting classification based on local and non-local features introduced by links. To be more specific, relaxation labeling starts off by applying a text-based classifier to the document's neighborhood (i.e. documents linked to the target document), and expands the local features with the resulting class labels to classify the target document. This process is performed iteratively until classification stabilizes.

To test the effectiveness of the relaxation labeling technique, Chakrabarti, Dom and Indyk used TAPER, the hierarchical text classifier discussed in the previous section, to classify IBM's patent database and Yahoo documents⁵⁴. The results showed that relaxation labeling significantly boosted performance (i.e. high classification accuracy) compared to the naïve approach of using the raw text from neighbors. They also examined the classification performance of using link-based features alone (i.e. class labels of neighbors) without any local features, and discovered that such a link-only classifier works well only when a large fraction of the neighborhood has known classes. In other words, the integration of local and linked features is important in leveraging links for TC, if the document neighborhood itself is largely unclassified. Conversely, if the reasonable portion of the document neighborhood happens to be already classified, non-local features of neighboring documents' class labels would be sufficient information for classification.

4.2.5 Text Categorization: Evaluation

Since TC studies report findings that sometimes seem to conflict with one another and often cannot be compared directly, it might be useful to iterate at this point the various TC evaluation measures and then review some of the evaluation studies that compare TC methods.

Classifiers, after being trained on a set of training documents with known classes, are typically tested on a distinct set of test documents, also with known categories. Evaluation of a classifier is usually based on such test documents, and can employ one or more of the performance measures of recall, precision, fallout, accuracy, and error (Lewis, 1991; Aas & Eikvil, 1999). These measures, defined for each category, are as follows:

⁵³ Salton and Zhang found that inclusion of cited titles could add spurious words that degrade retrieval quality. Chakrabarti, Dom and Indyk found that TC using texts of linked documents performed worse than using local text alone.

⁵⁴ IBM's patent database had 3 first level nodes and 12 leaves (4 per node) with 630 documents per leaf for training and 300 documents for testing. Yahoo data used consisted of 20,000 documents in 13 classes.

$$\text{recall} = \frac{a}{a+c} \quad (21)$$

$$\text{precision} = \frac{a}{a+b} \quad (22)$$

$$\text{fallout} = \frac{b}{b+d} \quad (23)$$

$$\text{accuracy} = \frac{a+d}{a+b+c+d} \quad (24)$$

$$\text{error} = \frac{b+c}{a+b+c+d} \quad (25)$$

The numbers denoted by a , b , c , and d in above equations come from the cells of a contingency table of true category membership by classifier assignment, where a is the number of documents correctly assigned to the category, b is the number of documents incorrectly assigned to the category, c is the number of documents incorrectly rejected from the category, and d is the number of documents correctly rejected from the category. $(a + c)$ in equation 21 represents the total number of documents belonging to the category, $(a + b)$ in equation 22 represents the total number of documents assigned to the category, and $(a + b + c + d)$ in equations 24 and 25 represents the total number of documents evaluated for the category.

For evaluating performance average across categories, there are the “macro-averaging” method, which computes the global performance scores by averaging the per category scores, and the “micro-averaging” method that first computes the contingency table values for all categories to compute the global performance scores. Micro-averaging tends to be dominated by classifier’s performance on common categories, while macro-averaging tends to be influenced by rare categories (Yang & Liu, 1999). There are also measures that combine recall and precision, such as “break-even” point (Lewis, 1992) and the F -measure (van Rijsbergen, 1979). The break-even point is simply the point where recall and precision values are the same. The F -measure is defined as:

$$F_{\beta} = \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R}, \quad (26)$$

where P denotes precision, R , recall, and β is a parameter that defines the relative importance of recall and precision. For example, F_0 (i.e. $\beta=0$) is simply the precision, F_∞ is recall, and F_1 is the Dice coefficient that gives precision and recall equal weights.

Yang and Pedersen (1997) conducted comparison studies of feature selection methods and found that Information Gain (IG) and χ^2 -statistic (CHI) methods were most effective. They also found that Document Frequency Thresholding (DF), the simplest feature selection method with the lowest computational cost, performed competitively despite its ad-hoc approach. Further analysis revealed that DF, IG, and CHI term scores were strongly correlated, from which they concluded that DF is more than just an ad-hoc method previously assumed in the TC literature, but a reliable measure for feature selection. Experiments with different numbers of selected features report somewhat conflicting findings. Lewis et al. (1996) and Yang and Pedersen (1997) found that the best results are generally obtained using only a small number of features, but others have contended that larger feature sets could be more advantageous (Han & Karypis, 2000; Joachims, 1997; McCallum et al., 1998).

There are many classifier comparison studies in the literature, most of which seem to agree that most classifiers perform comparably with a slight advantage by the Support Vector Machine (SVM) classifier. Dumais et al. (1998) compared Find Similar, Native Bayes (NB), Decision Tree, and SVM classifiers using the Reuters dataset and found SVM to be the most accurate of all classifiers tested. Also using Reuters dataset, Yang and Liu (1999) found that SVM, k-Nearest Neighbor (KNN), and Linear Least Squares Fit (LLSF) mapping significantly outperformed Neural Network and NB classifiers when the number of positive training examples per category were small (less than ten), but all the methods performed comparably when the categories were sufficiently common (over 300 training examples). Based on an analysis of results from 5 previous studies, Aas and Eikvil (1999) concluded that all methods performed reasonably well and none was markedly superior to others. An exception is the study by Han and Karypis (2000), who reported that their centroid-based classifier consistently and substantially outperformed NB, KNN, and C4.5 (decision tree) classifiers on various datasets⁵⁵. Unfortunately, the centroid classifier was not compared to the VSM classifier.

⁵⁵ Datasets used were TREC-5, 6, 7, WestGroup, Reuters-21578, OHSUMED, and WebACE collections.

4.3 Organizing Web Documents

Traditional text-based IR research uses homogeneous corpora with coherent vocabulary, high quality content, and congruous authorship. The Web corpus, however, introduces the challenges of diverse authorship, vocabulary, and quality. Furthermore, some Web documents are intentionally fragmented to facilitate navigation and hyperlinking, making it difficult to determine their topics from local contents alone. Leveraging hyperlinks in the Web is also more problematic than in traditional hypertext IR research due to the diversity in link types that are hard to classify automatically.

Information organization (IO) on the Web inherits all the problems and challenges generally associated with Web IR. For instance, it is difficult to cluster or classify the whole Web due to its massive size and diversity. Even if such a feat were possible, most clustering approaches, which are not incremental, and text categorization approaches, which are based on a static classification scheme, will not be able to deal with the dynamic nature of the Web corpus. Consequently, methods of organizing Web documents need to be efficient, flexible and dynamic. Moreover, post-retrieval organization of retrieved documents may be a more desirable as well as realistic approach than trying to organize the entire Web.

4.3.1 Clustering the Web

We have already reviewed some studies that explored clustering of Web documents, such as cocitation analysis to identify topic clusters (Larson, 1996), the “trawling” process to find emerging Web communities (Kumar et al., 1999), and topic management approaches that collect and organize Web pages related to a particular topic (Modha & Spangler, 2000; Mukherjea, S., 2000). The post-retrieval clustering approach of Scatter/Gather (Cutting et al., 1992; Hearst & Pederson, 1996) also has been applied to the Web setting to dynamically produce topic-coherent clusters of retrieved documents (Sahami, Yusufali & Baldonado, 1998)

One of the few studies that focused on tailoring their IO approaches specifically for the Web, which demand fast, effective, and dynamic algorithms that produce concise and accurate descriptions, was conducted by Zamir and Etzioni (1998). To satisfy the rigorous requirements of the Web environment, they proposed an incremental, linear time clustering algorithm called *Suffix Tree Clustering* (STC), which clusters documents based on shared phrases. The STC algorithm

works in three stages. The first step involves stopping and stemming, which is followed by the identification of base clusters. A base cluster is defined as a set of documents that share a phrase. A data structure called a suffix tree is used to efficiently identify all base clusters, after which they are combined by merging clusters with high degrees of overlap in their document sets. This final step relaxes the stringent requirement of base clusters that require all documents in a cluster to share a common term, and by doing so creates more semantically coherent clusters by grouping documents that share related phrases with one another. STC, due to the nature of its algorithm, creates overlapping clusters, can be applied incrementally, and uses phrases that consider both the order and position of words with which to cluster, which are novel properties not found in other clustering methods.

In an experiment with Grouper, the clustering interface to the HuskySearch metasearch engine, Zamir and Etzioni (1999) use STC to dynamically group the metasearch results into clusters. Grouper uses snippets returned from search engines rather than the full-text of documents to avoid fetching documents, which is usually the biggest bottleneck in Web IR⁵⁶. Grouper presents cluster summaries in a table, ordered by estimated coherence⁵⁷. A cluster summary consists of the cluster size (i.e. number of documents in the cluster), shared phrases and words ordered by coverage (i.e. % of documents in which a term appear), and sample titles. The options from the cluster summary table allows users to either view documents in clusters or “refine” the search based on clusters, which use the shared phrases of clusters to reformulate the query.

Zamir and Etzioni evaluated Grouper based on users’ search behaviors recorded by its system logs. To investigate their first hypothesis, namely that users would tend to follow documents from relatively few clusters, they computed the average number of followed clusters as a function of the number of documents followed and compared it to that of random clustering. The result was disappointing in that users examined 7 documents from 3 clusters on the average, which was not markedly better than the behaviors of users with random clusters. Zamir and Etzioni suggest a possible explanation for this finding by saying that users seek overview rather than exhaustive review of retrieved documents. Needless to say, another possible explanation is that the clustering method used simply is not effective enough to make a difference.

⁵⁶ A prior study comparing the clusters using snippets to those using full-texts has shown the snippet approach to have only a moderate degradation of 15% average precision loss (Zamire & Etzioni, 1998).

⁵⁷ The Coherence of a cluster is estimated by the number of cluster (i.e. shared) phrases it contains.

The second hypothesis, namely that Grouper's clustering interface is a better alternative to the traditional ranked list display of retrieval results, were evaluated using the number of documents followed to estimate the amount of retrieved information by the user, time spent on traversing the result set as a measure of search efficiency, and the click distance, which is the physical display distance between clicked links, to measure the cost of finding relevant documents. Zamir and Etzioni used these three metrics as rough measures with which to identify patterns in search behavior, and observed that finding the first few relevant documents require more effort in Grouper, but gets easier after the first two or three documents.

In addition to applying the text-based clustering method to better organize and present Web search results to users, Grouper also integrates the concept of relevance feedback in its "refine" feature. The idea of integrating query expansion and conceptual relevance feedback with clustering is explicitly discussed by Chang and Hsu (1998). They propose a method where the cluster instead of individual documents of a metasearch result is used as a feedback unit, much like Grouper's "refine". They differ from Grouper in that the feedback query is expanded by cluster vectors using the Rocchio formula (Rocchio, 1971), where the cluster vector is composed of the top k weighted terms. Regardless of which formulas are used to expand the query and to weight the terms, the basic idea is to expand the query by the best features of relevant clusters, which may be more representative of overall retrieved results than individual documents.

4.3.2 Classifying the Web

Though post-retrieval clustering of Web documents offers a viable and effective alternative to traditional ranked list retrieval, it lacks the clarity and purpose of a structured organizational hierarchy. Classification of Web documents not only produces a useful organization of information that can be browsed or searched but also offer a standardized way to describe or refer to the content of Web pages like a thesaurus, which can be leveraged to enhance the retrieval performance. For example, Srinivasan (1996a, 1996b) explored combining category labels and keyword search and found that expanding the query with MeSH terms led to significant improvement in precision and recall. The relaxation labeling method (Chakrabarti, Dom & Indyk, 1998) is also based on the similar idea of using category labels as high-quality keywords.

Hearst & Karadi (1997a, 1997b) extended the idea of category label as a retrieval enhancement tool by separating the representation of category labels from documents. Instead of

associating documents with each node of a classification hierarchy as conventional classification approaches do, the Cat-a-Cone system by Hearst & Karadi associate category labels with each document and display the portion of the classification hierarchy that corresponds to the category label of interest. As different documents are displayed, the associated classification hierarchy changes dynamically, which highlight the differences in key concepts of documents. Cat-a-Cone's classification hierarchy display also enables users to see the context in which category labels occur, where unfamiliar or ambiguous concepts are made clearer by the neighboring concepts (i.e. ancestors, siblings, descendants) in the hierarchy.

Geffner et al. (1999) suggest another method of exploiting the classification hierarchy. By representing a classification hierarchy as a data cube, where leaves become singleton ranges and nodes become the union of all ranges of its children, the data cube approach summarizes and encapsulates the organizational structure in a multidimensional database of attributes. Yahoo's display of its categories with associated sizes and subcategories, for example, can be thought of as an application of the data cube idea to present a compact summary of the categories.

So far in this section, we examined how classification hierarchy can be leveraged for the purpose of Web IR. Such studies implicitly assume the existence of a classification hierarchy. The truly challenging task of establishing an organization scheme is beyond the scope of this paper, so we did not explore the issue of classification hierarchy construction other than by the virtue of hierarchical agglomerative clustering approaches. Even if we assume an existing organization scheme, how to leverage it to organize Web documents is still a challenging matter. Some of the text categorization literature reviewed in section 4.2 used existing taxonomies such as Yahoo to test their classification algorithms. The taxonomy, however, was not the focus of their research, but simply a tool of validation. For the remainder of this section, we will review several works whose explicit aim is to leverage existing taxonomy to organize documents.

Larson's LCC project is one of the earliest studies on leveraging an existing taxonomy (Larson, 1992). Larson represented 30,471 MARC records from U.C. Berkley Library as vectors of LCC number, title, and subject headings, and clustered them by the "normalized" class number, which is the topical portion of the LCC number (i.e. the \$a subfield) to create 8435 classification clusters. To test the validity of such class hierarchy, he applied a Rocchio-based classifier to 283 new books as queries against cluster vectors and ranked clusters by cluster-document similarity scores. Based on the result, which showed 46.6% of correct clusters ranked as the top cluster and

74.4% ranked in the top 10, Larson suggested that his classification method may not be ideal for fully automatic categorization but could be used as a tool for semi-automatic classification, where correct categories might be manually selected from the top 10 categories.

Similar to Larson's LCC project is the Scorpion research project by OCLC (Thompson et al., 1997). Thompson et al.'s investigation of Dewey Decimal Classification (DDC) as a classification scheme with which to perform automatic subject assignment of documents differs from Larson's approach in that they use the concatenated concept phrases of DDC class paths directly as class representatives without any training data, instead of deriving the class representatives from the positive training examples as is typically done in supervised machine learning. The purpose of their study was to investigate DDC as a viable source for automatic classification, so their report is restricted to testing the integrity of DDC classification. They tested the class integrity of DDC by querying the DDC database, which consisted of class representatives, with every class representative and retrieving the top 20 classes. The analysis of results, which involved examination of top-ranked classes to see how closely related they were to the actual class, showed that most top-ranked matches were in the same discipline as the input query concept, thus validating DDC as a classification scheme with a high degree of class integrity.

The study by Jenkins et al. (1998) takes DDC to the next level and describes a method for using DDC to classify Web pages for WWLib⁵⁸. WWLib is organized by Automated Classification Engine (ACE), which use class representatives constructed from DDC to hierarchically classify documents. The class representatives, consisting of DDC classmark and accompanying header text as well as a manually selected set of keywords and synonyms, are compared to documents using Dice's similarity coefficient. ACE is similar to TAPER (Chakrabarti et al., 1997) in that it employs the hierarchical classification approach using customized class representatives at each node of the DDC hierarchy. The ACE approach has the advantage of using a universal classification scheme that covers all subject areas in high levels of granularity, but is burdened with the manual process of class representative construction.

Lima et al. (1998) proposes a method of classifying medical documents that take advantage of the hierarchical structure of the International Code of Disease (ICD). They construct a class hierarchy from the ICD scheme by representing each ICD code as a node with associated medical

⁵⁸ WWLib (<http://www.scit.wlv.ac.uk/wwlib/>), a virtual library at the University of Wolverhampton, is a searchable, classified catalogue of Web pages in the United Kingdom.

terms, and describe two algorithms for assigning ICD codes to documents. The vector-space classifier uses concatenated path labels as class representatives, and computes the cosine similarity⁵⁹ between document and class vectors to classify medical documents. The hierarchical classifier searches for matching terms between documents and the class hierarchy in a top-down fashion, taking into consideration co-occurrence of terms. The results of an experiment that compared the vector-based classifier with the hierarchical classifier showed the hierarchical classifier to be more accurate.

Chekuri et al. (1996) describe a method of leveraging Yahoo categories with the objective of increasing the precision of the Web search. Their main idea is to use a Yahoo-trained classifier in conjunction with the traditional keyword search in an integrated search interface. In their proposed system, where users can specify both keywords and category terms, the classifier categorizes the keyword search results and presents filtered and organized documents to the user. In keeping with Larson's idea of automatic classification as a semi-automatic classification tool, Chekuri et al. suggest presenting the top k matching categories to the user and contend that a reasonably accurate classifier is sufficient in such a setting. To validate this contention, they trained a Rocchio classifier using a sample of 2000 Web pages from 20 high-level Yahoo categories and classified 500 random documents from Yahoo. The result showed that the correct category of more than 50% of test documents were at rank 1, more than 80% at top 3, and more than 90% at top 5, from which they concluded that displaying the top 10 categories for each of the search results would be quite sufficient for an interactive search interface.

A slightly different method of leveraging Yahoo categories is described by Grobelnik and Mladenic (1998). Instead of using the actual Web documents associated with Yahoo categories, They use Yahoo's descriptions of the categorized pages to train a hierarchical Naive Bayes classifier (Koller & Sahami, 1997). Since the feature space induced by such descriptions can be sparse, they propagate the description terms upwards in the hierarchy with weights proportional to the node size where they appear. In classifying documents, they use a pruning strategy where categories with less than the required number of features in common with the target document are pruned using an inverted index of features by categories. They tested their method by training the classifier on 3 sets of training documents selected from the top 14 yahoo categories and classifying 100 to 300 randomly selected actual Web pages categorized by yahoo. The result showed that the

⁵⁹ The maximum similarity score is used when there exist multiple code paths for an ICD code.

correct category assignment probability was over 0.99 using only a small number of features⁶⁰. They also found that pruning over half the categories resulted in misclassification of only 10 to 15% of the documents.

Labrou and Finin (1999) also leverage the Yahoo terms to train a classifier called Teltale. Their investigation of combining category labels, summaries and titles of links in a Yahoo page, and contents of the actual Web pages referenced by the Yahoo links, to describe Yahoo categories revealed that best results would occur when using very brief descriptions of yahoo category entries (i.e. Yahoo summaries and titles).

⁶⁰ Using n-grams instead of single words as features, they found the category feature vector size of 3 to 4 to be effective in most cases.