Survey of Text Clustering

Liping Jing
lpjing@eti.hku.hk

Department of Mathematics, The University of Hong Kong, HongKong, China

Abstract.

1 Introduction

Clustering text documents into different category groups is an important step in indexing, retrieval, management and mining of abundant text data on the Web or in corporate information systems. Text clustering task can be intuitively described as finding, given a set vectors of some data points in a multi-dimensional space, a partition of text data into clusters such that the points within each cluster are similar to each other. Good text clustering enables better information services by browsing and organizing documents into meaningful cluster hierarchies and provides a useful complement for traditional text search engines when key-word based search returns too many documents.

Among others, the challenging problems of text clustering are big volume, high dimensionality and complex semantics. In this paper we are interested in the solutions to these problems. For the first two problems, subspace clustering can provide bright future, ontology can provide promising solution to the third one.

2 Basic VSM model [1]

There are some text source materials in different forms, such as: plain tex files, HTML Web pages, Microsoft Office files, PDF and PostScript documents, e-mail, and certain proprietary formats and for generic tasks such as recursively traversing directory structures containing such documents. The most basic mechanism used to represent the variety of text source is vector space model *(VSM)* [2, 3] of information retrieval. In the vector space model, a vector is used to represent each item or document in a collection. Each component of the vector reflects a particular concept, key word, or term associated with the given document. The value assigned to that component reflects the importance of the term in representing the semantics of the document. Typically, the value is a function of the frequency with which the term occurs in the document or in the document collection as a whole [4, 5].

A data set containing a total of d documents described by t terms is represented as a $d \times t$ *term-by-document matrix* A. The d vectors representing the d documents from the rows of the matrix. Thus, the matrix element a_{ij} is the weighted frequency at which term j occurs in document i [6]. In the parlance of the vector space model, the columns of A are the document vectors, the rows of A are the term vectors. The semantic content of the data set is wholly contained in the column space of A, meaning that the document vectors span that content. Not every vector represented in the column space of A has a specific interpretation in terms of the document collection itself. The important thing is to exploit geometric relationships between document vectors to model similarities and differences in content and then compare term vectors geometrically in order to identify similarities and differences in term usage.

In order to satisfy text clustering method, the original unstructured documents collection need to be preprocessed. As the approach mentioned above, the standard representation of the text is term vectors [7]. let $tf(d, t)$ be the absolute frequency of term $t \in T$ in document $d \in D$, where D is the set of documents and $T = \{t_1, \ldots, t_m\}$ is the set all different terms occurring in D and then we denote $t_d = (tf(d, t_1), \ldots, tf(d, t_m))$. As a slightly more advanced approach, several task: tokenizering [8, 9], removing stopwords, stemming of terms, pruning of terms that appear infrequently, weighting term [10] and even concatenating ontology concepts are combined into the creation of document vectors. The centroid of a set X of term vectors can be defined as the mean value $t_X := \frac{1}{|X|} \sum_{t_d \in X} t_d$.

Tokenizer performs on the raw document collection in order to compress the collection based on individual word. It removes the punctuation and some un-detected character and extracts words which exist in the basic dictionary repository.

Stopwords are words which are considered as non-descriptive within a bag-of-words approach. They typically comprise prepositions, articles, etc. Following common practice, a standard list with stopwords[1] is always used in many researches.

Using the Porter stemmer introduced in [11], text documents are computed the frequency of stemmed terms instead of the original terms in the collections. and used them to construct a vector representation t_d for

[1] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

each text document. The length of the resulting vectors is given by the number of different stemmed terms in the text corpus.

For some empirical investigations, all terms appearing rarely or more frequently should be entirely discarded. There are two criteria to evaluate the term occurrence. First one is term frequency: a term t is discarded from the representation (i.e. from the set T), if $\sum_{d \in D} tf(d,t) \leq \delta_1$ or $\sum_{d \in D} tf(d,t) \geq \delta_2$. Second one is document frequency: a term t is discarded from the representation (i.e. from the set T), if $\sum_{d \in D} \psi(tf(d,t)) \leq \beta_1$ or $\sum_{d \in D} \psi(tf(d,t)) \geq \beta_2$. Where parameters δ_1, δ_2 and β_1, β_2 are predefined thresholds, and they should be adjusted in the experiment.

Salton, in [12], reported weighting method: term frequency-inverted document frequency *tfidf*. *Tfidf* weights the frequency of a term in a document with a factor that discounts its importance when it appears in almost all documents. Therefore terms that appear too rarely or too frequently are ranked lower than terms that balance between the two extremes and, hence, are expected to be better able to contribute to clustering results. The *tfidf* of term t in document d is defined by:

$$tfidf(d,t) := tf(d,t) \cdot log\left(\frac{|D|}{df(t)}\right) \tag{1}$$

where $df(t)$ is the document frequency of term t that counts in how many documents term t appears. If *tfidf* weighting is applied then the term vectors will be replaced by $t_d := (tfidf(d,t_1), \ldots, tfidf(d,t_m))$.

Finally, for the integration of concepts, the methods are described in detail in the next section. It is easy to obtain various representations of text documents by performing the above steps in all different combinations. Recently, Lavelli et al. [13] compared two distributional term representation methods: document occurrence representation and term co-occurrence representation for text clustering and categorization. They showed there was a substantial difference in effectiveness between the two representation styles, also they gave an intuitive explanation and an information-theoretic justification for these different behaviours.

3 Ontology [14]

Usually, as mentioned in the previous section, the text collection is represented as a set of vectors $\mathbf{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$. Each vector $\mathcal{X}_j$ is characterized by a set of m terms or words, $(t_1, t_2, \ldots, t_m)$ [7]. However, the bag of words representation used for text clustering methods is often unsatisfactory as it ignores relationships between important terms that do not co-occur literally.

As we all known, meaningful sentences are composed of meaningful words, any system that hopes to process natural languages as people do must have information about words and their meanings. This information is traditionally provided through dictionaries, especially, machine-readable dictionaries, such as *WordNet* [15]. There are many methods to construct and expand the machine-readable dictionary, such as: *Kaon* [16], and others mentioned in [17–19].

The traditional text representation method is based on the words which occur in the relative documents, and then the clustering methods compute the similarity between the vectors. However, many documents contain the similar semantic information, even though they do not contain common words. For instance, if one document describes the "beef" issue, it should be turned up "meat" issue even though the document does not contain word "meat". In order to deal with such problem, a concept-based model [20, 21] using ontologies is necessary.

There have been many attempts to define what constitutes an ontology, perhaps the best know being due to Gruber [22]: an ontology is a conceptual framework for defining the basic classes of entities in some domain of knowledge, the relationships these entities have to each other, and the organization of concepts in terms of higher-lever concepts, typically taxonomic in nature. The term ontology is often used to refer to a range of linguistic and conceptual resources, from thesauri and dictionaries containing records defining key terms, synonyms, and so forth in some domain, to taxonomies that classify names and phrases in higher-level categories, and formal knowledge representations that might support automatic inferences and certain types of reasoning.

According to the definition, ontology is a collection of concepts and their interrelationships which can collectively provide an abstract view of an application domain [23]. So ontologies have to be built and maintained before using them. And this is a difficult, labor-intensive, and expertise-intensive task. Recently, some approaches to constructing ontology [24, 25] and integrating ontology into information [26] are presented. While, the most frequently used collection of ontologies in research area is *WordNet*[2] supported by the cognitive science laboratory of Princeton University. The invitation of ontology could handle with the semantic issue in e-learning [27] and in text mining, including document classifying [28] and clustering [29]. The principle ideal of ontology integration is that the

[2] http://www.cogsci.princeton.edu/ wn/obtain.shtml.

variance of documents within one category is reduced by representation with ontology, thus improving results of text mining.

Ontology was firstly applied to text clustering in [30]. A core ontology based on *WordNet* provides background knowledge to enrich the original term vector representation. Then they used *Bisection-KMeans* clustering the modified vectors into smaller different groups. And experiments demonstrated that the best strategies that involve such ontology are never worse than the baseline, but often times better, especially, when including word sense disambiguation and feature weighting.

As already mentioned above, high quality ontologies are crucial for many applications, and their construction, integration, and evolution greatly depends on the availability of a well-defined semantics and powerful reasoning tools. Till now, description logics (DLs) [31, 32] could provide for both, so they should be ideal candidates for ontology representation and the popular expressive is DL *SHIQ* [33]. A core ontology used in text clustering [34] is represented according to the DL *SHIQ*. The following will give some knowledge about definition of the core ontology and lexicon.

Definition: A *core ontology* is a tuple $\mathcal{O} := (C, \leq_C)$ consisting of a set C whose elements are called *concept identifiers*, and a partial order $\leq_C$ on C, called *concept hierarchy* or *taxonomy*.

Definition: If $c_1 <_C c_2$, for $c_1, c_2 \in C$, then c_1 is a *subconcept* of c_2, and c_2 is a *superconcept* of c_1. If $c_1 <_C c_2$ and there is no $c_3 \in C$ with $c_1 <_C c_3 <_C c_2$, then c_1 is a *direct subconcept* of c_2, and c_2 is a *direct superconcept* of c_1, noted by $c_1 \prec c_2$.

Definition: A *lexicon* for an ontology $\mathcal{O}$ is a tuple $Lex := (S_C, Ref_C)$ consisting of a set S_C whose elements are called *signs for concepts*, and a relation $Ref_C \subseteq S_C \times C$ called *lexical reference for concepts*, where $(c, c) \in Ref_C$ holds for all $c \in C \cap S_C$.

Based on Ref_C, for $s \in S_C$, $Ref_C(s) := \{c \in C | (s, c) \in Ref_C\}$ and, for $c \in C$, $Ref_C^{-1}(c) := \{s \in S_C | (s, c) \in Ref_C\}$.

An *ontology with lexicon* is a pair $(\mathcal{O}, Lex)$ where $\mathcal{O}$ is an ontology and Lex is a lexicon for $\mathcal{O}$.

Using the background knowledge: *WordNet*, ontology can be constructed and applied into text vector representation. There are different strategies for compiling ontology into the text representations, focusing on concepts, disambiguation, and hypernyms. Three strategies are provided for concepts: adding related concepts to the term vector, replacing terms by concepts or only concept vector without terms; three for disambiguation: including all concepts, first concept (according to the rank reflecting how common the concept is in "standard" English language), or disambiguation by context; more strategies for hypernyms referring to the parameter *Hypdepth*, which show the concepts corresponding to the given concept.

Next we focus on the strategies of concepts application.

Add Concepts. When applying this strategy, each term vector $\mathbf{t_d}$ will be extended by new entries for concepts c appearing in the document set. Thus the vector $\mathbf{t_d}$ is replaced by the concatenation of $\mathbf{t_d}$ and $\mathbf{c_d}$, where $\mathbf{c_d} := (cf(d, c_1), \ldots, cf(d, c_l))$is the concept vector with $l = |C|$ and $cf(d, c)$ denotes the frequency that a concept $c \in C$ appears in a document d as indicated by applying the reference function Ref_C described in the definition to all terms in the document d.

Replace Terms by Concepts. This strategy works like 'add concepts' but it expels all terms from the vector $\mathbf{t_d}$ for which at least one corresponding concept exists.

Concept Vector Only. This strategy works like 'replace terms by concepts' but it expels all term frequencies from the vector. Only $\mathbf{c_d}$ is used to represent the document d.

For the selection of concepts, we can refer to the strategies of disambiguation and hypernyms [29, 34]. Finally a new vector will be constructed for each original term vector. The re-adjusted vector set with concepts from the core ontology described above has two benefits. First it resolves synonyms; and second it introduces more general concepts which help identifying related topics. From the experimental results[3] , it is evident that clustering the modified representations based on ontology obtains better performance than clustering the baseline term vector representation.

4 Subspace Clustering[35]

Text data are high dimensional with a big problem: *curse of dimensionality*. It has been shown that in a high dimensional space the distance between every pair of points is almost the same for a wide variety of data distributions and distance functions [36]. This motivates the attempt of reducing dimension of the input data. Many feature selection techniques have been applied to search through various feature subsets and evaluating each of these subsets using some criterion [37–41]. However, the correlations among the dimensions are often specific to data locality, in the sense that some data points are correlated with a given set of features and others are correlated with respect to different features. Namely, in high dimensional space, like text data, each cluster usually has its own subspace structure.

[3] http://www.aifb.uni-karlsruhe.de/WBS/aho/clustering.

We can use a simple dataset to illustrate the need for subspace clustering. The data set X has 3 features t_1, t_2, t_3. Two normally distributed clusters are found in (t_1, t_2) (see Figure 1(a)). t_3 is a random variable with a uniform distribution. No cluster structure can be found in (t_1, t_3) and (t_2, t_3) (see Figure 1(b) and 1(c)). When we applied the *standard k-means* algorithm to X, the two clusters in (t_1, t_2) may not be discovered because of the noise variable t_3. In higher dimensional datasets, especially for text data, always ten or hundred thousand dimensions, this problem becomes even worse and the clusters become impossible to find, suggesting that we should consider fewer dimensions for each cluster. For example, if we assign weights 0.47, 0.40, 0.13 to features t_1, t_2, t_3 respectively, then two clear clusters will be found as plotted in Figure 1(a).

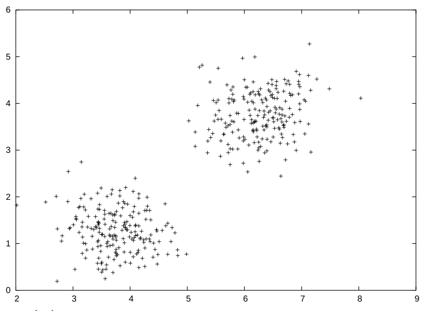
4.1 General subspace clustering

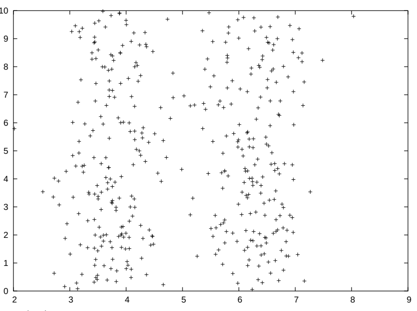
Traditional clustering techniques focus on one-sided clustering and they can be classified into partitional, hierarchical, density-based, and grid-based [42, 43]. Partitional clustering attempts to directly decompose the data set into k disjoint classes such that the data points in a class are nearer to one another than the data points in other classes. Hierarchical clustering proceeds successively by building a tree of clusters. Density-based clustering is to group the neighboring points of a data set into classes based on density conditions. Grid-based clustering quantizes the object space into a finite number of cells that form a grid-structure and then performs clustering on the grid structure. However most of these algorithms are not effective in high dimensional spaces.

Recent research work has focused on subspace clustering, discovering clusters embedded in the subspaces of a high dimensional data set. As demonstrated in [44], subspace clustering algorithms are divided into two categories based on how they determine a measure of locality with which to evaluate subspaces.

Bottom-up subspace search methods

This approach takes advantage of the downward closure property of density to reduce the search space, using an APRIORI style approach. It determines locality by creating bins for each dimension and using those bins to form a multi-dimensional grid. There are two group consists of *CLIQUE* [45] and *ENCLUS* [46] which both use a static sized grid to divide each dimension into bins. The second group contains *MAFIA* [47], *CBF* [48], *CLTree* [49] and *DOC* [50]. In the second group, we can distinguish the methods by using data driven strategies to determine the


(a) Plot in the subspace of t_1, t_2.


(b) Plot in the subspace of t_1, t_3.

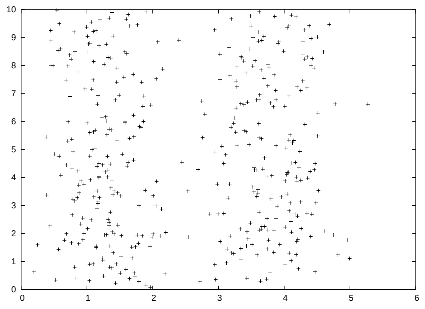

(c) Plot in the subspace of t_2, t_3.

Fig. 1: Sample data plotted in different dimension collection

cut-points for the bins on each dimension. The first two methods use histograms to analyze the density of data in each dimension. The third one uses a decision tree based strategy to find the best cut-point. A maximum width and minimum number of instances per cluster are used to guide a random search in *DOC*.

Iterative top-down subspace search methods

The top-down subspace clustering approach starts by finding an initial approximation of the clusters in the full feature space with equally weighted dimensions. Next each dimension is assigned a weight for each cluster. The updated weights are then used in the next iteration to regenerate the clusters. This approach requires two critical parameters: the number of clusters, the size of the subspaces. Locality in top-down methods is determined by some approximation of clustering based on the weights for the dimensions obtained so far. *PROCLUS* [51], *ORCLUS* [52], *FINDIT* [53], and *δ-Clusters* [54] determine the weights of instances for each cluster. *COSA* [55] is unique in that it uses the k nearest neighbors for each instance in the dataset to determine the weights for each dimension for that particular instance.

Additionally, [56] and [57] presented subspace clustering algorithm by performing clustering and feature weighting simultaneously. They identify clusters from subspaces by automatically assigning large weights to the variables that form the subspaces in which the clusters are formed. Whilst the method is based on the extensions to the *standard k-means* algorithm so it is efficient and scalable to large data set. Recently, some researchers [58] bring ontology into subspace clustering on gene expression data and demonstrated that combining ontology with subspace clustering significantly improve the clustering performance with minimal degradation of the cluster quality.

4.2 Text Subspace Clustering

The more detail content about the general subspace clustering is provided in [44]. In this survey, we focuses on the latest subspace clustering on text document clustering.

Adaptive subspace iteration (ASI)

Li et al. [59] introduced a new clustering algorithm: ASI that allows explicit modelling of the subspace structure associated with each cluster. ASI simultaneously performs two tasks: data reduction (assigning data points into clusters) and subspace identification (identifying subspace structure associated with each cluster). The tasks are carried out by an iterative optimization procedure that alternates between identification of the subspace structure from current cluster partitions and updating of the clusters based on the identified new subspace structures. The simultaneous clustering of both points and their attributes by way of utilizing the canonical duality contained in the point-by-attribute data representation is similar to the optimization procedure in co-clustering mentioned in [60, 61]. This procedure is also applied to the problem of clustering gene expression data [62].

In experiments, Li compared ASI with the traditional K-means algorithm and the algorithms provided in the CLUTO[4] package [63, 64]. From the experimental results, Li observed that ASI achieves the best performance on most datasets except WebKB4. Although ASI is not so good as P1 on WebKB4, ASI is a viable and competitive algorithm in document clustering domain, because except for the higher clustering accuracy, ASI provides meaningful word sets corresponding to each document cluster to represent their contents.

Simultaneous keyword identification and clustering of text documents (SKWIC)

Frigui [65] proposed a new approach: SKWIC, to unsupervised clustering text documents based on a coupled process of clustering and cluster-dependent keyword weighting. Considering that the distribution of words in most real document collections vary drastically from one group of documents to another, Frigui presented automatic method to identify clusters that differ widely in their best keyword sets by assigning different weights to the keywords [66, 67] for each cluster. Namely, one document cluster is located in a special keyword set rather than the whole keyword space.

SKWIC has three advantages compared to existing document clustering techniques. First, its continuous term weighting provides a much richer feature relevance representation than binary feature selection: not all terms are considered equally relevant in a single cluster of text documents. Second, a given term is not considered equally relevant in all cluster. We can find this point in our simple example in this section. Finally, SKWIC learns a different set of term weights for each cluster in an unsupervised manner which can be used to represent meaning of the final text cluster.

Also, Frigui conducted experiments on Newsgroups [68] to demonstrate that SKWIC obtains the collection of terms receiving highest feature relevance weights in each cluster reflected the general topic of the

[4] http://www-users.cs.umn.edu/ karypis/cluto/

category winning the majority of the documents that were assigned to the cluster. Additionally, SKWIC was extended into fuzzy clustering method in [65]. As expected, fuzzy SKWIC performs better than the unweighted counterpart: fuzzy C-means [69], where cosine-based distance are used in the experiments.

5 Semi-Supervised Clustering

Learning with both labeled and unlabeled data, also called semi-supervised learning or transductive learning, has recently been studied by many research with great interest, mainly as a way of exploiting information in unlabeled data to enhance the performance of a classification model (tranditionally trained using only labeled data). A bit improvement of classification result has been obtained, however, Zhang [70] and Krogel [71] theoretically and experimentally questioned the usefulness and reliability of transductive SVMs for semi-supervised learning. Their analysis is limited due to the assumptions that all classes are available in the labeled data and the distributions for both labeled and unlabeled data are the same.

Since semi-supervised classification are not well positioned to detect new classes, researchers are motivated to look at semi-supervised clustering, which exploits labeled data to enhance clustering results on unlabeled data. Semi-supervised clustering employs user-provided knowledge such as: data labels or data relationship, to bias the grouping so that it corresponds closely to user's view of similar and dissimilar objects. Several techniques exist for incorporating supervision into clustering, such as seeding (initializing clusters with labeled objects) [72], distance metric learning (training a similarity measure using the labeled objects) [73], pairwise constrained clustering [74] and feedback approaches [75, 76]. Additionally, special topic knowledge can also direct the unsupervised clustering process [77].

5.1 Seeded Approaches

Seeded approaches use labeled data only to help initialize cluster centroids. Proper seeding biases clustering towards a good region of the search space, thereby reducing the chances of it getting stuck in poor local optima, while simultaneously producing a clustering similar to the user-specified labels. Basu et al. used seed clustering to initialize the KMeans algorithm rather than initializing KMeans from k random means, the mean of the lth cluster is initialized with the mean of the lth partition S_l of the seed set. The seed clustering is only used for initialization, and the seeds are not used in the following steps of the algorithm. So the used-specified labeling of the seed data may be changed in the course of the algorithm. They experimentally demonstrated that KMeans clustering benefits from the standard seeding methods.

5.2 Constrained Approaches

Constrained approaches keep the grouping of labeled data unchanged (as fixed constraints) throughout the clustering process. The cluster memberships of the data points in the seed set are not re-computed in the clustering steps, that is, the cluster labels of the seed data are kept unchanged, and only the labels of the non-seed data are re-estimated. However, these approaches will use pairwise and distance metric constraint to modify the clustering process including the seed data label.

Generally, pairwise constrained are more practical than provided class labels in the clustering process. There are two constraints: must-link and cannot-link between points in a data set, in addition to having distances between the points. That is, these approaches explicitly modify the objective function or make certain constraints during the clustering process [78]. They parametrize distance metric and learn the metric parameters in a manner, so that the distance between objects connected by must-links is smaller and the distance between objects connected by cannot-links is large in general [79, 80].

Basu et al. designed a model to store the pairwise constraints and use them to formulate the objective function in Kmeans algorithm. In this model, $\mathcal{M}$ is the set of unordered must-link pairs such that $(x_i, x_j) \in \mathcal{M}$ implies x_i and x_j should be assigned to the same cluster; and $\mathcal{C}$ is the set of unordered cannot-link pairs such that $(x_i, x_j) \in \mathcal{C}$ implies x_i and x_j should be assigned to different clusters. Next, they defined the weights corresponding to the two pairwise constraints $\mathcal{W} = \{w_{ij}\}$ and $\bar{\mathcal{W}} = \{\bar{w}_{ij}\}$. Finally, the objective function is founded as:

$$\mathcal{J} = \sum_{x_i \in \mathcal{X}} \|x_i - \mu_{l_i}\|^2 + \sum_{(x_i, x_j) \in \mathcal{M}} w_{ij} I[l_i \neq l_j] + \sum_{(x_i, x_j) \in \mathcal{C}} \bar{w}_{ij} I[l_i = l_j] \quad (2)$$

where I is the indicator function ($I[ture] = 1$, I[false]=0) and l_i are the cluster labels.

In [72], seeded approaches and constrained approaches are compared based on the spherical k-means [81]. They observed that the constrained version fares at least as well as the seeded one. Next year, [82] combined these two methods and presented a unified approach based on the standard k-means and experimental results demonstrate that the combined approach generally performs better than either of the individual approaches.

5.3 Feedback-based Approaches

Feedback-based approaches first run a regular clustering process and then adjust resulting clusters based on labeled data. This approach assumes that the human user has in their mind criteria that enable them to evaluate the quality of a clustering. The user interacts with the clustering system which attempts to learn a criterion that yields clusters the user is satisfied with. There are many different types of feedback that users might provides to a semi-supervised clustering system, for example, the constraints on individual data points. From this perspective, the constrained approaches should belong to feedback-based semi-supervised learning. Additionally, the user can tell the system that the current clustering is too coarse or too fine, which will in return direct the clustering process: if the user indicate that a cluster is good, the future re-clusterings of the data should attempt to maintain the structure of this good cluster.

Under a model-based clustering framework [83], Zhong [84] presented deterministic annealing extensions of the three semi-supervised clustering methods – seeded clustering, constrained clustering and feedback clustering and compares their performance on real text datasets. Experimental results show that feedback-based clustering is superior when the label data contain only a partial set of text categories, which exactly balances the drawback of semi-supervised classification.

5.4 Topic-based Approaches

Except for useing labeled data semi-supervise clustering process, special knowledge, such as: topics collection, can also be exploited in text clustering. Zhao and Karypis [77] firstly defined and implemented the topic-driven clustering approach. This approach is motivated to cluster documents into the given topic and simultaneously guarantee that documents within one topic are more similar to each other than the documents from two different topics.

Because of the insufficient information about each topic, traditional classification methods can not solve this problem. Because the prior knowledge: topic is not in the format of labeled data, the semi-supervised clustering methods mentioned above fail to address it too. Considering of these two facets, Zhao formulated the problem by combining unsupervised optimization process and supervised optimization process. They presented three efficient combined scheme: weighted scheme, normalized scheme, and hybrid scheme that consider the similarity between the document to its topic and the relationship between the documents themselves. Of course, experiments on five document datasets demonstrated that this approach outperforms the traditional supervised classification, unsupervised clustering and seed-based semi-supervised clustering.

6 Existing System [85]

On the application side, many existing works focused on document classification or clustering, which has become an increasingly important technique for supervised/unsupervised document organization, automatic topic extraction, and fast information retrieval or filtering. In this information world, a web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories like Vivisimo[5].

The result of text clustering is the identification of set of not only keywords but also concept derived from the text by the applied analysis engines for a given group of text. These keyword and concept sets can be considered to be the main concepts connecting the corresponding groups. Ultimately, the clusters would represent a domain and their corresponding subspaces would indicate the key concepts of the domain. This information could be further utilized by training a classifier with domains and representative concepts. This classifier could then be used to classify or categorize previously unseen documents.

In fact, the ideal mentioned above is reasonable and promising. IBM develops *unstructured information management architecture* (UIMA) along with this principle.

6.1 UIMA - Unstructured Information Management Architecture

IBM's UIMA[6] [86] is an architecture and software framework that helps you build that bridge. It supports creating, discovering, composing and

[5] http://www.vivisimo.com
[6] http://www.alphaworks.ibm.com/tech/uima

deploying a broad range of analysis capabilities and linking them to structured information services.

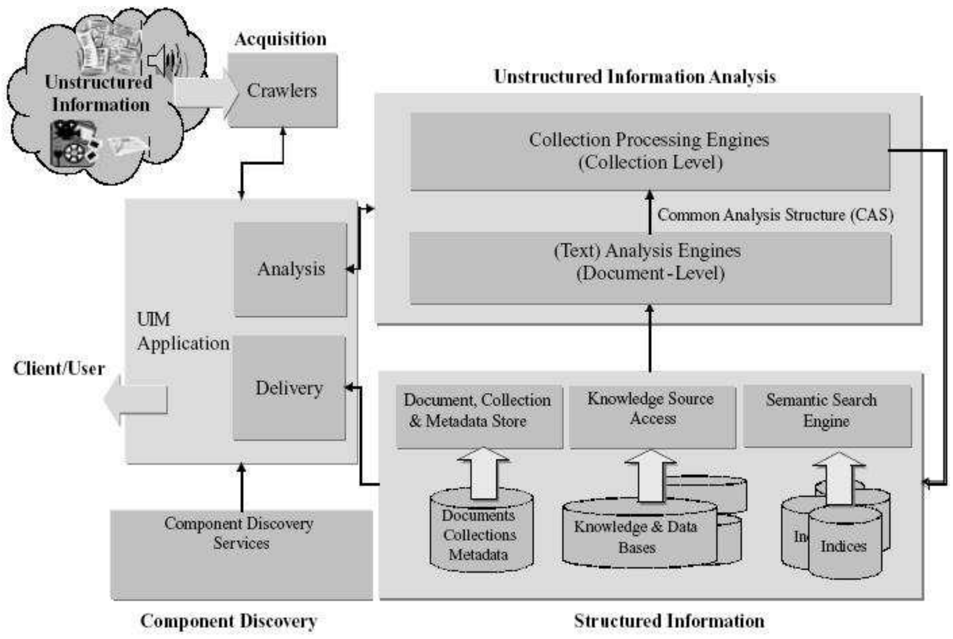

Unstructured Information
Acquisition
Crawlers
Unstructured Information Analysis
Collection Processing Engines (Collection Level)
Common Analysis Structure (CAS)
(Text) Analysis Engines (Document-Level)
Analysis
UIM Application
Delivery
Client/User
Document, Collection & Metadata Store
Knowledge Source Access
Semantic Search Engine
Documents Collections Metadata
Knowledge & Data Bases
Indices
Component Discovery Services
Component Discovery
Structured Information

Fig. 2: UIMA high-level architecture

Figure 2 illustrates the roles, interfaces and communications of large-grained components essential for UIMA. These include components capable of analyzing unstructured artifacts, integrating and accessing structured sources and storing, indexing and searching for artifacts based on discovered semantic content resulting from analysis. Till now, UIMA has been applied in life science, medical documents analysis and other areas, because it provides many easy-connecting API to extend the functions.

6.2 KAON - The KArlsruhe ONtology and Semantic Web tool suite

Except for IBM's UIMA, AIFB institute in Univ. of Karlsruhe also developed KAON[7] [87]: an ontology and semantic web tool suite.

KAON is an open-source ontology management infrastructure targeted for business applications. It includes a comprehensive tool suite al-

[7] http://kaon.semanticweb.org/

lowing easy ontology creation and management and provides a framework for building ontology-based applications. An important focus of KAON is scalable and efficient reasoning with ontologies.

6.3 GATE - A General Architecture for Text Engineering

GATE[8] [88] is designed by the Dept. of Computer Science in Univ. of Sheffield.

GATE is a framework and graphical development environment which enables users to develop and deploy language engineering components and resources in a robust fashion. The GATE architecture has enabled us not only to develop a number of successful applications for various language processing tasks (such as Information Extraction), but also to build and annotate corpora and carry out evaluations on the applications generated. The framework can be used to develop applications and resources in multiple languages, based on its thorough Unicode support.

6.4 BOW

BOW [89] is a library of *C* code useful for writing statistical text analysis, language modelling and information retrieval programs. The current distribution includes the library, as well as front-ends for document classification (rainbow), document retrieval (arrow) and document clustering (crossbow). In Bow, many facilities are provided such as: performing N-grams among the tokens, mapping strings to integers and back again efficiently, building a sparse matrix of document/token counts, pruning vocabulary by word counts or by information gain, building and manipulating word vectors, setting word vector weights according to *Naive Bayes*, *TFIDF*, and several other methods, scoring queries for retrieval or classification, writing all data structures to disk in a compact format, reading the document/token matrix from disk in an efficient, sparse fashion, performing test/train splits, and automatic classification tests and etc.

7 Conclusion

[8] http://gate.ac.uk

References

1. “**VSM Model**.”
2. C. Buckley, G. Salton, J. Allan, and A. Singhanl, “Automatic query expansion using smart: Trec 3,” *Overview of the 3rd text retrieval conference*, pp. 69–80, 1995.
3. G. Salton and M. McGill, “Introduction to modern information retrieval,” *McGraw-Hill*, 1983.
4. S. Dumais, “Improving the retrieval of information from external sources,” *Behavior research methods, instruments, and computers*, vol. 23, pp. 229–236, 1991.
5. K. Jones, “A statistical interpretation of term specificity and its applications in retrieval,” *J. Documentation*, vol. 28, pp. 11–21, 1972.
6. M. Berry and G. Dumais, S.and Obrien, “Using linear algebra for intelligent information retrieval,” *SIAM Rev.*, vol. 37, pp. 573–595, 1995.
7. G. Salton, A. Wong, and C. Yang, “A vector space model for automatic indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.
8. W. Teahan and J. Cleary, “Models of english text.”
9. A. Moffat, “Word-based text compression,” *Software-Practice and Experience*, pp. 185–198, 1989.
10. G. Amati, C. Carpineto, and G. Romano, “Fub at trec-10 web track: A probabilistic framework for topic relevance term weighting,” *The Tenth Text Retrieval Conference*, p. online publication, 2001.
11. M. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.
12. G. Salton, “Automatic text processing: the transformation, analysis and retrieval of information by computer,” *Addison-Wesley*, 1989.
13. A. Lavelli, F. Sebastiani, and R. Zanoli, “Distributional term representations: An experimental comparison,” *CIKM’04*, pp. 615–624, 2004.
14. “**Ontology Lecture**.”
15. G. Miller, “Wornet: A lexical database for english,” *CACM*, vol. 38, no. 11, pp. 39–41, 1995.
16. E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, and V. Zacharias, “Kaon - towards a large scale semantic web,” *In Proceedings of Ec-web,*, pp. 304–313, 2002.
17. H. Ayad and K. M., “Topic discovery from text using aggregation of different clustering methods,” *15th Conference of the Canadian Society for Computational Studies of INtelligence*, pp. 161–175, 2002.
18. H. Avancini, A. Lavelli, B. Magnini, F. Sebastiani, and R. Zanoli, “Expanding domain-specific lexicons by term categorization,” *In Proceedings of SAC-03, 18th ACM Symposium on Applied Computing*, pp. 793–797, 2003.
19. H. Avancini, F. Lavelli, A. Sebastiani, and R. Zanoli, “Automatic expansion of domain-specific lexicons by term categorization,” *Submitted*, 2004.
20. L. Khan, “Ontology-based information selection,” *PhD Thesis*, 2000.
21. L. Khan and D. McLeod, “Audio structuring and personalized retrieval using ontology,” *Proc. of IEEE Advances in Digital Libraries*, 2000.
22. T. Gruber, “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.
23. ——, “Toward principles for the design of ontologies used for knowledge sharing,” *Proc. of International Workshop on Formal Ontology*, 1993.

24. L. Khan and F. Luo, "Ontology construction for information selection," *Proc. of 14th IEEE International Conference on Tools with Artificial Intelligence*, pp. 122–127, 2002.
25. L. Khan, F. Luo, and I. Yen, "Automatic ontology derivation from documents," *Proc. of the 15th Conference On Advanced Information Systems Engineering (CAISE '03)*, 2003.
26. H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner, "Ontology-based integration of information - a survey of existing approaches," *In Stuckenschmidt, H., ed., IJCAI-01 Workshop*, pp. 108–117, 2001.
27. J. Tane, C. Schmitz, G. Stumme, S. Staab, and R. Studer, "The courseware watchdog: an ontology-based tool for finding and organizing learning material," *Fachtagung Mobiles Lernen und Forschen*, vol. 6, no. 11, 2003.
28. M. Rodriguez, J. Hidalgo, and B. Diaz-Agudo, "Using wordnet to complement training information in text categorization," *Recent Advances in Natural Language Processing II*, vol. 189, 2000.
29. A. Hotho, S. Staab, and G. Stumme, "Wordnet improves text document clustering," *In Proc. of the Semantic Web Workshop at SIGIR-2003, 26th Annual International ACM SIGIR Conference*, 2003.
30. A. Hotho, S. Staab, and A. Madche, "Ontology-based text clustering," *Workshop Text Learning: Beyond Supervision IJCAI 2001*, 2001.
31. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Schneider, "The description logic handbook: theory, implementation and applications," *Cambridge University Press*, 2002.
32. F. Baader and U. Sattler, "An overview of tableau algorithms for description logics," *Studia Logica*, vol. 69, no. 1, pp. 5–40, 2001.
33. I. Horrocks, U. Sattler, and S. Tobies, "Practical reasoning for very expressive description logics," *the Interest Group in Pure and Applied Logic*, vol. 8, no. 3, pp. 239–264, 2000.
34. A. Hotho, S. Staab, and G. Stumme, "Text clustering based on backgroud knowledge," *Technical Report 425, Univ. of Karlsruhe, Institute AIFB*, 2003.
35. "**Subspace Clustering**."
36. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbors meaningful?" *Proc. of 7th International Conference on Database Theory (ICDT'99)*, pp. 217–235, 1999.
37. M. Dash and H. Liu, "Feature selection for clustering," *PAKDD-00*, pp. 110–121, 2000.
38. J. Pena, J. Lozano, P. Larranaga, and T. Murali, "Dimensionality reduction in unsupervised learning of conditional gaussian networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 590–603, 2001.
39. P. Mitra, C. Murthy, and S. Pal, "Unsupervised feature selection using feature similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 301–312, 2002.
40. L. Yu and H. Liu, "Feature selection for high-dimensional data: a fast correlation-based filter solution," *Proc. of the 20th International Conference on Machine Learning*, pp. 856–863, 2003.
41. J. Dy and C. Brodley, "Feature selection for unsupervised learning," *The Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.
42. K. Jain and C. Dubes, "Algorithms for clustering data," *Prentice Hall*, 1988.
43. J. Han and M. Kamber, "Data mining: Concepts and techniques," *Morgan Kaufmann Publishers*, 2000.

44. L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *SIGKDD Explorations*, vol. 6, no. 1, pp. 90–105, 2004.
45. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *ACM SIGMOD-98*, pp. 94–105, 1998.
46. C. Chen, A. Fu, and Y. Zhang, "Entropy-based subspace clustering for mining numerical data," *ACM SIGKDD-99*, pp. 84–93, 1999.
47. S. Goil and H. Nagesh, "Mafia: Efficient and scalable subspace clustering for very large data sets," *Technical Report CPDC-TR-9906-010, Northwestern Univ.*, 1999.
48. J. Chang and D. Jin, "A new cell-based clustering method for large, high-dimensional data in data mining applications," *Proc. of the 2002 ACM symposium on Applied computing*, pp. 503–507, 2002.
49. B. Liu, Y. Xia, and P. Yu, "Clustering through decision tree construction," *Proc. of the 9th international conference on Information and knowledge management*, pp. 20–29, 2000.
50. C. Procopiuc, M. Jones, P. Agarwal, and T. Murali, "A monte carlo algorithm for fast projective clustering," *ACM SIGMOD-02*, pp. 418–427, 2002.
51. C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park, "Fast algorithms for projected clustering," *ACM SIGMOD-99*, pp. 61–72, 199.
52. C. Aggarwal and P. Yu, "Finding generalized projected clusters in high dimensional spaces," *ACM SIGMOD-00*, pp. 70–81, 2000.
53. I. Witten and J. Lee, "Findit: a fast and intelligent subspace clustering algorithm using dimension voting," *PhD thesis, Korea Advanced Institute of Science and Technology*, 2002.
54. J. Yang, W. Wang, H. Wang, and P. Yu, "δ-clusters: capturing subspace correlation in a large data set," *In Data Engineering-02*, pp. 517–528, 2002.
55. J. Friedman and J. Meulman, "Clustering objects on subsets of attributes," *Journal of the Royal Statistical Society: Series B*, 2004.
56. Y. Chan, K. Ching, K. Ng, and Z. Huang, "An optimization algorithm for clustering using weighted dissimilarity measures," *Pattern recognition*, vol. 37, no. 5, pp. 943–952, 2004.
57. H. Frigui and O. Nasraoui, "Unsupervised lerning of prototypes and attribute weights," *Pattern recognition*, vol. 37, no. 3, pp. 567–581, 2004.
58. J. Liu, W. Wang, and J. Yang, "A framework for ontology-driven subspace clustering," *In Proc. of the 10th ACM SIGKDD*, pp. 623–628, 2004.
59. T. Li, S. Ma, and M. Ogihara, "Document clustering via adaptive subspace iteration," *SIGIR-04*, pp. 218–225, 2004.
60. I. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," *Technical Report 2001-05, Dept. of Computer Science, Univ. of Texas at Austin*, 2001.
61. ——, "Information theoretic co-clustering," *ACM SIGKDD Conference*, pp. 89–98, 2003.
62. Y. Cheng and M. Church, "Biclustering of expression data," *In proc. of the 8th International conference on Intelligent systems for molecular biology*, pp. 93–103, 2000.
63. Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," *Technical Report. Dept. of Computer Science, Univ. of Minnesota*, 2001.
64. ——, "Evaluation of hierarchical clustering algorithms for document datasets," *Technical Report. Dept. of Computer Science, Univ. of Minnesota*, 2003.

65. H. Frigui and O. Nasraoui, "Simultaneous clustering and dynamic keyword weighting for text documents," *Survey of text mining, Michael Berry, Ed, Springer*, pp. 45–70, 2004.
66. H. Almuallim and G. Dietterich, "Learning with many irrelevant features," *in 9th National Conference on Artificial Intelligence*, pp. 547–552, 1991.
67. D. Wettscherech, W. Aha, and T. Mohri, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence*, vol. 11, pp. 273–314, 1997.
68. CMU, "20 newsgroup data set." [Online]. Available: www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html
69. J. Bezdek, "Pattern recognition with fuzzy objective function algorithms," *Plenum Press, New York*, 1981.
70. T. Zhang and F. Oles, "A probabilistic analysis on the value of unlabeled data for classification problems," *In proc. 17th Int. Conf. Machine Learning*, pp. 1191–1198, 2000.
71. M. Krogel and T. Scheffer, "Multi-relational learning, text mining, and semi-supervised learning for functional genomics," *Machine Learning*, vol. 57, pp. 61–81, 2004.
72. S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," *In Proc. of 19th International Conference on Machine Learning (ICML-2002)*, pp. 19–26, 2002.
73. M. Bilenko and R. Mooney, "Learning to combine trained distance metrics for duplicate detection in databases," *Technical Report AI 02-296, AI Lab., Univ. of Texas at Austin*, 2002.
74. S. Basu, A. Banerjee, and R. Mooney, "Active semi-supervised for pairwise constrained clustering," *In proc. of 2004 SIAM International Conference on Data Mining (SDM-04)*, 2004.
75. H. Kim and S. Lee, "A semi-supervised document clustering technique for information organization," *In Proc. of the 9th international conf. on Information and knowledge management*, pp. 30–37, 2000.
76. D. Cohn, R. Caruana, and A. McCallum, "Semi-supervised clustering with user feedback," *Technical Report TR2003-1892, Cornell University*, 2003. [Online]. Available: citeseer.ist.psu.edu/cohn03semisupervised.html
77. Y. Zhao and G. Karypis, "Topic-driven clustering for document datasets," *In SIAM-05 Data mining Conf.*, 2005.
78. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," *In proc. of 18th internation Conf. on Machine Learning*, pp. 577–584, 2001.
79. S. Basu, M. Bilenko, and R. Monney, "A probabilistic framework for semi-supervised clustering," *In proc. of the 10th Intl. Conf. on knowledge discovery and data mining*, 2004.
80. E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," *Advances in Natural Information Processing Systems*, vol. 15, pp. 505–512, 2003.
81. S. Dhillon and S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine learning*, pp. 143–175, 2001.
82. S. Basu, M. Bilenko, and R. Mooney, "Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering," *In Proc. of the ICML-03 workshop*, pp. 42–49, 2003.

83. S. Zhong and J. Ghosh, “A unified framework for model-based clustering,” *Jorunal of Machine Learning Research*, vol. 4, pp. 1001–1037, 2003.
84. S. Zhong, “Semi-supervised model-based document clustering: A comparative study,” *SDM-03 Workshop*, 2003. [Online]. Available: http://www.cse.fau.edu/ zhong/pubs.htm
85. “**Text Processing System**.”
86. “Uima: Unstructured information management architecture.” [Online]. Available: http://www.alphaworks.ibm.com/tech/uima
87. “Kaon: The karlsruhe ontology and semantic web tool suite.” [Online]. Available: http://kaon.semanticweb.org/
88. “Gate: A general architecture for text engineering.” [Online]. Available: http://gate.ac.uk
89. A. K. McCallum, “Bow: a toolkit for statistical language modeling, text retrieval, classification and clustering,” 1996. [Online]. Available: http://www.cs.cmu.edu/ mccallum/bow