

MINING TEXT DATA USING SIDE INFORMATION BY CONSIDERING META DATA

Monica.M¹ Ganesh.J²

*¹PG Scholar, Department of IT, ²Assistant Professor, Information Technology,
Dr.Sivanthi Aditanar College of Engineering, Tiruchendur,(India)*

ABSTRACT

The text document in the web content may contain many additional information referred as side information. These side information may be the origin information of the document or the uniform resource locator or the user access behavior of the document. Such information may contain tremendous amount of information for clustering or either it can add noise to the clustering process. In order to make the clustering process efficient side information is considered with its meta data. First, clustering is done and this approach is extended to classification process. In this classification process the attributes are been considered and experimented by the real data sets. Moreover to show the effectiveness of this process, a comparison is been made. By this method coherent clusters are created and it is experimented by using real numbers of data sets.

Keywords: *Text Mining, Clustering, Side-information.*

I. INTRODUCTION

The problem of text clustering arises in the context of many application domains such as the web, social networks, and other digital collections. The rapidly increasing amounts of text data in the context of these large online collections has led to an interest in creating scalable and effective mining algorithms. A tremendous amount of work has been done in recent years on the problem of clustering in text collections [5], [11], [27], [30], [37] in the database and information retrieval communities. However, this work is primarily designed for the problem of pure text clustering, in the absence of other kinds of attributes. In many application domains, a tremendous amount of side information is also associated along with the documents. This is because text documents typically occur in the context of a variety of applications in which there may be a large amount of other kinds of database attributes or meta information which may be useful to the clustering process. Some examples of such side-information are as follows:

- In an application in which we track user access behavior of web documents, the user-access behaviour may be captured in the form of web logs. For each document, the meta-information may correspond to the browsing behavior of the different users. Such logs can be used to enhance the quality of the mining process in a way which is more meaningful to the user, and also application-sensitive. This is because the logs can often pick up subtle correlations in content, which cannot be picked up by the raw text alone.
- Many text documents contain links among them, which can also be treated as attributes. Such links contain a lot of useful information for mining purposes. As in the previous case, such attributes may often provide insights about the correlations among documents in a way which may not be easily accessible from raw content.

- Many web documents have meta-data associated with them which correspond to different kinds of attributes such as the provenance or other information about the origin of the document. In other cases, data such as ownership, location, or even temporal information may be informative for mining purposes. In a number of network and user-sharing applications, documents may be associated with user-tags, which may also be quite informative.

While such side-information can sometimes be useful in improving the quality of the clustering process, it can be a risky approach when the side-information is noisy. In such cases, it can actually worsen the quality of the mining process. Therefore, we will use an approach which carefully ascertains the coherence of the clustering characteristics of the side information with that of the text content. This helps in magnifying the clustering effects of both kinds of data. The core of the approach is to determine a clustering in which the text attributes and side-information provide similar hints about the nature of the underlying clusters, and at the same time ignore those aspects in which conflicting hints are provided. While our primary goal in this paper is to study the clustering problem, we note that such an approach can also be extended in principle to other data mining problems in which auxiliary information is available with text. Such scenarios are very common in a wide variety of data domains. Therefore, we will also propose a method in this paper in order to extend the approach to the problem classification. We will show that the extension of the approach to the classification problem provides superior results because of the incorporation of side information. Our goal is to show that the advantages of using side-information extend beyond a pure clustering task, and can provide competitive advantages for a wider variety of problem scenarios.

II. RELATED WORK

The problem of text-clustering has been studied widely by the database community [18], [25], [34]. The major focus of this work has been on scalable clustering of multidimensional data of different types [18], [19], [25], [34]. A general survey of clustering algorithms may be found in [21]. The problem of clustering has also been studied quite extensively in the context of text-data. A survey of text clustering methods may be found in [3]. One of the most well known techniques for text-clustering is the scatter-gather technique [11], which uses a combination of agglomerative and partitional clustering. Other related methods for text-clustering which use similar methods are discussed in [27], [29]. Co-clustering methods for text data are proposed in [12], [13]. An Expectation Maximization (EM) method for text clustering has been proposed in [22]. Matrix-factorization techniques for text clustering are proposed in [32]. This technique selects words from the document based on their relevance to the clustering process, and uses an iterative EM method in order to refine the clusters. A closely related area is that of topic-modeling, event tracking, and text-categorization [6], [9], [15], [16]. In this context, a method for topic-driven clustering for text data has been proposed in [35]. Methods for text clustering in the context of keyword extraction are discussed in [17]. A number of practical tools for text clustering may be found in [23]. A comparative study of different clustering methods may be found in [30]. The problem of text clustering has also been studied in context of scalability in [5], [20], [37]. However, all of these methods are designed for the case of pure text data, and do not work for cases in which the text-data is combined with other forms of data. Some limited work has been done on clustering text in the context of network-based linkage information [1], [2], [8], [10], [24], [31], [33], [36], though this work is not applicable to the case of general side information attributes. In this paper, we will provide a first approach to using other kinds of attributes in

conjunction with text clustering. We will show the advantages of using such an approach over pure text-based clustering. Such an approach is especially useful, when the auxiliary information is highly informative, and provides effective guidance in creating more coherent clusters. We will also extend the method to the problem of text classification, which has been studied extensively in the literature. Detailed surveys on text classification may be found in [4], [28].

III. SYSTEM ANALYSIS

3.1 Proposed System

3.1.1 Clustering With Side-Information

In this section, we will discuss an approach for clustering text data with side information. We assume that we have a corpus S of text documents. The total number of documents is N , and they are denoted by $T_1 \dots T_N$. It is assumed that the set of distinct words in the entire corpus S is denoted by W . Associated with each document T_i , we have a set of side attributes X_i . Each set of side attributes X_i has d dimensions, which are denoted by $(x_{i1} \dots x_{id})$. We refer to such attributes as *auxiliary* attributes. For ease in notation and analysis, we assume that each side-attribute x_{id} is binary, though both numerical and categorical attributes can easily be converted to this format in a fairly straightforward way. This is because the different values of the categorical attribute can be assumed to be separate binary attributes, whereas numerical data can be discretized to binary values with the use of attribute ranges. Some examples of such side-attributes are as follows:

- In a web log analysis application, we assume that x_{ir} corresponds to the 0-1 variable, which indicates whether or not the i th document has been accessed by the r th user. This information can be used in order to cluster the web pages in a site in a more informative way than a techniques which is based purely on the content of the documents. As in the previous case, the number of pages in a site may be large, but the number of documents accessed by a particular user may be relatively small.

3.2 Algorithm

3.2.1 The Coates Algorithm

In this section, we will describe our algorithm for text clustering with side-information. We refer to this algorithm as *COATES* throughout the paper, which corresponds to the fact that it is a *COntent and Auxiliary attribute based Text cluStering* algorithm. We assume that an input to the algorithm is the number of clusters k . As in the case of all text-clustering algorithms, it is assumed that stop-words have been removed, and stemming has been performed in order to improve the discriminatory power of the attributes. The algorithm requires two phases:

- **Initialization:** We use a lightweight initialization phase in which a standard text clustering approach is used without any side-information. For this purpose, we use the algorithm described in [27]. The reason that this algorithm is used, because it is a simple algorithm which can quickly and efficiently provide a reasonable initial starting point. The centroids and the partitioning created by the clusters formed in the first phase provide an initial starting point for the second phase. We note that the first phase is based on text only, and does not use the auxiliary information.
- **Main Phase:** The main phase of the algorithm is executed after the first phase. This phase starts off with these initial groups, and iteratively reconstructs these clusters with the use of *both* the text content and the auxiliary information. This phase performs alternating iterations which use the text content and auxiliary attribute information in order to improve the quality of the clustering. We call these iterations as *content* iterations and

auxiliary iterations respectively. The combination of the two iterations is referred to as a *major iteration*. Each major iteration thus contains *two minor iterations*, corresponding to the auxiliary and text-based methods respectively.

The focus of the first phase is simply to construct an initialization, which provides a good starting point for the clustering process based on text content. Since the key techniques for content and auxiliary information integration are in the second phase, we will focus most of our subsequent discussion on the second phase of the algorithm. The first phase is simply a direct application of the text clustering algorithm proposed in [27]. The overall approach uses alternating minor iterations of content-based and auxiliary attribute-based clustering. These phases are referred to as *content-based* and *auxiliary attribute-based* iterations respectively. The algorithm maintains a set of seed centroids, which are subsequently refined in the different iterations. In each content-based phase, we assign a document to its closest seed centroid based on a text similarity function. The centroids for the k clusters created during this phase are denoted by $L1 \dots Lk$. Specifically, the cosine similarity function is used for assignment purposes. In each auxiliary phase, we create a probabilistic model, which relates the attribute probabilities to the cluster-membership probabilities, based on the clusters which have already been created in the most recent text-based phase. The goal of this modeling is to examine the coherence of the text clustering with the side-information attributes. Before discussing the auxiliary iteration in more detail, we will first introduce some notations and definitions which help in explaining the clustering model for combining auxiliary and text variables. We assume that the k clusters associated with the data are denoted by $C1 \dots Ck$. In order to construct a probabilistic model of membership of the data points to clusters, we assume that each auxiliary iteration has a *prior* probability of assignment of documents to clusters (based on the execution of the algorithm so far), and a *posterior* probability of assignment of documents to clusters with the use of auxiliary variables in that iteration. We denote the prior probability that the document Ti belongs to the cluster Cj by $P(Ti \in Cj)$. Once the pure-text clustering phase has been executed, the *a-priori* cluster membership probabilities of the auxiliary attributes are generated with the use of the last content-based iteration from this phase. The *apriori* value of $P(Ti \in Cj)$ is simply the fraction of documents which have been assigned to the cluster Cj . In order to compute the *posterior* probabilities $P(Ti \in Cj|Xi)$ of membership of a record at the end of the auxiliary iteration, we use the auxiliary attributes Xi which are associated with Ti . Therefore, we would like to compute the conditional probability $P(Ti \in Cj|Xi)$. We will make the approximation of considering only those auxiliary attributes (for a particular document), which take on the value of 1. Since we are focussing on sparse binary data, the value of 1 for an attribute is a much more informative event than the default value of 0. Therefore, it suffices to condition only on the case of attribute values taking on the value of 1. For example, let us consider an application in which the auxiliary information corresponds to users which are browsing specific web pages. In such a case, the clustering behavior is influenced much more significantly by the case when a user *does* browse a particular page, rather than one in which the user *does not* browse a particular page, because most pages will typically not be browsed by a particular user. This is generally the case across many sparse data domains such as attributes corresponding to links, discretized numeric data, or categorical data which is quite often of very high cardinality (such as zip codes). Furthermore, in order to ensure the robustness of the approach, we need to eliminate the noisy attributes. This is especially important, when the number of auxiliary attributes is quite large. Therefore, at the beginning of each auxiliary iteration, we compute the *gini-index* of each attribute based on the clusters created by the last contentbased iteration. This gini-index provides a quantification of the discriminatory power of each attribute with respect to the clustering process.

IV. EXTENSION TO CLASSIFICATION

In this section, we will discuss how to extend the approach to classification. We will extend our earlier clustering approach in order to incorporate supervision, and create a model which summarizes the class distribution in the data in terms of the clusters. Then, we will show how to use the summarized model for effective classification. First, we will introduce some notations and definitions which are specific to the classification problem.

We refer to our algorithm as the *COLT* algorithm throughout the paper, which refers to the fact that it is a *C*ontent and *aux*iliary attribute-based *T*ext classification algorithm. The algorithm uses a supervised clustering approach in order to partition the data into k different clusters. This partitioning is then used for the purposes of classification. The steps used in the training algorithm are as follows:

- **Feature Selection:** In the first step, we use feature selection to remove those attributes, which are not related to the class label. This is performed both for the text attributes and the auxiliary attributes.
- **Initialization:** In this step, we use a *supervised kmeans* approach in order to perform the initialization, with the use of purely text content. The main difference between a supervised k -means initialization, and an unsupervised initialization is that the class memberships of the records in each cluster are pure for the case of supervised initialization. Thus, the k means clustering algorithm is modified, so that each cluster only contains records of a particular class.
- **Cluster-Training Model Construction:** In this phase, a combination of the text and side-information is used for the purposes of creating a cluster-based model. As in the case of initialization, the purity of the clusters is maintained during this phase.

Once the set of supervised clusters are constructed, these are used for the purposes of classification. We will discuss each of these steps in some detail below. Next, we will describe the training process for the *COLT* algorithm.

```
Algorithm COATES(NumClusters:  $k$ , Corpus:  $T_1 \dots T_N$ ,  
Auxiliary Attributes:  $X_1 \dots X_N$ );  
begin  
  Use content-based algorithm in [27] to create  
  initial set of  $k$  clusters  $C_1 \dots C_k$ ;  
  Let centroids of  $C_1 \dots C_k$  be  
  denoted by  $L_1 \dots L_k$ ;  
   $t = 1$ ;  
  while not(termination_criterion) do  
    begin  
      { First minor iteration }  
      Use cosine-similarity of each document  $T_i$  to  
      centroids  $L_1 \dots L_k$  in order to determine  
      the closest cluster to  $T_i$  and update the  
      cluster assignments  $C_1 \dots C_k$ ;  
      Denote assigned cluster index for  
      document  $T_i$  by  $q_c(i, t)$ ;  
      Update cluster centroids  $L_1 \dots L_k$  to the  
      centroids of updated clusters  $C_1 \dots C_k$ ;  
      { Second Minor Iteration }  
      Compute gini-index of  $\mathcal{G}_r$  for each auxiliary  
      attribute  $r$  with respect to current  
      clusters  $C_1 \dots C_k$ ;  
      Mark attributes with gini-index which is  
       $\gamma$  standard-deviations below the  
      mean as non-discriminatory;  
      { for document  $T_i$  let  $R_i$  be the set of  
      attributes which take on the value of 1, and for  
      which gini-index is discriminatory;}  
      for each document  $T_i$  use the method discussed  
      in section 2 to determine the posterior  
      probability  $P^*(T_i \in C_j | R_i)$ ;  
      Denote  $q_c(i, t)$  as the cluster-index with highest  
      posterior probability of assignment for document  $T_i$ ;  
      Update cluster-centroids  $L_1 \dots L_k$  with the  
      use of posterior probabilities as discussed in  
      section 2;  
       $t = t + 1$ ;  
    end  
  end
```

Figure 1. Coates Algorithm.

The first step in the training process is to create a set of supervised clusters, which are then leveraged for the classification. The first step in the supervised clustering process is to perform the feature selection, in which only the discriminative attributes are retained. In this feature selection process, we compute the gini-index for each attribute in the data with respect to the class label. If the gini index is γ standard deviations (or more) below the average gini index of all attributes, then these attributes are pruned globally, and are never used further in the clustering process. With some abuse of notation, we can assume that the documents T_i and auxiliary attributes X_i refer to these pruned representations. We note that this gini index computation is different from the gini-index computation with respect to the auxiliary attributes. The latter is performed during the main phase of the algorithm. Once the features have been selected, the initialization of the training procedure is performed only with the content attributes. This is achieved by applying a k -means type algorithm as discussed in [27] to the approach, except that class label constraints are used in the process of assigning data points to clusters. Each cluster is associated with a particular class and all the records in the cluster belong to that class. This goal is achieved by first creating unsupervised cluster centroids, and then adding supervision to the process. In order to achieve this goal, the *first two iterations* of the k -means type algorithm are run in exactly the same way as in [27], where the clusters are allowed to have different class labels. After the second iteration, each cluster centroid is strictly associated with a class label, which is identified as the majority class in that cluster at that point. In subsequent iterations, the records are *constrained* to only be assigned to the cluster with the associated class label. Therefore, in each iteration, for a given document, its distance is computed only to clusters which have the same label as the document. The document is then assigned to that cluster. This approach is continued to convergence. Once the initialization has been performed, the main process of creating supervised clusters with the use of a combination of content and auxiliary attributes is started.

```

Algorithm COLT(NumClusters:  $k$ , Corpus:  $T_1 \dots T_N$ ,
    Auxiliary Attributes:  $X_1 \dots X_N$ ,
    Labels  $l_1 \dots l_N$ ):
begin
    Perform feature selection on text and auxiliary
    attributes with the use of class labels and gini index
    as explained in section 3;
    Use supervised version of algorithm in [27]
    to create initial set of  $k$  clusters denoted by
     $C_1 \dots C_k$ , so that each cluster
     $C_i$  contains only records of a particular class;
    Let centroids of  $C_1 \dots C_k$  be
    denoted by  $L_1 \dots L_k$ ;
     $t = 1$ ;
    while not(termination_criterion) do
    begin
        { First minor iteration }
        Use cosine-similarity of each document  $T_i$  to
        centroids  $L_1 \dots L_k$  in order to determine
        the closest cluster to  $T_i$  (which belongs to same
        class) and update the cluster
        assignments  $C_1 \dots C_k$ ;
        Denote assigned cluster index for
        document  $T_i$  by  $q_i(i, t)$ ;
        Update cluster centroids  $L_1 \dots L_k$  to the
        centroids of updated clusters  $C_1 \dots C_k$ ;
        { Second Minor Iteration }
        Compute gini-index of  $G_i$  for each auxiliary
        attribute  $x$  with respect to current
        clusters  $C_1 \dots C_k$ ;
        Mark attributes with gini-index which is
         $\gamma$  standard-deviations below the
        mean as non-discriminatory;
        { for document  $T_i$ , let  $B_i$  be the set of
        attributes which take on the value of 1, and for
        which gini-index is discriminatory;}
        for each document  $T_i$  use the method discussed
        in section 2 to determine the posterior
        probability  $P^{(t)}(T_i \in C_j | B_i)$ ;
        Denote  $q_i(i, t)$  as the cluster-index with highest
        posterior probability of assignment for document  $T_i$ 
        which also belongs to the same class;
        Update cluster-centroids  $L_1 \dots L_k$  with the
        use of posterior probabilities as discussed in
        section 2;
         $t = t + 1$ ;
    end
end
    
```

Figure 2. COLT training process.

As in the previous case, we use two minor iterations within a major iteration. One minor iteration corresponds to content-based assignment, whereas another minor iteration corresponds to an auxiliary attribute-based assignment. The main difference is that class-based supervision is used in the assignment process. For the case

of content-based assignment, we only assign a document to the closest cluster centroid, which belongs to the same label. For the case of the auxiliary minor iteration, we compute the prior probability $Pa(Ti \in Cj)$ and the posterior probability $Ps(Ti \in Cj|Ri)$, as in the previous case, except that this is done only for cluster indices which belong to the same class label. The document is assigned to one of the cluster indices with the largest posterior probability. Thus, the assignment is always performed to a cluster with the same label, and each cluster maintain homogeneity of class distribution. As in the previous case, this approach is applied to convergence.

V. EXPERIMENTAL RESULTS

In this section, we compare our clustering and classification methods against a number of baseline techniques on real and synthetic data sets. We refer to our clustering approach as *Content and Auxiliary attribute based Text clustering (COATES)*. As the baseline, we used two different methods: (1) An efficient projection based clustering approach [27] which adapts the *k*-means approach to text. This approach is widely known to provide excellent clustering results in a very efficient way. We refer to this algorithms as *SchutzeSilverstein [text only]* in all figure legends in the experimental section. (2) We adapt the *k*-means approach with the use of both text and side information directly. We refer to this baseline as *K-Means [text+side]* in all figure legends.

For the case of the classification problem, we tested the *COLT* methods against the following baseline methods: (1) We tested against a *Naive Bayes Classifier* which uses only text. (2) We tested against an *SVM classifier* which uses only text. (3) We tested against a supervised clustering method which uses both text and side information. Thus, we compare our algorithms with baselines which are chosen in such a way that we can evaluate the advantage of our approach over both a pure text-mining method and a natural alternative which uses both text and side information. In order to adapt the *k*-means approach to the case where both text and side-information is used, the auxiliary attributes were simply used as text-content in the form of “pseudo-words” in the collection. This makes it relatively simple to modify the *k*-means algorithm to that case. We will show that our approach has significant advantages for both the clustering and classification problems.

5.1 Data Sets

We used three real data sets in order to test our approach. The data sets used were as follows:

5.1.1 Cora Data Set: The Cora data set1 contains 19,396 scientific publications in the computer science domain. Each research paper in the Cora data set is classified into a topic hierarchy. On the leaf level, there are 73 classes in total. We used the second level labels in the topic hierarchy, and there are 10 class labels, which are *Information Retrieval, Databases, Artificial Intelligence, Encryption and Compression, Operating Systems, Networking, Hardware and Architecture, Data Structures Algorithms and Theory, Programming and Human Computer Interaction*. We further obtained two types of side information from the data set: citation and authorship. These were used as separate attributes in order to assist in the clustering process. There are 75,021 citations and 24,961 authors. One paper has 2.58 authors in average, and there are 50,080 paper-author pairs in total.

5.1.2 DBLP-Four-Area Data Set: The *DBLP-Four-Area data set* [31] is a subset extracted from DBLP that contains four data mining related research areas, which are database, data mining, information retrieval and machine learning. This data set contains 28,702 authors, and the texts are the important terms associated with the papers that were published by these authors. In addition, the data set contained information about the conferences in which each author published. There are 20 conferences in these four areas and 44,748 author-

conference pairs. Besides the author conference attribute, we also used co-authorship as another type of side information, and there were 66,832 co author pairs in total.

5.1.3 IMDB Data Set: The **Internet Movie DataBase (IMDB)** is an online collection² of movie information. We obtained ten-year movie data from 1996 to 2005 from IMDB in order to perform text clustering. We used the plots of each movie as text to perform pure text clustering. The genre of each movie is regarded as its class label. We extracted movies from the top four genres in *IMDB* which were labeled by *Short*, *Drama*, *Comedy*, and *Documentary*. We removed the movies which contain more than two above genres. There were 9,793 movies in total, which contain 1,718 movies from the *Short* genre, 3,359 movies from the *Drama* genre, 2,324 movies from the *Comedy* genre and 2,392 movies from the *Documentary* genre. The names of the directors, actors, actresses, and producers were used as categorical attributed corresponding to side information. The *IMDB* data set contained 14,374 movie-director pairs, 154,340 movie-actor pairs, 86,465 movie-actress pairs and 36,925 movie-producer pairs.

5.2 Evaluation Metrics

The aim is to show that our approach is superior to natural clustering alternatives with the use of either pure text or with the use of both text and side information. In each data set, the class labels were given, but they were not used in the clustering process. For each class, we computed the cluster purity, which is defined as the fraction of documents in the clusters which correspond to its dominant class. The average cluster purity over all clusters (weighted by cluster size) was reported as a surrogate for the quality of the clustering process. Let the number of data points in the k clusters be denoted by $n_1 \dots n_k$. We denote the dominant input cluster label in the k clusters by $l_1 \dots l_k$. Let the number of data points with input cluster label l_i be denoted by c_i . Then, the overall cluster purity P is defined by the fraction of data points in the clustering which occur as a dominant input cluster label in the k clusters by $l_1 \dots l_k$.

5.3 Sensitivity Analysis

We also tested the sensitivity of the *COATES* algorithm with respect to two important parameters. We will present the sensitivity results on the *Cora* and *DBLP-Four-Area data sets*. As mentioned in the algorithm in Fig. 1, we used threshold γ to select discriminative auxiliary attributes. While the default value of the parameter was chosen to be 1.5, we also present the effects of varying this parameter. The results are constant for both baseline methods because they do not use this parameter.

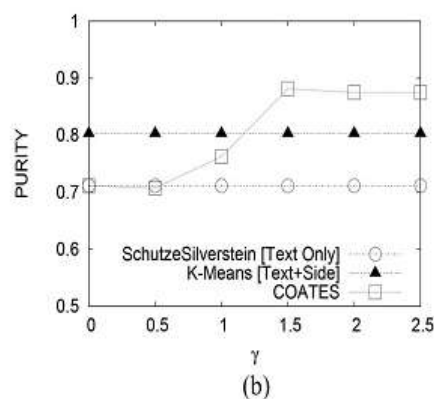
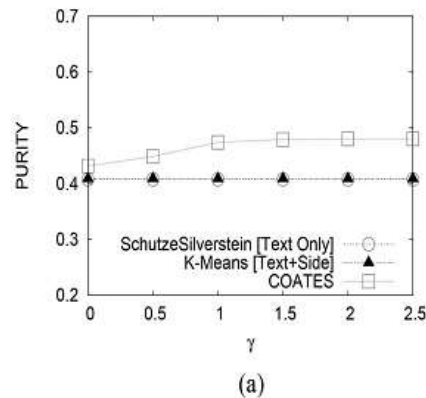


Figure 3. Sensitivity Analysis With Threshold γ . (A) *Cora* Data Set. (B) *DBLP Four- Area Data Set*

It is evident from both figures that setting the threshold γ too low results in purity degradation, since the algorithm will prune the auxiliary attributes too aggressively in this case. On both data sets, the *COATES*

algorithm achieves good purity results when γ is set to be 1.5. Further increasing the value of γ will reduce the purity slightly because setting γ too high will result in also including noisy attributes. Typically by picking γ in the range of (1.5, 2.5), the best results were observed. Therefore, the algorithm shows good performance for a fairly large range of values of γ . This suggests that the approach is quite robust.



VLEXTENSION TO CLASSIFICATION

We also tested the classification accuracy of the *COLT Classify* method, which uses both text and side information. As baselines, the following algorithms were tested (a) A Naive Bayes Classifier³, (b) An SVM Classifier⁴, and (c) A supervised *k*-means method which is based on both text and side information. In the last case, the classification is performed by using the nearest cluster based on text+side. For each of the data sets, we used 10-fold cross validation to evaluate the classification model. Clearly, the accuracy of such a model would depend upon the underlying model parameters.

The reasons are both methods process more data including text as well as side information, and they are iterative approaches extended from clustering methods. In addition, *COLTClassify* is a more complex model than the supervised *k*-means algorithm, and therefore consumes more time. However, considering the effectiveness gained by *COLTClassify*, the overhead in running time is quite acceptable. We also tested the sensitivity of the classification scheme to the parameters γ and $_$.

The sensitivity of the scheme with respect to the parameter γ for the *Cora* and *DBLP* data sets is presented. The threshold γ is illustrated on the X-axis, and the classification accuracy is illustrated on the Y-axis. The baselines are also illustrated in the same Figure. It is evident that for most of the ranges of the parameter γ , the scheme continues to perform much better than the baseline. The only case where it does not do as well is the case where the feature selection threshold is chosen to be too small. This is because the feature selection tends to be too aggressive for those cases, and this leads to a loss of accuracy. Further increasing the value of γ beyond 1.5 will reduce the accuracy slightly because setting γ too high will result in also including noisy attributes. However, for most of the range, the *COLTClassify* technique tends to retain its effectiveness with respect to the other methods. In general, since the value of γ is expressed in terms of normalized standard deviations, it is expected to not vary too much with data set. In our experience, the approach worked quite well for γ in the range (1.5, 2.5). This tends to suggest the high level of robustness of the scheme to a wide range of the choice of threshold parameter γ . The smoothing parameter $_$ is illustrated on the X-axis, and the accuracy is illustrated on the Y-axis. We note that the smoothing parameter was not required for the other schemes, and therefore the accuracy is presented as a horizontal line in those cases.

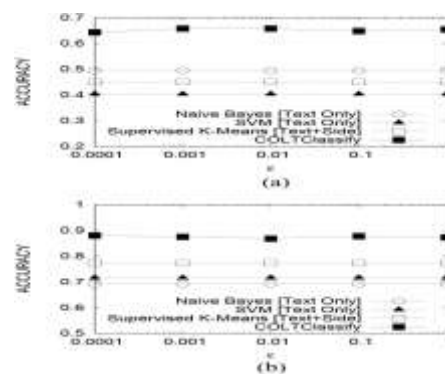


Figure 4. Variation of Classification Accuracy With The Smoothing Factor. (A) Cora Data Set. (B) DBLP-Four-Area Data Set.

For the entire range of values for the smoothing parameter α , the *COLTClassify* method performs much more effectively with respect to the other schemes. In fact, the classification accuracy did not change very much across the entire range of the smoothing parameter. Therefore, it is robust for smoothing.

VII. CONCLUSION AND SUMMARY

In this paper, we presented methods for mining text data with the use of side-information. Many forms of text databases contain a large amount of side-information or meta-information, which may be used in order to improve the clustering process. In order to design the clustering method, we combined an iterative partitioning technique with a probability estimation process which computes the importance of different kinds of side-information. This general approach is used in order to design both clustering and classification algorithms. We present results on real data sets illustrating the effectiveness of our approach. The results show that the use of side-information can greatly enhance the quality of text clustering and classification, while maintaining a high level of efficiency.



REFERENCES

- [1] C. C. Aggarwal and H. Wang, *Managing and Mining Graph Data*. New York, NY, USA: Springer, 2010.
- [2] C. C. Aggarwal, *Social Network Data Analytics*. New York, NY, USA: Springer, 2011.
- [3] C. C. Aggarwal and C.-X. Zhai, *Mining Text Data*. New York, NY, USA: Springer, 2012.
- [4] C. C. Aggarwal and C.-X. Zhai, "A survey of text classification algorithms," in *Mining Text Data*. New York, NY, USA: Springer, 2012.
- [5] C. C. Aggarwal and P. S. Yu, "A framework for clustering massive text and categorical data streams," in *Proc. SIAM Conf. Data Mining*, 2006, pp. 477–481.
- [6] C. C. Aggarwal, S. C. Gates, and P. S. Yu, "On using partial supervision for text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 2, pp. 245–255, Feb. 2004.
- [7] C. C. Aggarwal and P. S. Yu, "On text clustering with side information," in *Proc. IEEE ICDE Conf.*, Washington, DC, USA, 2012.
- [8] R. Angelova and S. Siersdorfer, "A neighborhood-based approach for clustering of linked document collections," in *Proc. CIKM Conf.*, New York, NY, USA, 2006, pp. 778–779.

- [9] A. Banerjee and S. Basu, "Topic models over text streams: A study of batch and online unsupervised learning," in *Proc. SDM Conf.*, 2007, pp. 437–442.
- [10] J. Chang and D. Blei, "Relational topic models for document networks," in *Proc. AISTASIS*, Clearwater, FL, USA, 2009, pp. 81–88.
- [11] D. Cutting, D. Karger, J. Pedersen, and J. Tukey, "Scatter/Gather: A cluster-based approach to browsing large document collections," in *Proc. ACM SIGIR Conf.*, New York, NY, USA, 1992, pp. 318–329.
- [12] I. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. ACM KDD Conf.*, New York, NY, USA, 2001, pp. 269–274.
- [13] I. Dhillon, S. Mallela, and D. Modha, "Information-theoretic coclustering," in *Proc. ACM KDD Conf.*, New York, NY, USA, 2003, pp. 89–98.
- [14] P. Domingos and M. J. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Mach. Learn.*, vol. 29, no. 2–3, pp. 103–130, 1997.
- [15] M. Franz, T. Ward, J. S. McCarley, and W. J. Zhu, "Unsupervised and supervised clustering for topic tracking," in *Proc. ACM SIGIR Conf.*, New York, NY, USA, 2001, pp. 310–317.
- [16] G. P. C. Fung, J. X. Yu, and H. Lu, "Classifying text streams in the presence of concept drifts," in *Proc. PAKDD Conf.*, Sydney, NSW, Australia, 2004, pp. 373–383.
- [17] H. Frigui and O. Nasraoui, "Simultaneous clustering and dynamic keyword weighting for text documents," in *Survey of Text Mining*, M. Berry, Ed. New York, NY, USA: Springer, 2004, pp. 45–70.
- [18] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proc. ACM SIGMOD Conf.*, New York, NY, USA, 1998, pp. 73–84.
- [19] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," *Inf. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.
- [20] Q. He, K. Chang, E.-P. Lim, and J. Zhang, "Bursty feature representation for clustering text streams," in *Proc. SDM Conf.*, 2007, pp. 491–496.
- [21] A. Jain and R. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc., 1988.
- [22] T. Liu, S. Liu, Z. Chen, and W.-Y. Ma, "An evaluation of feature selection for text clustering," in *Proc. ICML Conf.*, Washington, DC, USA, 2003, pp. 488–495.
- [23] A. McCallum. (1996). *Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering* [Online]. Available: <http://www.cs.cmu.edu/mccallum/bow>
- [24] Q. Mei, D. Cai, D. Zhang, and C.-X. Zhai, "Topic modeling with network regularization," in *Proc. WWW Conf.*, New York, NY, USA, 2008, pp. 101–110.
- [25] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proc. VLDB Conf.*, San Francisco, CA, USA, 1994, pp. 144–155.
- [26] G. Salton, *An Introduction to Modern Information Retrieval*. London, U.K.: McGraw Hill, 1983.
- [27] H. Schutze and C. Silverstein, "Projections for efficient document clustering," in *Proc. ACM SIGIR Conf.*, New York, NY, USA, 1997, pp. 74–81.
- [28] F. Sebastiani, "Machine learning for automated text categorization," *ACM CSUR*, vol. 34, no. 1, pp. 1–47, 2002.
- [29] C. Silverstein and J. Pedersen, "Almost-constant time clustering of arbitrary corpus sets," in *Proc. ACM SIGIR Conf.*, New York, NY, USA, 1997, pp. 60–66.

- [30] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *Proc. Text Mining Workshop KDD*, 2000, pp. 109–110.
- [31] Y. Sun, J. Han, J. Gao, and Y. Yu, "iTopicModel: Information network integrated topic modeling," in *Proc. ICDM Conf.*, Miami, FL, USA, 2009, pp. 493–502.
- [32] W. Xu, X. Liu, and Y. Gong, "Document clustering based on nonnegative matrix factorization," in *Proc. ACM SIGIR Conf.*, New York, NY, USA, 2003, pp. 267–273.
- [33] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proc. ACM KDD Conf.*, New York, NY, USA, 2009, pp. 927–936.
- [34] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Conf.*, New York, NY, USA, 1996, pp. 103–114.
- [35] Y. Zhao and G. Karypis, "Topic-driven clustering for document datasets," in *Proc. SIAM Conf. Data Mining*, 2005, pp. 358–369.
- [36] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/ attribute similarities," *PVLDB*, vol. 2, no. 1, pp. 718–729, 2009.
- [37] S. Zhong, "Efficient streaming text clustering," *Neural Netw.*, vol. 18, no. 5–6, pp. 790–798, 2005.

About Author

	Monica.M received the B.Tech degree in Information Technology from Anna University, Chennai, in 2013. Currently, she is pursuing M.Tech degree in Information Technology from Anna University, Chennai. Her research areas of Interests include Data Mining, Operating system, and Networking.
	Ganesh.J is currently working in Dr.Sivanthi Aditanar College of Engineering as Assistant Professor . His research areas of Interests include Cloud computing and Data mining.