Unsupervised Learning from URL Corpora

Deepak P[1*], Deepak Khemani[2]

[1]IBM India Research Lab, Bangalore, India
[2]Dept. of CS& E, Indian Institute of Technology Madras, Chennai, India
deepak.s.p@in.ibm.com, khemani@iitm.ac.in

Abstract

This paper illustrates the utility of URL information in unsupervised learning. We outline the motivation behind the usage of URL information upfront, and present two techniques for unsupervised learning from URL corpora. First, we devise a similarity measure for URL pairs putting down the intuitions behind the same and verify its goodness by using it for clustering. Further, we outline a method for keyword identification using the similarity measure. Then, we explore the usage of character N-grams for flat clustering of URL corpora. The motivation to keep URLs compact forces the usage of a lot of variations of the same word, which is a very unique kind of forcibly inserted noise. N-gram based models are very tolerant to such noise and are very compact too. We then compare the two similarity measures using a rank correlation measure, which reveals that their similarity is very subjective on the corpus used and depends on the distinctiveness of the clusters in the corpus. Given that URLs are small entities, our techniques are magnitudes faster than unsupervised techniques on full-text corpora and require far less information than the latter. To the best of our knowledge, this is the first attempt on unsupervised learning from URL information.

1. Introduction

Techniques for unsupervised learning from web documents (and hypertext) have gained a lot of attention over the past few years. Information sources for such techniques usually include hyperlink structure, the text of the web document, the latent structure in the markup language used (such as differential weighting for titles, table headings etc. in HTML), anchor texts of links to and from the web document etc. Most of the techniques proposed have ignored the URL information. Here we attempt the novel problem of unsupervised learning from corpora of URLs. First, we present a similarity metric for URL pairs, which is very different from its counterparts for document (text or hypertext) pairs. Devising similarity measures for structured representations is a well studied topic [26] in itself, mainly, in the field of Case Based Reasoning systems. We then use the similarity metric for hierarchical agglomerative clustering. It may be mentioned here that clustering [5] is a very natural choice for demonstrating the goodness of similarity measures when labeled data is available for evaluation. Further, we go on to show that the measure is useful for keyword identification from topical URL corpora and heterogeneous URL corpora. Next, we use character n-grams for clustering URL corpora. The choice of a URL involves a trade-off between compactness and expressive power, which are in a way, contradictory to each other. The unavailability of domain names with expressive terms (coupled with other regulations) forces the usage of unnatural abbreviations and less often, results in usage of meaningless homophones. Character n-grams have been shown to be very tolerant to such noise [29]. We go on to show the goodness of the two different techniques proposed by means of clustering tasks on various datasets. Having proposed two similarity measures which are very different in terms of representations and underlying intuitions, we compare them using the Spearman's Rank Correlation coefficient [36]. The comparative study reveals that the similarity/correlation between the techniques is highly dependent on the corpus used. The similarity between these techniques is found to be directly related to the distinctiveness of the clusters in the corpus.

Section 2 surveys the related work in web document clustering. Section 3 describes the dataset used. Section 4 details the intuition behind using URL as an information

International Conference on Management of Data
COMAD 2006, Delhi, India, December 14-16, 2006
©Computer Society of India, 2006

[*] This work was done while the author was with the Indian Institute of Technology Madras

source. Section 5 describes the proposed URL-Sim measure and outlines the tasks that we attempt for unsupervised learning using the similarity measure. Section 6 talks about our technique of clustering URL corpora based on character n-gram vectors. Section 7 describes the various experiments using the two techniques proposed and presents their results. Section 8 describes our study on comparing the two representations proposed. Section 9 concludes the paper and lists possible future work too.

2. Related Work

Techniques for web document clustering are very mature in the context of search engines, where clustering of results for a query has been widely and successfully experimented ([1], [2], [3], [4], [5], [6], [7]). Vivisimo[2], iBoogie[3] and Clusty[4] are search engines that do clustering of search results. Although clustering of web document corpora (in contrast to web search result corpora) have received far less attention, an evolutionary algorithm to find the most important features in web document clustering [8] has been proposed. A large benchmark dataset [9] has been published to aid clustering tasks on web corpora, although the same has been used more for classification ([10], [11]) than clustering related tasks [12]. This follows the increased attention that web document classification has traditionally been getting, compared to its unsupervised counterpart ([13], [14]). The first attempt on harnessing URL based information for a machine learning task is the MeURLin[5] system, a URL based Web Page Classifier ([15], [16]). Our work involves unsupervised clustering of URL corpora, and is very different from MeURLin in terms of the problem that is being addressed, in that the latter is a supervised task, doesn't involve a pair wise URL similarity measure and that the tokenization of URLs in the latter is biased by the training set used.

Topic Detection and Tracking [17] is an emerging field in text mining. The aim is to annotate a document. There have been studies on topic identification from textual data ([27], [28]) as well. We, in the course of our experiments, attempt to summarize homogenous corpora of URLs by sets of keywords (or keyword fragments). Further, we apply the approach to heterogeneous corpora to evaluate the performance. Although summarization of URL corpora is a novel problem, multi-document summarization ([18],[19],[20],[21],[22]) has been gaining increasing significance over the past many years. WebInEssence[6] and MEAD[7] are tools of interest in this field.

Character n-grams [29] have been studied extensively in the context of text mining. They have been used for a wide variety of tasks which include document clustering [30], authorship attribution [31] and malicious code [32]. TCatNG[8] is a toolkit for n-gram analysis of text data. Here, we illustrate the usage of character n-grams for clustering URL corpora.

3. Datasets Used

For our experiments, we use subsets of the standard corpora such as the WebKB[9] dataset, BankSearch[10] Dataset [9] (the subset used is detailed with the results in Section 7) and some corpora which we collected using Google[11]. We decided to collect our own datasets rather than completely relying on standard corpora because of the fact that the standard corpora were intended to be used as document clustering datasets, and hence good/bad performance of our techniques on them may/may not reflect the quality of our techniques. For instance, our clustering techniques work very well for the WebKB dataset, because the URLs themselves contain the category labels (cornell.edu is part of every URL categorized under Cornell). A motivating example would be to say that *www.india.gov.in*, can at best be clustered into a category of Indian or Governmental websites, but never into a cluster of information on the "*Right to Information Act*" even though the current version of the webpage may warrant its inclusion into such a cluster. Contents may change, but URLs don't (justifying their usage as URLs). For gathering corpora by means of Google, we used a query Q and obtained the top 30 results for that query from Google. Many such queries were issued, and result sets from various queries were merged to form URL corpora, each entry labeled with the query for which it was obtained as a result. Table 1 gives the details of each of the datasets used in our experiments.

Table 1. Datasets Used

Dataset Name	#docs	Description
BankSearch	150	First 50 documents of categories {Motorsport, Java, Banks} from BankSearch Dataset
BS0469	4000	Subset of pages for categories {Banks, Java, VisualBasic, Motorsport} from the BankSearch Dataset
BS46	2000	Subset of pages for categories {Java, VisualBasic}
Corpus1	90	Top 30 search results for {Tennis,

[2] http://www.vivisimo.com

[3] http://www.iboogie.com

[4] http://www.clusty.com

[5] http://wing.comp.nus.edu.sg/meurlin/

[6] http://tangra.si.umich.edu/clair/home/web.html

[7] http://tangra.si.umich.edu/clair/home/mead.html

[8] http://tcatng.sourceforge.net/

[9] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

[10] http://www.banksearchdataset.info

[11] http://www.google.com

		Kerala, Computer Science}
Corpus2	90	Top 30 search results for {IIT, Cricket, London}
Corpus3	90	Top 30 search results for {Government, Jobs, Sports}
WebKB	159	Course Pages for Cornell, Texas and Washington Universities
CRNL	192	WebKB subset of pages related to Cornell

4. URLs as an Information Source

4.1 Information Content of a Single URL: The Use of Background Knowledge

We would like to state upfront that the discussion in this subsection is to enable appreciate the information content of a single URL, given some background knowledge. Although it is more relevant for a supervised task which organizes and uses background knowledge, we include this to emphasize that URLs are an important information source and thereby justify our intention of mining URL corpora. Consider the URL *www.cs.cmu.edu*. Given the background knowledge that *edu* refers to educational institutions and that *cs* refers to computer science, we could readily infer that it is the homepage of the *CS* department/school in a university which bears the acronym *CMU*. This is illustrative about the info that a URL (coupled with the background knowledge) can hold, and is suggestive of the effectiveness of supervised learning techniques on URLs. This motivates the usage of learning from URLs.

4.2 URL Corpora: Motivation for Unsupervised Learning

Unsupervised learning involves learning from an unlabeled collection of entities; we choose to delve into the possibilities of unsupervised learning from URL corpora. The set of URLs <*www.iisc.ernet.in*, *www.iitkgp.ernet.in*, *www.iitm.ernet.in* > all belong to research/educational institutions in India and *ernet* is a suffix for the Indian "*Education and Research Network*". The occurrence of *ernet* can enable the above URLs to cluster together due to the common factor that they belong to the same class of institutions. Further, *www.abc.ernet.in* can be inferred to be somehow related to the above URLs due to the common suffix. For a cluster of URLs that have the *ernet* suffix, *ernet* could be a descriptive keyword. URLs are a scanty resource for mining and hence mining just the URL words (delimited) would not possibly suffice. Such mining would not recognize the similarity between *www.kerala.gov.in* and *www.newkerala.com* (in that both relate to the same state called *Kerala* in *South India*). This motivates the usage of common substring based similarity measurements for clustering. Although very hostile cases, such as *www.whitehouse.gov* (US Government Site) and *www.whitehorse.com* (A website design company), would surely plague the similarity measure, our experiments show that such cases aren't frequent enough to dissuade one from deploying this technique.

4.3 Variable Information Content of URL Segments

URLs are variably delimited (as opposed to all delimiters carrying the same meaning or significance) sequences (as opposed to sets) of words. We refer to a delimited segment of text in the URL as a *URL segment*, with every non-alphanumeric character treated as a delimiter (with exceptions for ASCII characters, such as *%20* representing whitespace).

Consider the example of a website with the URL: *http://www.cs.abc.edu/courses/current/cs511/assignments* a *synthetic* example for the assignments page of a course CS511 offered in the current semester by the *CS* department of the *ABC* University. *Current* can be made sense of, only in the context of *courses*. Similarly, *CS511* makes sense only in the context of *courses*. The vice versa isn't always true. For instance, *courses* makes sense even in the absence of *CS511* and *assignments*. The best determiner would be the segment that conveys most information regarding which cluster the URL would go into. Thus, *current* would have been the best determiner if the separate clusters were *current courses* and *past courses*. Our argument is based on our observation (from inspection of general web document clustering datasets) that even the initial segments of the URL (except the stop-words such as *www*, *net* etc) would usually determine the class which it should go to. Thus, as we walk thru the segments of the URL left-to-right, the specificity (of the cluster for which the current segment is a good determiner) increases. As we are interested in only the top-level (most abstract/general) clusters most of the time (as observed from popular web document datasets), we argue that *courses* is a better determiner than *current* (and *cs511*) and conveys more information regarding the cluster that the URL should go into (as *courses* is indicative of the initial context upon which the segments later in sequence build). *Our discussion translates to an interesting hypothesis; that information content decreases as we go down the URL.* The first sequence of URL segments, the one starting right after the protocol specifier and going on till the next non-dot delimiter, commonly referred to as a hostname, notably presents an exception with regard to information content breakup between segments. In hostname *www.cs.abc.edu*, '*CS*' makes sense only in the context of *abc* whereas *abc* makes sense only in the context of *edu*. It can be readily inferred that the vice versa isn't true. We sum up our conclusions in this regard to say that *information content of a segment decreases as we go down the URL, whereas it is exactly the opposite in the case of the hostname*.

5. URL-Sim: A Similarity Measure for URLs

The similarity measure (for pairs of URLs) that we devise, called URL-Sim, is detailed in the first subsection. Further, we go on to show how URL-Sim can be used in unsupervised learning tasks. The description of the tasks comprises the second subsection.

5.1 URL-Sim Computation

5.1.1 Preprocessing. We remove the scheme/protocol tag from each URL in the pair for which we want to compute the similarity. This is based on the intuition that scheme information is very rarely a determiner for that category that a URL falls in. Further, we remove stopwords from the URLs. Stopwords are words with little determining power. We have currently identified the set of stopwords as {*com*, *net*, *www*}. Further, as the last part of phase 1, the order of URL segments in the hostname are reversed, so that the URL segments are ordered in the decreasing order of information content (Ref. Sec 4.3). Thus, we get a newer trimmed URL from the original input URL. Some examples are cited below.

Table 2. Phase 1 Tasks as Input-Output pairs

Input to Phase 1	Output from Phase 1
http://www.iitm.ac.in/students	in.ac.iitm/students
http://www.cs.cmu.edu/afs	edu.cmu.cs/afs

5.1.2 Tokenizing and Weight Tagging. Tokenizing involves splitting up the URLs into segments. We refer to a URL segment as a delimited segment of text in the URL, with every non-alphanumeric character treated as a delimiter (with exceptions for ASCII characters, such as %20 representing whitespace). Each segment has an associated level, which stands for the number of '/' (slashes) occurring before it in the pre-processed URL, and an associated rank, which is the sequential order of the segment in the pre-processed URL among the segments which are in its level. We have a hard-coded function, *total_weight(level)*, which (represented as input output pairs in table 3) gives the total weight that a level is assigned with. *total_weight(level)* gives the sum of the weights of the segments in that level. The weight assignments were arrived at empirically, experimenting with widely different corpora. Although widely varying corpora peaked in clustering performance at different weight assignments, we settled on an assignment (which is given below) where the clustering performance was not bad for any corpora and was better than average for the majority of the corpora. This is based on the intuition that the weight assignment should carry the bias that URLs are structured in such a fashion, and should not be biased on the specific training set (any specific corpus) of URLs used. The weight break-ups for the different segments in the same level was arrived at by choosing the best break-up (in the same fashion as for weight assignments) from a set of break-up methods popular in literature.

Table 3. Weight Assignments to URL Levels

Level	Total_Weight(Level)
0	10
1	8
2	6
3	4
4	2
5 – upwards	0

The split-up of weights between the segments in a level is dependent on the rank of each segment and the total number of segments in the level. The function to determine the weight of a segment (the total number of segments in a level represented by *total_segs(level)*) is given as in (1) below. The expression is quite simple in that, it distributes weights in a level in the reverse order of ranks. Thus, if there are two tokens in a level, with '*a*' bearing rank 1 and '*b*' bearing rank 2, the ratio of weights of '*a*' and '*b*' would be 2:1, with the added (obvious) constraint that the total weight would add-up to the total weight allocated to the level. Thus, at the end of this level, each segment of the URL would be tagged with a weight. An example is included in table 4 below.

Equation 1. Segment Weighting Function

Weight(Segment s) = (total_segs(s.level()) + 1 – s.rank())*total_weight(s.level()) /(1+2+…+total_segs(s.level())	(1)

Table 4. Weight Break-ups for a Sample URL

Pre-processed URL	Segment	Weight
in.ac.iitm/students	in	5.00
	ac	3.33
	iitm	1.67
	students	8.00

5.1.3. Similarity Computation. The final similarity computation for URL pairs, involves pair wise matching of each segment from the first URL, URL_1 with each segment from the second URL URL_2. The similarity value is initialized to zero, with each segment pair adding a value to the similarity depending on the length of the segments, weights associated with the segments, and the length of the largest common substring. The function is briefly summarized in Figure 1. The increment is computed as the weighted average of the '*amount*' of match between the segments. If both the segments are the same (strings), increment would be the average of their weights. Figure 1 gives the pseudocode of the algorithm.

```
Similarity Computation for URL Pairs
URL-Sim(URL $URL_1$, URL $URL_2$)
{
  URL-Sim-Val = 0;
  for each pair <$seg_1$, $seg_2$>, $seg_1$ • $URL_1$
  and $seg_2$ • $URL_2$
  {
    Let C = Length of the Largest Common
    Substring, str, between $seg_1$ and $seg_2$;
    Increment =
      ((weight($seg_1$)*C/$seg_1$.length()) +
     (weight($seg_2$)*C/seg2.length()))/2.0;
    URL-Sim-Val = URL-Sim-Val + Increment;
  }
  return URL-Sim-Val;
}
```

Figure 1. Similarity Computation Algorithm

5.2 Unsupervised Learning using URL-Sim

5.2.1 Clustering. Having obtained the pair wise similarity measures for every URL pair in the URL corpus, we apply the well-known hierarchical agglomerative clustering algorithm [23] on the corpus. It starts of with as many singleton clusters as there are URLs and goes on to merge two closest clusters per iteration. The similarity between clusters are taken as the average of the similarity between URL pairs <URL_1, URL_2> the first item in the pair from one cluster and the second item from the other (average-link measure). The average purity [24] of the clusters is determined using the formula in Equation 2 below. Purity is considered to be a better evaluation measure (for clustering algorithms) [25] than other measures such as entropy.

Equation 2. Purity of a Clustering

$$\text{Purity(Clustering C)} = \left(\sum_{C_i \in C} \{\text{Cardinality of the most frequent label in } C_i\} \right) / |\text{Total elements}| \quad (2)$$

5.2.1 Keyword Detection. The second unsupervised learning task that we attempt is that of keyword detection on a homogeneous corpus. Homogeneous corpus is a corpus which contains URLs centered on a single topic. For instance, the first 30 results for the web search query "*A*" intuitively forms a homogeneous corpus for the topic "*A*". This is according to the assumption that web search engines typically return authoritative pages for a search query among the top results. This follows from the assumptions in the popular HITS [33] and PageRank [34] algorithms for web search. The algorithm is summarized in the pseudocode in Figure 2. This algorithm reuses most of the URL-Sim algorithm. It outputs a scored list of words for the input corpus. A ranked list of keywords could be considered to be accurate if it contains the topic keyword within the top k ranks. Also note that, the strings output may not be full-keywords, but may just be keyword fragments, as we choose to score the largest common substrings rather than segments themselves. The technique is outlined in Figure 2.

```
Keyword Detection Algorithm
Keyword_Detection(Corpus C)
{
  for every possible string s, s.score = 0;
  for every pair <$URL_1$, $URL_2$>, $URL_1$,$URL_2$ ∈ C
  {
    Pre-process $URL_1$ and $URL_2$ (Section
        5.1.1) and weight-tag their segments
        (Section. 5.1.2);
    for each pair <$seg_1$, $seg_2$>, $seg_1$ ∈ $URL_1$
      and $seg_2$ ∈ $URL_2$
    {
      Let C = Length of the Largest Common
      Substring, str, between $seg_1$ and $seg_2$;
      Increment =
       ((weight($seg_1$)*C/$seg_1$.length()) +
       (weight($seg_2$)*C/$seg_2$.length()))/2.0;
      str.score = str.score + Increment;
     }
  }
  Output the list of strings in the
   descending order of scores
}
```

Figure 2. Keyword Detection Algorithm

6. N-Gram Based Clustering for URL Corpora

6.1 Why Character N-Grams for URL Clustering?

URLs are unique in many aspects. Apart from being a globally unique identifier for a website (or page), they serve to describe the page as well (to some extent). The webmaster usually wants to make his URL as expressive and compact as possible. The former lends more descriptive power to the URL, whereas the latter is motivated by the need to make the URL simple for users to remember. The expressiveness requirement can be satisfied either by choosing the most common of words (for example, *greetings.com* for a greeting card site) or by using more than one word to describe it (e.g., *rentacoder.com*, *win2pdf.com* etc). Choosing the most common of words is not always possible because many such domains may be unavailable. *Thus the usual parameters for URL selection are compactness of the URL and also the expressive power.* They are more often than not, conflicting requirements and thus, people tend to choose very unnatural names which are usually abbreviations (*gov* for government sites, *cric* for cricket related sites) or less often, meaningless homophones (*2* for *to* as in *contests2win.com* and *win2pdf.com*, *indya* for *India* as in *indya.com* etc) of meaningful words. Finding all such abbreviations for a given word from a corpus explicitly is obviously a difficult task in an unsupervised setting. Character n-grams capture such dependencies implicitly and hence are very well suited for a problem such as the one that we are addressing. URL-Sim also captures such dependencies implicitly due to the substring component, but it is suitable only for algorithms which can work with pair-wise similarities such as hierarchical agglomerative clustering. K-Means [35] clustering and its variants have the added advantage that they are linear in the number of elements to cluster (as opposed to hierarchical agglomerative clustering which is quadratic on the number of elements to cluster), but require a vector space embedding of the entities. *Summing up the discussion, we find that character n-grams are interesting due to two factors, that they are very tolerant to noise and that they give a vector space embedding of the data so well-known partitional clustering algorithms such as K-Means can be used.*

6.2 Character N-Gram Vectors

Character N-grams are character sequences of length n. Thus bi-grams are sequences of length 2 and trigrams are those of length 3. For example, the bigrams for *SIX:* would be *SI*, *IX* and *X:*. Character n-grams for text data usually take non-alphabetic characters, such as punctuation characters, into account. In the domain of URLs, usage of non-alphabetic characters is more defined by the structural requirements of URLs as opposed to free text. Further, as mentioned in Section 4.3, the consecutive segments of a URL may differ in the level of generality. Thus, for a given URL, the set of n-grams would be taken as the union of the set of n-grams for each segment in the URL. To cite an example, the set of bigrams for *abc.org* would be {*ab*, *bc*, *or*, *rg*} as *abc* and *org* are the two segments associated with the URL. The task of building n-gram vectors would be straightforward from hereon. The monogram vector for the same URL *abc.org*, would be of length 26 (as we are considering non-case-specific English alphabets only) and the k^{th} element in the vector would assume the value equal to the frequency of k^{th} monogram (in the case of monograms, the k^{th} alphabet) in the URL. This extends to n-grams for any value of *n*. For a given task, we fix the value of n and get the corresponding n-gram vectors for the URLs. We normalize the vectors so that the elements of each vector sum up-to unity so that all URLs have the same influence on the clustering algorithm irrespective of their lengths.

We chose to use a fixed value of *n* to build the n-gram vectors because using multiple values of *n* would require us to devise weighting schemes for the different features. When comparing two URLs, the occurrence of the same 4-gram in both should ideally strengthen the similarity score much more than the occurrence of the same bigram in both. Empirically determining such weighting schemes would have an inherent training set bias, removing which would be highly non-trivial, if at all possible. Further, we chose not to include the URL level based differential weighting (Ref: Section 4.3) because multiple words in multiple levels of the URL can contribute the same n-gram which makes empirically estimating the weights for different levels difficult.

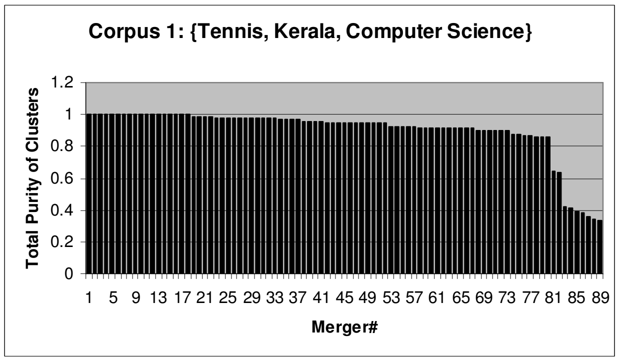
7. Experiments and Results

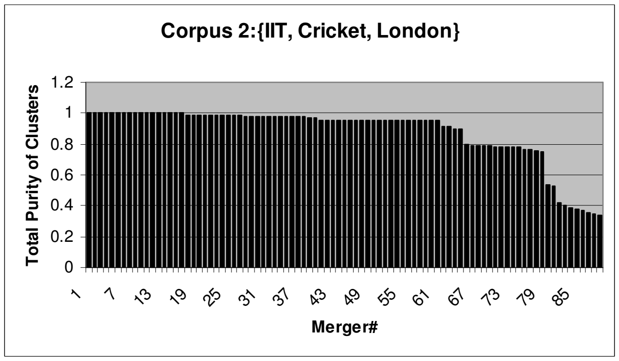
This section is a compilation of the experiments that we performed using the techniques outlined in Sections 5 and 6, their results and related observations. The first subsection focuses on the experiments which involve the techniques proposed in Section 5 whereas the second sub-section includes those concerning the character n-gram vector based clustering as proposed in Section 6.

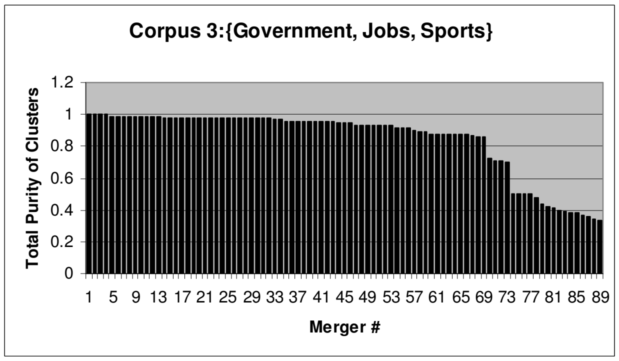
7.1 Experiments using URL-Sim

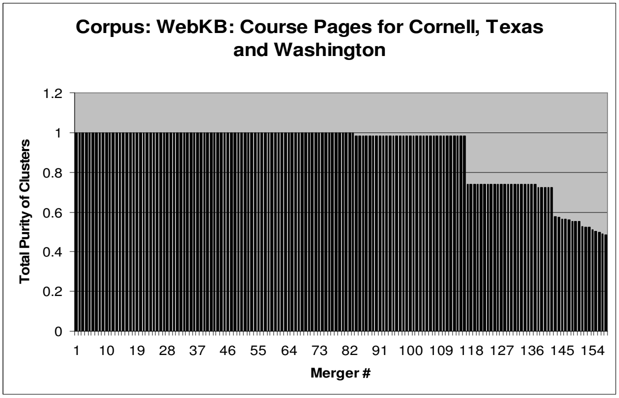
7.1.1 Clustering Results. We present a sequence of charts herewith summarizing the results of our clustering experiments on various corpora. For a cluster having 90 documents, 30 each of 3 different labels, the final and initial clustering purities will always be 0.33 (=30/90) and 1.0 (all singleton clusters) respectively. We consistently use corpora containing exactly three labels for our experiments to aid visual comparison of result graphs. The quality can (intuitively) be assessed by means of how late the curve declines (or starts to decline). *The later (i.e., the more towards right) and the sharper the decline of the purity curve, the better the clustering (and hence the similarity measure used for clustering).* For ease of evaluation, we list the number of *mistakes* by the clustering algorithm in the first 30% and 50% of the mergers. A *mistake* is defined as a merger that results in a decrease of total purity. *Merging accuracy can be defined as the number of non-mistake mergers to the total number of mergers done. Our results show a merging accuracy of 94.82% for the initial 30% mergers and 92.74% for the initial 50% mergers* across the various corpora experimented with.

Figure 3. Purity Plots for Various Datasets









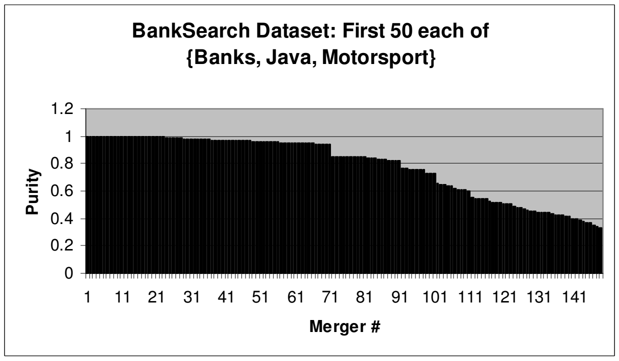


Table 5. Log of "Mistakes"

Corpus	Count (No. of documents in the corpus)	Mistakes in first 30% mergers	Mistakes in first 50% mergers
Corpus 1	90	1	3
Corpus 2	90	1	3
Corpus 3	90	2	5
WebKB	159	0	0
BankSearch	150	5	10
Average Merging Accuracies First 30% mergers: *94.82%* First 50% mergers: *92.74%*			

7.1.2. Keyword Identification Results. We present the results (word, score tuples in descending order of scores; as many of the top *k* tuples as we deem to be relevant) of the keyword identification experiments, both for homogenous corpora and heterogeneous corpora already used. *A ranked list of keywords is considered to be accurate if it contains the topic keyword (or a very representative segment of the same) within the top k ranks (k should be as close to unity as possible).* Our algorithm satisfies the accuracy criterion at an average value of *k = 1.933 (std. dev. = 1.43)* among the homogeneous datasets that we tested. An interesting fact to be noted here is that WebKB dataset is the one where the keyword identification algorithm performance is the worst, whereas the clustering gave maximal purity for the WebKB corpus. The keyword identification algorithm meets the accuracy criterion at an average value of *k = 1.33 (std. dev. = 0.65)*, for the set of homogeneous corpora excluding the WebKB corpus. For quantifying the results on our experiments on heterogeneous corpora, we define *accuracy as the number of topical keywords that have occurred in the top 5 results (in the ranked list) upon the total number of legitimate keywords in the heterogeneous corpus in question*. The average accuracy within the top 5 ranks is found to be 67% with the algorithm reporting a very low accuracy of 33% for the WebKB and BankSearch datasets

Table 6. K/w Identification on Homogeneous Corpora

Homogeneous Corpus	List of Sorted Keyword Score Tuples
Cricket	<**cricket**, 1161> <co, 243>
Computer Science	<**cs**,1094> <edu,313>
Government	<**gov**,1007> <go,106>
IIT	<**iit**,1292> <in,125>
Jobs	<**jobs**,706> <co,86>
Kerala	<**kerala**,1003> <keral,114>
London	<uk,300> <**london**,192>
Sports	<**sports**,402> <sport,224>
Tennis	<**tennis**,748> <en,230>
Cornell (WebKB)	<info, 6560> <cs,5222> <courses, 3780> <**cornell**, 3095>
Texas (WebKB)	<cs,7263> <users,5624> <**utexas**,2343> <edu,1171>
Washington (WebKB)	<education,22800> <edu,17555> <courses,17100> <cs,16026> <**washington**,9880>

Banks (BankSearch)	<co,3057> <uk,1258> <**bank**,677>
Java (BankSearch)	<**java**,2443> <ava,602> <ja,245>
MotorSport (BankSearch)	<in,473> <**motorsport**,342> <or,324> <race,285> <sport,246>
Accuracy Criterion satisfied at k = 1.933 (Average)	

Table 7. Keyword Identification on Heterogeneous Corpora

Heterogeneous Corpus	**Accuracy** (top 5)	Keywords
Corpus 1	100%	<**cs**,1321> <**kerala**,1003> <**tennis**,748> <co,575> <en,507>
Corpus 2	67%	<**iit**,1292> <**cricket**,1161> <co,957> <uk,459> <ac,390>
Corpus 3	100%	<**gov**,1348> <**jobs**,772> <**sports**,402> <co,365>
WebKB	33%	<cs,110190> <education, 22800> <courses,17864> <**washington**,9753>
BankSearch	33%	<co,5183> <in,3910> <**java**,2443> <al,2042> <es,2031>
Average Accuracy within the top 5 ranks: 67%		

7.2 Clustering of N-Gram Vectors: Results

We hereby present the results of K-Means clustering of n-gram vectors for each value of n. The Euclidean distance measure is used consistently in the course of our experiments as it is among the most popular and well-studied distance measures in literature. For each dataset (which has an associated value of *k*, the number of classes/clusters that it has), we apply K-Means for *K=k*, *2k* and *3k* and get the average purity (across clusters) [24] measures for each clustering task across multiple iterations of the algorithm (with different seeds). We experimented with different values of *n* (Ref: Figure 4) and found that performance doesn't improve any further (and in most cases, deteriorates) when *n* is set to a value greater than 3, mainly because setting *n* to a value greater than 3 misses any segment of length 3 (which are abundant among URLs). Further, keeping *n* as low as possible is motivated by the fact that the size of the vector space (it may be noted that K-Means is linear in the number of dimensions of the vector space) increases exponentially with increasing *n*.

The striking observation is that the increase in purity of clusters is only marginal when we move from bigram vectors to trigram vectors. The bigram vectors are very sparse and the average number of non-zero entries over each corpus across corpora was around 250. The purities are very high for the WebKB dataset, which was due to the presence of the class label in the URL itself. Further, it reports a purity of more than 80% for the web search datasets, which is very interesting in itself. The web search corpora yielded an average purity of 97% when clustered using document texts, but the vector spaces were typically in the range of 3000-5000 dimensions (words). Clustering based on document texts took roughly 300 times more time as compared to clustering using n-gram vectors. Figure 5 gives the variation of purity of bi-gram based clustering when *K* was set to different multiples of *k* (where *k* is the number of natural clusters, i.e., the number of unique labels in the dataset).

Figure 4. Purities of K-Means Clusters

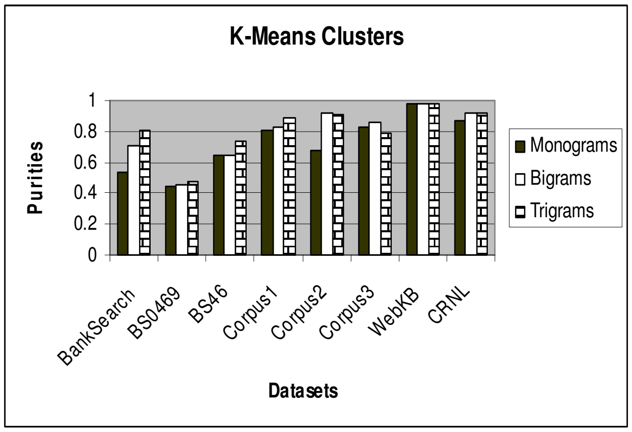
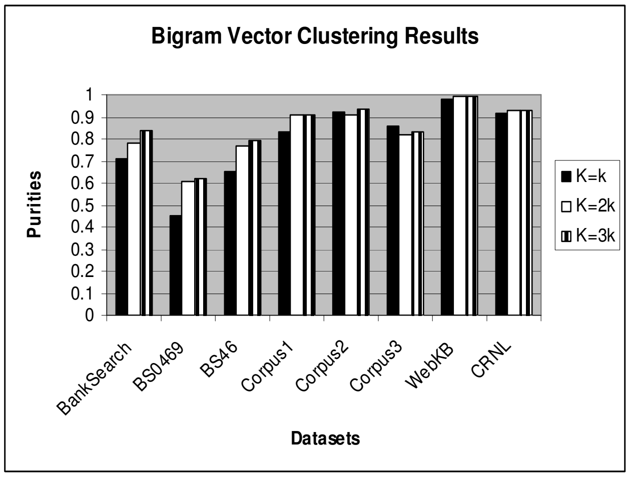


Figure 5. Bigram Clustering Purities for Varying K



The increase in purity with increasing *K* for the web search datasets as well as the small datasets (those with a small number of documents) is marginal. But, BS0469, which is a dataset with 4000 documents, shows a large increase in purity from *K=k* to *K=2k*. For large datasets, applying K-Means to produce a large number of clusters followed by agglomeration would be a better choice as opposed to small datasets, where flat clustering with *K=k* would provide results which cannot be bettered by the introduction of an agglomerative phase.

8. Comparing the two approaches

In this paper we have proposed two techniques for computing similarities over URLs. One of them, URL-Sim is a function for computing pair-wise similarities between URLs, whereas N-Gram vector is an embedding of the URLs on a vector space (over which any distance function can be applied) where we choose to measure distances using the Euclidean distance (L2 norm). The first subsection lays down the motivation for a comparative study. The second subsection herein describes the methodology for the study, whereas the third subsection outlines the inferences and the observations.

8.1 Motivation

Given that we have proposed two techniques which have been shown to be good individually, we would like to compare them to find how similar they are. Further, given that the individual techniques are very different from each other, use very different representations for URLs, and are based on totally unrelated assumptions/intuitions, it would be an interesting exercise whatever the output of the study were to be. If they are uncorrelated, both being good in their own respects, it would make sense to examine as to whether they can be combined in some way to achieve a higher degree of accuracy in clustering; i.e., some method that can capture the goodness of both the techniques. Although arriving at such a combination would not be as easy as it sounds, the chances of being able to arrive at such a combined technique would intuitively increase with the increasing dissimilarity between the techniques. If the two techniques are similar (i.e., show a high degree of correlation), it would be interesting too since the techniques themselves are based on very different building blocks and different views of the same URL. Further, it would be interesting to evaluate how much role the corpus plays in determining the similarity between the two techniques.

8.2 Methodology

URL-Sim can provide only pair wise similarities and only one similarity measure, whereas the NGram vector space embedding could potentially result in various distance measures depending on the distance function used. We have been consistently using the L2 norm (Euclidean distance) for distance computation using NGram vectors through this paper. We describe herewith our methodology for comparing the two techniques by means of the pseudo code in Figure 6.

```
Comparing NGram+L2 and URL-Sim
For each Corpus
{
     Generate the list of URL-pairs
     sorted in the ascending order of
     distances according to NGram+L2 as
     $List_1$;
     Generate the list of URL-pairs
     sorted in the descending order of
     similarities according to URL-
     Sim as $List_2$;
     Compute the SpearMan Rank
     Correlation Coefficient [36] for
     the two lists $List_1$ and $List_2$ as
     both of them contain the same
     URL-pairs, but in a different
     order;
}
```

Figure 6. Comparing the two Measures: Methodology

Thus, we get one correlation score for each corpus that we would be experimenting with. We chose to use the Spearman Coefficient [36], because it is the most popular technique to compare two ranked lists and has gained much more acceptance among the statisticians as compared to other techniques such as Kendall Tau Distance [37]. Spearman Coefficient would give a value between -1 and +1, wherein a numerically higher value would mean that there is a higher agreement between the two lists.

8.3 Results and Observations

We summarize our results as in Table 8.

Table 8. Correlation Analysis Results

Corpus	#docs	Correlation
Corpus1	90	0.252847
Corpus2	90	0.32945
Corpus3	90	0.056278
BankSearch	150	0.467305
WebKB	159	-0.14001

The results show that the correlation (between NGram+L2 and URL-Sim) is stongly dependent on the corpus and varies widely among the corpora. The corpora that we have used are very different in their characteristics and the level of generality of the clusters involved. For instance, *Government* and *Jobs* had a lot of documents (URLs) with a high-level of similarity among them (from a general analysis of similarities between clusters). The BankSearch dataset contained the widely varying categories of {*Banks*, *Motorsport*, *Java*}. The WebKB dataset, where the technique disagree maximally, has URLs with the same structure as they are from University web pages. This leads to a very interesting observation, *that the techniques tend to agree more when the clusters or classes involved are very different/distinct. They tend to disagree more when the different clusters are related (presence of some parameters which cause URLs in different clusters to be similar) in some sense or the other.*

From the observations, the methodology for combining the two techniques in order to get the best out of both isn't very clear given that the relationship/correlation between the techniques is dependent on the corpus. We tried combining the ranked list of URL pairs from NGram+L2 and URL-Sim by summing up the ranks from the lists for each URL pair and then sorting the URL pairs in the ascending order of total ranks. The average-link measure was used for computing similarities between clusters and hierarchical agglomerative clustering was tried using the similarity measure. The results were not very encouraging with the performance deteriorating marginally as compared to using only URL-Sim, in most cases. This probably points to the fact that there has to be a more intelligent technique of combining the techniques if we are to get the best of both techniques.

9. Conclusions, Contributions and Future Work

9.1 Conclusions

Based on our experiments, we conclude that URL-Sim performs very well as a similarity measure for URL pairs. The number of *mistakes* that HAC makes (with the URL-Sim measure) is minimal among the first few mergers. Although URL-Sim doesn't take the content of the target web-pages into account (which is by far, the major knowledge source for text clustering tasks), we could use HAC with URL-Sim and perform the first few, say 30%, of the mergers and arrive at initial clusters which would provide a good deal of insight into the kind of clusters present in the corpus. The keyword identification experiments also have performed exceedingly well on homogeneous corpora with the main topic-word being ranked as the highest scoring word in nine out of the 15 corpora that we chose to test with. As can be seen from the results, the techniques worked well for heterogeneous corpora too. Further, we have shown that n-gram vectors are useful for clustering based on URL information and that they can fetch a purity of above 80% on web search corpora. Bigram vectors are consistently better than monogram vectors, but the value addition (over bigram vectors) obtained by considering trigram vectors, is minimal. At times, bigram vectors perform better than trigram vectors as well, which may be explained by the fact that trigram vectors completely disregard URL segments of length two. This, when read in tune with the fact that trigram vectors are in a space which is orders of magnitude larger than the bigram vector space, leads us to the conclusion that *bigram vectors are most suited for clustering URL corpora*. Further, when the URL dataset is huge, it is better to set *K* to a high value in K-Means and let an agglomerative phase do the merging to the desired number of clusters. Further, having proposed two techniques, our observation that the correlation between the techniques is directly related to the distinctiveness of the clusters in the corpus would be a very useful pointer for any future work on combining the techniques.

9.2 Contributions

Our contribution in this paper is five-fold. *Firstly*, we have introduced the novel task of unsupervised learning from URL corpora and shown that URLs are a *good enough* information source. *Secondly*, we devise a similarity measure for URL pairs, URL-Sim, which makes use of the intuitive structure and differential information content of the URL and illustrate its goodness by applying it to a clustering task. *Thirdly*, we present an approach utilizing the URL-Sim measure for keyword extraction from homogeneous URL corpora. *Fourthly*, we present character n-gram vectors as a representation of URLs which induce a high clustering accuracy and go on to observe that bigram vectors are most suited to represent URLs. *Lastly*, we show, by means of a comparative evaluation of the representations proposed, that the correlation between the techniques is largely dependent on (and directly related to the distinctiveness of) the clusters in the corpus used.

9.3 Future Work

Future work in this regard could address the application of further unsupervised learning techniques such as association rule mining on URL corpora. Usage of ontologies to enhance the URL-Sim function could be explored. Techniques to project the URLs into a vector space could be useful as it would aid usage of partitional clustering algorithms such as K-Means. For any given dataset, we can have multiple views of data, some of which may include some others. This is especially true of web data where the latent structure in the HTML tags, the URLs, the titles of the pages etc all provide useful information. Clustering using most of these would be much cheaper as compared to traditional text clustering where there would be a vector space of thousands of dimensions. Refinement of such low-cost clusters using some additional information would be an interesting problem to look into. Further, URL information may be used to find URL-level similar pages for a given page. This would be interesting in many contexts because of the static nature of URLs as opposed to page content.

References

1. Zamir, Etzioni, Madani, Karp, "Fast and Intuitive Clustering of Web Documents", SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 1997), 1997
2. Oren Zamir, Oren Etzioni, "Web Document Clustering: A Feasibility Demonstration", SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998), 1998.

3. Oren Zamir, Oren Etzioni, Madanim, “Grouper: A dynamic clustering interface to web search results”, World Wide Web Conference (WWW 1999), 1999
4. Xiaofeng He, Hongyuan Zha, Chris H. Q. Ding, Horst D. Simon, “Web document clustering using hyperlink structures”, Computational Statistics and Data Analysis, 2002
5. Alexander Strehl, Joydeep Ghosh, Raymond Mooney, “Impact of Similarity Measures on Web-Page Clustering”, Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search, 2000
6. Deepak P, Jyothi John and Sandeep Parameswaran, “Context Disambiguation in Web Search Results”, IEEE International Conf On Web Services (ICWS 2004), 2004
7. Wang, Kitsuregawa, “Link-based Clustering of Web Search Results”, Web Age Information Management (WAIM 2001), 2001.
8. Mark P Sinka, David W Corne, “Evolving Document Features for Web Document Clustering: A Feasibility Study”, Proceedings of the IEEE Congress on Evolutionary Computation, 2004
9. Mark P Sinka, David W Corne, “The BankSearch web document dataset: investigating unsupervised clustering and category similarity”, Journal of Network and Computer Applications, 2005
10. Benbrahim, Bramer, “An empirical study for hypertext categorization”, IEEE International Conference on Systems, Man and Cybernetics, 2004, pp.5952-5957
11. M. Kovacevic, M. Diligenti, M. Gori, V. Milutinovic. “Visual Adjacency Multigraphs - a Novel Approach for a Web Page Classification.” Proceedings of the ECML/PKDD Workshop on Statistical Approaches to Web Mining, 2004
12. Mark P. Sinka, and David W. Corne, “Measuring Effectiveness of Text-Decorated HTML Tags in Web Document Clustering”, IADIS Conference on WWW/Internet, 2004
13. Makoto Tsukada, Takashi Washio, Hiroshi Motoda “Automatic Web-Page Classification by Using Machine Learning Methods”, Proceedings of the First Asia-Pacific Conference on Web Intelligence: Research and Development, 2001
14. Hwanjo Yu, Jiawei Han, Kevin Chen-Chuan Chang , “PEBL: Web page classification without negative examples”, IEEE Transactions on Knowledge and Data Engineering (TKDE), 2004
15. Min-Yen Kan, Hoang Oanh Nguyen Thi, “Fast webpage classification using URL Features”, ACM Conference on Information and Knowledge Management (CIKM 2005), 2005
16. Min-Yen Kan, “Web Page Classification without the Web Page”, World Wide Web Conference (WWW 2004), 2004
17. Allan, “Introduction to Topic Detection and Tracking”, Kluwer Information Retrieval, 2002
18. Radev, Jing, Sys, Tam, “Centroid-based summarization of multiple documents”, Information Processing and Management: An International Journal, Elsevier, 2004
19. M Saravanan, S Raman, B Ravindran, “A Probabilistic Approach to Multi-Document Summarization for Generating a Tiled Summary”, Proc of the 6th International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2005), 2005
20. Horacio Saggion, Dragomir Radev, Simone Teufel, Wai Lam, Stephanie M Strassel, “Developing infrastructure for the evaluation of Single and Multi-Document Summarization systems in a cross-lingual environment”, Proceedings of LREC-2002, 2002
21. Seki, Eguchi, Kando, “User-Focussed Multi-Document Summarization with Paragraph Clustering and Sentence-Type Filtering”, Working Notes of NTCIR-4, 2004
22. Deepak P & Jyothi John, “Identifying the subject of small, sparsely linked collections from a web community”, International Journal on Web Based Communities, Inderscience, 2004
23. Peter Willet, “Recent Trends in Hierarchical Document Clustering: A Critical Review”, Information Processing and Management: An International Journal, 1988
24. Ying Zhao, George Karypis, “Criterion Function for Document Clustering: Experiments and Analysis”, Dept. of CS, University of Minnesota, TR#01-40
25. Mandhani, Joshi, Kummamuru, “A Matrix Density Based Algorithm to HierarchicallyCoCluster Documents and Words”, World Wide Web Conference (WWW 2003), 2003
26. Bunke, Messmer, “Similarity Measures for Structured Representations”, Proc. Of the First European Workshop on Topics in Case Based Reasoning, 1993
27. Clifton, Cooley, Rennie, “TopCat: Data Mining for Topic Identification in a Text Corpus”, IEEE Transactions on Knowledge and Data Engineering (TKDE), 2004
28. Stein, Eissen, “Topic Identification: Framework and Application”, Proc of International Conference on Knowledge Management (I-KNOW 2004), 2004
29. Canvar, Tenkle, “N-Gram based text categorization”, Proc of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR 1994), 1994
30. Miao, Keselj, Milios, “Document clustering using character N-grams: a comparative evaluation with term-based and word-based clustering”, Proc. Of ACM Conference on Information and Knowledge Management (CIKM 2005), 2005
31. Vlado Keselj, Fuchun Peng, Nick Cercone, and Calvin Thomas, “N-gram based author profiles for authorship attribution”, Proc. Of the Annual Meeting of the

Pacific Association for Computational Linguistics, (PACLING 2003), 2003

32. T Abou-Assaleh, N Cercone, V Keselj, R Sweidan, "N-gram based detection of malicious code", Computer Software and Applications Conference (COMPSAC 2004), 2004
33. Jon M Klienberg, "Authoritative Sources in a Hyperlinked Environment", Journal of the ACM, Vol. 46, No. 5, 1999, pp. 604-632
34. Sergey Brin, Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Computer Networks and ISDN Systems, 1998
35. MacQueen, JB, "Some methods for classification and analysis of multivariate observations", Proc. Of the 5th Symposium on Math, Statistics and Propability, Berkeley, CA
36. Lehmann, EL and D'Abrera HJM, "Nonparametrics: Statistical Methods based on ranks", rev. ed., Englewood Cliffs, NJ, Prentice-Hall, 1998
37. Diaconis, P, 1988, "Group Representation in Probability and Statistics", IMS Lecture Series, Institute of Mathematical Statistics