

# CTMS: A Comparative Text Mining System

Peng Zang  
University of Illinois at Urbana-Champaign  
pengzang@uiuc.edu

ChengXiang Zhai  
University of Illinois at Urbana-Champaign  
czhai@uiuc.edu

## ABSTRACT

In many applications, there is often a need for comparing multiple text collections to find commonalities and differences in topical themes, a task we refer to as *comparative text mining*. In this paper, we present a general comparative mining system (CTMS). The CTMS system takes any two collections of text and generates a list of cross-collection themes and their associated individual collection-specific themes. The themes are linked to representative passages in each collection. The themes are represented as word distributions, and the underlying comparative mining algorithm is based on a probabilistic mixture model. The system carries out all the stages of text mining from data cleaning and preprocessing to the actual mining and post-processing, allowing users to perform comparative analysis between any two collections and navigate through the extracted theme space. This system can potentially be applied to a broad range of areas including opinion summarization, business intelligence, and summarization of text.

## Categories and Subject Descriptors

H.4.3 [Information System Applications]: Communications Applications—*comparative text browser*; H.3.7 [Information Storage and Retrieval]: Digital Libraries—*comparative document browsing, text mining*

## Keywords

comparative text mining, multi-document summarization, clustering

## 1. INTRODUCTION

In many applications, there is often a need for comparing multiple text collections to find commonalities and differences in topical themes, a task we refer to as *comparative text mining*. For example, a user trying to choose between buying a Dell laptop or an IBM one would be interested in comparing their reviews to see what has been said about them in common and what product specific things have been said about each laptop. Ideally we would want to be able to automatically generate something similar to table 1.

Consider another example. Suppose we have some recent news articles about the Iraq war, some originating from the US and some from Europe. It may be interesting to compare these two sets of news articles to see what is in common between these two groups of articles, which presumably represents the common opinions or some facts about the war. One may also be interested in determining what is unique to a particular news source.

Yet another example is comparison of course descriptions of computer science programs. Through the comparison, we can expect to find a common set of courses taught in all the programs, which can serve as an empirically defined core computer science curriculum.

Comparative text analysis is also useful for the researcher. For example, we can imagine that the publications returned when searching for a topic can be partitioned according to time period. We can then compare the publications in different periods to reveal their commonalities and each's unique content. Another use for the researcher might be for novelty detection. When examining any particular document, the researcher can compare the document in question with information already collected to see if any new information of interest exists in the document.

In all these examples, our problem is to extract common and unique themes from a set of comparable text collections. The CTMS is developed to provide support for such comparative text analysis. The CTMS system takes any two collections of text and generates a list of cross-collection themes and their associated individual collection-specific themes. The themes are linked to representative passages in each collection. The themes themselves are represented as word distributions, and the underlying comparative mining algorithm is based on a probabilistic mixture model. The system carries out all the stages of text mining from data cleaning and preprocessing to the actual mining and post-processing, allowing a user to perform comparative analysis between any two collections and navigate through the extracted theme space. This system can potentially be applied to a broad range of areas including opinion summarization, business intelligence, and summarization of text. We evaluate the tool on two different data sets (course descriptions and laptop reviews). The results show that CTMS can be very useful as a tool for *interactively* exploring and browsing comparable text collections. It is also effective for extracting common and unique themes from these collections, which can be directly useful in text mining applications.

The rest of this paper is organized in the following manner. In the following section, we will formally outline the problem definition. We then examine methods for solving the problem before

**Table 1: An ideal comparative summary**

Subtopics	Dell Latitude 800	IBM T40
Battery life	approximate battery life of three hours fresh battery can get through a movie about three to three-and-a-half hours	approx. 5.0 hours with standard battery 4 hours of life with light usage nine-cell battery that delivers superb battery life
Display	king-size 1920x1200 screen 1920x1200 WUXGA wide aspect display	1400x1050 SXGA display 14.1- inch display
...	...	...

discussing our system architecture. Finally, we will examine some sample results, briefly compare with some related work, and conclude.

## 2. THE COMPARATIVE TEXT MINING PROBLEM

While it would be ideal if we could generate a table like Table 1, this would require in-depth natural language understanding, which is impossible given the current state of the art in natural language processing. We thus first define an easier goal where we attempt to capture various kinds of word/term distributions shown in Table 2. This goal can be thought of as a kind of “stepping stone” towards our ideal.

**Table 2: Distribution-based Comparative Analysis**

Subtopic	Dell Latitude 800	IBM T40
<b>battery 0.129</b>	three 0.150	5 0.125
<b>hour 0.080</b>	hours 0.130	long 0.100
<b>life 0.060</b>	fresh 0.076	hours 0.089
...	...	...
<b>display 0.134</b>	1920x1200 0.125	14.1in 0.144
<b>screen 0.120</b>	wide 0.112	SXG 0.115
...	...	...

The distributions approximate and provide a platform for further processing to take us the rest of the way to achieving the ideal mining result shown in Table 1. For example, once we have these word distributions, we can take the top few keywords from the first column (which corresponds to common themes) as a brief description (a label) for the theme, we may have “battery hour” to label the first theme and “display screen” to label the second theme. We may also use the collection-specific word distributions to identify the passages in the corresponding collection that are about this particular theme. For example, we may use the word distribution corresponding to Dell’s battery life to scan the reviews for Dell laptops and identify which part of the reviews is about Dell’s battery life. In CTMS, we support all these functions, making it useful in two ways: (1) as a tool for supporting a user to interactively explore and browse comparable collections; (2) as a tool for generating hidden semantic themes that can characterize the commonality and difference of the two collections.

The reason why such word distributions are somehow easier to extract/generate than the exact answers is because the distributions allow us to keep rather than resolve the uncertainties. However, comparative mining (even as in the realistic setup) is challenging in several ways: (1) It is a completely unsupervised learning task; no training data is available. (Ironically it is for this same reason that comparative mining can be very useful for many different purposes – it makes minimum assumptions about the collections and

in principle we can compare any arbitrary partition of text.) (2) We need to identify themes *across* different collections, which is more challenging than identifying topic themes in a single collection. (3) The task involves a discrimination component – for each discovered theme, we also want to identify the unique information specific to each collection in addition to the information common to all the collections. Such a discrimination task is difficult given that we do not have training data. Later we will describe how we can learn all these distributions simultaneously by fitting a mixture probabilistic model to all the text data [15].

## 3. COMPARATIVE TEXT MINING METHODS

In order to develop a complete comparative text mining system, we need to solve three technical challenges:

1. **Theme mining and extraction:** Mine/extract the common and collection-specific themes (i.e., word distributions) from the raw collections.
2. **Theme passage extraction:** Segment the original text into passages and categorize the passages into their appropriate themes (either the common or one of the collection specific themes).
3. **Theme summarization:** Summarize all the passages that belong to a theme and generate a concise description suitable for conveying what the theme is.

We now describe how we solve each of these challenges.

### 3.1 Theme mining and extraction

We use the cross-collection mixture model proposed in [15] to mine and extract the common and unique (i.e., collection-specific) themes from comparable text collections. In a way, what we want is some special way of clustering the text data, and the idea of the cross-collection mixture model is to assume a generative model for the text data so that the task of clustering can be reduced to model parameter estimation via an EM algorithm; the parameters correspond to the word distributions. The generative model is a mixture model of various unigram language models (i.e., word distributions) each modeling a different thematic aspect of the text data.

The generative model considers a general background model,  $k$  latent common themes and a set of  $k$  collection-specific themes for each collection as shown in (illustrated in Figure 1) [15]. The sampling distribution for a word in document  $d$  from collection  $C_i$  is collection-specific. Specifically, it assumes each word is generated independently in the following manner:

$$p_d(w|C_i) = (1 - \lambda_B) \sum_{j=1}^k [\pi_{d,j} (\lambda_C p(w|\theta_j) + (1 - \lambda_C) p(w|\theta_{j,i}))] + \lambda_B p(w|\theta_B)$$

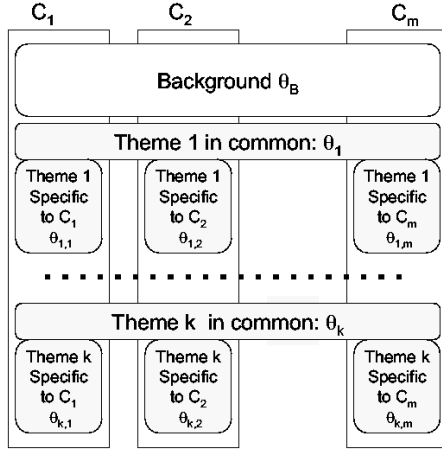


Figure 1: The Cross-Collection Mixture Model

Intuitively, when we “generate” a word, we first decide whether to use the background model  $\theta_B$  according to  $\lambda_B$ . If we decide not to use  $\theta_B$ , then we need to decide which theme to use; this is controlled by  $\pi_{d,j}$ , the probability of using theme  $j$  when generating words in  $d$ . Finally, once we decide which theme to use, we determine whether we should use the common theme model or the collection-specific theme model, and this is controlled by  $\lambda_C$ , the probability of using the common model.[15] This can be seen graphically in Figure 2.

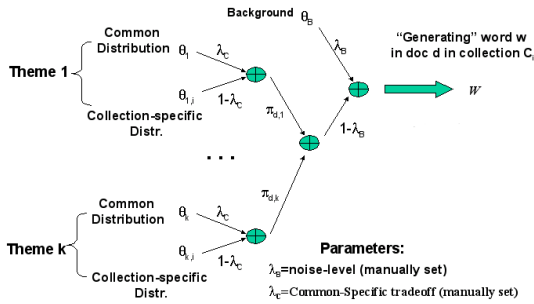


Figure 2: Generating word  $w$  in document  $d$  in collection  $c_i$ .

By assuming this generative model, clustering can become a simple process of model estimation via an EM algorithm. In particular all the language models which make up the themes, common themes and collection-specific themes are estimated as well as  $\pi_{d,j}$  – the probability of a particular theme occurring – for all  $j$ . In [15], this model is shown to be effective for extracting common and unique themes from comparative collections, though the common themes tend to have higher quality than the unique themes, perhaps due to the sparseness of data for accurately estimating specific themes.

While the theme word distributions extracted are themselves informative, their utility is limited. In particular, they are not very effective for conveying the similarities and differences between two collections of documents.

How do we better make use of such distributions? One possibil-

ity is to find documents from the original corpus that match each theme and push those documents to users as supplementary information. However, documents usually discuss more than one subtopic/theme, which makes ranking of relevant documents inaccurate and also makes it necessary for users to sift through a whole document to find the relevant passages.

To really make the themes useful we must extract relevant passages from the original text for them and summarize it. We now present our methods for extracting such theme passages and summarizing them.

### 3.2 Theme passage extraction

Our task is to segment the text into mono-themed passages and categorize them to the appropriate theme cluster. Thus we decided to use the learned theme word distributions to construct a Naive Bayes classifier [9] to classify and find the best passages.

We assume sentences are mono-themed and hence for classification purposes, atomic. Passage extraction and classification is then a process of classifying each sentence of each document of the original text. For each sentence  $S$ , we classify it into its appropriate cluster via the conditional probability that the sentence from collection  $i$  was generated from theme cluster  $T_j$ , i.e.,  $P(T_j|S, C_i)$ . In other words the sentence goes to the cluster given by  $\text{argmax}_j P(T_j|S, C_i)$ .  $P(T_j|S, C_i)$  is given by

$$\begin{aligned} P(T_j|S, C_i) &= \frac{P(S|T_j, C_i)P(T_j|C_i)}{P(S|C_i)} \\ &\propto \log P(S|T_j, C_i) + \log P(T_j|C_i) \end{aligned}$$

where  $p(T_j|C_i)$  is computed as  $\frac{\sum_{d \in C_i} p(T_j|d)}{\sum_j \sum_{d \in C_i} p(T_j|d)}$  and according to the model in [15],  $P(T_j|S, C_i)$  can be computed as

$$P(S|T_j, C_i) = \prod_{w \in S} [\lambda_B p(w|\theta_B) + (1 - \lambda_B)(\lambda_C p(w|\theta_j) + (1 - \lambda_C)p(w|\theta_{j,i}))]$$

In addition to deciding what theme cluster the sentence belongs to we must also decide which model – whether the common or its collection  $i$  model – has been used to generate it. For this we must compare the the likelihood that the common model is the sole source of generation (aside from the background model), versus the likelihood that the collection-specific model is the sole source of generation for the sentence (aside from the background model).

Given that we know sentence  $S$  belongs to cluster  $T_j$ , the probability that  $S$  is generated using the common theme model  $\theta_j$  is

$$\begin{aligned} P(\theta_j|S, T_j, C_i) &= \frac{P(S|\theta_j, T_j, C_i)P(\theta_j|T_j, C_i)}{P(S|T_j, C_i)} \\ &\propto p(S|\theta_j, T_j, C_i)P(\theta_j|T_j, C_i) \\ &\propto \prod_{w \in S} [\lambda_B p(w|\theta_B) + (1 - \lambda_B)p(w|\theta_j)]\lambda_C \end{aligned}$$

The probability that  $S$  is generated using the collection-specific theme  $\theta_{j,i}$  is

$$\begin{aligned} P(\theta_{j,i}|S, T_j, C_i) &= \frac{P(S|\theta_{j,i}, T_j, C_i)P(\theta_{j,i}|T_j, C_i)}{P(S|T_j, C_i)} \\ &\propto p(S|\theta_{j,i}, T_j, C_i)P(\theta_{j,i}|T_j, C_i) \\ &\propto \prod_{w \in S} [\lambda_B p(w|\theta_B) + (1 - \lambda_B)p(w|\theta_{j,i})](1 - \lambda_C) \end{aligned}$$

We can then compare  $p(\theta_j|S, T_j, C_i)$  with  $p(\theta_{j,i}|S, T_j, C_i)$  and assign  $S$  to the common theme model if  $p(\theta_j|S, T_j, C_i) > p(\theta_{j,i}|S, T_j, C_i)$  and to the collection-specific theme model otherwise.

Once the sentences are classified in their appropriate themes, passages are formed simply from consecutive sentences. We can see this whole process in the pseudocode in Figure 3

**Figure 3: Passage Extraction and Classification Pseudocode**

```

for each Collection {
  for each Document in collection {
    for each Sentence in document {
      topCluster = max cluster by
        P(cluster|sentence, collection);
      topCluster.add(sentence);

      if( P(topCluster.CommonModel|sentence, collection) >
        P(topCluster.collectionModel|sentence, collec) )
      {
        topCluster.CommonModel.add(sentence);
      }else{
        topCluster.collectionModel.add(sentence);
      }
    }
  }
}

```

The end result of this process are sentences classified into various theme clusters and passages with a coherent theme can be easily extracted from consecutive sentences of the same cluster. In exchange for the simplicity and speed of this method we expose ourselves to one weakness. That is, unlike in HMM, context is not taken into account in our method and sentences are assumed to be independently generated. As a result,  $P(T_j|S, C_i)$  is in no way affected by the classification of the previous sentence.

### 3.3 Theme Summarization

The previous step has classified sentences into their appropriate theme clusters and the appropriate models within their cluster, resulting in each theme cluster and model holding a set of belonging passages.

To be useful to users however, we need to summarize/organize these sets of passages so that we can present text that will quickly and intuitively convey the meaning of the model and its contents. If we treat the passages as documents, we quickly see that this is a traditional multi-document summarization problem.

While there are many approaches to multi-document summarization, our situation is unique because we know the distribution from which the documents are generated. Hence with little loss of generality we can find representative passages by ranking the passages via maximum likelihood and keeping the top  $k$ . This process can be very fast because it can be done during the passage extraction process adding only constant time.

For finding cluster summaries we rank on  $\log P(S|T_j, C_i) - \log P(S|\theta_B)$  which represents how much the cluster's common and collection specific models can explain the passage beyond the base background model.

For finding the summaries for the word distributions that make up the clusters we use  $\log P(S|\theta_j, T_j, C_i)$  for the common and  $\log P(S|\theta_{j,i}, T_j, C_i)$  for the collection-specific models.

As we are ranking passages of differing length, length normalization is used. This also provides us a way to control the desired length of the representative passages retrieved. Length normalization is done by dividing the probability scores by the length of the sentence. A small penalty to the probability score is added prior to the division to allow biasing toward either shorter or longer length passages.

## 4. CTMS SYSTEM DESIGN

CTMS is a complete system for doing automatic multi-collection compare and contrast analysis. It consists of two parts. The GUI front end is responsible for setting up mining sessions, running sessions and browsing results. The back end is responsible for doing the actual processing from cleaning to mining.

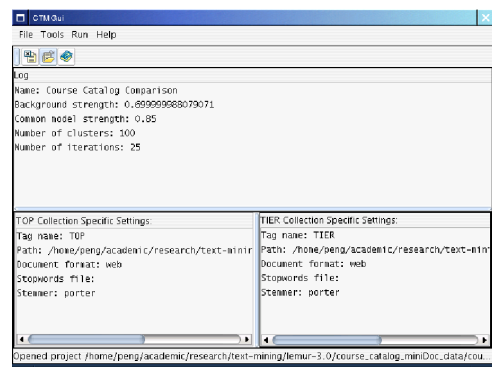
### 4.1 GUI Front-end

The GUI front-end is designed to integrate all the separate components needed to do compare and contrast of different collections – from collecting mining options and data locations to running the actual steps of the mining process like data cleaning, indexing and results analysis.

The front-end consists of three main parts: a primary frame which is the initial startup window, a results frame for browsing mining results, and a document viewer frame to allow users to pull up the original document.

#### 4.1.1 Primary frame

The primary frame is for setting up projects, manipulating project settings and starting the mining process. The primary frame consists of two portions. The top half of the frame is a panel dedicated to serving as a console/logging facility so users can see the mining process as it undergoes its various stages. Initially this panel contains information on the project loaded such as the name of the project and various general project options such as the number of clusters, or strength of the background model. The second half of the frame is comprised of two panels displaying the collection specific options of the two collections in question. See Figure 4.



**Figure 4: Comparative Text Miner's Initial Screen Upon Loading a Project**

When a user has started the mining process, the console panel (top) will show the mining progress.

All actions' primary access is via the menu bar of the primary frame. A few commonly used actions have buttons on the tool bar

for convenience and provide a secondary access method. The primary frame is relatively self-explanatory. Some actions such as starting a new project will bring up dialogs for user input.

### 4.1.2 Results browser

The results browser frame is responsible for displaying the results of the comparative text mining process and shows a number of clusters, each containing a common language model representing the common theme and  $k$  collection-specific language models representing the unique aspects of each collection with respect to the common theme. To accommodate such, the results browser frame is designed with a split-pane. See Figure 5. The left side is a tree structure of the clusters and its sub-models, provided for easy navigation of the results. The right side is used to show a summary of the selected cluster or language model. Since the summary of a cluster or language model is a list of some of its most representative passages from the original text, the right side is in the form of a list.

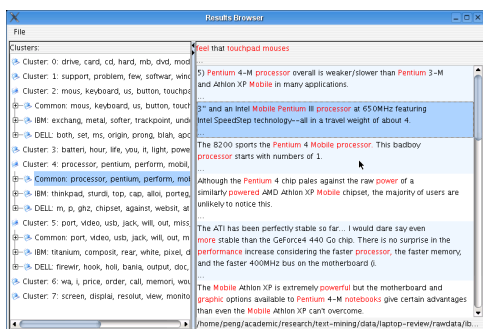


Figure 5: CTMS: Results Browser

A few characteristics of the results browser are worth mentioning. Words from the summary passages that are in the top  $n$  words of the corresponding language model are highlighted to guide user eyes to the most relevant sections of the passage. For each cluster a short three or four word blurb is presented at the top of the right pane as a quick fragment for insight into the cluster. In addition, as users browse through the various passages, highlighting a particular passage will update the status bar at the bottom with the location of the original file from which the passage was extracted.

### 4.1.3 Document viewer

The document viewer is where extracted passages that prove interesting enough in the results browser to pursue is shown in detail. When users are interested in pursuing an extracted passage they may double-click on it to bring up the original text from which the passage was extracted. The original text is brought up in a separate window called the Document Viewer or DocViewer. The DocViewer allows users to see the extracted passage in its original context. Similar to the results browser, common and collection specific words are highlighted to help users quickly locate the similarities and differences of the document. See Figure 6

In the above example, we can see that the common theme is computer organization and the collection specific uniqueness is that this university's class on computer organization has a strong assembler language component.

DocViewer is modeled much like a simplified web browser. It has backwards and forwards buttons to allow back and forth naviga-

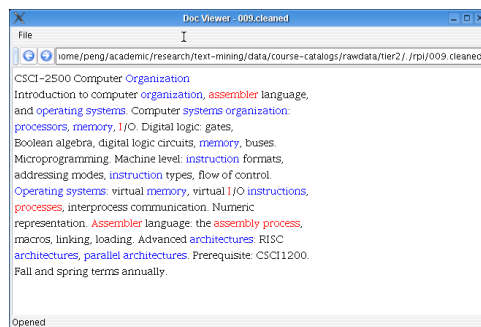


Figure 6: CTMS: Document Viewer

tion. It also has a navigation bar so users can directly open any file or webpage users may wish to view. This is to facilitate users' exploration of the data should something in the particular document be of interest and warrants further probing.

## 4.2 Back-end

The back-end is where the actual mining occurs. It is divided into four basic steps:

1. Data cleaning and preprocessing: In this stage, documents are (1) converted into plain text, (2) numbers and extraneous symbols removed, (3) documents are optionally stop listed stemmed. The result is that each document is converted into a list of words one per line.
2. Indexing. This stage is responsible for generating an index of the documents and words so that mining analysis can be done efficiently.
3. Mining. The cross-collection mixture model clustering is performed here. The result is a list of the clusters, their probabilities and their models.
4. Post-processing. Here we return to the original text and extract passages for each model and cluster so that users are presented with representative passages for them instead of word lists.

Mining sessions are setup by users through the GUI but are run by the back end. When mining begins, the first three steps are initially run. Post-processing is done on-the-fly when users explore the results. This allows flexibility in how results are viewed by letting users to vary characteristics such as the passage length, number of representative passages to display and any ranking adjustments necessary.

## 5. EVALUATION

We identify four evaluations of interest.

1. Holistic evaluation of the tool's effectiveness in aiding research tasks via user testing. Due to time constraints this evaluation could not be performed.
2. Evaluation of the clustering algorithm's ability to discover latent common themes across collections and collection-specific differences along those themes. This evaluated has been done

via comparison with a simple mixture model clustering method in [15]. Results form that comparison show our clustering algorithm is significantly more effective at discovering latent common themes. The extent of our evaluation in this paper then is in discussion of some sample results. They show aspects of the effectiveness of the algorithm not addressed in previous evaluations.

3. Evaluation of the passage extraction and classification algorithm’s ability to find representative passages for clusters and models. This evaluation is accomplished via comparison the the extracted passages with the corresponding original language model.
4. Evaluation of the effectiveness of cluster summaries at conveying the meaning of the cluster. Ideally this evaluation would also be done via user testing. Forgoing that, our evaluation is limited to discussion of some sample results.

We performed experiments on two datasets. The first dataset is a comparison of computer science course catalogs of the top ten ranked CS universities with a random sample of ten lower ranked universities the authors were familiar with.. The second dataset is a comparison of reviews for two laptops, a Dell 8200 Inspiron and an IBM T23.

For each dataset we compare the original unigram language model results with with our passage summaries version. We also discuss the overall effectiveness of the tool, highlighting the clustering algorithm’s and summary form’s role.

## 5.1 Course Catalog Experiments

To generate this dataset, we downloaded the course descriptions for all the computer science classes of the top ten ranked computer science departments. This serves as our first collection named TOP. The second collection consists of course descriptions for all the computer science classes of ten other randomly selected CS departments. Some sample results are provided below. We will first examine the clusters as a whole before discussing individual clusters in detail.

Examination of the top clusters (see Table 3) show our tool is able to extract common themes across collections well. The clusters are coherent, aligning on subject areas and each is able to identify an independent subtopic with no collisions. They are able to convey to users the top classes common to all university CS programs. For the researcher trying to analyze or develop a university’s course offerings this information is invaluable.

Cluster sizes provide us with another insight. Cluster sizes are spread across the course catalogs of twenty universities so the top cluster size of 86 suggests that on average, CS programs offer four classes on computer systems organization. Examining the top clusters suggests that more classes are offered on fundamental computer science topics such as systems, numerical methods and algorithms than the more advanced research areas such as artificial intelligence, distributed systems or HCI. On average only 2.5 classes are offered for any particular advanced area. These results while initially surprising, make sense. While a particular university may be strong in a few research areas and offer a variety of classes in those areas, *all* universities offer the fundamental computer science classes. Hence we should expect higher averages for those core classes. In fact, a cluster’s size can provide us an indication of how

**Table 3: Top clusters found (brief)**

Size (docs)	Subject
86	Computer systems organization
72	Numerical methods
70	Analysis of algorithms
65	Software engineering
59	Formal models of computation
53	Independent studies
50	Programming languages
50	Artificial intelligence
49	Database systems
49	Distributed systems
45	Networking
44	HCI
41	Data structures

basic or fundamental a subject area is. Alternatively, for advanced research areas, cluster sizes show how popular that research area is across universities. This is useful for spotting under-developed research areas or alternatively, “hot topics”. If old course catalogs were available, our tool would be able to compare and contrast how course offerings have shifted over time. It would show us how the focus of CS has changed and allow tracking of “hot topics” and their development.

We now turn our analysis to the individual clusters. We show several clusters in detail. For each cluster, we provide for all its models both the original word list and a few of the top passages our method retrieved. Retrieved passages are positioned directly beneath the word list for easy comparison.

From these results we can see that our passage extraction is effective. Our algorithm is able to identify and extract the relevant course descriptions for the cluster’s subject area. More importantly for the CTM system, the passages provide much more depth to our understanding of the cluster. They also serve the additional purpose of allowing users a peek at some of the courses. Double-clicking on a passage will present the originating document with relevant words highlighted. This combination enhances the results browsing experience, enabling users to leverage the mined information. As a scanning aid, top words from the language model are also presented to facilitate fast recognition of general cluster themes.

Examining the cluster contents themselves, we see our hypothesis that the TOP collection would have more advanced courses is verified. More often than not, the TOP collection would have many more advanced classes. The TIER collection classes in contrast, tend to be more introductory or more focused on practical applications. For example in Sample Cluster 1 (see Table 4), we see the TOP collection has classes covering “Modular kernels” and “kernel/user abstraction weakening”. In contrast the corresponding TIER collection covers more basic principles such as “datapath, control, pipelining” and “Files, directories. Input/output, buffering.” We see similar behavior in Sample Cluster 2 (see Table 5). The TOP collection covers self-healing systems, medical consulting and data mining. The TIER collection, while having a few advanced classes also focuses on practical work experience.

Sample Cluster 3 (see Table 6) however shows us a unique observation. In this example, the TIER collection model is unrelated to the common model. The cluster’s common theme is natural language processing but the TIER model is on visual programming and operating systems. The TIER model is suppose to highlight

**Table 4: Course Data. Sample Cluster 1**

Common Model	parallel, system, memori, architectur, oper, processor, organ, manag, instruct, pipelin
	Pipelining, vector processors, cache memory, high bandwidth memory design, virtual memory, input and output.
	Operating systems: virtual memory, virtual I/O instructions, processes, interprocess communication
	Memory organization with cache, virtual memory.
Top10 Model	machin, virtual, synchron, messag, protect, implement, multithread, perform, clinic, kernel
	Fundamental design: naming, synchronization, latency, and bandwidth.
	Operating systems, monolithic and microkernels, virtual machines. Naming, memory management, segmentation, paging, and virtual memory.
	Modular kernels, kernel/user abstraction weakening. Naming, distributed security policy, resource management.
Tier2 Model	architectur, oper, assembl, system, file, processor, storag, process, i, concept
	datapath, control (hardwired, microprogrammed), pipelining, input/output.
	Files, directories. Input/output, buffering.
	Assembler language: the assembly process, macros, linking, loading.

**Table 5: Course Data. Sample Cluster 2**

Common Model	learn, machin, decis, method, adapt, statist, neural, bayesian, cluster, classif
	Computer Learning and Human Information Processing: Techniques of computer learning...
	Machine Learning Principles, techniques, and algorithms in machine learning from the point of view of statistical inference;
	Foundations of Machine Learning Robert Schapire Introduces the mathematical foundations of machine learning, including theoretical models of machine learning, and the design and analysis of learning algorithms.
Top10 Model	mine, we, consult, analyz, infer, there, will, nanotechnolog, make, medic
	Method of Evaluation: Grading will be based on presentations in class and a report covering a specific aspect of self-healing systems and medical consulting, determination of cost-optimal classification rules, inferential information systems, and computer vision.
	Course will cover: introduction, data warehouse and OLAP technology for data mining, data preprocessing, primitives, languages, system architectures for data mining ...
Tier2 Model	learn, field, x, practic, reinforc, experi, capabl, issp, have, semest
	Professional Practice 0 Full time practical computer science work experience, which is related to the student field of study, and is of a semester's duration.
	Theory of Learning Algorithms (ISSP 3520) Description. The purpose of this course is to present fundamental results regarding the learning capabilities of computers...
	Learning techniques and methods developed by researchers in this field have been successfully applied to a variety of learning tasks in a broad range of areas ...

**Table 6: Course Data. Sample Cluster 3**

Common Model	cognit, languag, natur, process, of, linguist, human, semant, psycholog, the
	Natural Language and the Computer Representation of Knowledge Relationship between computer representation of knowledge and the structure of natural language.
	Computational Linguistics: Introduction to computational models of understanding natural languages.
	Introduction to Natural Language Processing. (3) Principles of computational linguistics, formal syntax, and semantics, as applied to the design of software...
Top10 Model	genet, think, ling, analog, question, understand, oop, brain, focu, same
	Genetic Algorithms and Genetic Programming–Genetic algorithm is a domain-independent algorithm for search, optimization, and machine learning. ...
	Much of the course will focus on the Neural Theory of Language (NTL), which seeks to answer these questions in terms of architecture and mechanism, using models and simulations of language and learning phenomena.
	This is a course on the current status of interdisciplinary studies that seeks to answer the following questions: (1) How is it possible for the human brain, which is a highly structured network of neurons, to think, learn, use, and understand language?
Tier2 Model	visual, oper, kernel, system, intern, icon, i, practic, driver, histor
	Visual Language & Visual Programming. Fundamentals of formal language theory, iconic and symbolic representation, formal theory of iconic systems, icon operators ...
	OPERATING SYSTEMS INTERNALS This course uses an operating system (such as UNIX) as an example to teach the internal workings of Operating Systems.
	Rather than using the complex syntax of a production language such as C or C++, this course will use Visual Basic.

**Table 7: Top clusters found (brief)**

Size (docs)	Subject
92	Drive bays
87	Installation/technical support
86	Input devices
84	Battery life
79	Processor speeds/power
71	I/O ports

differences *along* the common theme. Why then is the TIER model on a completely different topic?

We see two reasons – both are artifacts of our clustering method. First, our assumed generative model forces each common theme to have  $m$  corresponding models, one for each collection. This means that for any particular subtopic, we assume every collection has something unique to contribute. This is not always the case unfortunately. In such cases the clustering algorithm will pull weak clusters into the collection-specific model. Second, our assumed generative model provides little enforcement of alignment. That is, our model does not enforce content sharing between the collection-specific models and the common model. These two factors combined means that if the collection is weak on a particular subtopic or has significant unique content not tied to any common theme, we will see collection-specific models that do not match the common theme. Hence in our particular example in Table 6, we see that the TIER collection has little content on NLP but instead has content on visual programming. Visual programming is most likely content that uniquely belongs to the TIER collection and hence no common theme for it exists, forcing it to be shunted into a collection-specific model that’s tied to an unrelated common theme. The fact that it picked the NLP common theme further indicates that the TIER collection has little content on NLP. This affirms our hypothesis.

## 5.2 Laptop Review Dataset

For this dataset we collected reviews on two different laptops for comparison, the Dell Inspiron 8200 and the IBM T20. Roughly three hundred reviews on each were collected. Again we see from the top clusters (see Table 7) that CTMS is very effective at identifying the common themes.

Examining the clusters in detail show that the passage extraction is similarly accurate. However, it’s effectiveness is more evident in this example. In our previous experiment, the passages only provided extra depth to our understanding because the top words of the language model were surprisingly readable. In contrast, the word lists in this case are often confusing and do not convey the cluster’s content. For example, the words “large, skip, list and slimmer” (see Table 8) make little sense for IBM’s collection-specific model. A quick scan of the extracted passages however, will reveal what the words by themselves could not – that the model is on features of the IBM display. Not only are the passages often required to understand the cluster, but because we want to compare the two products by learning the differences, we need to be able to read them ourselves. The extracted passages are key for this. They point readers directly to the relevant passages where unique elements of the products are discussed, enabling readers to quickly learn the differences between the products and make an informed decision. This is not solely applicable to product comparisons, but to any comparisons where learning and understanding of the differences are desired.

In Sample Cluster 2 (see Table 9), we see another case of poor

model alignment. In this case, there are some comments on the IBM model’s light weight and accessory lights that do not have their own cluster because none of the DELL reviews mentioned accessories in detail. Similarly, the DELL model has some specific issues that are mentioned only rarely and not shared in terms of topic with any IBM reviews. The result is that they have been shunted into the battery cluster. We note that in these cases where there is model misalignment a division of the common model passages by collection can provide results close to what we desire.

Overall we feel that the results combined with the flexible “results browser” enable users to quickly identify and learn the differences between the two laptops. While it cannot yet provide the fully automated compare and contrast analysis, it already is a valuable tool for aiding user analyses.

## 6. RELATED WORK

Some work has already been done on comparative text mining. Besides the work done in [15] that we are extending, Mani et. al did some early work [6] that is very similar. Both aim to automatically summarize similarities and differences among related documents. The main difference lies in approach as they use a spreading activation scheme. [6] is also focused more on a retrieval oriented compare and contrast summarization as their work is query oriented. We see our method as more discovery oriented better suited for exploration.

Work on multi-document summarization [13, 3] is often very similar to comparative text mining. In multi-document summarization care must be taken to avoid redundancy and yet ensure coverage. Such goals require at least rudimentary comparative analysis. In work such as [8] the entire approach for summarization is a comparative one.

Despite the many similarities and overlap comparative text mining has with the more general multi-document summarization, our work is unique because it takes into account intrinsic differences between collections and recognizes those collection boundaries. For example one would not want to use multi-document summarization on laptop reviews of two different laptops because we’re interested in how they are different – we want to summarize along their boundaries.

Comparative text mining also often overlap in methods with multi-document summarization. For example passage extraction and summarization methods are necessary components of both types of work. Indeed such methods have been studied extensively themselves and innumerable comparisons can be made. Suffice to say however, that while we use relatively well known methods, what makes our process unique is its context. We do passage extraction, classification and summarization with an a priori defined language model.

## 7. CONCLUSIONS AND FUTURE WORK

Uses of automated compare and contrast summarization are wide, from aiding research to business intelligence. We developed CTMS for these purposes. CTMS is a complete system for comparative text mining from initial data cleaning to results browsing. It features:

1. Identification of latent common themes across collections and collection-specific differences along those themes.

**Table 8: Laptop Review Data. Sample Cluster 1**

Common Model	screen, displai, resolut, view, monitor, playback, uxga, x, notebook, lcd
	The latest screen technology (the UltraSharp UXGA TFT screen) gives the same resolution as the older UXGA 15inch screen but the brightness of the screen is extremely noticable.
	With the UltraSharp screen, you can view your laptop screen at a wider angle than the basic UXGA screen;
	Most LCD screens look best when viewed straight on, but the Ultra Sharp gives you 170 degrees of horizontal viewing, added brightness and increased contrast for these viewing angles.
IBM Model	larg, skip, list, slimmer, arm, auction, crisp, thinner, plasma, deeper
	It also renders deep, rich colors.
	This constitutes for problems such as black spots and the multiple lucent bars during powersaving.
	Basically this is a new micosilicon adapted version of the T10 that didn't get much critical acclaim a while back. However, some improvements have been made to the ancient Z-32 motherboard. In the new model the horrid TFT display has been replaced by a liquid plasma LCD screen with dual attachments.
DELL Model	uxga, inspiron, ultrasharp, sxga, angl, poor, radeon, eyesight, sit, ultra
	You have a choice of SXGA , UXGA or Enhanced UXGA (Dell UltraSharp).
	The Screen: The UltraSharp UXGA and plain vanilla UXGA screens are both 1600 x 1200 and present problems to older eyes.
	Fine details are maintained throughout with excellent colour reproduction and vivid colour displays.

**Table 9: Laptop Review Data. Sample Cluster 2**

Common Model	batteri, hour, life, you, it, light, power, if, time, do
	Battery life is fairly good. Approximately 3 hours on a fully charged battery.
	The battery life was great because I purchased dual batteries, thus giving me extra battery life (a total of about 4-5 hours depending on use).
	IBM says the battery should last around 4 hours.
IBM Model	weigh, pound, shut, chik, dsl, automat, lightweight, stabl, ac, lug
	working on the plane or in dim situations a nice light shines down onto the keyboard without flooding the screen with light.
	Lightweight even with the extras you are never to knackered from lugging it around.
	It's sleek, it's very durable, and it's really lightweight, I can throw it into any backpack or suitcase and I barely even notice the weight difference. With 700 mhz behind this baby, nothing is a pain to load.
DELL Model	optic, requir, hous, minut, latch, depend, under, cpu, bug, ran
	It ran Star Trek Voyager: Elite Force, without so much as a hiccup.
	The speed of your CPU should be in a slower state under lower CPU loads, and should INCREASE when something begins to utilize more processor resources. Currently, there's a BIG FAT BUG with this on the 8200's.
	On a single battery and an average load, the unit is capable of operating for roughly three hours before requiring to be attached to a permanent source of power or requiring battery recharge. This time should be most sufficient for most remote projects where the permanent power source is not available, but if you do require more battery life you can easily remove the floppy drive bay and convert it to a second battery bay for a dual battery capability that should extend your battery life to almost six hours of continuous use.

2. Automatic extraction of passages from the original corpus to illustrate the identified common themes and collection differences, which also serve as pointers to relevant sections to interest for readers.
3. Document viewer that presents passages in their full original context to enable user investigation.
4. GUI for doing the comparative mining and exploration of results by clusters, models and passages.

CTMS is based on combining the cross-collection mixture model clustering first introduced in [15] with post-processing and an exploration oriented GUI to bring to the fore extracted passages from the original text for compare and contrast investigation.

In our evaluation of CTMS we find that our algorithms for passage extraction, classification and summarization are generally very accurate. Comparison of cluster word lists and passages show agreement and highly ranked passages usually provide good context for understanding the cluster. Passages provide richer understanding of cluster contents and also are excellent primers – identifying interesting differences for users to explore. Evaluation of the clusters generated show CTMS is effective in its ability to identify common themes and collection-specific differences though there are occasional misalignments. Though not mature enough to be a complete automated compare and contrast tool, we feel it is invaluable as an aid for comparison analyses.

Our future work starts first with a revision of the model we use for clustering to address the misalignment previously seen. Cluster summarization also has planned revisions. We currently use top representative passages as the cluster summary, but we are interested in providing an additional “cluster abstract” via a summary generation approach.

## 8. REFERENCES

- [1] R. Barzilay and M. Elhadad. Using lexical chains for text summarization, 1997.
- [2] S. R. Das and M. Y. Chen. Yahoo! for amazon: Sentiment parsing from small talk on the web. *EFA 2001 Barcelona Meetings*, 2001.
- [3] J. Goldstein, V. Mittal, J. Carbonell, and J. Callan. Creating and evaluating multi-document sentence extract summaries. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 165–172. ACM Press, 2000.
- [4] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM Press, 2004.
- [5] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence*, pages 755–760. AAAI Press / The MIT Press, 2004.
- [6] I. Mani and E. Bloedorn. Summarizing similarities and differences among related documents. *Inf. Retr.*, 1(1-2):35–67, 1999.
- [7] I. Mani, D. House, G. Klein, L. Hirschman, T. Firmin, and B. Sundheim. The tipster summarization text summarization evaluation. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 77–85. Association for Computational Linguistics, 1999.
- [8] K. McKeown, J. Klavans, V. Hatzivassiloglou, R. Barzilay, and E. Eskin. Towards multidocument summarization by reformulation: Progress and prospects. In *Proceedings of AAAI/IAAI 1999*, pages 453–460.
- [9] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [10] M. Mitra, A. Singhal, and C. Buckley. Automatic text summarization by paragraph extraction.
- [11] A. Nadamoto and K. Tanaka. A comparative web browser (cwb) for browsing and comparing web pages. In *Proceedings of the twelfth international conference on World Wide Web*, pages 727–735. ACM Press, 2003.
- [12] D. R. Radev and K. R. McKeown. Generating natural language summaries from multiple on-line sources. *Comput. Linguist.*, 24(3):470–500, 1998.
- [13] G. Stein, T. Strzalkowski, B. G. Wise, and . Bagga. Evaluating summaries for multiple documents in an interactive environment. In *Proceedings of 2nd International Conference on Language Resources and Evaluation*, 2000.
- [14] S. Teufel and M. Moens. Sentence extraction as a classification task, 1997.
- [15] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 743–748. ACM Press, 2004.