A High-quality Digital Library Supporting Computing Education: The Ensemble Approach

Yinlin Chen

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science & Application

July 27, 2017
Blacksburg, Virginia 24061

Keywords: Educational Digital Library; ACM Classification System;
Amazon Mechanical Turk; Classification; Transfer learning; Active learning;
YouTube; SlideShare; Digital Library Service Quality; Cloud Computing

A High-quality Digital Library Supporting Computing Education: The Ensemble Approach

Yinlin Chen

ABSTRACT

Educational Digital Libraries (DLs) are complex information systems which are designed to support individuals' information needs and information seeking behavior. To have a broad impact on the communities in education and to serve for a long period, DLs need to structure and organize the resources in a way that facilitates the dissemination and the reuse of resources. Such a digital library should meet defined quality dimensions in the 5S (Societies, Scenarios, Spaces, Structures, Streams) framework - including completeness, consistency, efficiency, extensibility, and reliability - to ensure that a good quality DL is built.

In this research, we addressed both external and internal quality aspects of DLs. For internal qualities, we focused on completeness and consistency of the collection, catalog, and repository. We developed an application pipeline to acquire user-generated computing-related resources from YouTube and SlideShare for an educational DL. We applied machine learning techniques to transfer what we learned from the ACM Digital Library dataset. We built classifiers to catalog resources according to the ACM Computing Classification System from the two new domains that were evaluated using Amazon Mechanical Turk. For external qualities, we focused on efficiency, scalability, and reliability in DL services. We proposed cloud-based designs and applications to ensure and improve these qualities in DL services using cloud computing. The experimental results show that our proposed methods are promising for enhancing and enriching an educational digital library.

This work received support from ACM, as well as the National Science Foundation under Grant Numbers DUE-0836940, DUE-0937863, and DUE-0840719, and IMLS LG-71-16-0037-16.

A High-quality Digital Library Supporting Computing Education: The Ensemble Approach

Yinlin Chen

GENERAL AUDIENCE ABSTRACT

Educational Digital Libraries (DLs) are designed to serve users finding educational materials. To have a broad impact on the communities in education for a long period, DLs need to structure and organize the resources in a way that facilitates their dissemination and reuse. Such a digital library should be built on a well-defined framework to ensure that the services it provides are of good quality.

In this research, we focused on the quality aspects of DLs. We developed an application pipeline to acquire resources contributed by the users from YouTube and SlideShare for an educational DL. We applied machine learning techniques to build classifiers in order to catalog DL collections using a uniform classification system: the ACM Computing Classification System. We also used Amazon Mechanical Turk to evaluate the classifier's prediction result and used the outcome to improve classifier performance. To ensure efficiency, scalability, and reliability in DL services, we proposed cloud-based designs and applications to enhance DL services. The experimental results show that our proposed methods are promising for enhancing and enriching an educational digital library.

This work received support from ACM, as well as the National Science Foundation under Grant Numbers DUE-0836940, DUE-0937863, and DUE-0840719, and IMLS LG-71-16-0037-16.

Acknowledgments

It has been a long journey; I feel so grateful that I have had the opportunity to work with my wonderful advisor, Dr. Edward A. Fox. Without him, I would never have been able to finish my Ph.D. study. I am so thankful for his guidance, support, and inspiration during these years. He is my mentor and role model; he taught me many things, not only in research, but also about life experiences. I cannot thank him enough.

I would like to thank my committee members: Dr. Weiguo Fan, Dr. Ricardo Da Silva Torres, Dr. Christopher North, and Dr. Sanmay Das for their assistance and valuable feedback.

I would like to thank Dr. Lillian ("Boots") Cassel for her support through the Ensemble project, and all the professors and students I have worked with during these years. It was a remarkable research experience in my life.

I would like to thank the Association for Computing Machinery (ACM) for sharing the metadata from the ACM Digital Library for this research. This research work received support from the National Science Foundation (NSF) and the Institute of Museum and Library Services (IMLS) under Grant Numbers DUE-0836940, DUE-0937863, DUE-0840719, and IMLS LG-71-16-0037-16.

I would like to thank Uma Murthy, Monika Akbar, and Seungwon Yang who worked with me on research projects during these years. I also thank Susan Marion for her assistance, and my colleagues at the Digital Library Research Laboratory, for their feedback and friendship throughout my study.

I would like to thank my parents for their endless support and encouragement throughout my life. I am thankful for my two lovely children, Nathan and Nina, who always bring pleasure to me and give me strength. Finally, I would like especially to thank my wife, Chungwen Hsu, for her sacrifice, patience, and understanding during all these years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital Libraries (DLs) are complex information systems that aim to fulfill users' information needs and information seeking behavior. Digital librarians could follow formal reference models, such as the DELOS Digital Library Reference Model [26] or the Streams, Structures, Spaces, Scenarios, and Societies (5S) formal framework [60] [55] to establish fully functional DLs. [99] proposed that particular DLs would be extensions from a minimal DL under the 5S framework which should have key components including: catalog, collection, digital object, metadata specification, repository, and services. To evaluate the quality of DLs, there is 5SQual [98], a tool to assess key components and measure the quality of DLs automatically. Although we can build individual DLs by adapting these models, frameworks, and tools, DLs also need to have key content collections in order to serve their users; it is hard to prove usefulness with limited records and resources. Clearly, DLs that are only for small focused groups, with few particular topics of interest, are unlikely to have a broad impact.

Many DLs aggregate resources from other content providers, leading to challenges with metadata management and metadata integration problems. Metadata quality control is another challenging task; when DLs harvest more and more resources from different content providers, managing these resources becomes extremely difficult because of various schema and heterogeneous structures that come from different content providers. How a DL can continue aggregating and managing new resources while consistently maintaining overall DL quality at a high level is an important and interesting research topic.

A DL's reputation depends on the services it provides. If it provides incorrect information to the user, its reputation may be harmed and the user might never want to use its services again. Thus, a DL should always make sure that all of the records that are stored in its digital repositories have high quality, and that its services fetch correct records and provide them to the users. A DL should be able to catalog its records in repositories correctly using a uniform classification system and also have the ability to provide quality assured services persistently.

To summarize, we need to develop new approaches, architecture designs, and techniques that are solving these challenges to build an effective digital library with broad impact.

1.1 Research motivation

The research motivation in this dissertation is derived from the following three challenges identified by the Library of Congress [105].

1. The challenge in the **Effective Access** category: Develop approaches that can present heterogeneous resources in a coherent way.

2. The challenge in the **Building the Resource** category: Design tools that facilitate the enhancement of cataloging or descriptive information by incorporating the contributions of users.

3. The challenge in the **Sustaining the Resource** category: Develop economic models to support national digital libraries.

Our motivation is to address these challenges using our proposed approaches. We want to develop a framework that can gather content created by users and also categorize these resources precisely using a uniform classification system. User created content is not limited to the resources that users upload directly to the DL, but also the content from other parts of Web 2.0, such as YouTube and SlideShare, that contain valuable educational materials contributed by users.

In order to present heterogeneous resources in a coherent way, we want to develop a new technique, based on machine learning, that can efficiently, effectively, and systematically categorize heterogeneous resources using a uniform classification system. We want to integrate these functionalities into a DL to reduce the amount of cataloging and labeling work for the DL administrator/librarian. With an enhanced cataloging system, a DL can provide better services for the user to search and browse collections and fulfill a user's information needs.

We also are looking into cloud computing as an approach to address the third challenge. We want to develop a cloud-based architecture design that will not only sustain long-term DL operations but also ensure the quality of services in the DL.

1.2 Research goals

Guided by our research motivation, there are several research goals that we want to achieve in this dissertation:

1. Develop a method to build classifiers that can efficiently and effectively categorize heterogeneous resources using a uniform classification system. We chose the 2012 ACM Computing Classification System (CCS - see Chapter 3) [36] as our uniform classification system.

2. Develop a framework that can gather educational resources in computing from Web 2.0 in order to enrich the collections in a DL. We chose YouTube and SlideShare as our target content providers, since these two sites are currently major user-generated content sites that contain educational resources.

3. Develop a cloud-based architecture design that ensures the quality of service in a DL, utilizes computing resources only as needed, and reduces the overall cost through the advantages provided by cloud computing.

1.3 Research questions and hypotheses

To accomplish our research goals, we develop two research questions that we want to answer in this study and propose our hypotheses for the research questions correspondingly.

1.3.1 Research questions

Research question 1: From the universe of user generated online resources, how can we precisely identify and categorize educational content?

Research question 2: How can we ensure high quality in an educational digital library when:

a. building collections, and

b. continuing operations and services.

1.3.2 Hypotheses

Hypothesis 1: An effective way to identify and categorize content from YouTube and SlideShare for a digital library of computing education resources integrates categorization, machine learning, and crowdsourcing. This hypothesis corresponds to research question 1.

Hypothesis 2: Topic modeling on an existing broad coverage DL can drive acquiring candidate resources so building new DLs will have similar broad coverage. This hypothesis corresponds to research question 2a.

Hypothesis 3: Cloud capabilities allow a digital library to be more scalable, reliable, and sustainable relative to on-premise environments. For example, response time can be equalized around the world, and advantages in computing can be quickly utilized. This hypothesis corresponds to research question 2b.

1.4 Research approach

Our approach for the first research question is to develop a machine learning method, using auxiliary sources to exploit and transfer the knowledge to new domains. We propose an iterative process to continue updating classifiers by supplying new training data results from Amazon Mechanical Turk (MTurk) [2]. We study and conduct experiments aiming to verify the flexibility that results from enlisting the general public to do labeling tasks, instead of hiring computing professionals.

Our approach to the collection building task is to use topic modeling on the existing DL collections in order to find the hidden semantic structures and to use that information to gather new related resources from other content providers. In this research, we focus on educational resources in computing and use an ACM DL metadata dataset as our starting point. We develop an application pipeline to gather educational resources in computing from Web 2.0.

Our approach to enable continuing operations and services of the DL is focused on ensuring efficiency, reliability, and scalability of DL services. We develop and compare several cloud-based designs and finalize a design that ensures these qualities using cloud computing services.

To sum up, we research approaches to achieve our goals by enhancing both the internal (especially completeness and consistency) and external (especially efficiency, reliability, and scalability) qualities of a DL. To achieve DL collection and repository completeness, we develop an application pipeline to gather new educational resources in computing from Web

2.0. To achieve DL consistency, we improve user satisfaction by classifying collections in a DL using a uniform classification system, organized according to the ACM CCS. We use machine learning techniques to build classifiers that can categorize records relative to the ACM CCS. Lastly, to achieve DL efficiency, reliability, and scalability, we develop cloud-based designs and software to establish a DL where external qualities are assured.

Table 1.1 relates the research questions with the chapters that address them.

Table 1.1: Research questions and chapter organization

Topic	**Chapter**
Analyzing ACM DL dataset and building ACM classifiers	Chapter 3
Using topic modeling to aid record acquisition from Web 2.0	Chapter 4
Building collections in computing for an educational DL	Chapters 3, 4
Building classifiers for Web 2.0 educational records	Chapter 5
Using MTurk to evaluate and improve the performance of classifiers	Chapter 5
Using cloud computing to construct an efficient and reliable DL	Chapter 6

The contributions of this study can be listed as follows:

1. Based on the framework presented in this dissertation, a digital library could be developed that would be able to gather new resources about a variety of computing topics from multiple Web 2.0 sites.

2. The algorithms and the classifiers developed in the dissertation can be integrated into an existing DL and used to help categorize the collections in the DL. It would bring benefits to both librarians who manage the DL and the users of the DL services.

3. The proposed cloud-based architecture design may bring improvements to existing DL services. As cloud computing is a mature technology and is quickly becoming a mainstream source of technology in multiple areas, it may broaden the research horizons in the DL community.

1.5 Dissertation organization

The rest of the dissertation is organized as follows. In Chapter 2, we provide background for each of the research goals and present an overview of the relevant literature. Chapter 3 presents an analysis of the ACM dataset, the procedure of building ACM classifiers, and improvement of the performance of classifiers. In Chapter 4, we use topic modeling to acquire new resources from Web 2.0 and compare the results with other methods. In Chapter 5, we propose our machine learning approach and conduct MTurk experiments to evaluate new classifiers for new domain datasets from YouTube and SlideShare. In Chapter 6, we present our cloud-based design to ensure the quality of services in a DL. Finally, Chapter 7 presents a summary of the tasks accomplished in this dissertation work, along with future research directions.

Chapter 2

Background and Related work

In this chapter, we provide an overview of the Digital Libraries background, and prior research and techniques that are related to different phases of this dissertation work, to help readers better understand the content of later chapters.

2.1 Background

In this section, we start with the background knowledge of digital libraries followed by quality aspects of DLs. We also describe an educational DL, Ensemble, which is used to conduct our experiments and is closely related to our work.

2.1.1 Digital Libraries

Digital libraries (DLs) provide services for users to search the contents of their system. A good digital library should have enough content (metadata, possibly supplemented by digital objects) to cover a broad range of topics in order to fulfill the users' information needs. Therefore, librarians and curators aggregate metadata from different resources, store and index the metadata, and create services intended for users who then can get the information they want including the original resources. Thus, a DL is heavily dependent on the quality of the metadata in order to provide useful services. A DL's reputation also is built on the metadata stored. If a user finds out that one of the records is incorrect and cannot be trusted, the reputation of this DL is damaged and a long time may be needed before trust returns.

For a metadata resource to become included in a collection and to be stored in the DL repository requires several steps. First comes resource acquisition, wherein librarians find content providers, establish workflows to get their metadata content, and store metadata in the DL repository. DLs aggregate metadata from content providers through different means; the most common standard method is using OAI-PMH [103] to harvest metadata resources from participating content providers. Although through OAI-PMH metadata can be quickly harvested from content providers, the quality of these metadata resources from different content providers will vary. Moreover, there is no guarantee that metadata quality is always the same even when metadata resources come from the same content provider. Besides, DLs often don't harvest metadata resources from a single content provider; thus when DLs harvest metadata resources from more and more different content providers, due to various metadata schema and standards, managing metadata resources and controlling metadata quality becomes a complex task. DLs also need to continue evaluating new resources in order to ensure the overall collections' quality in the DL repository. Many research studies [104] [38] focus on solving this problem through automatic and scalable approaches in order to supplement traditional human manual approaches.

Besides aggregating resources from outside content providers, DLs can create metadata resources from their own digital content. Through a manual evaluation process, librarians can create high quality metadata resources. This human evaluation process is suitable for small collections. When DLs get more and more resources, however, it would take much time to do data cleaning through a human evaluation process [14].

To acquire a large number of resources, having a DL rely solely on metadata harvesting and/or create metadata resources merely on its own content is not efficient in today's environment. User generated content has been increasing dramatically in recent years and contains valuable resources which are suitable to be harvested and preserved. Furthermore, the user generated content contains features such as views, comments, ratings, and tags which can be analyzed to determine the usefulness of the content for the target audience. The quality of user generated content varies; it is also a challenge to identify good resources from user generated content and include them in a DL repository.

All of the collection building processes, which we mentioned above, can be categorized into four types. These are: metadata harvesting, user submission, community content submission, and focused crawling. These methods require having quality assessment in order to ensure high quality records. For example, when a DL harvests resources from other content providers through OAI-PMH, there is no guarantee that content providers will provide high quality records. Content providers may provide incomplete information and thus the service providers need to check harvested resources and make necessary improvements [25]. Users may submit unrelated content through the user submission process and thus a DL needs a group of people to moderate new submissions [91]. Community content created by members of a DL community varies in type and quality. It is a difficult challenge to determine whether such content is an educational resource or not. Finally, many researchers developed techniques to automatically generate metadata from Web resources [107], but there still must be an evaluation process to purify generated metadata. When the collection size is increasing rapidly, these issues become even more difficult to solve. It is very expensive to maintain collection quality solely by using human evaluation. There is a need to develop a scalable approach to handle both collection acquisition and collection quality.

To sum up, to establish a good digital library requires accomplishing many complex tasks [61]. In order to engender trust and provide useful services, DLs should take responsibility to ensure that all the resources they have are of high quality. Moreover, DLs should keep their collections current and acquire more resources, which fit their users' expectations, and further engage them to contribute more resources and form a continuously evolving information environment. We believe that a DL with a mass of high quality educational resources can attract many users and make a positive impact on education, including in the computing domain. It could bring users to see different resources and gain benefit from them. We argue that once we solve the complex metadata quality control task, we can use the methodologies we proposed to build a new DL with appropriate collections, in a very efficient way.

2.1.2 5S framework

The 5S framework is a unified formal theory for Digital Libraries (DLs) [60] [55]. It is composed of Streams, Structures, Spaces, Scenarios, and Societies (5S). Streams describe properties of static or dynamic digital objects in the DL such as text, audio, and video. Structure specifies the organizational scheme of the digital objects in the DL, such as ontologies, taxonomies, links, tags, and labels. Space defines logical and presentational views of several DL components, such as vector space, index, and user interface. Scenario details the behavior of DL services. Society describes different roles and groups for operating or using the DL services.

2.1.3 Dimensions of quality

Extending from the 5S framework, we can define elements and construct a minimal DL [99]. Some key concepts of a minimal DL are catalog, collection, digital object, metadata specification, repository, and services. To ensure the good quality of a minimal DL, [61] defines a number of dimensions of quality. These dimensions of quality are accessibility, accuracy, completeness, composability, conformance, consistency, effectiveness, efficiency, extensibility, pertinence, preservability, relevance, reliability, reusability, significance, similarity, and timeliness.

In this dissertation work, we focus on both external and internal qualities for DL services

[132]. The internal DL qualities refer to the completeness and consistency quality dimensions in collection, catalog, and repository. The external DL qualities refer to the efficiency and reliability quality dimensions in services. Figure 2.1 shows relationships in the 5S framework, regarding connecting DL concepts with the dimensions of quality we consider in this research work.

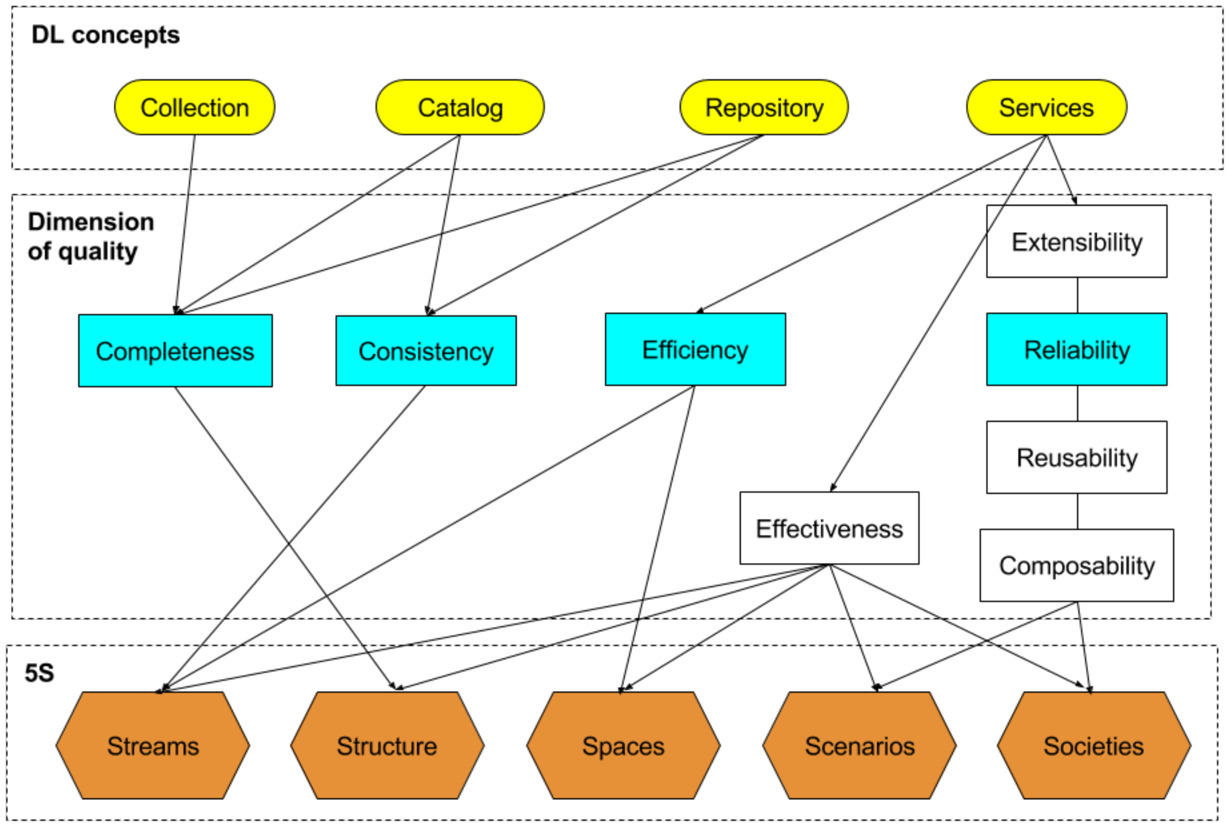


Figure 2.1: Relationships in the 5S framework – connecting DL concepts and dimensions of quality

2.1.4 DL interoperability

There are two ways to implement DL interoperability: through distributed search or harvesting. The distributed search, or metasearch, is a service that provides unified query interfaces to multiple search engines. It requires each search engine to implement a joint distributed search protocol; moreover, as it needs to post-process search results in real time, it presents significant scalability problems. The distributed search approach is studied in NCSTRL [47] and Stanford Infobus [113].

A harvesting approach collects data from heterogeneous sources in advance; therefore, it is more realistic in dealing with a large number of digital libraries. Harvesting approaches have the additional attractive property that they allow data enhancing procedures to be run on the collected data. Enhancements such as normalization, augmentation, and restructuring are applied to data originating from different sources in order to create consistent end-user services. In a harvesting scenario, these activities can be dealt with in a batch manner. The harvesting approach is studied in Harvest [21] and is the basis of OAI-PMH.

2.1.5 Dublin Core metadata terms

The Dublin Core Metadata Initiative (DCMI) and the Institute for Electrical and Electronics Engineers (IEEE) Learning Object Metadata (LOM) Working Group are both concerned with metadata, such as for educational resources. Manually creating metadata is a time-consuming process; several research studies have discussed the possibility of reducing the cost of traditional libraries by providing equivalent library services in a digital library. [62] proposed a digital library in which all tasks are carried out automatically. These tasks include selection, cataloguing, and indexing, seeking for information, reference services, etc. Although it is not possible to automate all these tasks now, what tasks can be automated deserve much attention. Since metadata is the fundamental building block in digital library systems, many research studies explore the potential for automating metadata generation. [107] developed an automated name authority control (ANAC) tool which used a Naïve Bayes classifier to assign a name to a Library of Congress (LC) metadata record. Klarity and DC-dot [5] are two Dublin Core (DC) automatic metadata generation applications. Both applications only read the top level of the resource (Web page) and use both metadata extraction and harvesting methods to generate DC metadata. [63] compared and evaluated these two applications and concluded that metadata generators have the potential to create useful metadata by using these two methods.

2.1.6 OAI-PMH

The Open Archives Initiative (OAI) is an effort to address interoperability issues among many existing and independent DLs [81] [126]. An OAI-compliant Data Provider (see [103] for OAI notation and terms) exposes metadata through an HTTP/XML based protocol.

The original sources can be a Web site or another DL. DLs often use OAI-PMH to expose their collection as well as to harvest from other DL collections. Although OAI-PMH metadata harvesting has been proven to be a low-barrier and low cost way for DLs to exchange their collections [79], this process requires the original source to have an OAI data provider setup ready for others to harvest the metadata. However, not every content provider has an OAI data provider for us to harvest its collections. When we want to harvest their collections, the usual approach is to ask content owners to build their own data provider. We can provide them with metadata guidelines and software to help them to build the data provider. Due to uncertainty regarding content provider's technical abilities, how much time they will need to have their OAI data provider ready for harvesting is unable to be determined. Moreover, the OAI data provider's functionalities and the metadata quality may not be consistent across different content providers. When a DL harvests metadata from different content providers, the DL might face several interoperability issues. For example, OAI data providers might produce invalid metadata records, misunderstand the DC field definitions and so provide the wrong value, not support "deleted" records, or fail to provide incremental harvesting due to incorrect timestamps associated with records [80]. Hence a DL may need to spend additional time and significant human intervention to solve these interoperability issues with content providers. This passive model is usually slow and may not be scalable [80]. Furthermore, if the content provider doesn't intend to build an OAI data provider, we have no way to

Table 2.1: Partial content provider list and number of records we have harvested

Digital libraries and research projects	**# of records**
ACM Women in Computing	897
Algorithm Visualization	528
CITIDEL	260
Digital Library Curriculum	36
iLumina	58
PAWS	25
STEMRobotics	343
SWENET	64

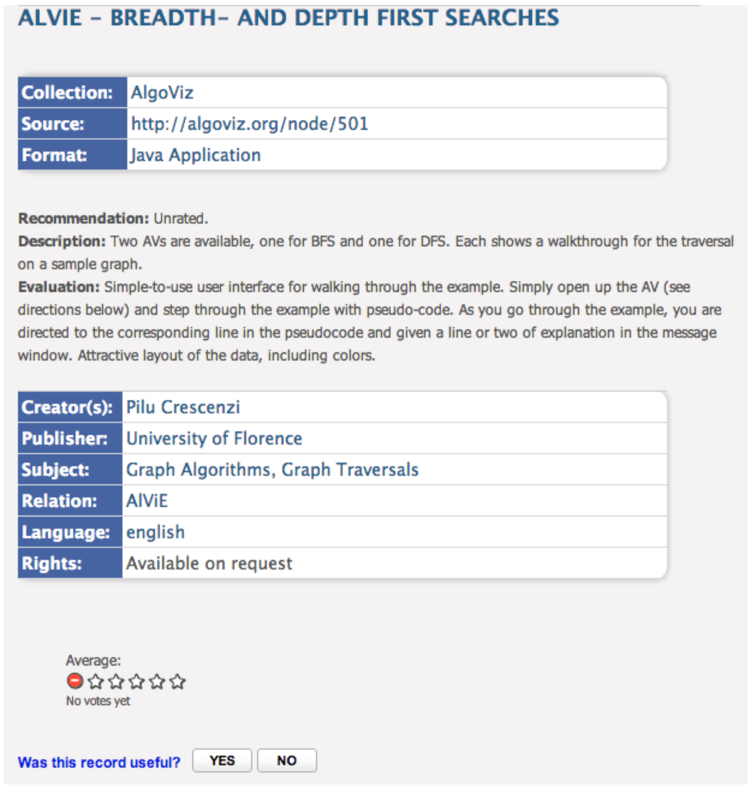
harvest its metadata. Even if content providers are willing to set up an OAI data provider, they still need to transform their collection to provide DC metadata. The process is not a trivial task, and needs time to make it correct.

2.1.7 Ensemble: the computing portal

Ensemble is a National Science Digital Library (NSDL) Pathways project working to establish a national, distributed digital library for computing education. The Ensemble Portal aggregates resources through OAI-PMH metadata harvesting, user contributions, and crawlers. Currently Ensemble has over 30 educational collections; some of these collections are ACM Women in Computing, Algorithm Visualization, The Beauty and Joy of Computing, Stanford Nifty Collection, BPC, CITIDEL, Computer History Museum, CSERD, CSTA, Digital Library Curriculum, iLumina, PAWS, PlanetMath, STEMRobotics, SWENET, VKB, Walden's Paths, and Ensemble user contributions. Table 2.1 lists some of the current digital libraries and research projects which provide Ensemble with resources through OAI-PMH.

Ensemble is built upon the 5S digital library framework. We can describe Ensemble according to the 5S framework as follows:

Streams. Computing educational contents in Ensemble include textual information (HTML, PDF, Word, and PowerPoint), screenshots of educational tools, a short introductory video describing educational tools, etc. Figure 2.2 shows a metadata record in Ensemble.


ALVIE – BREADTH– AND DEPTH FIRST SEARCHES

Collection:	AlgoViz
Source:	http://algoviz.org/node/501
Format:	Java Application

Recommendation: Unrated.
Description: Two AVs are available, one for BFS and one for DFS. Each shows a walkthrough for the traversal on a sample graph.
Evaluation: Simple-to-use user interface for walking through the example. Simply open up the AV (see directions below) and step through the example with pseudo-code. As you go through the example, you are directed to the corresponding line in the pseudocode and given a line or two of explanation in the message window. Attractive layout of the data, including colors.

Creator(s):	Pilu Crescenzi
Publisher:	University of Florence
Subject:	Graph Algorithms, Graph Traversals
Relation:	AlViE
Language:	english
Rights:	Available on request

Figure 2.2: A metadata record in the Ensemble Portal

Structures. Ensemble acts as a data harvester and as a data provider. All the metadata in Ensemble has two formats; one is Dublin Core metadata and the other is NSDL_DC metadata. All user-contributed content and educational materials collected through crawlers are transformed into these two formats, and other DLs can harvest Ensemble metadata through the Ensemble data provider. All of the metadata harvested from other DLs is cataloged and represented in a user readable record page in Ensemble that also is accessible to the general public. Each record page contains text, links, labels, images, videos, etc.

Spaces. All of the content in Ensemble is indexed using Solr, working with feature and vector spaces. Ensemble uses the functionalities provided by Solr to support faceted search with browsing capabilities. Those operate through user interfaces on two-dimensional spaces, rendered in windows on computer or mobile device screens.

Scenarios. Users can find educational materials in Ensemble through browsing and searching services. They can create groups in Ensemble and invite others with similar interests to join those communities. Users also can contribute their educational content and add to the Ensemble user contributions collection. Further, Ensemble provides several interactive services to gather user feedback such as comments, tags, and ratings.

Societies. Ensemble defines different user roles to manage the operation of the Ensemble site. For example, administrators manage site-wide configuration and service upgrades, conference organizers manage user contributed content, and group managers handle the collections related to a group among the Ensemble communities.

2.1.8 Web 2.0

Web 2.0 differs from Web 1.0 is that Web 2.0 websites are designed to emphasize the usability and interoperability of their Web services and to engage users in uploading content, yielding user-generated content. Web 2.0 examples include social networking sites (e.g., Facebook and Twitter), video sharing sites (e.g., YouTube), and slide hosting services (e.g., SlideShare). These websites provide Web Service APIs for other applications or Web sites to retrieve their content in different formats – such as XML, RDF, and JSON – and broadly share their content with communities across the Internet. In this study, we chose YouTube and SlideShare as our target content providers since there are many videos and presentation slides that have high educational value in computing.

2.2 Related work

To the best of our knowledge, this study is the first attempt to use machine learning techniques to classify educational records from Web 2.0 using ACM CCS. In this section, we perform a general literature review of these research areas: DL quality, text classification, feature selection methods, topic modeling, transfer learning, and active learning. We further describe related works that have similarities with our proposed approaches, in subsequent chapters.

2.2.1 Quality in the digital library

There are many research studies that have focused on solving quality problems in digital libraries. [121] described the key issues of evaluation and integration in DLs. [23] listed seven metadata quality criteria: completeness, accuracy, provenance, conformance, consistency, timeliness, and accessibility. [61] defined a number of dimensions of quality, which are accessibility, accuracy, completeness, composability, conformance, consistency, effectiveness, efficiency, extensibility, pertinence, preservability, relevance, reliability, reusability, significance, similarity, and timeliness. Although librarians can build on these models to determine metadata quality, there are still various ways to measure metadata quality in DLs using different means. Many of these are developed case by case and require much human effort. Assessing DL collections based on these quality criteria requires both human intervention and tool support [14]. Using experts to evaluate metadata quality is not scalable and is very expensive. Thus, developing tools and applications that can automate evaluation processes and reduce the need for experts are promising research topics. Several researchers therefore prepared custom-written software to evaluate OAI repositories [101].

Quality of metadata is a critical concern when building successful digital libraries. In order to provide high quality DL services, metadata needs to be evaluated. Cataloging and metadata professionals use different mechanisms to assess metadata resources. These mechanisms include staff training, following metadata creation guidelines, using metadata creation tools, or forming metadata review groups to assess metadata. However, all these approaches require humans to be involved, so it is hard to be scalable. For example, the National Science Digital Library (NSDL) faced the problem of managing metadata records from heterogeneous resources, and addressed it by delegating the enforcement of quality controls to each project team included, such as those running the Ensemble portal. Thus, there is a need to develop a scalable approach to help DL managers measure data quality easily, so they can focus only on data that needs remediation.

Several research groups have created tools to evaluate DL quality, into different levels. 5SQual [98] measures DL quality using seven metadata quality criteria: completeness, accuracy, provenance, conformance, consistency, timeliness, and accessibility. [101] is a lightweight measurement tool which evaluates records collected through OAI-PMH harvesting. [24] implemented a Greenstone plugin tool to provide a quality analysis component. Using human judgments to assess quality (expert review), MERLOT uses comprehensive peer-review processes to evaluate user-contributed resources [91]. Wikipedia, which depends entirely on user contributions, has spawned numerous research efforts exploring algorithmic approaches for automatically assessing article quality [7]. However, human judgment based approaches have several inter-related issues, such as the consistency of humans when making potentially subjective assessments. They also face scalability issues, and it is hard to keep up with the rate of user contributions. Thus some researchers focus on using machine learning techniques to do automatic reviews [25] and quality visualizations [52]. Custard and Sumner trained machine learning models to judge overall quality using low-level features like website domain names, the number of links on a page, how recently a page was updated, and whether or not videos or sound clips were present [38]. Zhu has developed algorithms for improving Internet search results that draw on quality metrics such as the time since last update of a Web page, presence of broken links, and the amount of textual content [117]. [107] used machine learning techniques to detect decision patterns from annotated resources in order to support domain experts making quality judgments on new resources. Although these researchers all aim to solve the quality issues in DLs, there is no complete solution that connects the resource acquisition and resource quality enhancement process all together to construct this kind of DL.

2.2.2 Classification techniques

The first research task of this dissertation is to build classifiers using an ACM supplied dataset. This task is a classical text classification problem and a supervised learning task. In a supervised learning task, the algorithm learns from a labeled training dataset so it can correctly predict the class labels for elements in that training dataset as well as a labeled validation dataset. The learning continues until the algorithm achieves an acceptable level of performance. A classification problem is to determine that an object belongs to a certain category, for example "book" or "movie". If the number of categories in a classification problem is two, it is called binary classification. If the number of categories in a classification problem is larger than two, it is called multi-class classification. In the ACM CCS, a CCS node contains zero to many child nodes and thus we address both binary and multi-class classification problems in this dissertation.

There are many algorithms that have been developed to solve the binary classification problem. From among those algorithms, in this research we use decision tree, k-nearest neighbor, Naïve Bayes, Perceptron, Ridge regression, Support Vector Machine (SVM), and Neural Network (see also Chapter 4).

A decision tree is a tree-like graph. It is used to get the final outcome by asking a question at each node (in the middle of the tree) until a leaf node is reached. Decision tree learning uses a decision tree as a model to predict an object's class membership. Researchers developed decision tree algorithms to solve text classification problems, such as [73] [11].

k-nearest neighbors (k-NN) is a non-parametric method used for classification. The input to a decision consists of the k closest training examples in the feature space and the output is a class membership. An object is classified to a class based on the result of majority votes from its k nearest neighbors. The k is a positive integer to indicate the number of neighbors voting. k-NN has been applied to solve text classification problems in [125] [66] [139].

The Perceptron is a type of linear classifier. The Perceptron algorithm was one of the first artificial neural networks to be produced [133]. It does not require a learning rate and updates its model only on mistakes.

Naïve Bayes classifiers are simple probabilistic classifiers based on Bayes' theorem. They are known for being simple yet efficient [112]. It has been reported that they work well in different real-world situations, such as document classification [57] and spam filtering [115] [74].

A Support Vector Machine (SVM) is a popular supervised learning model that is widely used to solve text classification problems. SVM learns a linear model, determining the maximum margin hyperplane which gives the highest separation between the decision classes. It has been reported that SVM's performance is better than other text classification algorithms in many situations [137].

In addition to the methods summarized above, many other classification techniques are proposed to solve binary classification problems. In order to use these classifiers to solve the multi-class classification problems, we use the one-vs-all strategy. The one-vs-all strategy transforms a multi-class classification problem into n separate binary classification problem by taking one class as positive samples and all other classes as negatives. The classifier then trains on each dataset, makes a prediction with a confidence score, and outputs the final class membership based on the prediction with the highest confidence score.

2.2.3 Feature selection

Feature selection is a process to automatically select a subset of attributes in the dataset in order to acquire better classifier performance [43] [67]. Feature selection techniques are developed to create filtered versions of the dataset and improve the accuracy of the classifier.

There are several reasons to use feature selection: 1. Training time will be reduced with a small number of features. 2. Overfitting problems are avoided by eliminating irrelevant attributes. 3. Misleading decisions are avoided by reducing redundant attributes.

There are three general classes of feature selection algorithms [97]. There are embedded methods, filter methods, and wrapper methods. The filter methods use a statistical measure to select the features with high correlation to the labels regardless of the model [140]. The filter methods include the Chi-squared test [92] [100], information gain [136], and correlation coefficient scores [100]. The wrapper method tries to find the best subset of features for a learning algorithm [77]. The computation time would be quite long when the number of variables is large and different sets of features would be generated by different learning algorithms. The recursive feature elimination algorithm is an example of a wrapper method, and is commonly used with SVM [15]. The embedded method is incorporated with learning algorithms and tries to find the best subset of features contributing to accuracy [67]. The common type of embedded feature selection methods are regularization methods. LASSO [130], Elastic Net [145], and Ridge Regression [145] are some of the most popular examples.

There are several feature selection methods applied in text classification, such as document frequency [138], information gain [118], mutual information [93], and Chi-square [143]. Finally, we also use a bag-of-words model and an n-gram model to train our ACM classifiers.

2.2.4 Topic modeling

Topic modeling is used to find the hidden semantic structure in text documents. The most common topic model is Latent Dirichlet Allocation (LDA) [16]. The graphical model for LDA is shown in Figure 2.3. The outer plate represents documents and the inner plate represents the repeated choice of topics and words within a document. M is the number of documents, N is the number of words in a document, α is the parameter of the uniform Dirichlet prior on the per-document topic distributions, β is the parameter of the uniform Dirichlet prior on the per-topic word distribution, θ_m is the topic distribution for document

m, φ_k is the word distribution for topic k, Z_{mn} is the topic for the n-th word in document m, and w_{mn} is the specific word [64].

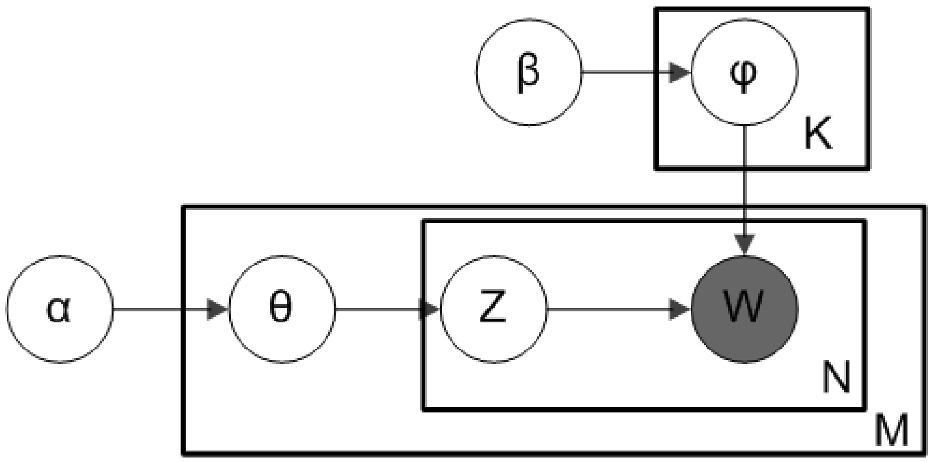


Figure 2.3: Plate notation for LDA with Dirichlet-distributed topic-word distributions [37]

LDA has been used to discover semantic topics from different types of corpora, such as Web 2.0 social media [123] [142], research papers [64], etc. LDA has also been used to assess similarity measures [27] [102].

2.2.5 Transfer learning

In traditional machine learning, there is a common assumption that both training and test data have the same feature space and distribution. However, in many real-world situations, this assumption does not hold. Moreover, in many situations there are no labeled training data, or it would be difficult to collect these training data to build the models. In such cases, transfer learning [106] involves leveraging the knowledge of source domains to help train a classifier for a target domain. There are many real-world applications that use transfer learning techniques, such as text classification [8] [116] [41]. Transfer learning assumes that there are some common features between the source and target domains, that can be learned from the source domain. Transfer learning has been applied to solve classification, regression, and clustering problems.

There are three major settings in transfer learning: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning [106]. Table 2.2 shows an overview of transfer learning settings [106]. There are four main categories in the transfer learning approaches: instance-transfer, feature-representation-transfer, parameter-transfer, and relational-knowledge-transfer. The instance-transfer approach assumes that certain parts of the data in the source domain can be reused for learning in the target domain by re-weighting. The feature-representative-transfer approach is to learn a "good" feature representation that reduces the difference between a source domain and a target domain. The parameter-transfer approach assumes that the source and the target tasks share some parameters or prior distributions of the hyper-parameters of the models and transfers the knowledge by discovering the shared parameters and priors. The relational-knowledge-transfer approach assumes that the relationships among the data in both source and target domain are similar. The knowledge that can be transferred describes the relationships among the data. Table 2.3 shows these transfer learning approaches and some of the related literature.

2.2.6 Active learning

Active learning, sometimes called optimal experiment design, is a form of supervised machine learning [35]. In the situation that labeled data is not easy to obtain, an active learning approach is to ask an "oracle" (domain expert) to label a small set of instances from the dataset and to use as few labeled instances as possible to build a high accuracy classifier [42] [119]. Active learning has been studied to reduce labeling cost in classification problems; many advanced methods have been developed to solve the problem in different scenarios. To name a few, there are multi-instance active learning, multi-label active learning, multi-task active learning, multi-view active learning, multi-oracle active learning, multi-objective active learning, and active transfer learning [120].

Table 2.2: An overview of transfer learning settings [106]

Transfer Learning Settings	**Related Areas**	**Source Domain Labels**	**Target Domain Labels**	**Tasks**
Inductive Transfer Learning	Multi-task Learning	Available	Available	Regression, Classification
	Self-taught Learning	Unavailable	Available	Regression, Classification
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias, Co-variate Shift	Available	Unavailable	Regression, Classification
Unsupervised Transfer Learning	Self-taught Clustering, Multi-task Clustering	Unavailable	Unavailable	Clustering, Dimensionality Reduction

Table 2.3: Transfer learning approaches and some related references

Transfer learning approaches	**Literature**
Instance-transfer	[41] [40] [108] [70] [141]
Feature-representation-transfer	[39] [10] [18] [44] [12] [83]
Parameter-transfer	[82] [20] [53] [59]
Relational-knowledge-transfer	[95] [96] [48]

Active transfer learning combines settings in transfer learning and active learning to build more effective models [135] [28]. The central idea is using the transfer learning approach to exploit the labeled data from the source domain in order to facilitate the learning process in the target domain, while using active learning with selective sampling of unlabeled data in the target domain [122]. There are two approaches to perform the domain knowledge transfer. One is to transfer an instance from the source domain to the target domain using the transfer learning methods. Another is to query the “oracle” with an instance for the label [144].

In our study, the ACM dataset is our source domain, and a Web 2.0 dataset is our target domain. The source domain labels are available, but target domain labels are unavailable. The source and target domains are different while the source and target tasks are the same. Our scheme falls into a transductive transfer learning setting which can use instance-transfer [41] and feature-representation-transfer [109] approaches. More specifically, our study is related to domain adaptation for knowledge transfer in text classification [46]. Our proposed approach is related to active learning. We use MTurk as our “oracle” to evaluate the classifier prediction result and obtain a new training dataset to update the existing classifier in order to achieve good performance. In Chapter 5, we present our work on transferring what we learned from the auxiliary source domain and applying it to the target domain, along with active learning to build the classifiers for the Web 2.0 dataset.

Chapter 3

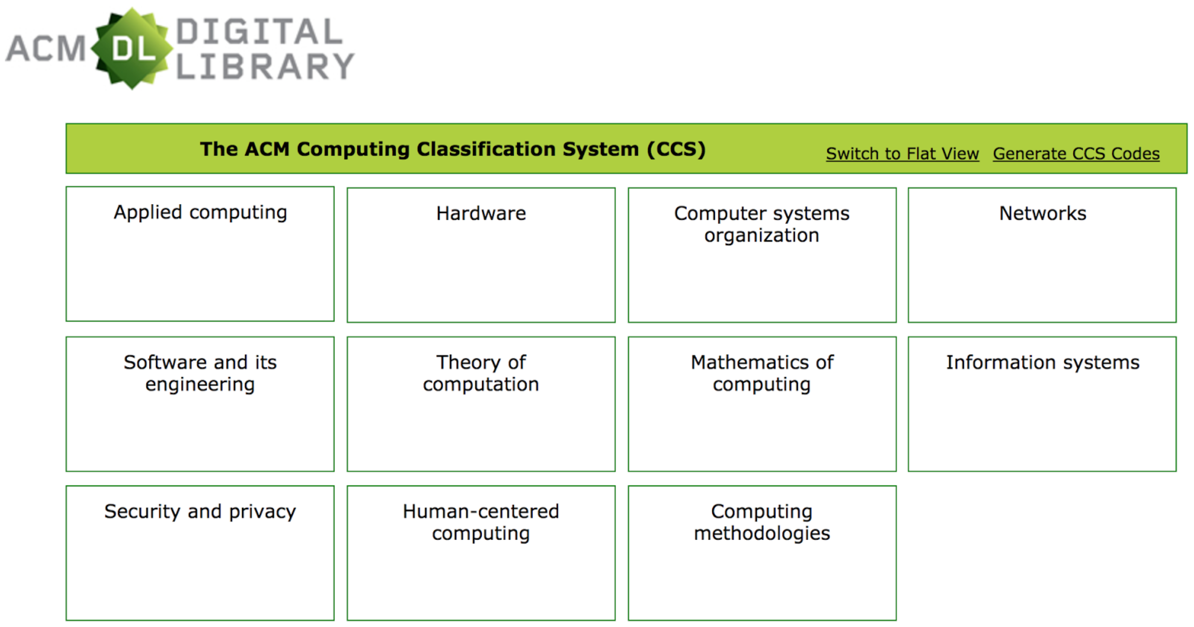
Building ACM digital library classifiers

3.1 Introduction

In this chapter, we first introduce the 2012 ACM computing classification system (CCS) and ACM DL [51] metadata dataset (ACM dataset). We perform a detailed analysis and identify current issues in the ACM dataset. We then present our work on building ACM DL classifiers using the ACM dataset. We describe our approaches to build ACM classifiers with high performance. We apply several state of the art classification techniques with suitable features to train the ACM classifiers and identify the best machine learning algorithm for the ACM dataset. The dataset that has been cleaned and used to build classifiers becomes the dataset we use in the following studies reported in the later chapters. The trained models can be used to predict labels for future articles going into the ACM DL.

3.1.1 The 2012 ACM Computing Classification System (CCS)

The 2012 ACM Computing Classification System (CCS) is a standard classification system for the computing field. It is a six-level poly-hierarchical structure. The top level contains thirteen branches, and each branch has at most six-levels. Each branch contains multiple CCS terms, and each CCS term reflects a concept in the computing discipline. Among these branches, “General and reference” and “Social and professional topics” include cross-cutting computing concepts and thus we used the other eleven branches in this research. Figure 3.1 shows an overview of these selected eleven top-level branches we used throughout this dissertation work.

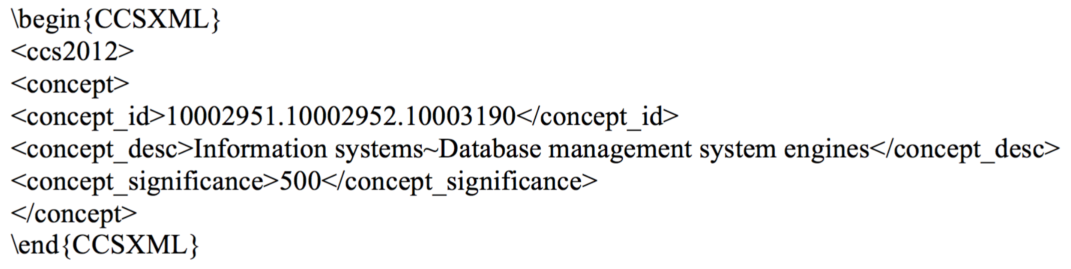


The ACM Computing Classification System (CCS)

Applied computing	Hardware	Computer systems organization	Networks
Software and its engineering	Theory of computation	Mathematics of computing	Information systems
Security and privacy	Human-centered computing	Computing methodologies	

Figure 3.1: Eleven CCS top level branches used in this research

ACM recommends that an author use the lowest node in the CCS tree available and preferably use the leaves of the tree when applicable to label an article. The author needs to assign weights to each CCS term that he is using. ACM uses a three level weighting system: High, Medium, and Low. High relevance is indicated by a score of 500, Medium relevance is indicated by a score of 300, and Low relevance is indicated by a score of 100. For example, **Information systems—Database management system engines;** 500 indicates that the CCS term chosen for this work has high relevance to the subject matter of the publication. Figure 3.2 shows a sample TeX code from the above example.



```
\begin{CCSXML}
<ccs2012>
<concept>
<concept_id>10002951.10002952.10003190</concept_id>
<concept_desc>Information systems~Database management system engines</concept_desc>
<concept_significance>500</concept_significance>
</concept>
\end{CCSXML}
```

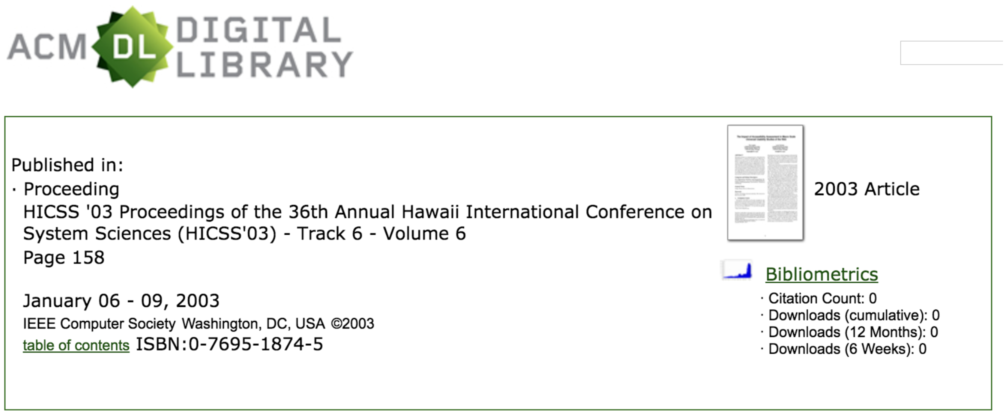
Figure 3.2: An ACM CCS TeX code sample

Therefore, a published work using ACM CCS should contain at least one CCS term with associated relevance weight.

3.1.2 ACM DL metadata dataset (ACM dataset)

On February 7, 2014, in support of our research, we received the ACM DL metadata dataset from the Association for Computing Machinery (ACM). This dataset contains 1,761,956 journal and conference proceeding metadata records in XML format. Each metadata record contains title, abstract, CCS, general term, author, and reference of the article. It does not include the full text of the article. This dataset contains thousands of records classified by professionals in computing, and thus is a valuable labeled training dataset for us to build classifiers in this study.

We processed all of the records in the ACM dataset to obtain our training dataset. Only about 40 percent of the records are labeled in CCS; the total number of labeled records is 700,213. Among these records, there are 265,092 records without abstract and 37 articles without a title. Figure 3.3 shows a record page in the ACM digital library website without title information [6]. Figure 3.4 shows a record page with empty abstract [54].

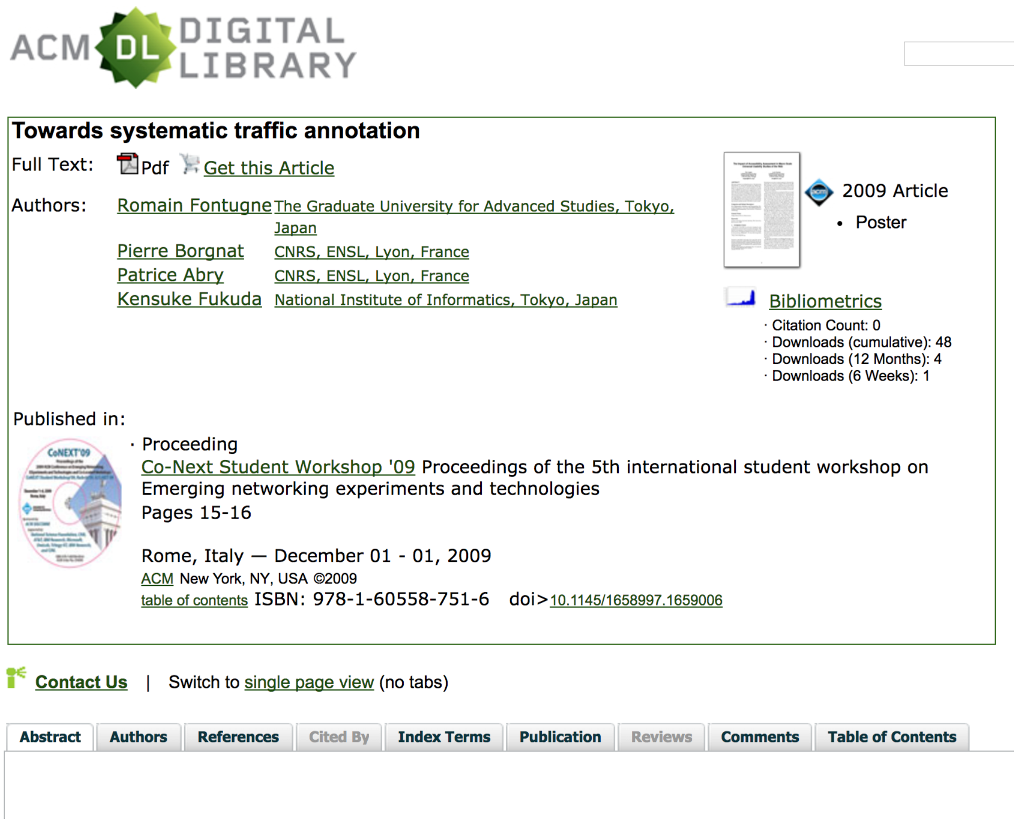

Published in:
· Proceeding
HICSS '03 Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 6 - Volume 6
Page 158

January 06 - 09, 2003
IEEE Computer Society Washington, DC, USA ©2003
table of contents ISBN:0-7695-1874-5

2003 Article

Bibliometrics
· Citation Count: 0
· Downloads (cumulative): 0
· Downloads (12 Months): 0
· Downloads (6 Weeks): 0

Figure 3.3: A record page without title information in ACM DL


ACM DL DIGITAL LIBRARY

Towards systematic traffic annotation

Full Text: Pdf Get this Article

Authors:	Romain Fontugne	The Graduate University for Advanced Studies, Tokyo, Japan
	Pierre Borgnat	CNRS, ENSL, Lyon, France
	Patrice Abry	CNRS, ENSL, Lyon, France
	Kensuke Fukuda	National Institute of Informatics, Tokyo, Japan

2009 Article
- Poster

Bibliometrics
- Citation Count: 0
- Downloads (cumulative): 48
- Downloads (12 Months): 4
- Downloads (6 Weeks): 1

Published in:

- Proceeding
Co-Next Student Workshop '09 Proceedings of the 5th international student workshop on Emerging networking experiments and technologies
Pages 15-16

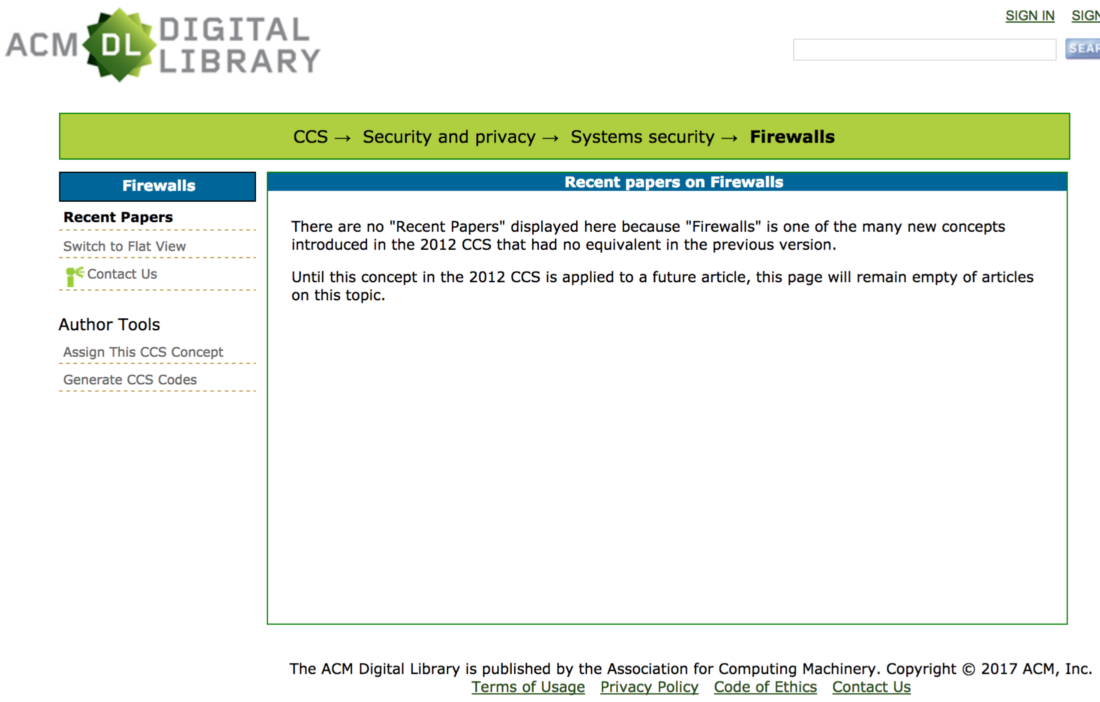
Rome, Italy — December 01 - 01, 2009
ACM New York, NY, USA ©2009
table of contents ISBN: 978-1-60558-751-6 doi>10.1145/1658997.1659006

Contact Us | Switch to single page view (no tabs)

Figure 3.4: An article that shows incomplete title and missing abstract

Finally, we obtained 435,119 records that have title, abstract, and are labeled in CCS, to use in this study. These records were stored in a relational database. The database schema is described in Appendix A.

Table 3.1 shows the distribution of the selected records in the chosen eleven ACM top level branches. The "Computing methodologies" branch contains the most articles, while "Security and privacy" contains the least. There are many CCS nodes with zero articles and thus not every top level branch is a six-level tree. Figure 3.5 shows an example where an ACM CCS concept contains no paper.



There are no "Recent Papers" displayed here because "Firewalls" is one of the many new concepts introduced in the 2012 CCS that had no equivalent in the previous version.

Until this concept in the 2012 CCS is applied to a future article, this page will remain empty of articles on this topic.

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2017 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Figure 3.5: An ACM CCS concept contains no paper

According to the ACM dataset we acquired, "Networks" and "Software and its engineering" are six-level trees. "Mathematics of computing," "Information systems," "Theory of computation," and "Computing methodologies" are five-level trees. "Security and privacy," "Human-centered computing," "Applied computing," "Computer systems organization," and "Hardware" are four-level trees.

With regard to the three-level weighting system, the ACM dataset has no articles labeled with an ACM term with medium relevance. An article can have multiple CCS terms associated with it with different relevance weights. Table 3.2 shows the number of relevance weights used in each article. For example, there are 252,364 articles labeled with one high relevance weight CCS term, and there are 21,536 articles labeled with six low relevance weight CCS terms. Figure 3.6 shows a histogram corresponding to Table 3.2. We can see that the majority of the articles (90 percent of the ACM dataset) contain only one or two "High"

Table 3.1: Number of papers in the ACM top level branches

CCS Identifier	**CCS Name**	**# of articles**	**ACM tag**
0.10002950	Mathematics of computing	59,281	CCS1
0.10002951	Information systems	67,156	CCS2
0.10002978	Security and privacy	16,108	CCS3
0.10003033	Networks	39,062	CCS4
0.10003120	Human-centered computing	33,435	CCS5
0.10003752	Theory of computation	56,828	CCS6
0.10010147	Computing methodologies	127,302	CCS7
0.10010405	Applied computing	59,683	CCS8
0.10010520	Computer systems organization	26,479	CCS9
0.10010583	Hardware	38,352	CCS10
0.10011007	Software and its engineering	71,612	CCS11

Table 3.2: Number of relevance weights used in each article

	1	**2**	**3**	**4**	**5**	**6**
Low relevance	108,062	96,399	75,641	55,970	35,388	21,536
High relevance	252,364	112,431	33,996	2,431	889	123

relevance weight CCS terms. Since our goal is to build classifiers to predict what CCS term is most relevant to an article, we use the "High" relevance weight CCS term as our class label, and prepared our training datasets accordingly.

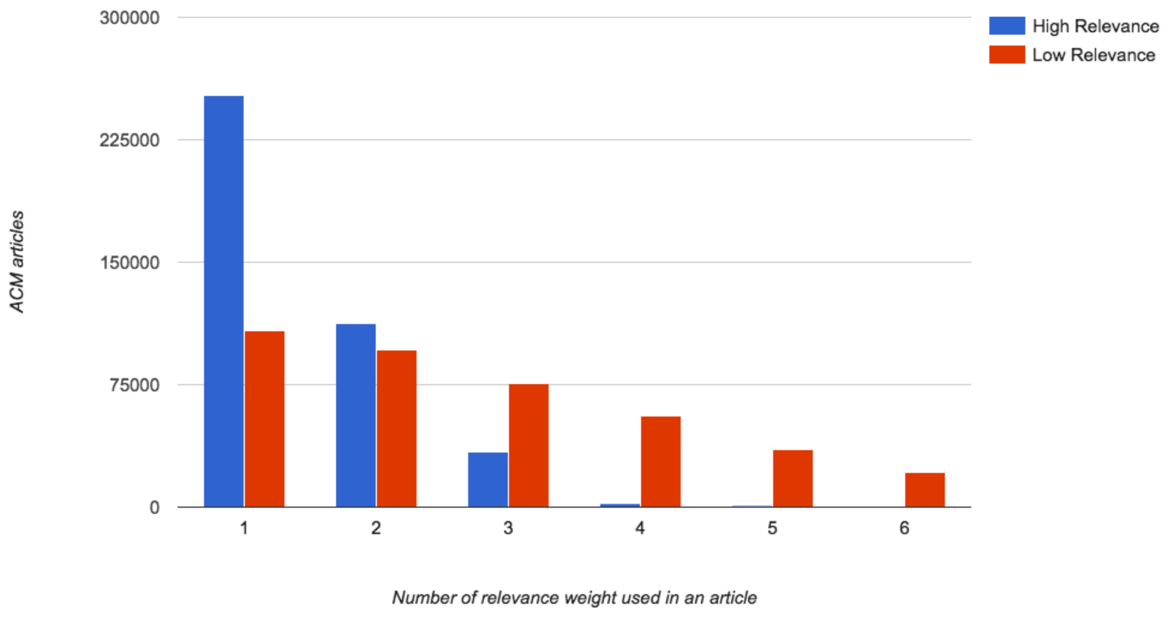


Figure 3.6: Number of relevance weights used in an article

During our data preparation, we identified several issues with the ACM dataset. First, there are articles that have been associated with a large number of different CCS terms; one article is labeled with 56 different CCS terms.

Second, there are sibling CCS nodes that contain exactly the same articles. Figure 3.7 gives an example that shows two sibling CCS nodes that have exactly the same articles and thus their parent node has the same number of articles. This raises the question: does CCS need that lower level or just the upper, parent node? Is the parent node enough to capture that computing discipline? For example in Figure 3.7, is the "Network range" CCS term sufficient to represent both "Local area networks" and "Wide area networks"?

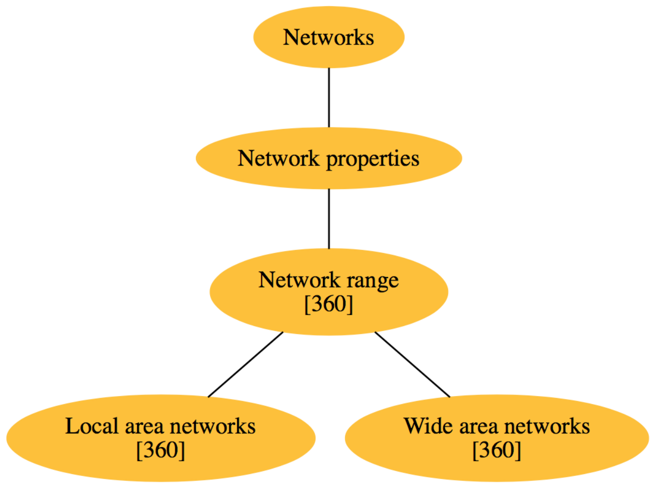


Figure 3.7: Two CCS nodes have exactly the same articles

Third, CCS is a poly-hierarchical structure and thus a CCS node can have multiple parents. Figure 3.8 shows a CCS node "Calculus" that has two parents; one is "Continuous mathematics" while the other one is "Mathematical analysis." Since the records in "Calculus" can be included in either parent node, we didn't include records in "Calculus" when we prepared the "Continuous mathematics" dataset and "Mathematical analysis" dataset for building a classifier for the CCS node "Mathematics of computing."

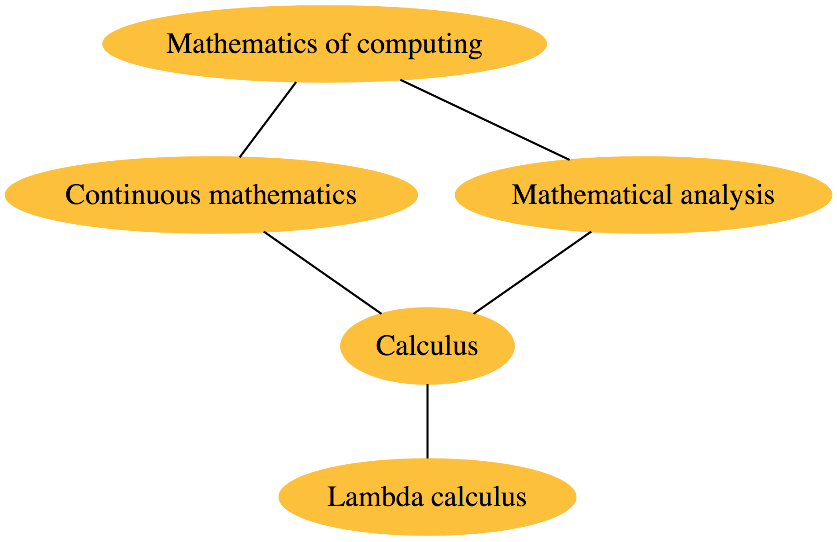


Figure 3.8: A CCS node “Calculus” has two parents

Last, an article can be labeled at any level of the node under a CCS branch. It is not necessary for labels to be associated only with the leaf nodes. For example, an article can be labeled in the middle node of the “Security and privacy” branch tree. Figure 3.9 shows a subtree in the “Security and privacy” branch node tree with the number of articles in each node.

Consider the node “Cryptography”; there are 6,635 articles in this node; however, the total number of articles assigned to any of its children nodes is only 4,569. Therefore there are 2,066 articles labeled in the middle node “Cryptography.”

3.2 Research Method

Our goal is to train a hierarchy of classifiers for computing areas using the ACM dataset. The ACM dataset is classified according to a six-level hierarchy (tree), and each internal node contains one to many children (CCS terms). If a node in the ACM CCS has two children, we can solve a binary classification problem. If a node in the ACM CCS has more than two children, we must solve a multi-classification problem, and use a one vs. all approach to train the classifier. We use the text content in the ACM dataset and consider categorizing its records as a text classification task. We use a bag-of-words approach to transform each of the corpora into a sparse matrix, and apply multiple classification algorithms to train the ACM classifiers.

3.2.1 Data preparation

We prepare a subset dataset for each ACM node for model training. Considering a subtree in the "Security and privacy" branch, for example, as shown in Figure 3.9, we prepare three ACM node datasets for this subtree. These three datasets are:

1. The "Security and privacy" dataset contains three labeled corpora, "Cryptography," "Security services," and "Intrusion/anomaly detection and malware mitigation."
2. The "Cryptography" dataset contains three labeled corpora, "Public key (asymmetric) techniques," "Cryptanalysis and other attacks," and "Mathematical foundations of cryptography."
3. The "Security services" dataset contains two labeled corpora, "Authentication" and "Access control."

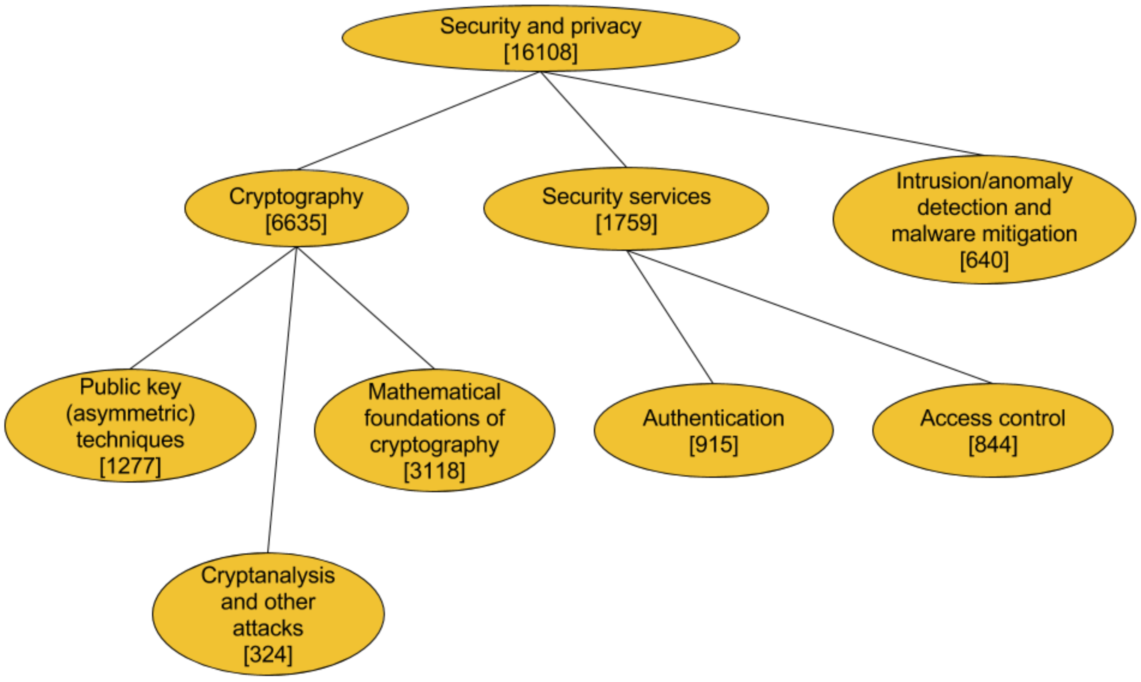
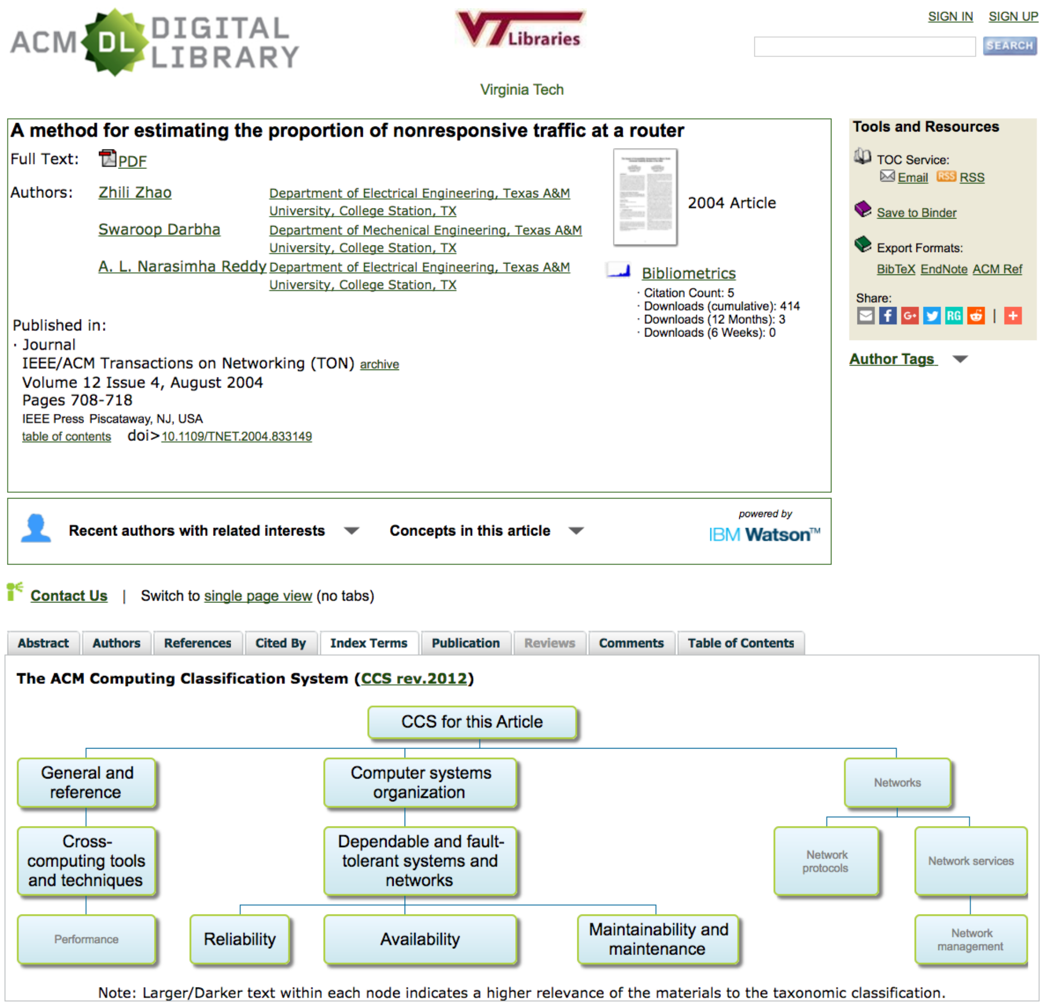

Security and privacy
[16108]
Cryptography
[6635]
Security services
[1759]
Intrusion/anomaly detection and malware mitigation
[640]
Public key (asymmetric) techniques
[1277]
Mathematical foundations of cryptography
[3118]
Authentication
[915]
Access control
[844]
Cryptanalysis and other attacks
[324]

Figure 3.9: ACM CCS "Security and privacy" branch subtree

Note that there are many ACM nodes containing zero articles, or less than 50 articles; we remove any child node which has less than 50 articles. If any of two or more child nodes contain the same records, we merge all these child nodes into one single child node. Thus we use a reduced or pruned version of the CCS tree during our classification study. For example, though in the ACM CCS there are six children under the "Security services" node, we consider only two children, i.e., "Authentication" and "Access control," that contain a good number of articles for training. Further, if a node in the reduced tree contains less than two children, then we don't create a dataset for that node, since no classifier is needed. Figure 3.10 shows an article that has been labeled in three siblings' CCS nodes, "Reliability," "Availability," and "Maintainability and maintenance."



SIGN IN SIGN UP

SEARCH

Virginia Tech

A method for estimating the proportion of nonresponsive traffic at a router

Full Text: PDF

Authors: Zhili Zhao — Department of Electrical Engineering, Texas A&M University, College Station, TX

Swaroop Darbha — Department of Mechenical Engineering, Texas A&M University, College Station, TX

A. L. Narasimha Reddy — Department of Electrical Engineering, Texas A&M University, College Station, TX

2004 Article

Bibliometrics
- Citation Count: 5
- Downloads (cumulative): 414
- Downloads (12 Months): 3
- Downloads (6 Weeks): 0

Published in:
- Journal

IEEE/ACM Transactions on Networking (TON) archive
Volume 12 Issue 4, August 2004
Pages 708-718
IEEE Press Piscataway, NJ, USA
table of contents doi>10.1109/TNET.2004.833149

Tools and Resources

TOC Service: Email RSS

Save to Binder

Export Formats: BibTeX EndNote ACM Ref

Share:

Author Tags

Recent authors with related interests — Concepts in this article — powered by IBM Watson™

Contact Us | Switch to single page view (no tabs)

Abstract | Authors | References | Cited By | Index Terms | Publication | Reviews | Comments | Table of Contents

The ACM Computing Classification System (CCS rev.2012)

CCS for this Article
- General and reference
  - Cross-computing tools and techniques
    - Performance
- Computer systems organization
  - Dependable and fault-tolerant systems and networks
    - Reliability
    - Availability
    - Maintainability and maintenance
- Networks
  - Network protocols
  - Network services
    - Network management

Note: Larger/Darker text within each node indicates a higher relevance of the materials to the taxonomic classification.

Figure 3.10: An article that has been labeled with multiple sibling ACM CCS nodes

In each ACM node, each child is a class that contains a corpus. A corpus is a collection of documents. Each document contains an article title and abstract. We use a top-down approach and prepare all the ACM node datasets. Table 3.3 shows the number of ACM node datasets in the eleven ACM branches.

Table 3.3: Number of ACM node datasets in each of the eleven ACM branches

ACM tag	**CCS1**	**CCS2**	**CCS3**	**CCS4**	**CCS5**	**CCS6**
# of datasets	15	24	5	5	9	19
ACM tag	**CCS7**	**CCS8**	**CCS9**	**CCS10**	**CCS11**	**Total**
# of datasets	28	15	9	14	28	171

3.2.2 Data cleaning

Although all the text is from the ACM DL dataset, there still is much noise and irrelevant information. We use these steps to clean the dataset:

1. Remove combining words, such as “describeabout,” or “Thisresearchis.” We remove any word that is not an English word.
2. Remove HTML tags, special characters, and math formula and equation symbols.
3. Tokenize the text and filter out stop words using the set of English stop words from the Glasgow Information Retrieval Group [87].
4. Stem the text using the NLTK stemmer interface [88]. A stemming algorithm is a computational procedure to reduce derivational and inflectional forms of a word to a common base form [89].

3.2.3 Imbalanced data

As shown in Figure 3.9, each ACM node dataset is an imbalanced dataset. This can lead to problems when accuracy is used for evaluation, and training occurs with the full imbalanced dataset. For example, consider a binary classification problem with 1000 instances: 10 instances in class A and 990 instances in class B. A simple classifier would assign all documents to class B, and achieve classifier accuracy of 99.9%. However, this would yield poor prediction results if new data were all in class A. Our approach, addressing this situation, is to use under-sampling to balance each class in the dataset.

3.2.4 Feature extraction

We used the bag-of-words model to represent our ACM dataset. The bag-of-words model learns a vocabulary from the corpus of documents, then models each document by counting the number of times each word appears. Thus, a corpus of documents is represented by a sparse matrix, one row per document and one column per word occurring in the corpus. The bag-of-words model is widely used in text classification.

There are some words that occur many times in the corpus of documents, but such words often fail to contain meaningful information about the actual content, and thus affect the performance of the classifier. The common term weighting approach used in information retrieval and document classification is the term frequency-inverse document frequency (TF-IDF) transform. Here TF means term frequency, while IDF means inverse document frequency. TF-IDF stands for term frequency (TF) times inverse document frequency (IDF).

In the ACM dataset, a document is an article that contains title and abstract. The tf(t, d) is the term frequency of feature t in article d. IDF is calculated using the following equation:

$$idf(t) = log\frac{1+N}{1+df(d,t)} + 1 \tag{3.1}$$

N is the total number of articles, and $df(d,t)$ is the number of articles that contain term t. The TF-IDF value is:

$$tf{-}idf(t,d) = tf(t,d) \times idf(t) \tag{3.2}$$

3.2.5 Feature selection

In order to train classifiers with high performance and reduce overfitting, we used the following feature selection techniques. Besides using the bag-of-words model, we also use an n-gram model. Specifically, we use bi-grams (n=2) and tri-grams (n=3) in our experiment. During preprocessing of the dataset, we compute Chi-square statistics between each non-negative feature and class and select some number of the highest scoring features for training. We use three different types of penalties in our linear model to avoid overfitting. The three penalties are L1, L2, and Elastic Net. The Elastic Net is the convex combination of L2 and L1.

3.2.6 Classifiers

The classifiers used in the experiments are Ridge regression [68], Perceptron [133], k-nearest neighbors (KNN) [58], Random Forest [86], SVM [65] [71], multinomial Naïve Bayes [75], and BernoulliNB [93]. These classifiers can efficiently handle sparse matrices for text classification.

3.2.7 Evaluation metrics

We used the following measures to evaluate the performance of the ACM classifiers: accuracy, precision, recall, and F1-score. We used a confusion matrix to investigate the prediction errors. Table 3.4 shows a confusion matrix table.

Table 3.4: Confusion matrix table

		Predicted class	
		Positive	Negative
Actual class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

$$F1{-}score = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)} \tag{3.6}$$

3.3 Experimental setup

The proposed system architecture is shown in Figure 3.11. First, we randomly split the ACM node dataset into 80% training and 20% testing sets. The system then extracts a full set of features. During the feature selection stage, we use regularization to reduce the generation error by introducing a penalty for large individual weights and thus reduce the complexity of the model. Depending on the algorithms, we use L1, L2, and Elastic Net (combination of L1 and L2) penalties.

$$\text{n-dimensional vector} : w = [w_1, w_2, ..., w_n] \tag{3.7}$$

$$L1_{norm} : \sum_{i=1}^{n} |w_i| \tag{3.8}$$

$$L2_{norm} : \sqrt{\sum_{i=1}^{n} |w_i|^2} \tag{3.9}$$

We use a one-vs-all strategy if the ACM node dataset contains more than two classes. After the classifier model is built, the system uses a testing set to evaluate the model performance.

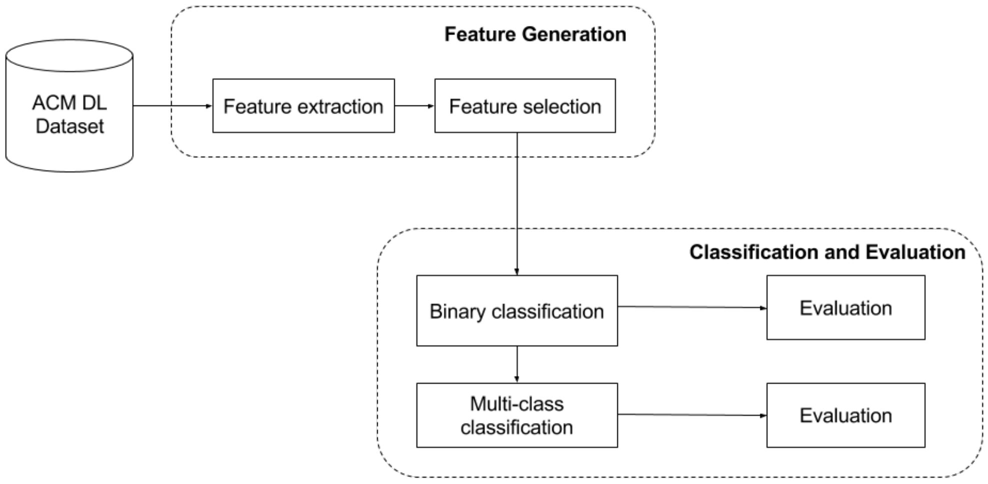


Figure 3.11: Data flow for building an ACM classifier

In order to find the optimal features and parameters for an algorithm and achieve the best performance result, we use the following steps, shown in Figure 3.12. After a built training model is evaluated, we use grid search with initial 3-fold cross-validation and followed by 10-fold cross-validation to acquire the optimal features and parameters. We use the k-fold cross-validation technique; the training dataset is randomly divided into k approximately equal sized groups, also called folds. Among these folds, one of the folds is selected as the validation set while the other $k-1$ folds are used as training set. The cross-validation process is then repeated k times, with each of the k folds used exactly once as the validation set. This process results in k estimates which are then averaged out and produce a final estimation [110]. A grid search is a traditional way of performing hyperparameter optimization in machine learning. Once the best performance is identified, the system saves the built model.

For the built models with F1-score below 0.8, the system did a document similarity check on all classes in the dataset and identified if there are any duplicated articles among these classes. If a duplication is detected, we update the dataset and train the new classifier model.

If there is no duplication found, the system use grid search to find the best performance and then saves the built model.

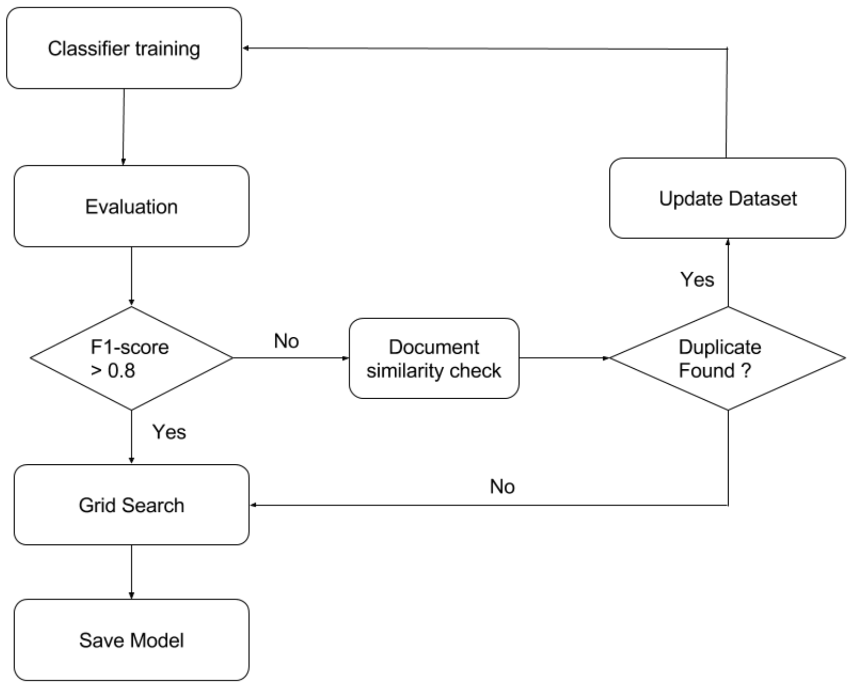


Figure 3.12: Workflow for training an optimal ACM classifier

The system uses several algorithms with different penalties to train the classifiers; these algorithms are listed in Table 3.5.

We used the cosine similarity to measure the document similarity between the classes in the ACM node dataset. Here A and B are the two vectors used to calculate the distance; A_i and B_i are the documents associated with vectors A and B, respectively; and n is the number of documents in the dataset.

$$\text{Document similarity} = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.10}$$

Finally, after the ACM node dataset is fully processed, we ranked the classification algorithms

Table 3.5: Algorithms used for building ACM classifiers

Algorithm Tag	**Algorithms**
A1	Ridge Classifier
A2	Passive-Aggressive
A3	Linear SVM with SGD training and Elastic Net penalty
A4	Linear SVM with SGD training and L1 penalty
A5	Linear SVM with SGD training and L2 penalty
A6	Linear Support Vector Classification with L1 penalty
A7	Linear Support Vector Classification with L2 penalty

by the F1-score, and identified the algorithms that perform best in each ACM branch.

3.4 Experimental results

ACM classifiers Initially, we built a total of 171 ACM classifiers from our prepared ACM node dataset in the experiment. We discuss one ACM node classifier result here, for demonstration purposes. "Software and application security" is a second level ACM CCS node in the "Security and privacy" ACM CCS branch. It has two child nodes, "Software security engineering" and "Domain-specific security and privacy architectures." The F1-score for different classifiers is shown in Table 3.6.

When we identified the algorithm with the best result, we used grid search and supplied different choices for feature selection, stop words, Chi-square, unigram, bigram, trigram, and penalty values; we generated the optimal features and parameters. The final F1-score of this classifier is 0.951. Table 3.7 shows the classification report.

Several ACM node classifiers have poor F1-score. We calculated the cosine similarity between each class in the dataset and identified if any two classes have a high cosine similarity score. We found that many ACM nodes contain two classes with a very high cosine similarity score.

Table 3.6: Software and application security classifier results

Algorithm Tag	**F1-score**
A3	0.919
A7	0.919
A1	0.919
A2	0.911
A4	0.911
A5	0.911
A6	0.886

Table 3.7: The classification report for “Software security engineering” classifier

CCS node	**Precision**	**Recall**	**F1**
Software security engineering	0.95	0.97	0.96
Domain-specific security and privacy architectures	0.95	0.91	0.93

Consider the ACM CCS node “Retrieval tasks and goals” for example; the performance of the classifier is very poor, i.e., 0.244. We then performed the document similarity check and found that “Document filtering” and “Information extraction” are used to label exactly the same articles. Figure 3.13 shows the “Retrieval tasks and goals” node tree, both “Document filtering” and “Information extraction” have exactly the same number of articles. After we merged “Document filtering” and “Information extraction” into a new node, “Document filtering/Information extraction”, and updated the dataset and trained a new classifier, the classifier performance was improved from 0.244 to 0.877. Table 3.8 shows the classification report.

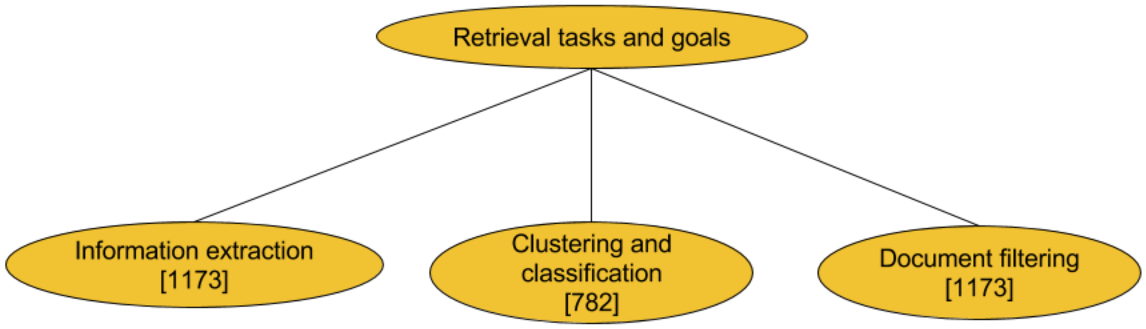

Retrieval tasks and goals
Information extraction [1173]
Clustering and classification [782]
Document filtering [1173]

Figure 3.13: "Retrieval tasks and goals" node tree

Table 3.8: The classification report for "Retrieval tasks and goals" classifier

CCS node	Precision	Recall	F1
Document filtering/Information extraction	0.83	0.93	0.88
Clustering and classification	0.88	0.73	0.80

Through several iterative processes as shown in Figure 3.12 over the entire ACM node dataset, we constructed a tree of ACM classifiers for each ACM CCS branch. Figure 3.14 shows a classifier tree for the "Security and privacy" branch.

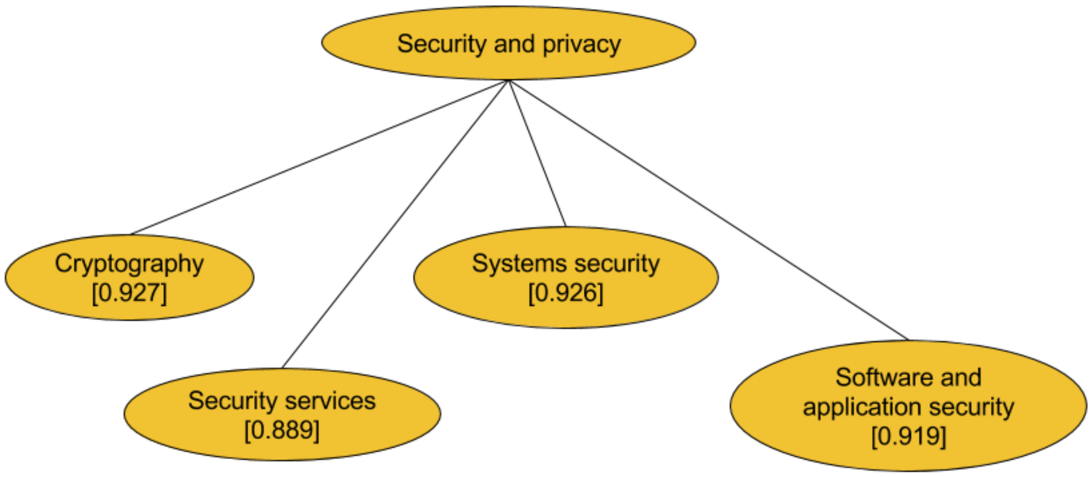

Security and privacy
Cryptography [0.927]
Systems security [0.926]
Security services [0.889]
Software and application security [0.919]

Figure 3.14: "Security and privacy" branch classifier tree

We further examined all the classifier results and identified whether there are algorithms that can be used to build classifiers for every node in an ACM CCS branch. For each ACM CCS branch, we identified the top two algorithms that outperform other algorithms. Table 3.9 shows the results.

Table 3.9: Top two algorithms in each ACM CCS branch

ACM tag	**Top two algorithms**	**ACM tag**	**Top two algorithms**
CCS1	A1, A7	CCS7	A3
CCS2	A5, A7	CCS8	A5, A6
CCS3	A4	CCS9	A7
CCS4	A4, A5	CCS10	A3, A5
CCS5	A1, A6	CCS11	A6
CCS6	A2, A4		

We further examined all the classifier results and determined if there are algorithms that can be used to train the entire set of branch nodes. For each ACM CCS branch, we identify the top two algorithms that outperform other algorithms. Table 3.10 shows the results. We further generalized that Linear Support Vector Classification and Linear SVM with SGD training with a penalty, can be used to train all of the ACM dataset, as shown in Table 3.10.

Finally, we produced a set of ACM classifiers with good F1-scores to be used to predict classes for ACM articles.

3.5 Summary

In this chapter, we presented our work on analyzing the ACM dataset, and developing a system architecture to build ACM classifiers. Experiments were conducted on the entire ACM dataset, and the results showed that Linear Support Vector Classification and Linear SVM with SGD training outperformed the other classification algorithms, with performance

Table 3.10: Algorithms for ACM dataset

Algorithms	**ACM CCS branch**
Linear Support Vector Classification using (L1 or L2) penalty	Mathematics of computing Information systems Human-centered computing Applied computing Computer systems organization
Linear SVM with SGD training using (L1, L2 or Elastic-Net) penalty	Information systems Security and privacy Networks Theory of computation Applied computing Hardware Computing methodologies Software and its engineering

that was comparable and effective. Using a penalty in our linear model can avoid overfitting. Bigrams were prominent features for classifier building among most ACM node datasets. We found that the F1-score is highly dependent on the articles in each node dataset. An ACM classifier tree with high performance classifiers for these major ACM categories is constructed and can be used to classify future articles.

We identified several issues in the current ACM dataset, such as missing title and abstract, overused CCS term, high document similarity between ACM CCS nodes in the same level, and duplicated articles in ACM CCS in the same level. These issues, identified in this study, might be employed to improve the ACM DL and ACM CCS. We found that when the document similarity is above 0.7, then the result is poor classifier performance. In the future, we can further investigate how high document similarity affected the ACM classifiers.

Finally, we chose the ACM CCS node dataset that contributed to high classifier performance

Table 3.11: Number of selected ACM node datasets in each of the eleven ACM branches

ACM tag	**CCS1**	**CCS2**	**CCS3**	**CCS4**	**CCS5**	**CCS6**
# of datasets	10	16	4	4	7	15
ACM tag	**CCS7**	**CCS8**	**CCS9**	**CCS10**	**CCS11**	**Total**
# of datasets	19	11	5	12	22	125

to use in the next chapter. Table 3.11 shows the number of ACM CCS node datasets that have been selected in each of the eleven ACM CCS branches. This reflects two removals: 1) ACM CCS node datasets that contribute to low classifier performance, and 2) ACM CCS node datasets that contain only one child node.

For example, as illustrated earlier in Figure 3.7, the original "Network range" node dataset contains two children nodes. However, the articles in these two children nodes are exactly the same, and thus we did not build a classifier for this ACM CCS node and did not select this ACM CCS node dataset.

Ultimately, then, 125 node datasets were used, and 125 classifiers were built for the pruned ACM CCS node tree. This tree and set of classifiers can be applied to other works in the ACM digital library. As explained in the next chapter, the 125 classifiers also can be adapted using transfer and adaptive learning, so resources in other domains, like YouTube and SlideShare, can have resources suitably classified according to the ACM CCS.

Chapter 4

Collection building for educational resources

In this chapter, we present our work on retrieving educational records, highly related to computing, from Web 2.0. We develop a framework to get resources from YouTube and SlideShare and compare three different approaches to retrieve metadata records that are relevant to computing. We extract features from the description content of these metadata records and measure the similarity with the ACM dataset in a computing topic. We identify the best approach based on our experimental result on acquiring new educational resources that are relevant to particular topics in computing. The resulting new datasets include promising candidates to be added into our educational DL, Ensemble. The framework shows that new educational resources are obtained, and increases the DL collections with broader coverage, which addresses our research question about building collections.

4.1 Introduction

In recent years, many online learning sites have been developed that provide educational resources for the general public. The majority of these sites have been developed by universities and technology companies. If an educational DL wants to have a metadata collection about a particular online learning site, the curator should work with the technical staff in that site to develop a channel to acquire these resources. The number of such collections is limited by the available staff time as well as the priorities and preferences of a potentially large number of possible sites. A new way of acquiring online educational resources is needed. In this study, our goal is to develop a framework to increase the educational collection for the Ensemble DL.

4.1.1 Ensemble architecture

The Ensemble DL is an educational DL that contains multiple internal services to manage educational content and provide an integrated service for users. Figure 4.1 shows the current Ensemble architecture. Ensemble gathers educational resources through multiple means; the major collection building method is through OAI-PMH. We customize an OAI data harvester from jOAI [50] to harvest metadata records from multiple OAI data providers which we work with from several educational digital libraries and research projects in computing education. Figure 4.2 shows this OAI data harvester interface.

After we harvest these metadata records, we use a metadata mapper and transformer to convert each metadata record into a customized record serving as an individual page in the Ensemble portal for users to view, comment, rate, and tag. Users can view the information in that page and go to the content provider website to read the original educational content. Each of these metadata records can represent a class syllabus, a course assignment, a programming tutorial Web page, a class lecture video recording, a blog article, a forum post, or another kind of learning object.

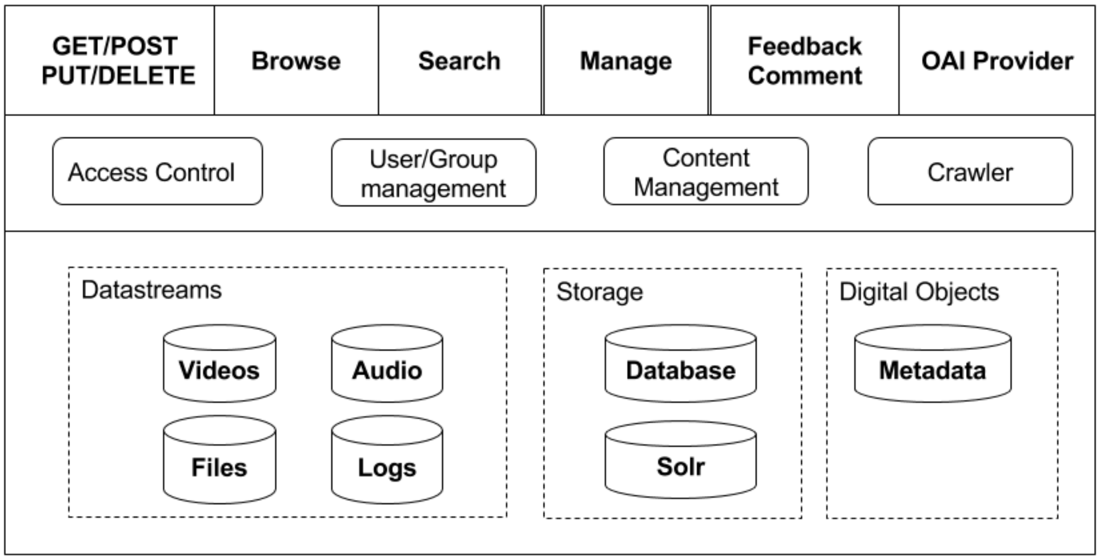
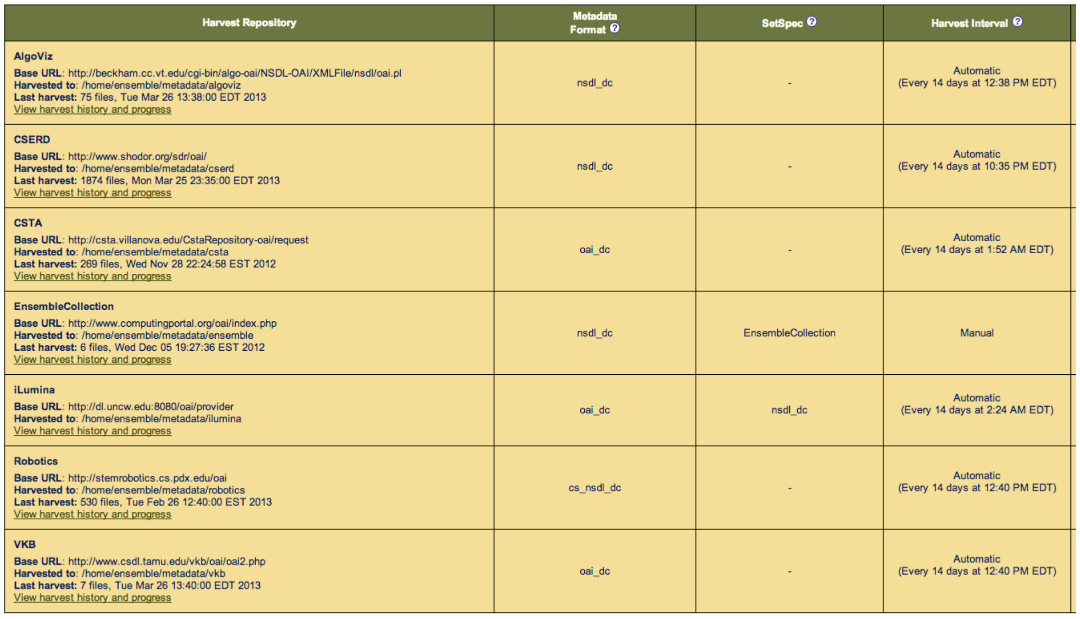


Figure 4.1: Ensemble architecture



Harvest Repository	Metadata Format	SetSpec	Harvest Interval
AlgoViz **Base URL**: http://beckham.cc.vt.edu/cgi-bin/algo-oai/NSDL-OAI/XMLFile/nsdl/oai.pl **Harvested to**: /home/ensemble/metadata/algoviz **Last harvest:** 75 files, Tue Mar 26 13:38:00 EDT 2013 View harvest history and progress	nsdl_dc	-	Automatic (Every 14 days at 12:38 PM EDT)
CSERD **Base URL**: http://www.shodor.org/sdr/oai/ **Harvested to**: /home/ensemble/metadata/cserd **Last harvest:** 1874 files, Mon Mar 25 23:35:00 EDT 2013 View harvest history and progress	nsdl_dc	-	Automatic (Every 14 days at 10:35 PM EDT)
CSTA **Base URL**: http://csta.villanova.edu/CstaRepository-oai/request **Harvested to**: /home/ensemble/metadata/csta **Last harvest:** 269 files, Wed Nov 28 22:24:58 EST 2012 View harvest history and progress	oai_dc	-	Automatic (Every 14 days at 1:52 AM EDT)
EnsembleCollection **Base URL**: http://www.computingportal.org/oai/index.php **Harvested to**: /home/ensemble/metadata/ensemble **Last harvest:** 6 files, Wed Dec 05 19:27:36 EST 2012 View harvest history and progress	nsdl_dc	EnsembleCollection	Manual
iLumina **Base URL**: http://dl.uncw.edu:8080/oai/provider **Harvested to**: /home/ensemble/metadata/ilumina View harvest history and progress	oai_dc	nsdl_dc	Automatic (Every 14 days at 2:24 AM EDT)
Robotics **Base URL**: http://stemrobotics.cs.pdx.edu/oai **Harvested to**: /home/ensemble/metadata/robotics **Last harvest:** 530 files, Tue Feb 26 12:40:00 EST 2013 View harvest history and progress	cs_nsdl_dc	-	Automatic (Every 14 days at 12:40 PM EDT)
VKB **Base URL**: http://www.csdl.tamu.edu/vkb/oai/oai2.php **Harvested to**: /home/ensemble/metadata/vkb **Last harvest:** 7 files, Tue Mar 26 13:40:00 EDT 2013 View harvest history and progress	oai_dc	-	Automatic (Every 14 days at 12:40 PM EDT)

Figure 4.2: OAI data harvester interface

The Ensemble portal provides several services for users to submit educational resources.

Table 4.1: User contributed collections

User contributed collections	**# of records**
Ensemble Communities Content	128
Ensemble User Contributions	30
TECH	34
The Beauty and Joy of Computing	79
Stanford Nifty Collection	74

First, users can submit individual items of educational content through the user submission service's form. Second, users can join Ensemble community groups and post educational content in each group's forum. Finally, Ensemble administrators and curators can manually create educational content that is added into Ensemble collections. The content can be in various forms, but all records lead to a similar presentation layout. Table 4.1 shows a portion of user contributed records in Ensemble.

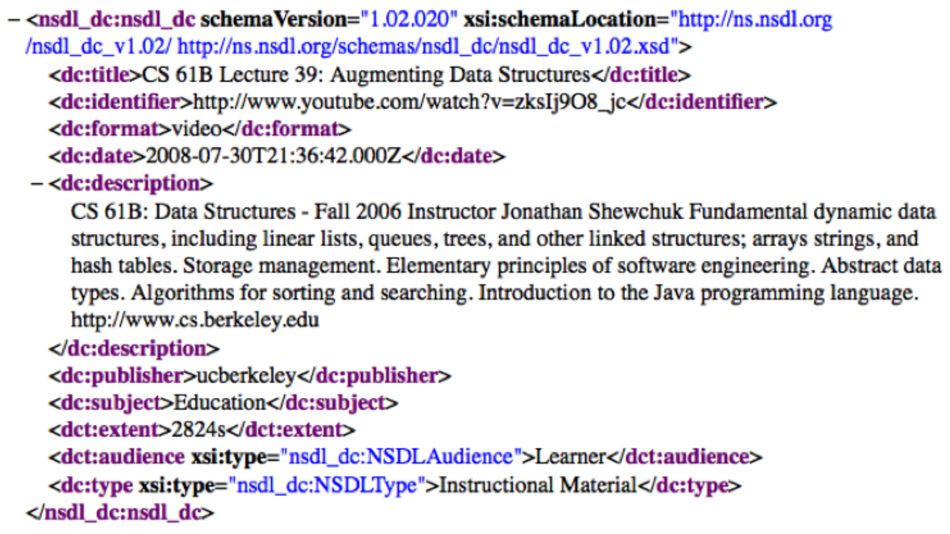
After we implemented these services to collect educational resources from users, partner research projects, and institutions, we focused on Web 2.0 for additional educational resources.

4.1.2 Educational resources in Web 2.0

Web 2.0 contains much user-generated content, including educational resources. An online educational resource is usually in the form of document (HTML, PDF, .ppt, etc.), image, audio, or video. SlideShare (https://www.slideshare.net) is the most popular platform for users to share documents and presentation slides, with 18 million uploads in 40 content categories. SlideShare has education and technology categories that contain many computer science related slides uploaded by students and faculty in colleges and universities.

YouTube (https://www.youtube.com) is the major video sharing website. YouTube has more than 1.2 billion videos and over one billion users. YouTube has an education channel that contains educational videos uploaded by accredited universities. For example, there were 4,405 videos in MIT OpenCourseWare, 3,251 videos from Stanford, and 1,975 videos in UCI Open in July 2017.

Moreover, both services contain many videos and documents with high educational value content uploaded by universities, technology companies, and professionals. Therefore, we choose these two online sharing sites as our target content providers for online educational resources in this study. Both sites provide public access APIs for us to access their educational materials in structured formats, such as JSON or XML. Accordingly, we use the approach of submitting multiple focused queries to these APIs, to obtain records likely to be both educational and related to computing. We implement a tool to transform these metadata records into a metadata record shown in Ensemble. Figure 4.3 shows a YouTube metadata record in the Ensemble portal.



```
– <nsdl_dc:nsdl_dc schemaVersion="1.02.020" xsi:schemaLocation="http://ns.nsdl.org
  /nsdl_dc_v1.02/ http://ns.nsdl.org/schemas/nsdl_dc/nsdl_dc_v1.02.xsd">
    <dc:title>CS 61B Lecture 39: Augmenting Data Structures</dc:title>
    <dc:identifier>http://www.youtube.com/watch?v=zksIj9O8_jc</dc:identifier>
    <dc:format>video</dc:format>
    <dc:date>2008-07-30T21:36:42.000Z</dc:date>
  – <dc:description>
      CS 61B: Data Structures - Fall 2006 Instructor Jonathan Shewchuk Fundamental dynamic data
      structures, including linear lists, queues, trees, and other linked structures; arrays strings, and
      hash tables. Storage management. Elementary principles of software engineering. Abstract data
      types. Algorithms for sorting and searching. Introduction to the Java programming language.
      http://www.cs.berkeley.edu
    </dc:description>
    <dc:publisher>ucberkeley</dc:publisher>
    <dc:subject>Education</dc:subject>
    <dct:extent>2824s</dct:extent>
    <dct:audience xsi:type="nsdl_dc:NSDLAudience">Learner</dct:audience>
    <dc:type xsi:type="nsdl_dc:NSDLType">Instructional Material</dc:type>
  </nsdl_dc:nsdl_dc>
```

Figure 4.3: A YouTube metadata record

Thus, as explained below, in order to collect relevant resources from YouTube and SlideShare, we propose a topic modeling approach described in the following sections. That allows us to prepare and submit the multiple focused queries required to obtain resources that are educational and related to each of the 125 categories of computing for which we built classifiers as explained in Chapter 3. As a proof of concept, and to demonstrate that the approach has high quality, we do this for 11 (one from each top level CCS node) of the 125.

4.2 Research Method

The goal of this study is to gather educational records in computing from YouTube and SlideShare. Our proposed design was to answer the following questions:

1. How to acquire educational resources that are relevant to computing, from YouTube and SlideShare?
2. How to generate keywords that are useful for searching computing related resources from YouTube and SlideShare?
3. How to determine resources from YouTube and SlideShare that could contain some set of features that are similar to the ACM dataset?

To answer these questions, we start from building crawlers to retrieve records from YouTube and SlideShare. We select ACM CCS as our pool of target computing topics, since it reflects the state of the art in the computing discipline. That being said, we aim to gather educational records from YouTube and SlideShare that relate to the computing topics in ACM CCS. For example, some course videos in YouTube are about "Supervised learning". Moreover, the gathered records contain some features that indicate they are similar to the articles under the ACM CCS node "Supervised learning."

4.2.1 Query construction

In order to generate applicable queries to retrieve related records in computing, we use three different methods to construct the queries for our study. The first method is to use an ACM CCS term as a query. For example, we use an ACM CCS term – "Mathematical foundations of cryptography" – as a query to search the records from YouTube and SlideShare. The second method is to use the topics generated by topic modeling as the queries. We generate topics from the ACM dataset. The third method is to use the most informative features (words) in the ACM CCS classifier (built for a node in CCS) as the query. Table 4.2 shows a list of generated queries from a selected ACM CCS node – "Network types" – using these three methods.

Table 4.2: List of queries generated by three different methods

CCS terms	**Classifier**	**Topic modeling**
Mobile networks	bluetooth 802 sensor mobile wireless	sensor base mobil scheme network
Public Internet	links red tcp ip internet	network switch base algorithm propos
Packet-switching networks	buffer packet switched switching	packet buffer traffic switch network

The ACM CCS terms constitute a fixed number of computing concepts. Thus the number of queries we can generate is limited by how many CCS terms are in that selected ACM CCS node. Take an ACM CCS node "Retrieval tasks and goals" for example, shown in Figure 3.13; This node contains three children nodes, "Clustering and classification", "Document filtering" and "Information extraction" and thus we can generate three queries, the same as the number of children CCS terms.

To expand on these sets of queries, since each ACM CCS node contains hundreds to thousands of articles, we can generate a number of topics to be used as queries. We describe this procedure in the next section.

As explained in our presentation in Chapter 3 of the work on building ACM classifiers, we use linear SVM. In linear SVM, a hyperplane is represented by weights identified by giving the coordinates of a vector which is orthogonal to that hyperplane. The absolute size of the coefficient relative to the other ones gives an indication of how important the feature was for the separation. For example, if only the first coordinate is used for separation, w will be of the form $(x, 0)$ where x is some non-zero number and then $|x| > 0$. Thus the most informative features mean the features that are most useful for separating the data [67]. In this study, we used the top 5 informative features per class in a classifier as shown in the second column of Table 4.2.

4.2.2 Topic modeling

We use a commonly used topic modeling algorithm, Latent Dirichlet Allocation (LDA) [16], to generate underlying topics from the ACM dataset. LDA takes a corpus as input and then generates the topics. A corpus is a collection of documents. We prepared several corpora from the ACM dataset and each corpus is a collection of articles in one ACM CCS node. We apply LDA to these corpora and generate topics. The workflow is shown in Figure 4.4.

An ACM node dataset contains a set of text documents, and the content in each text document is an article's title and abstract. We first tokenize the document string, remove any stop words from the tokens, and finally stem all the tokens. Second, we use bag-of-words to represent the document in a vector form. We assigned a unique integer ID to all words appearing in the corpus and create a mapping called a dictionary. Third, we count the number of occurrences of each distinct word and convert the word to its integer ID and return the result as a sparse vector for that corpus. Finally, we use the vectors as input and apply the LDA algorithm to create an LDA model [69] and generate a set of topics. For each ACM node dataset, we generate 20 topics, where each topic contains 5 words.

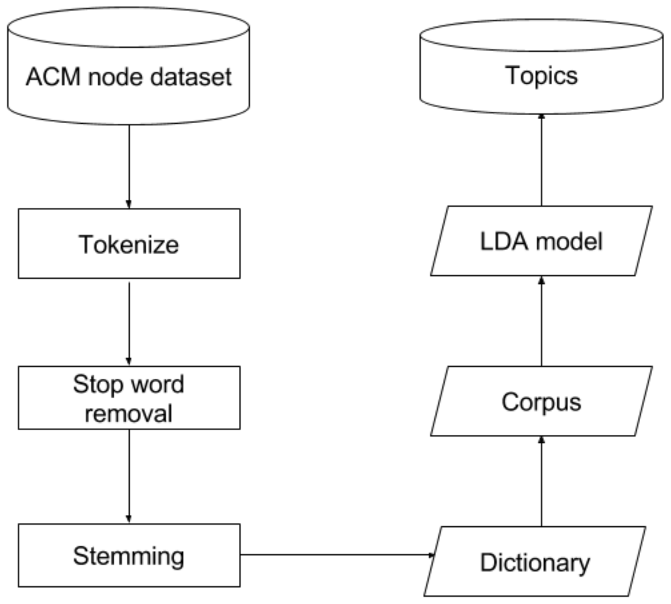
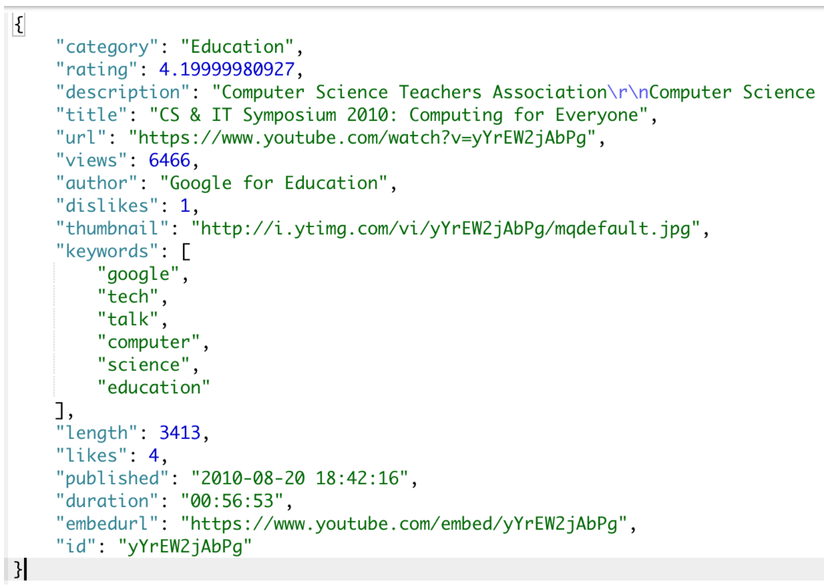


Figure 4.4: A workflow for generating topics using LDA

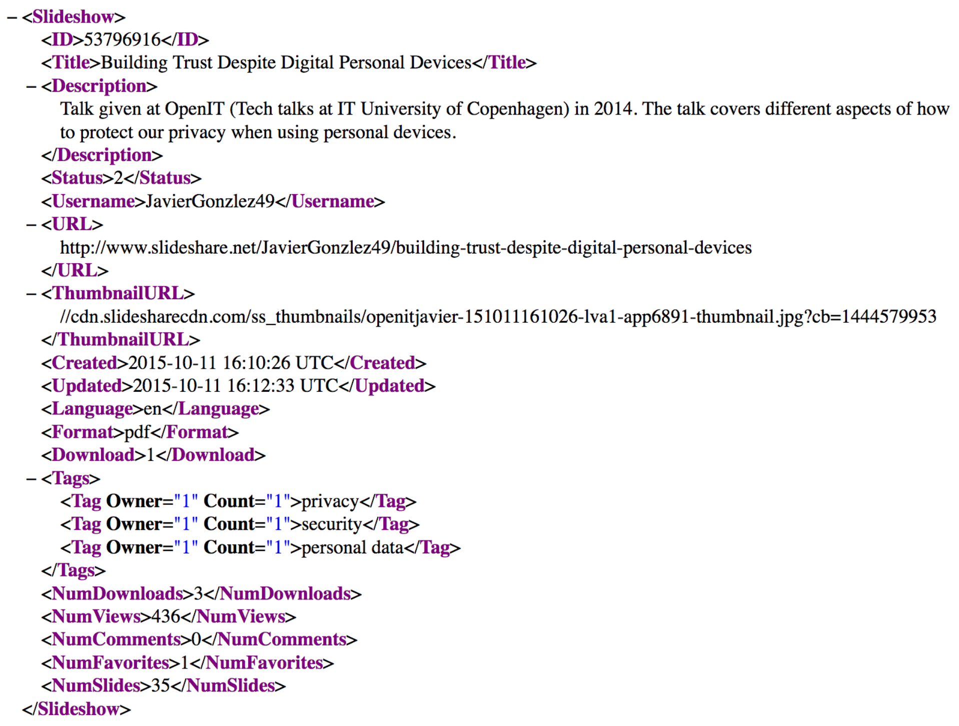
4.2.3 Crawlers

YouTube and SlideShare both provide an official API for developers to access their metadata records. We use the YouTube Data API [4] to implement a YouTube crawler to retrieve record metadata from YouTube. Figure 4.5 shows a search result response from YouTube in the JSON format. Each record has many attributes; we use title and description information for our study. Similar to YouTube, we use the SlideShare Developer API [3] to implement a SlideShare crawler to retrieve record metadata from SlideShare. Figure 4.6 shows a search result response from SlideShare in the XML format. We also use SlideShare title and description information for our study.



```
{
    "category": "Education",
    "rating": 4.19999980927,
    "description": "Computer Science Teachers Association\r\nComputer Science
    "title": "CS & IT Symposium 2010: Computing for Everyone",
    "url": "https://www.youtube.com/watch?v=yYrEW2jAbPg",
    "views": 6466,
    "author": "Google for Education",
    "dislikes": 1,
    "thumbnail": "http://i.ytimg.com/vi/yYrEW2jAbPg/mqdefault.jpg",
    "keywords": [
        "google",
        "tech",
        "talk",
        "computer",
        "science",
        "education"
    ],
    "length": 3413,
    "likes": 4,
    "published": "2010-08-20 18:42:16",
    "duration": "00:56:53",
    "embedurl": "https://www.youtube.com/embed/yYrEW2jAbPg",
    "id": "yYrEW2jAbPg"
}
```

Figure 4.5: A search result response from YouTube in JSON format



```
– <Slideshow>
    <ID>53796916</ID>
    <Title>Building Trust Despite Digital Personal Devices</Title>
  – <Description>
      Talk given at OpenIT (Tech talks at IT University of Copenhagen) in 2014. The talk covers different aspects of how
      to protect our privacy when using personal devices.
    </Description>
    <Status>2</Status>
    <Username>JavierGonzlez49</Username>
  – <URL>
      http://www.slideshare.net/JavierGonzlez49/building-trust-despite-digital-personal-devices
    </URL>
  – <ThumbnailURL>
      //cdn.slidesharecdn.com/ss_thumbnails/openitjavier-151011161026-lva1-app6891-thumbnail.jpg?cb=1444579953
    </ThumbnailURL>
    <Created>2015-10-11 16:10:26 UTC</Created>
    <Updated>2015-10-11 16:12:33 UTC</Updated>
    <Language>en</Language>
    <Format>pdf</Format>
    <Download>1</Download>
  – <Tags>
      <Tag Owner="1" Count="1">privacy</Tag>
      <Tag Owner="1" Count="1">security</Tag>
      <Tag Owner="1" Count="1">personal data</Tag>
    </Tags>
    <NumDownloads>3</NumDownloads>
    <NumViews>436</NumViews>
    <NumComments>0</NumComments>
    <NumFavorites>1</NumFavorites>
    <NumSlides>35</NumSlides>
  </Slideshow>
```

Figure 4.6: A search result response from SlideShare in XML format

4.2.4 Document similarity

It is not known whether the records obtained by crawling are relevant with regard to the particular ACM CCS node under consideration. What is needed is an efficient and effective way to determine this for a particular record from the YouTube and SlideShare collections.

Our approach is to build a low dimensional vector space for each of the ACM CCS nodes – low dimensional so that efficient distance computations are feasible. The construction of such spaces need only be done once for each ACM CCS node, and includes placing each of the ACM digital library records assigned to that node into the space.

A technique that captures well the semantics of the node is desired; we turn to the LSI algorithm [49]. LSI allows reducing the number of dimensions; for example for a million documents it suffices to have 300-500 dimensions [22]. Pilot experiments were undertaken and show that it suffices to have a 2-dimensional LSI vector space. When a YouTube or SlideShare record is mapped into this space, and is fairly similar to a large proportion of the points in that space (i.e., to the ACM digital library records for that node), then we have high confidence that the YouTube or SlideShare record relates to that ACM CCS node's topic.

In order to determine the similarity between records from the YouTube and SlideShare and ACM node datasets, we transform the record's metadata description into a query and compute the cosine similarity between this query (a specific document) and the ACM node dataset (a set of documents). In both YouTube and SlideShare, we then use title and description to construct the query, fit that query into that LSI vector space, and convert the query into a vector, V_q. In the future, we could use a larger number of latent dimensions and select a higher level ACM CCS node with a larger number of articles to conduct similar experiments. Among the eleven ACM CCS branches, "Mathematics of computing", "Information systems", "Human-centered computing", "Theory of computation", "Computing methodologies", "Applied computing", "Computer systems organization", and "Software and its engineering" have ACM CCS nodes that contain more than 10,000 articles.

Last, we use the cosine similarity function provided in Gensim [111] to compute the cosine similarity score between V_q and the crawled record in the LSI vector space. The cosine similarity function returns a score in the range from -1 to 1. A score close to 1 means that two documents are very similar. A record is compared with all the articles in the ACM node dataset. We define a threshold for the cosine similarity score of 0.6, calculate how many articles in the ACM node dataset have the cosine similarity score greater than or equal to 0.6 (the threshold we have identified through pilot studies to be suitable), and output the result. For example, if an ACM node dataset contains 500 articles, and there are 400 articles that have cosine similarity score greater than or equal to 0.6, then the output value is 0.8.

4.3 Experimental setup

This experiment is designed to verify that this proposed framework can acquire educational resources in various computing topics from YouTube and SlideShare. We examine what query generation strategies are best for this purpose and what level of similarity exists between the acquired resources and the ACM dataset.

4.4 Experimental design

4.4.1 ACM datasets

In Chapter 3, we described how we built a set of ACM CCS classifiers with high F1-scores from the ACM node datasets in the eleven ACM CCS branches. These selected ACM node datasets are our target pool to be used in this study; we selected 125 ACM node datasets from the eleven ACM CCS branches based on our analysis. For simplicity, we choose one ACM node dataset from each of the eleven ACM CCS branches in this experiment. We use these selected eleven ACM node datasets among the target pool and perform two tasks:

1. We apply our proposed method to these datasets and generate queries for our crawlers.
2. We use the implementations provided in Gensim [111] to create a 2-dimensional LSI vector space from each dataset to use for measuring similarity.

4.4.2 Experiment workflow

The experiment workflow is shown in Figure 4.7. In the first step (1) we perform the following tasks to generate queries from each selected ACM CCS node dataset:

1. We directly use ACM CCS terms to prepare the queries. This is the first strategy, S_1.
2. We acquire important features after the classifier is built and create a set of queries using these important features. This is the second strategy, S_2.
3. We generate topics using the workflow shown in Figure 4.4 and create a set of queries using these topics. This is the third strategy, S_3.

Table 4.3 shows a list of the three strategies used in this experiment.

Table 4.3: A list of query generation strategies used in this experiment

Strategy	**Symbol**	**Description**
ACM CCS	S_1	Use ACM CCS terms as queries
Class important features	S_2	Use important features in the class that contributed to the classifier with best performance as queries
Topic Modeling	S_3	Use topics generated by LDA from the ACM dataset as queries

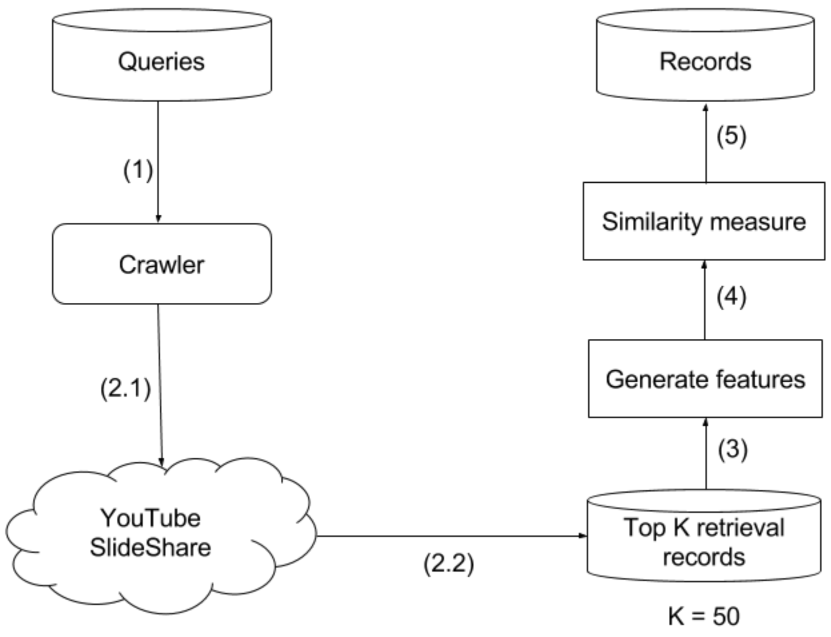


Figure 4.7: A workflow for retrieving records from YouTube and SlideShare that are similar to a given node in the ACM CCS tree

In step 2.1, the YouTube crawler and SlideShare crawler receive all of the sets of queries from step 1, and start issuing searches to the corresponding target site. Each crawler collects at most 50 search results per query (step 2.2). Each crawler extracts each record's metadata – title, description, and other attributes – and stores that in the backend database. The database schema is described in Appendix B.

In step 3, a program takes each record's title and description as features, converts words to a bag-of-words representation, and then creates a vector representation of the record. In step 4, a program computes the cosine similarity between the record's vector representation using the 2-dimensional LSI vector spaces and outputs the similarity score. In the final step 5, the program selects records that have similarity score over 0.7; the resulting set of records is collected through this workflow.

We evaluate each method on how many records have been gathered through the crawlers that are similar with the ACM node dataset. A valid record from YouTube or SlideShare should contain content that relates to a certain topic in computing in ACM CCS, for example, "Software system structures."

4.5 Experimental results

Depending on each selected ACM CCS node, the number of total queries is $N \times 3$. N is the number of children nodes, and we use three different methods to create the queries from each child node. Through our proposed workflow, each query gets at most 50 records from either YouTube and SlideShare. A number of those records are selected that contain features making them similar to the corresponding ACM CCS node. We conduct two experiments; one is to gather records from YouTube and another one from SlideShare. The results are discussed in the following sub-sections.

4.5.1 YouTube experiment result

In our proposed workflow, the YouTube crawler gets at most 50 search records from YouTube for each query. According to our experiment result with regard to strategy S_1, the crawler gets the maximum number, 50 search records, for each of the queries. There are many searches that yield below 50 records using the queries generated by S_2 or S_3. Since using S_3 can generate more topics to use as new search queries, we perform multiple searches to acquire more records from YouTube. We perform the similarity measurement on each record with the corresponding ACM CCS node representation through LSI (explained in the previous section), and select each record with similarity score over 0.7. Figure 4.8 shows an overview of the experiment result. Each block contains a graph representing one ACM CCS node selected from one of the eleven ACM CCS branches. The top left block shows a selected ACM CCS node from the CCS1 "Mathematics of computing" branch, followed by the CCS2 "Information systems", until one reaches CCS11 "Software and its engineering" in a row-wise ordering. Each graph contains the results using the three proposed strategies. The strategies are shown along the X axis; S_1 is shown in red (A), S_2 is shown in blue (B), and S_3 is shown in green (C). The number of records obtained from each strategy is shown on the Y axis. Incidentally, as shown in Figure 4.8, the largest value always is for S_3; either S_1 or S_2 has the lowest value. This means that many of the records from S_1 or S_2 do not contain many features that are similar to the corresponding ACM CCS node dataset. According to the result, using the S_3 method can get more related YouTube records than using the other two methods.

Each YouTube record contains a category field indicating which category that record belongs to; for example, the YouTube record shown in Figure 4.5 is categorized in the "Education" category. There are a total of 32 YouTube categories currently. In our implemented YouTube crawler, we can assign the crawler to search records from a specific YouTube category. Here the crawler searched for records in either of the "Education" and "Science & Technology" categories. We used this procedure for each of the 11 chosen ACM CCS categories to obtain YouTube records. After we finished a comparison of these strategies, we combined all the records, removed the duplicated records, and then collected a total of 6,762 YouTube records. These records are used as the experiment dataset discussed in the next chapter. In the future, we would include other categories in order to get more STEM education records from a larger candidate pool.

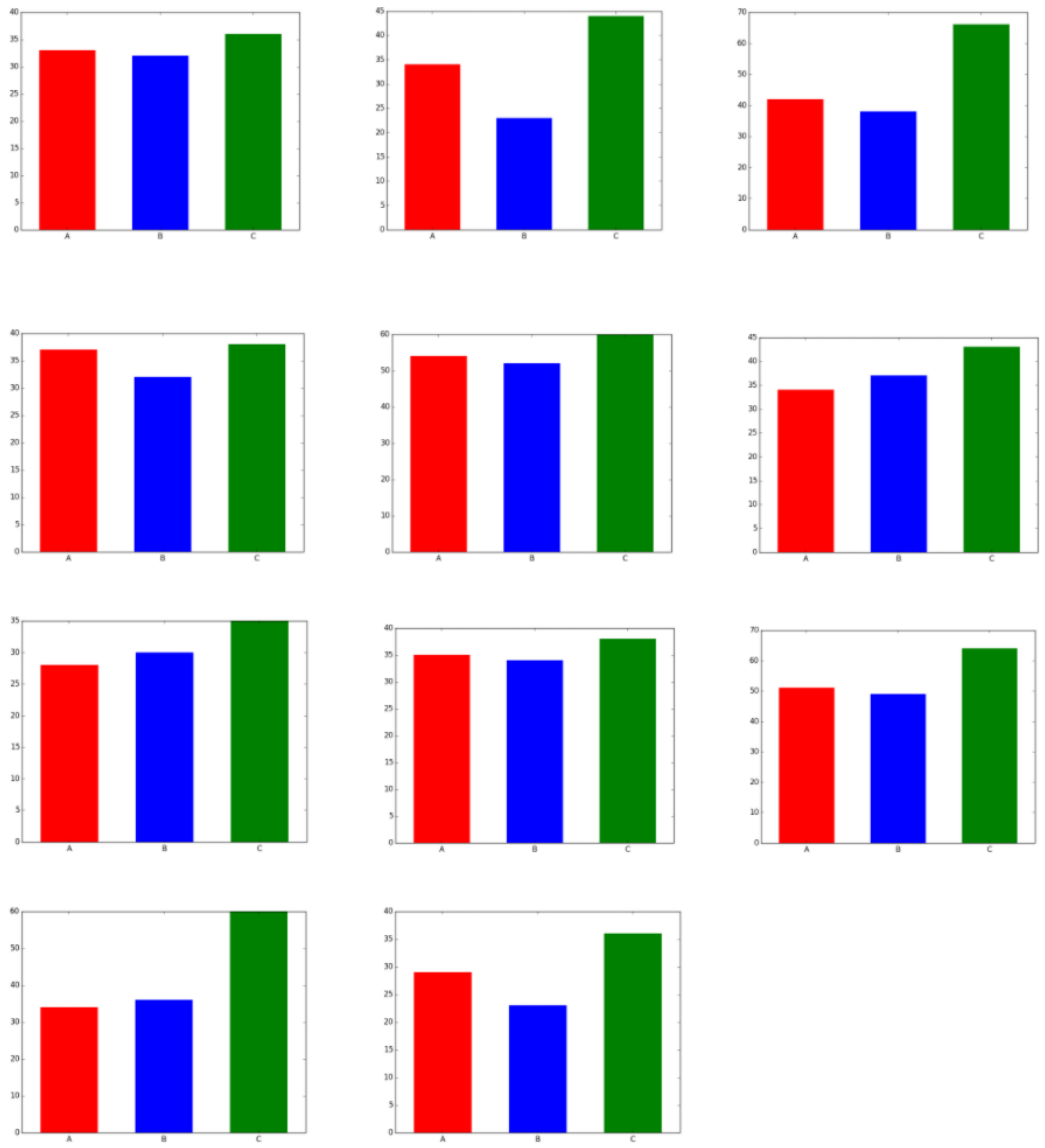


Figure 4.8: YouTube records obtained using proposed methods on selected ACM CCS node from the eleven CCS branches

4.5.2 SlideShare experiment result

We used the same procedure to gather SlideShare records using our SlideShare crawler. Differing from the situation in YouTube, our crawler can get fully 50 search results for most of the queries generated by any of the three methods. We use the same approach to perform the similarity measurement, and the final result is shown in Figure 4.9. Each block contains a graph representing one ACM CCS node selected from one of the eleven ACM CCS branches. The top left block shows a selected ACM CCS node from the CCS1 "Mathematics of computing" branch, followed by the CCS2 "Information systems", until one reaches the CCS11 "Software and its engineering" in a row-wise ordering. Each graph contains the results using three proposed strategies. The strategies are shown on the X axis; S_1 is shown in red (A), S_2 is shown in blue (B), and S_3 is shown in green (C). The number of records obtained from each of the strategies is shown on the Y axis.

The total number of records for these three methods is increased relative to the YouTube experiment. Although the difference between the total number of records in these three methods is reduced, the queries generated by method S_3 still perform better than the other two methods.

The obtained SlideShare records are in many different file formats. The different extensions are ppt, pps, pptx, odp, ppsx, key, docx, potx, xls, rtf, zip, and pdf. We remove the records that appear not to be presentation slides or documents, such as those in zip or xls format. A total of 8,089 SlideShare records were collected. These records are used as the experiment dataset discussed in the next chapter.

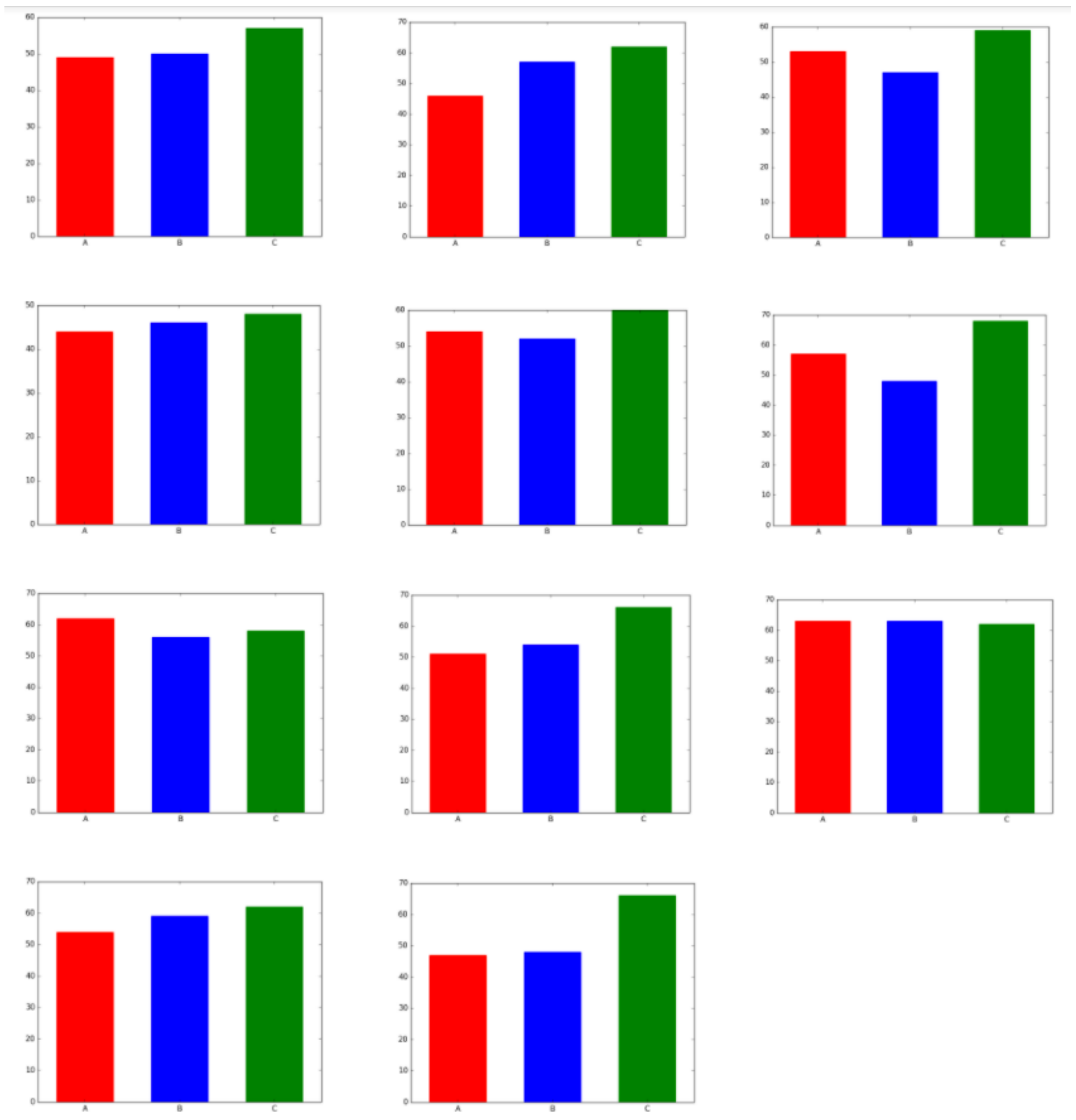


Figure 4.9: SlideShare records obtained using proposed methods on selected ACM CCS node

4.5.3 Discussion

We further examined the records that we collected from YouTube and SlideShare. There are some things worth mentioning, listed below.

1. We compared the number of records we collected through our crawlers from YouTube and SlideShare. We acquired more computing related records from SlideShare than YouTube.

2. Sets of videos uploaded by universities into YouTube, for a course, share the same description. For example, the course “Programming Methodology” uploaded by Stanford contains 28 lectures. The descriptions of all 28 videos are the same; only the titles are different. Thus, course descriptions have structure and consistency which sometimes can be analyzed and used to get more educational records. For example, we can extend our crawler to parse the playlist information from the description and get the full course videos.

3. Both YouTube and SlideShare provide record statistics information, such as number of likes, views, etc. These can be positive indicators to filter crawled records. In the future, we should consider including these features into our crawlers.

4.6 Summary

In this chapter, we compare three strategies to generate queries for searching computing education resources in specific topics from YouTube and SlideShare. We compared the performance of these strategies based on the number of records obtained in each strategy and identified that the topic modeling strategy outperforms the other two strategies. Our experiment result shows a promising indication that educational resources in specific computing topics can be collected from YouTube and SlideShare through our proposed workflow. Subsequent work will apply these methods to construct large collections of new records that are likely to be useful for Ensemble.

In the future, we will continue to extend this work by gathering educational content from other Web 2.0 sites that provide APIs for accessing their resources, such as Khan Academy, Vimeo, etc. We would like to see how the method performs on different sites and how it gathers more computing educational resources for the Ensemble portal.

Chapter 5

Transfer the learning to Web 2.0 classifiers

In this chapter, we present our work on combining transfer learning and active learning to build classifiers for labeling YouTube and SlideShare records using ACM CCS. We propose an active supervised domain adaptation approach that uses MTurk as "oracle" to help reduce the number of iterations to build the classifier. We use MTurk to examine the prediction result from the classifier and to create new training datasets, which in turn are used to improve the training of the existing classifier. Through an iteration process, the classifier can achieve good performance and better predict the classes for new records from YouTube and SlideShare.

5.1 Introduction

In our proposed framework, we integrate MTurk into the classifier building procedure and use this service to evaluate the classifier's prediction result. We conduct two experiments; one experiment is to verify that MTurk workers are capable of determining whether a resource is a computing educational resource and does belong to a specific computing topic. The other experiment is to validate our proposed framework and its performance.

The goal of this study is to address one of the DL challenges from the Library of Congress, to establish a coherent way to present heterogeneous collections harvested from multiple content providers. We take Ensemble as our testbed, and propose an approach to categorizing all the collection records using a well-defined classification system, ACM CCS.

Chapter 4 describes our framework to acquire new educational resources in computing from YouTube and SlideShare. These resources are only associated with a general category, such as "Education," "Science & Technique," etc. We want to further classify these records relative to a more specific computing topic, such as "Machine learning," "Data structure," or "Mobile computing."

Since we don't have records from YouTube and SlideShare labeled according to the ACM CCS, our goal is to develop a mechanism to be able to label these records using ACM CCS. In the ACM DL, there is labeled data provided by professionals in computing. This suggests using the labeled data to build classifiers for our YouTube and SlideShare records. Hence we come to the transfer learning approach.

Transfer learning is used to transfer knowledge from the auxiliary data in order to assist the learning task on the target data. It aims to reduce the effort to build new classifiers from scratch and to ease human work on labeling data from the target domain. In real-world settings, it is a difficult task to acquire newly labeled data from a new domain. In our case, to ask a professional to label a YouTube or SlideShare record using ACM CCS is unrealistic.

Thus, in this study, we propose an active transfer learning approach to using the ACM dataset as an auxiliary dataset to train classifiers. Our method combines with MTurk, which is used to evaluate classifier prediction results and create new training datasets. Through an iterative process we continue updating the classifier until we achieve good performance on labeling YouTube and SlideShare records using ACM CCS.

5.2 Related work

In this section, we start with describing the definition of transfer learning, domain adaptation, and finally MTurk. We review some of the state of the art methods that have addressed similar problems.

5.2.1 Definition of the transfer learning

We adopt the notations and definitions of transfer learning from [106], and use them in this study. A domain $\mathcal{D}$ contains a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, ...x_n\} \in \mathcal{X}$. A domain is denoted by $\mathcal{D} = \{\mathcal{X}, P(X)\}$. Two domains are considered different when either the feature space or the marginal probability distribution differ.

A task $\mathcal{T}$ contains a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$, which can be learned from the training data. The training data consist of pairs x_i, y_i, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The function $f(\cdot)$ can be used to predict the corresponding label, $f(x)$, of a new instance x. In our study, $\mathcal{Y}$ is the set of ACM CCS terms; the function $f(\cdot)$ can predict whether a new record, x, belongs to some y_i, e.g., "Data structures."

For simplicity, we consider that there is a source domain $\mathcal{D}_s$ and learning task $\mathcal{T}_s$. There is a target domain $\mathcal{D}_t$ and learning task $\mathcal{T}_t$. Transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_t$ using the knowledge in $\mathcal{D}_s$ and $\mathcal{T}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$, or $\mathcal{T}_s \neq \mathcal{T}_t$.

In our study, we aim to build a classifier to predict the correct ACM CCS term for a record using the knowledge learned from the ACM dataset. The source and target tasks are the same, while the source domain $\mathcal{D}_s$ and target domain $\mathcal{D}_t$ are different. Moreover, we don't have labeled data in the target domain available, but we have much labeled data in the source domain. Among transfer learning settings, ours is related to transductive transfer learning. More specifically, the feature spaces between the domains are the same, $\mathcal{X}_s = \mathcal{X}_t$, but the marginal probability distributions of the input data are different, $P(X_s) \neq P(X_t)$. Our case is related to domain adaptation for knowledge transfer in text classification [46].

5.2.2 Domain adaptation

The domain adaptation problem can be defined as follows: We have a set of labeled data from the source domain $\mathcal{D}_s$. We want to make predictions for the unlabeled data from the target domain $\mathcal{D}_t$. This problem setting for domain adaptation was considered in several research studies, such as image classification [109], sentiment analysis [17], and text categorization [41] [44]. Several methods have been proposed to solve this problem: feature-representation-transfer approaches [18], instance-transfer approaches [40], instance selection and weighting approaches [131], and supervised approaches [45]. The first two approaches focus on data representation and model building, while the other two focus on taking advantage of the labeled and unlabeled data to approximate the distribution to be learned for the target domain. In this study, our proposed approach is related to the supervised domain adaptation approach.

The supervised domain adaptation approach is to find the most useful instances from the source domain in order to classify the instances from the target domain. The hypothesis behind this approach is that these instances can be obtained through SVM and that they are in fact the support vectors since they are the nearest points to the decision boundary [72]. We present our proposed approach in Section 5.5.

5.2.3 Amazon Mechanical Turk (MTurk)

Amazon Mechanical Turk (MTurk) is a crowdsourcing service in which requesters can post tasks to be completed, and rewards are given to workers for satisfactory work. A requester is a person who creates and submits tasks to MTurk for workers to perform. A Human

Intelligence Task (HIT) is a task submitted by requesters. A requester can specify how many workers can submit completed work for the HIT. When a worker accepts a HIT, MTurk creates an assignment associated with that HIT for tracking the completion of the work. A worker is a person who accepts and submits an assignment to MTurk and gets rewards when his assignment has been approved by the requester. MTurk guarantees that a worker can engage in each task only one time. Figure 5.1 shows how MTurk functions. Researchers use the MTurk service to conduct behavioral research [76], and MTurk has been proven to perform well on labeling tasks [124]. In our previous MTurk experiment [31], MTurk workers achieved similar quality as experts with necessary background knowledge on the record labeling task. In this study, we use MTurk to evaluate classifier predicted results and to create new training datasets for updating existing classifiers.

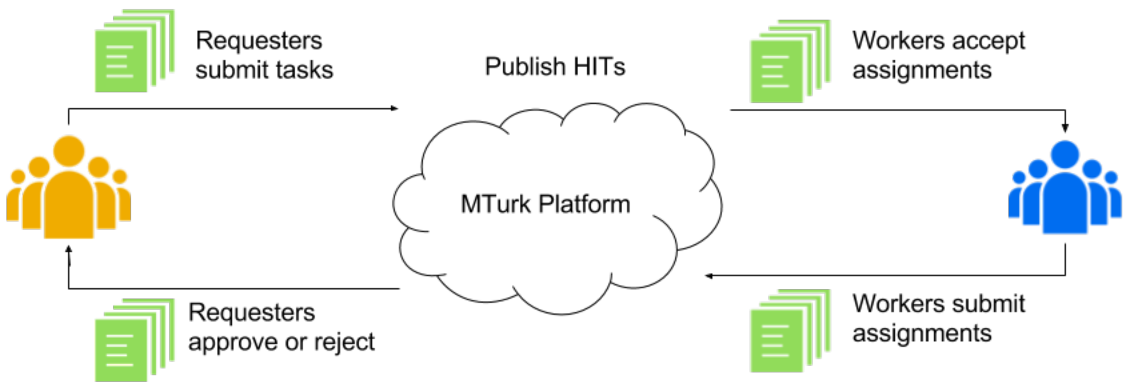

Requesters submit tasks
Publish HITs
Workers accept assignments
MTurk Platform
Requesters approve or reject
Workers submit assignments

Figure 5.1: An overview of MTurk workflow

5.3 Research method overview

In our proposed active supervised domain adaptation method, a small amount of labeled data from the target domain is needed. Our approach is to acquire these labeled data by combining the domain adaptation method and MTurk. We use the classifier that is built from the source domain to make predictions for a set of unlabeled records from the target domain. After a set of labeled records emerges as output from the classifier, we use MTurk to examine these records and determine if they are correctly labeled. We update the existing classifier with the new training dataset until the performance of the classifier achieves the level of quality set by us.

In Section 5.4, we describe how we conduct the MTurk experiment to evaluate predicted records from the classifiers and how the new training dataset has been created for the next iteration in classifier building. In Section 5.5, we present our proposed method and evaluate its performance by comparing with baseline classification algorithms.

5.4 MTurk

The purpose of using MTurk is to evaluate the records predicted by the classifier and identify the educational value of the records. In our previous study, we compared the labeling work for educational records between MTurk workers and computer science students and found out that both parties had similar performance [32]. We extend that work and design a new MTurk experiment in order to fulfill our goal of the current study. After the records are evaluated by the MTurk workers, the records with high educational value and correctly labeled with an ACM CCS term are prepared for addition into Ensemble. A new training dataset that includes both positive and negative records is created to be used for updating the existing classifier. Our idea is to use MTurk to gather a more accurate training dataset for the classifier and further reduce the number of iterations for updating the classifier. In the following sections, we give a detailed description of how we prepare the records to be evaluated by MTurk, as well as the MTurk experiment design, and the procedure of acquiring newly labeled records.

5.4.1 Dataset

We have two sets of datasets; one is YouTube datasets, while the other is SlideShare datasets. Each dataset contains a set of records labeled by the classifier. When the classifier predicts that a record belongs to an ACM CCS term, it provides a confidence score, from 0 to 100, for this prediction. We select records with confidence score over 80 and prepare these for the MTurk experiment.

5.4.2 Workflow

Figure 5.2 shows an overview of the MTurk workflow. In step 1, we extract the metadata information from the predicted records to create a HIT data file. A HIT data file is an Excel file that contains HIT records to be submitted to MTurk. In this Excel file, each row contains a record's title, description, URL, and the predicted ACM CCS term. In step 2, we upload the HIT data file to MTurk. MTurk receives the HIT data file, generates assignments using our designed HIT template stored in MTurk, and publishes these assignments to the public. MTurk workers can view these assignments, select an assignment they want to work on, and submit their answer to MTurk. In step 3, our MTurk result parser tool retrieves assignment results from MTurk. In step 4, the MTurk result parser tool extracts workers' answers and selects the record with an educational value in computing. In step 5, the MTurk result parser tool splits the selected records into two groups; one is for records with correct labels, while the other is for records with an incorrect label. In step 6, a new training dataset is created that contains both positive and negative data. In step 7, a set of new records is ready to be included in the Ensemble DL.

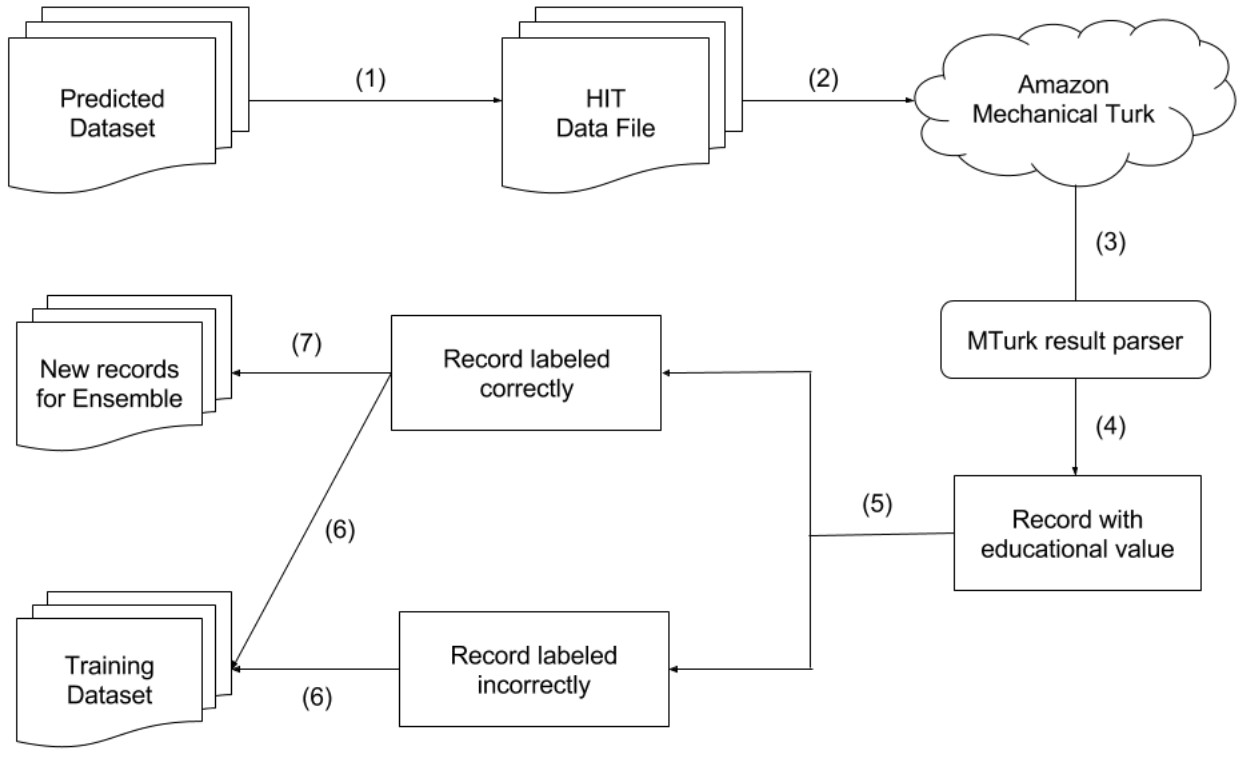


Figure 5.2: MTurk workflow

5.4.3 MTurk HIT template design

Our HIT template design goal is to acquire high educational value records in computing that are labeled with the correct ACM CCS term. Figure 5.3 shows the HIT template design workflow. Instructions are provided at the beginning of the HIT; the worker reads the instructions and then starts to review the record. The record is either a video or a set of presentation slides. The title, description, and ACM CCS term are also provided. Then a worker needs to answer all the questions in the HIT. In section 1, a worker needs to evaluate the educational value and quality of the record, and determines if this record is labeled correctly with the given ACM CCS term. The worker also needs to indicate his confidence level on these questions. If the worker doesn't think the ACM CCS term predicted by the classifier is correct, he can suggest another ACM CCS term which is more suitable for this record. In section 2, the worker needs to indicate how much time he spent on reviewing the record, what record attributes are most useful, and if he used any external resource to help make a judgment. Figure 5.4 shows a HIT example for a YouTube record.

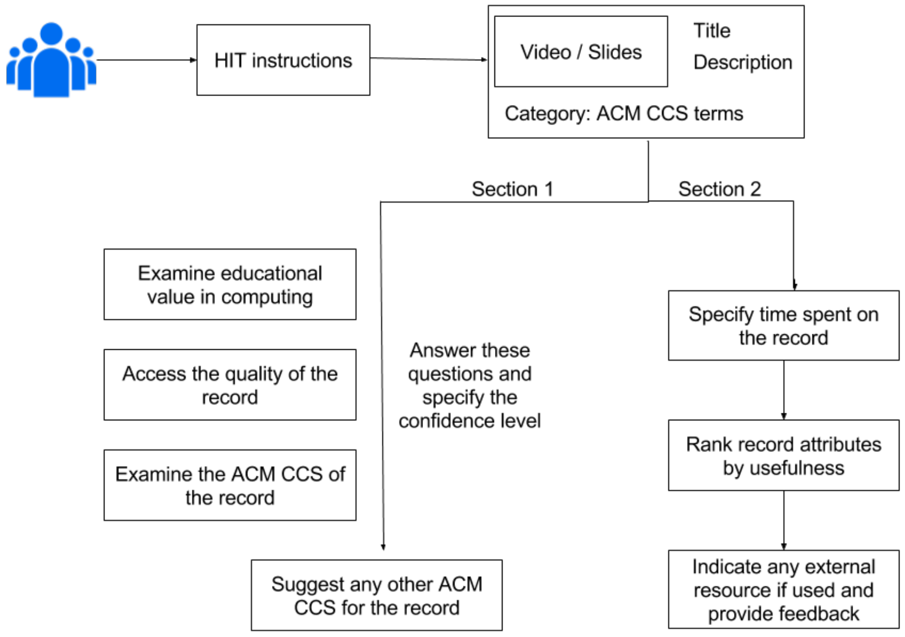
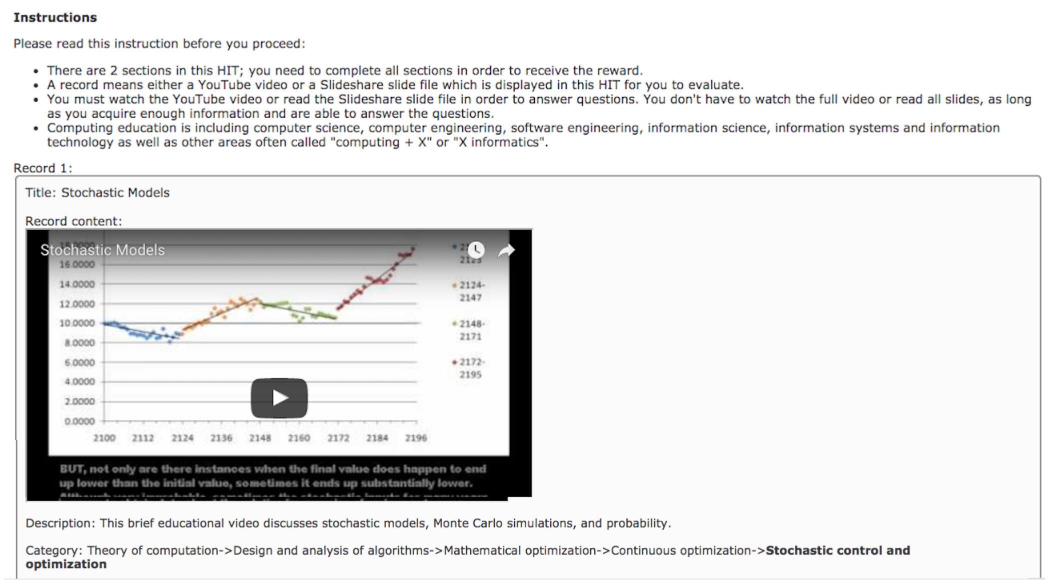

HIT instructions
Video / Slides
Title
Description
Category: ACM CCS terms
Section 1
Section 2
Examine educational value in computing
Access the quality of the record
Examine the ACM CCS of the record
Answer these questions and specify the confidence level
Suggest any other ACM CCS for the record
Specify time spent on the record
Rank record attributes by usefulness
Indicate any external resource if used and provide feedback

Figure 5.3: MTurk HIT template design workflow



Instructions

Please read this instruction before you proceed:

- There are 2 sections in this HIT; you need to complete all sections in order to receive the reward.
- A record means either a YouTube video or a Slideshare slide file which is displayed in this HIT for you to evaluate.
- You must watch the YouTube video or read the Slideshare slide file in order to answer questions. You don't have to watch the full video or read all slides, as long as you acquire enough information and are able to answer the questions.
- Computing education is including computer science, computer engineering, software engineering, information science, information systems and information technology as well as other areas often called "computing + X" or "X informatics".

Record 1:

Title: Stochastic Models

Record content:

BUT, not only are there instances when the final value does happen to end up lower than the initial value, sometimes it ends up substantially lower.

Description: This brief educational video discusses stochastic models, Monte Carlo simulations, and probability.

Category: Theory of computation->Design and analysis of algorithms->Mathematical optimization->Continuous optimization->**Stochastic control and optimization**

Figure 5.4: A YouTube record HIT example

5.4.4 MTurk experimental setup

Our MTurk experiment settings are as follows:

1. We assign, using different people, two assignments per HIT. A worker can only submit one assignment per HIT and can not submit the same assignment twice. Time allotted per assignment is 90 minutes.
2. We use these worker requirements to select workers: location, HIT approval rate (%), the number of HITs approved, and MTurk masters.
3. The HITs are available to workers for two weeks.

We defined the following rules to determine whether to either approve or reject assignments submitted by workers:

1. If there are any incomplete questions in the assignment then reject that assignment.

2. Time conflict: We included a question in the HIT, asking a worker how much time he spent reviewing the record. If a worker said that he spent 10 minutes on reviewing the record, but the time spent on that assignment shows only 5 minutes in the MTurk log, then we reject that assignment.

For every individual record, we expect to receive a consensus answer from both workers. For example, if both workers agree or strongly agree that this record is correctly labeled, then we close this HIT. If we receive contradictory answers from the workers, we publish that HIT again for more workers to evaluate that record until we receive two consistent answers from different workers.

We proposed requirements in order to identify what combination of worker's requirements, and how many rewards we should pay, to get our HITs (100 HITs, 200 assignments) to be finished by workers within a week. We conducted three MTurk experiments and used different worker requirements and rewards, and finally identified suitable requirements and rewards for our experiment. That led to requirement three shown in Table 5.1.

Table 5.1: MTurk worker requirements and rewards

	Requirement 1	**Requirement 2**	**Requirement 3**
HIT approval rates	98%	90%	90%
# of HITs approved	1000	500	100
Location	United States	United States	United States
Master required	Yes	No	No
Rewards	$0.20	$0.20	$0.20

5.4.5 MTurk experiment result

We conducted six different MTurk experiments and gathered all the results together. Three of the six are described in the previous section. The other three are follow up experiments to acquire two consistent answers from different workers. After we identified the best worker requirements and rewards, our HITs could be finished by the workers in a week. According to our result, shown in Figure 5.5, workers spent more time on reviewing a YouTube record than a SlideShare record. Most workers can finish reviewing a record in 10 minutes.

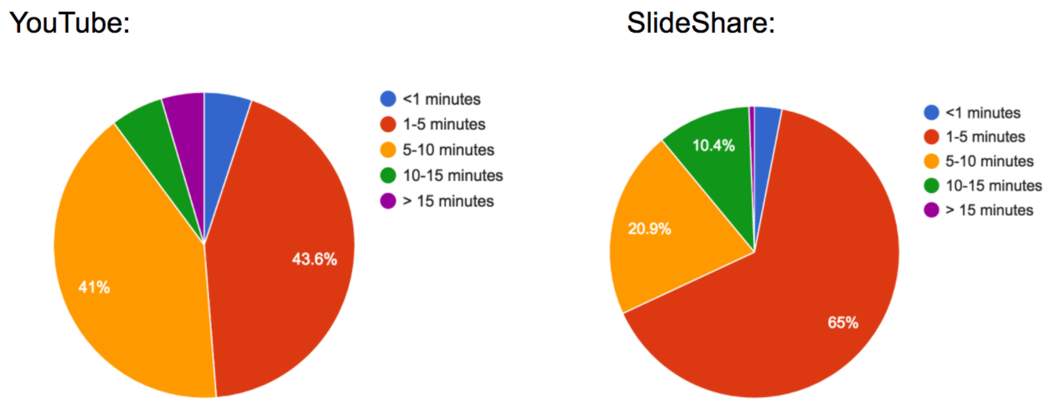


Figure 5.5: Time spent on reviewing a record

Regarding the educational value and quality of records, more than 70% of workers have strong confidence in their answers for both YouTube and SlideShare records. 84.8% of workers agree that the YouTube records have educational value in computing. 91% of workers agree that the SlideShare records have educational value in computing. 90.4% of workers agree that the quality of content in YouTube records is above average. 86.1% of workers agree that the quality of content in SlideShare records is above average. Figure 5.6 shows workers' evaluation results on assessing the educational value in computing of the records. Figure 5.7 shows workers' evaluation results on assessing the quality content of the records.

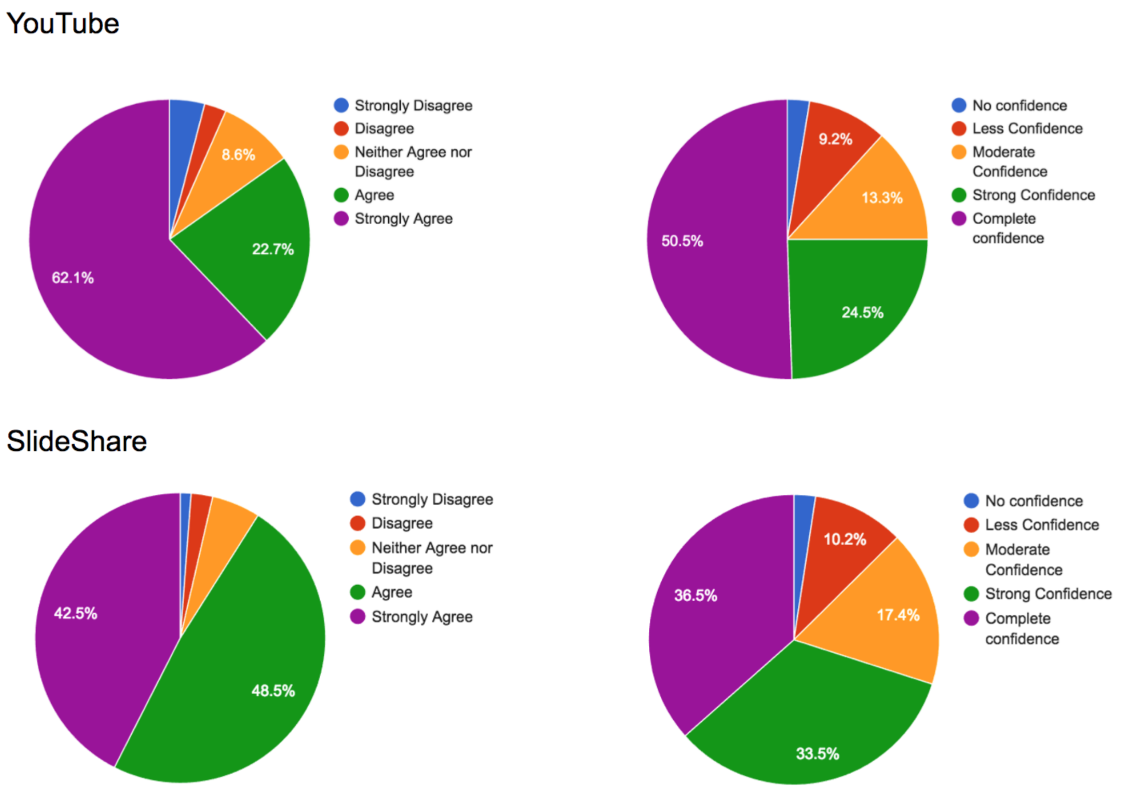


Figure 5.6: MTurk results on assessing educational value in computing education

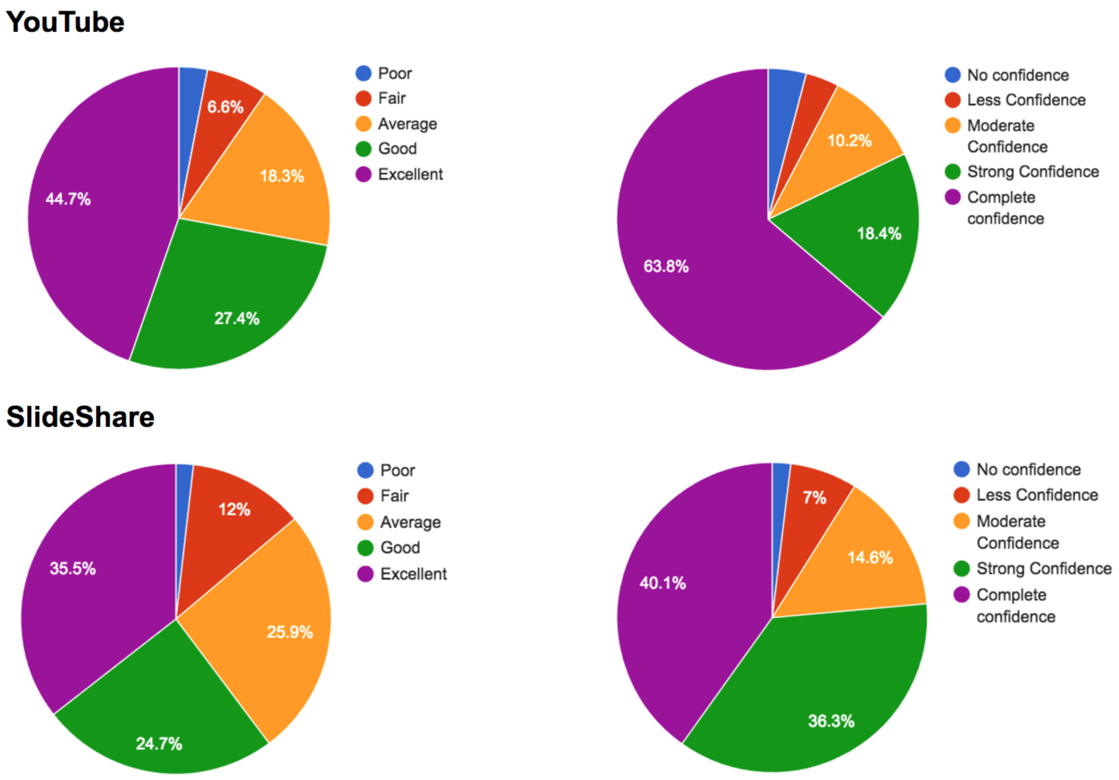


Figure 5.7: MTurk results on assessing content quality

Finally, we examine the MTurk assessments of the classifier predicted result on ACM CCS. Here 82.2% of workers agree that the YouTube records are labeled correctly with over 73.5% strong confidence or above. Also 74.1% of workers agree that the SlideShare records are labeled correctly with over 60.1% strong confidence or above. Figure 5.8 shows the results.

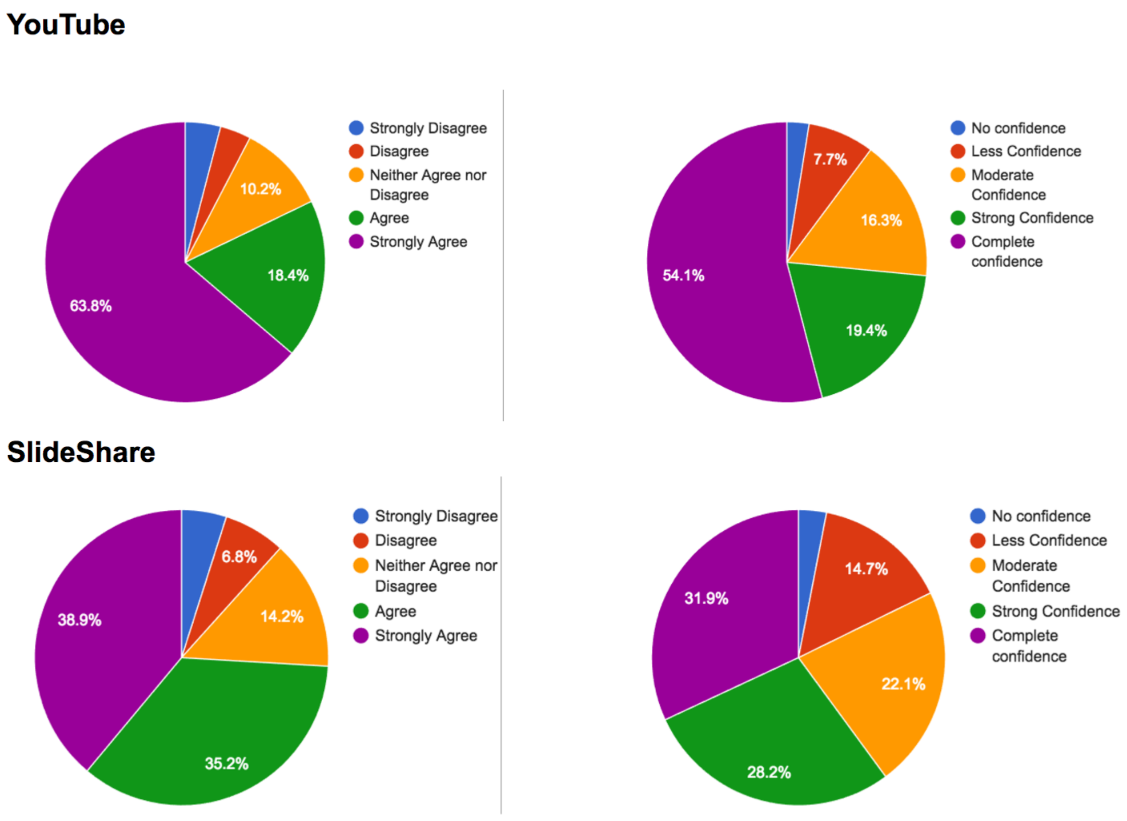


Figure 5.8: MTurk results on assessing ACM CCS labeling

According to their feedback, most workers can finish the assignment without any external help. There are just a few workers who used either Google search or Wikipedia for external help. The most useful record attributes for workers to assess the record are content and description; the record's title is least helpful. Although it is an option for a worker to provide thoughts on the assignment, most workers on YouTube records left comments on the assignment. Most of the comments state views of that record, for example, "excellent presentation," "Video is short, yet full of information," "The speaker explained his presentation well," "This was not an educational video. This was a presentation to sell software products." A few workers left comments on the SlideShare records, for example, "Very good presentation and easy to understand from user view" and "This hit is great to ensure accuracy and efficiency of the present work."

After the experiments were finished, we then followed our proposed workflow shown in Figure 5.2; we collected new records for Ensemble and prepared the new training dataset.

5.5 Domain adaptation classifier

5.5.1 Research method

Our proposed method is an active supervised domain adaptation for the text classification. Our idea is to bring the $\mathcal{D}_s$ distribution close to the $\mathcal{D}_t$ distribution by adding newly labeled data from the $\mathcal{D}_t$ into the training dataset in each iteration for classifier training. The new training dataset will have a more heterogeneous distribution in each iteration and further reduce the information gap between the two domains. Our method is a variant of co-training [19] [29].

The procedure of the algorithm includes the following steps: a classifier is trained using the labeled data from the source domain. The trained classifier predicts labels for the unlabeled data from the target domain. The records with predicted labels are selected based on the classifier's prediction confidence score; these records are prepared for MTurk for evaluation. The records that have been evaluated by MTurk and correctly classified are added into a new training dataset and used to train a new classifier. The process repeats until the testing dataset becomes empty or a stop criterion is met. Finally, a new classifier is used to classify unlabeled records from the target domain. We chose SVM in this study; as explained in Chapter 3, our experiment results with the ACM dataset show that we can achieve the best performance of classifiers using the Linear SVM with SGD training with a penalty.

The proposed domain adaptation method is shown in Algorithm 1.

For our situation, the training dataset is the ACM dataset, and the testing dataset is either YouTube or SlideShare. In each iteration, new records are acquired, and the classifier is updated. When there are no new records that have been labeled by the classifier or the

Algorithm 1 Proposed domain adaptation method algorithm

```
1:  **Data:** $\mathcal{D}_s$ is a set of labeled records from the source domain, each instance in $\mathcal{D}_s$ is a pair $(x_k, y_k)$ and $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. $\mathcal{Y}$ is the label space of the classification problem. $\mathcal{D}_t$ is a set of unlabeled records from the target domain to be classified.
2:  **Result:** A labeled set of records from $\mathcal{D}_t$ and a classifier $C_i$.
3:  $training\_dataset \leftarrow \mathcal{D}_s$
4:  $testing\_dataset \leftarrow \mathcal{D}_t$
5:  $i = 0$
6:  **repeat** $testing\_dataset \neq 0$ $or$ $selection\_dataset_i > 0$
7:      $C_i \leftarrow Train\ SVM\ classifier\ with\ training\_dataset$
8:      Classify testing_dataset with $C_i$
9:      $Selected\_dataset_i \leftarrow select\ record\ based\ on\ prediction\ score$
10:     Submit Selected_dataset to MTurk for evaluation
11:     $Evaluated\_dataset \leftarrow MTurk\ evaluation$
12:     $New\_training\_dataset \leftarrow training\_dataset \cup evaluated\_dataset$
13:     $training\_dataset \leftarrow New\_training\_dataset$
14:     $Testing\_dataset \leftarrow testing\_dataset - evaluated\_dataset$
15:     $i \leftarrow i + 1$
16: **until** No more predictions are confident
17: **return** $C_i$
```

classifier can achieve the defined performance, e.g., F1-score is 0.8, then the procedure is ended.

5.5.2 Experimental setup

This experiment is designed to evaluate our proposed transfer learning method and its performance. The outcome of built classifiers can predict YouTube and SlideShare records with good performance. We compare the performance with other baseline text classification algorithms.

5.5.3 Experimental design

The experiment is composed of three parts: ACM classifier prediction, MTurk evaluation, and domain adaptation classifier training and evaluation. The MTurk evaluation experiment is presented in the previous section, 5.4.

In Chapter 4, we described collecting computing records from YouTube and SlideShare, and preparing several groups of the dataset. Each group contains the records which are the search results from the queries that were generated from the ACM articles using topic modeling. These groups of the dataset are the testing datasets for our built ACM classifiers. In both YouTube and SlideShare records, we use title and description as features and use the same feature extraction and feature selection process as we discussed in Chapter 3.

The ACM classifier prediction workflow is shown in Figure 5.9. In Chapter 3, according to our experiment results, the Linear SVM with SGD training with a penalty can train the best classifier from the ACM dataset. We use the classifier we built in Chapter 3 to predict the ACM CCS term for the YouTube and SlideShare records. The classifier takes each record as an input and predicts the ACM CCS term for that record with a confidence score of this prediction. The confidence score is computed from the distance of the samples X to the separating hyperplane in the SVM model. We select predicted records with the classifier's confidence score over 80 percentage and acquire the ACM classifier predicted dataset. These predicted datasets are submitted to MTurk for evaluation, which is discussed in the previous section, 5.4.

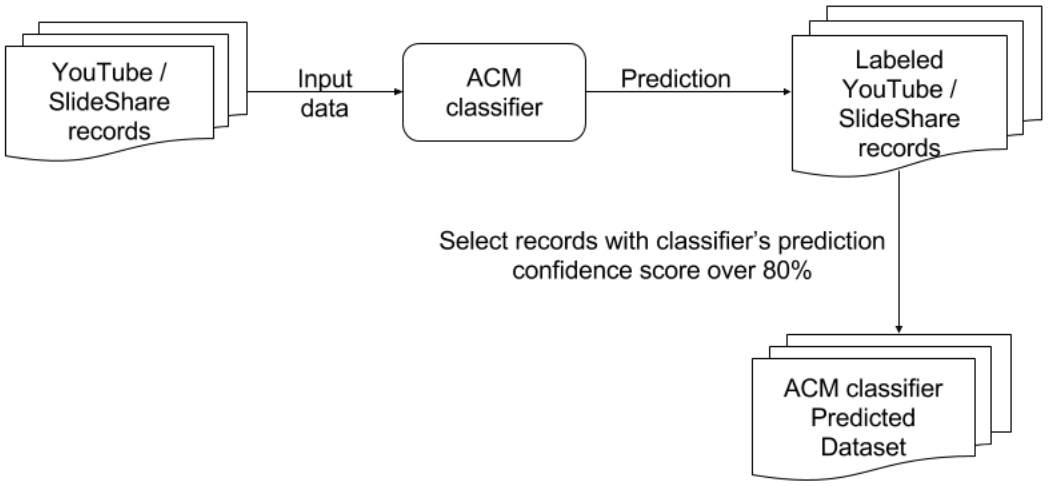

YouTube / SlideShare records
Input data
ACM classifier
Prediction
Labeled YouTube / SlideShare records
Select records with classifier's prediction confidence score over 80%
ACM classifier Predicted Dataset

Figure 5.9: ACM classifier prediction workflow

For simplicity, we select two ACM CCS nodes to use in this experiment, where we have trained a classifier with good performance using the articles in these two nodes as discussed in Chapter 3. These two nodes are “Data structure” and “Software system structures.” We prepared YouTube and SlideShare records as the testing dataset for these two ACM CCS terms using the procedure described in Chapter 4; there are four datasets in total.

In our proposed method, we add all the evaluated records from MTurk into the new training dataset for the next iteration. For simplicity and proof of concept purposes, we split the evaluated records in the separate dataset, ten records per dataset, and add ten records in the new training dataset, per iteration. For example, if we acquire 30 evaluated records from MTurk, then we split into three datasets and update the classifier three times and measure the performance.

We expect that the performance of the classifier that has been transferred should be no worse than the classifier without transfer. We compare our proposed method with current text classification algorithms.

5.5.4 Experimental results

Table 5.2 shows the experiment result. The baseline classification algorithms are multinomial Naïve Bayes (NB) and Linear SVM with SGD training (SVM). We denote our proposed method (DC) and use Linear SVM with SGD training. The last column (i) shows the number of iterations. According to the results, our proposed method outperformed the two baseline classification algorithms.

Table 5.2: F1-score of proposed method and other baseline classification algorithms

	CCS	**NB**	**SVM**	**DC**	**i**
YouTube	Data structure	0.75	0.79	0.86	3
	Software system structures	0.73	0.73	0.82	2
SlideShare	Data structure	0.67	0.74	0.85	3
	Software system structures	0.71	0.75	0.84	3

Through our proposed method, we can construct new classifier trees specific for YouTube and SlideShare records with good performance using the procedure we presented in this study.

5.6 Summary

In this chapter, we presented an active supervised domain adaptation integrated with MTurk, and built classifiers that can predict labels for YouTube and SlideShare records. These outperformed other baseline text classification algorithms. Using our proposed method on different ACM CCS datasets, we can construct classifier trees to label YouTube and SlideShare using ACM CCS.

The MTurk experiment result shows that MTurk workers perform the evaluation tasks well, and successfully verified that we could use MTurk to help to evaluate and acquire new training data for our classifiers.

In the future, we can apply our proposed method both on the current Ensemble records and on educational resources from other Web 2.0 sites, and further increase the number of educational resources in computing in Ensemble.

Chapter 6

Ensure the QoS in a digital library using cloud computing

In this chapter, we present our cloud-based architecture designs that ensure the quality of service in a DL. These cloud-based design approaches aim to ensure the external DL qualities, which are efficiency and reliability of the DL. We migrated an on-premise DL into the cloud-based DL using our proposed designs, compared the performance, and finalized the cloud-based architecture that ensures the external DL qualities with the best performance. Our experimental results show that the cloud-based architecture DL not only ensures these DL qualities but also that the performance is improved, compared to an on-premise DL – Ensemble portal. The proposed cloud-based architecture designs are flexible and general, and can be employed to establish a DL using different cloud computing providers.

6.1 Introduction

Currently, cloud computing plays an important role in the IT sector [13] [84]. Many companies move their on-premise infrastructure into a cloud-based infrastructure, e.g., Netflix (https://www.netflix.com/). Several DL research projects were considering migrating their applications and services into the cloud environment, e.g., CiteSeerX [129] [9]. Cloud computing offers many advantages that change the current IT infrastructure. Major cloud providers, such as Amazon AWS (https://aws.amazon.com/), Google Cloud Platform (https://cloud.google.com/), and Microsoft Azure (https://azure.microsoft.com/) provide many similar cloud computing services. They have been adapted into applications in both industry and academia. Currently, most DL cloud researchers focus on deploying the entire application into an instance (virtual machine) in the cloud environment [56] [114] [94]. To the best of our knowledge, this study is the first research that uses cloud computing to specifically address the DL qualities that are defined under the 5S framework.

The goal of this study is to develop and identify a cloud-based design that can migrate an on-premise DL into a cloud-based DL and can do well regarding these DL service quality indicators. After such an on-premise DL has been migrated to the cloud-based infrastructure, not only these DL quality indicators are ensured but also the performance of the DL is improved, along with cloud advantages being enabled.

6.2 Related work

There are several kinds of research using cloud computing in the digital library domain. Such research aims to explore the feasibility of using cloud computing services to extend the capability of their DLs, improve DL performance, and reduce the cost of operations. [30] moved their scholarly content into the cloud environment due to the lack of the disk space to store their increasing collections and also the limited fund for sustaining their DL. [129] and [90] host their digital library using a virtual machine provided by Amazon AWS and Google GAE. [127] and [134] investigated the cloud migration costs for their digital repository applications.

In our previous work [33], we compared the performance of different cloud-based designs for a digital library. In this study, we extend the scope of our previous work and further ensure efficiency and reliability quality indicators of the digital library in the cloud environment. With regard to efficiency, anyone can access DL collections through the Internet anywhere, with appropriate speed. With regard to reliability, such a digital library provides permanent access to its organized collections of selected digital resources. The collection and its associated digital resources are persisted over time, always accessible and unchanged. Similar to our study, [128] deployed their digital library search engine, SeerSuite, into the cloud environment and evaluated its scalability and availability.

6.2.1 Definition

Ensuring quality of service in a DL is an important topic [61] [78]. Under the 5S framework, a minimal DL implements key DL concepts (Collection, Catalog, Repository, and Services) and can be assessed with various quality indicators (Completeness, Consistency, Efficiency, and Reliability). Among these quality indicators, we focus on the efficiency and reliability quality indicators in this study. These quality indicators are the key factors that ensure the quality of the DL services. The definition of each quality indicator is presented in the following sections.

Efficiency

Efficiency in a digital library reflects the DL responding to a user's request in a short time period. A user's request could be browsing a metadata record page, searching records in a collection, playing a media file, or downloading a document file. A user should always receive a prompt response even if the digital library is working with heavy loads, and regardless of the user's location. [61] defined that the efficiency is measured regarding speed, i.e., the difference between request and response time. Let $t(e)$ be the time of event e, and let e_{ix} and e_{fx} be the initial and the final events of scenario sc_x in service Se. The efficiency of service Se is defined as

$$\textbf{Efficiency}(Se) = \frac{1}{max_{sc_x}\,\epsilon\, s_e\ (t(e_{fx}) - t(e_{ix})) + 1.0} \tag{6.1}$$

Example of use: The maximum response time for a user to view a metadata record in Ensemble is 2.40s. The efficiency of browsing in Ensemble is 0.3. The maximum response time for a user to search a record in Ensemble is 1.36s. The efficiency of searching in Ensemble is 0.42.

Reliability

Reliability indicates that a DL always responds to a user's response with zero downtime. Files, such as a document, audio, or video, stored in the DL, should always be unchanged and preserved for a long period. [61] defined the reliability of a service Se_x as

$$\textbf{Reliability}(Se_x) = 1 - \frac{\textbf{no. of failures}}{\textbf{no. of accesses}} \tag{6.2}$$

A failure is characterized as an event that occurs when:

1. The DL fails to respond to a user's request. Here a 404 response is considered a non-response.
2. A user can view the metadata record, but the associated files no longer exist.
3. A downloaded file is damaged and cannot be opened.

6.2.2 Cloud computing services

We did a thorough survey of current cloud computing services and identified services that we can include in the cloud-based architecture design, where we can do well regarding the DL quality indicators that we mentioned in the above section. We briefly describe these cloud computing services.

Compute cloud

A compute cloud provides a resizable compute capacity (CPU, memory, and network bandwidth) for the user to have a virtual instance in the cloud environment. For example, AWS EC2 provides different types of instances for users to chose, and users can install the operating system they preferred, such as Ubuntu.

Cloud storage

Cloud storage provides durable access to files anywhere around the Internet, provides multiple backups for each file, and detects data corruption in the files. For example, AWS S3 provides 99.999999999% durability of a file over a given year.

Auto scaling

Auto scaling provides a mechanism that can automatically increase or decrease the number of instances based on the conditions defined by the user.

Software as a Service (SaaS)

Cloud providers provide on-demand software for you to use on a subscription basis. For example, AWS RDS for MySQL gives the user access to a MySQL database engine and provides additional services, such as automated backups and database snapshots.

Cost model in cloud computing

The cost model in cloud computing is a pay-as-you-go model; you pay for what you use of the services. You can save much money if you optimize the cloud resource you acquired, otherwise you could waste your money if you over allocate cloud resources that you don't need.

Docker

Package your service into a docker image and deploy the container to separate the DL services into different components and communicate between them. Thus you can manage each component and adjustment based on the actual need.

Container

A container is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, and settings. It is a virtual environment that runs the Docker images.

Endpoint

An endpoint is the URL assigned by the service provider for you to access the service and retrieve responses in a defined data format. A cloud computing service can have multiple endpoints. An endpoint indicates a particular location for accessing a service using a specific protocol and data format.

Cloud Providers

The cloud computing services listed above are mostly provided by the major cloud providers (Amazon AWS, Google Cloud Platform, and Microsoft Azure) at the time of writing this dissertation. Table 6.1 lists the cloud services provided by these major cloud providers. Our proposed cloud-based designs are general, which can be implemented using these cloud services and not be limited to a particular cloud provider.

Table 6.1: List of cloud services provided by different cloud providers

Service	**Amazon AWS**	**Google Cloud Platform**	**Microsoft Azure**
Virtual Machine	Elastic Compute Cloud	Compute Engine	Virtual Machine
Container Service	EC2 Container Service	Container Engine	Container Service
Object Storage	S3	Cloud Storage	Blob Storage
CDN	CloudFront	Cloud CDN	CDN
Load Balancer	Elastic Load Balancing	Cloud Load Balancing	Load Balancer
Archive Storage	Glacier	Nearline	Cold Blob Storage
Function as a Service (FaaS)	AWS Lambda	Google Cloud Functions	Azure Functions

6.3 Design approach

Our design approach is first focusing on achieving each DL quality indicator and proposing corresponding cloud-based design solutions. Secondly, we measure the performance of each design and compare the result with an on-premise DL. Then, we describe the migration effort for an on-premise DL to adopt that cloud-based design and list the advantages gained from the cloud services. We use AWS as our cloud computing service to implement our cloud-based design and conduct experiments, since AWS is currently the market leader in cloud infrastructure as a service [85]. Figure 6.1 shows an overview of the current cloud providers as of June 2017. Finally, we present a complete integrated cloud-based architecture design that seems best for future DL development in the cloud environment.

We also considered that the migration effort for transitioning an on-premise DL to a cloud-based DL should be as minimal as possible. The cost for running such a cloud-based DL should be controllable, manageable, and adjustable depending on different use cases. As such, an economic model to support the long-term operations of the cloud-based DL could be developed using these design approaches.

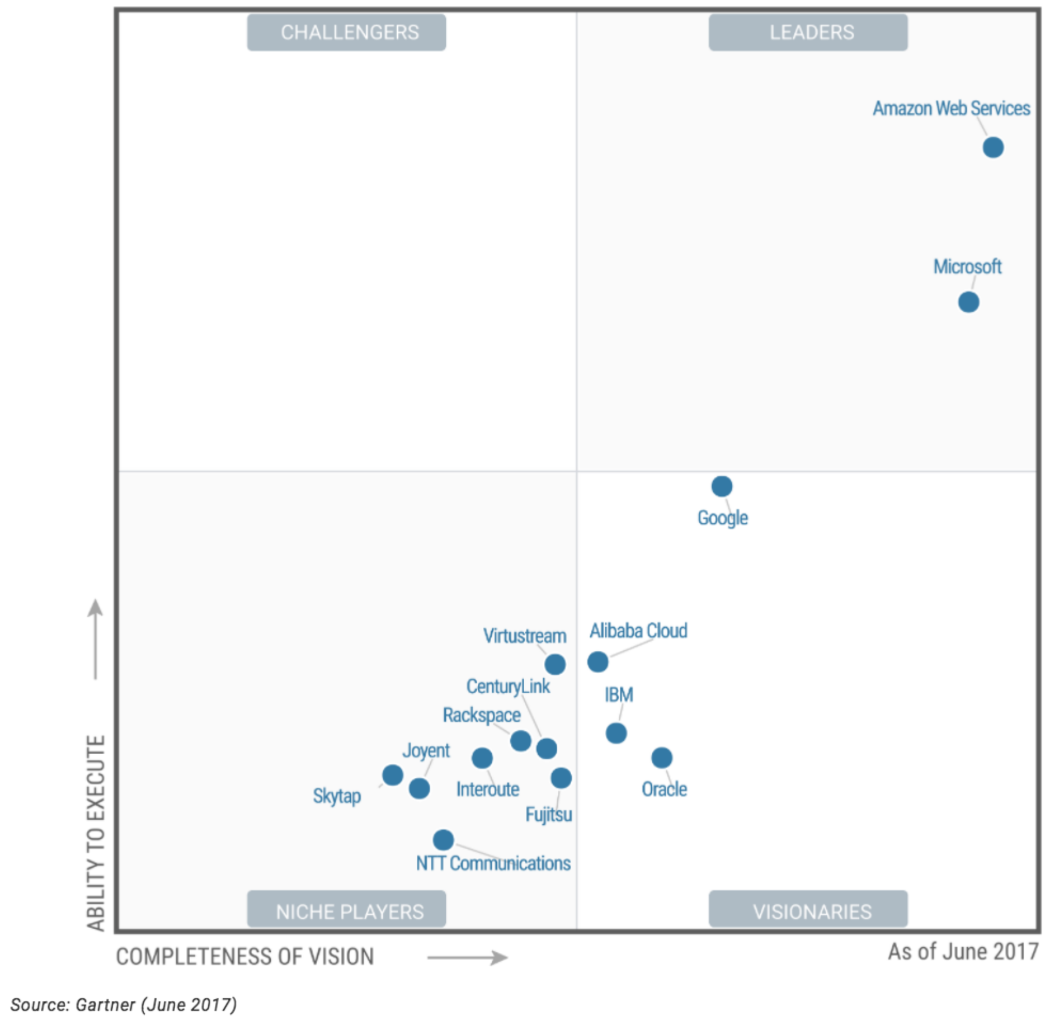


Figure 6.1: Cloud Infrastructure as a Service, Worldwide [85]

6.4 Evaluation metrics

To evaluate the efficiency, we measure the Web page response time and throughput of the DL services and the file download time. We consider that the performance of the cloud-based DL should either outperform or be similar to the existing performance of the on-premise DL. We select the top 100 most popular Web pages in Ensemble from the history log. These Web pages are either static or dynamic pages. The static page is usually an introduction page about a collection main page or a group main page. The dynamic page is usually a collection record page rendered by PHP, and data are fetched from the backend database or Solr server. We use these pages to measure the average response time. To measure the average file download time, we randomly select around 125 megabyte files from the Ensemble collection files for testing. These files are in various formats, such as audio, video, document, PDF, etc.

6.5 Architecture design for efficiency

Our cloud computing architecture designs presented in this section are to ensure the efficiency of the DL. We use average Web page response time and file download time as our measurements. To simulate real world scenarios, we set up clients from seven different locations around the world to conduct our experiments.

We propose three cloud computing architecture designs. These are Instance-based, Service-based, and Container-based. The Instance-based architecture creates exactly the same architecture as the on-premise architecture, Ensemble DL, in the cloud environment. The Container-based architecture packages each internal component in the Ensemble DL into several Docker images, and these Docker images are running in the container to support all DL services. The Service-based architecture replaces a system component in the Ensemble DL with cloud services if there are matches, or deploys a service into the cloud platform as an endpoint service. For example, a local MySQL database can be replaced by Amazon RDS for MySQL. To simplify the process to conduct experiments, we divide the Ensemble DL into three service components and recreate the Ensemble DL in the cloud environment according to each architecture design. These components are: Ensemble interface and repository (Application component), Solr server (Index component), and MySQL database (Storage component). The detailed architecture design is presented in the following sections.

6.5.1 Instance-based design

The most direct way to migrate an on-premise DL to a cloud-based DL is an instance-based design. The design is to create exactly the same environment (OS, software, etc.) in a cloud environment and then migrate the DL application, required software, and data into that environment.

The migration effort of this design is that we need to recreate the whole infrastructure in the cloud environment. The way we deploy the Ensemble DL in the AWS EC2 instance is as follows: First we select an instance type with the same CPU, Memory, and OS as the local Ensemble server and install all the software, such as MySQL and Solr server, and deploy the Ensemble applications. Second, we perform the data transfer task, which is to export data from the local MySQL server and import data into MySQL in the cloud instance. Last, we transfer all files from local storage to the cloud instance and storage.

Cloud advantages The AWS instance uses newer and faster computing equipment; thus the performance is immediately improved even using an instance with less CPU and memory. We can see the performance result with different instance types in Table 6.2.

Note that ARSP indicates the Average Response time for the top 100 pages in Ensemble. SRSP indicates the Total Response time for the 100 pages simultaneously. In AWS, the network bandwidth is assigned in different categories. “Low” means the network bandwidth

Table 6.2: The performance of different instance types and Ensemble

Instance Type	**vCPUs**	**Memory (GiB)**	**Network**	**ARSP (s)**	**SRSP (s)**
t2.medium	2	4	Low	1.28s	14.87s
t2.xlarge	4	16	Moderate	1.15s	13.22s
m4.xlarge	4	16	High	1.05s	10.79s
Ensemble	4	12	High	2.40s	75.23s

is from 50 MBit to 300 MBit, “Moderate” is from 300 MBit to 900 MBit, and “High” is from 900 MBit to 2.2 Gbit.

A t2.medium instance with 2 vCPUs and 4G memory performs better than the Ensemble DL with 4 vCPUs and 12GB memory. The Ensemble server CPU is 1900 MHz and is much less powerful compared to the current AWS instance CPU, which is 2400 MHz. One cloud advantage is that you can always use the most current compute power by re-provisioning your instance. Re-provisioning means to create an image of your instance and then provision a new instance using your image with the new instance type. Based on the DL loading and the performance you want to achieve, you can change your compute power on-demand.

Further, you can use two kinds of scaling approaches to improve DL performance. There are horizontal and vertical scaling. The vertical scaling, as we describe above, uses a larger instance with more compute power to improve the performance of the DL. The horizontal scaling means to use a load balancer in front of multiple instances, where each instance has the DL applications to provide services. Users’ requests will be distributed among these nodes and thus improve the performance. In order to know what kind of scaling would bring better performance we create multiple cloud-based Ensemble DLs in different instance types and measure the average response time for the top 100 pages in Ensemble. Table 6.2 shows the vertical scaling experiment result. A DL with more compute power has better performance.

We conduct another experiment using horizontal scaling. We use AWS auto scaling and define the number of instances to receive user requests using Elastic Load Balancing (ELB). We use a t2.medium instance for each node and host a DL. These instances receive user requests dispatched by the ELB. Figure 6.2 shows an AWS auto scaling example of this design.

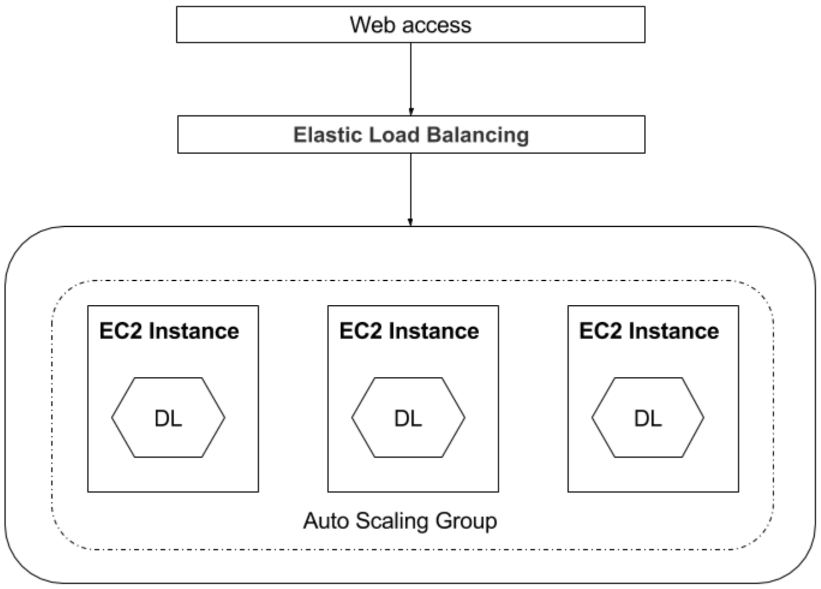


Figure 6.2: An AWS example for horizontal scaling instance-based DL

As the experiment result shows in Table 6.3 we can see the performance is improved when there are more nodes added under the ELB.

It is clear that adding more servers will improve the server response time when the DL faces high traffic. The approach also enhances the reliability of DL by avoiding a single point

Table 6.3: A stress test performance on an ELB with multiple instances

# of instances	**1**	**2**	**3**	**4**
SRSP (s)	14.87s	8.9s	6.85s	3.62s

of failure. Though local infrastructure can support similar tasks, you need to forecast the server loading and purchase servers ahead. When you are using cloud computing services, this kind of scaling can be done automatically and immediately. Take AWS for example, you can use a service such as Amazon CloudWatch [1] to monitor a server's loading and define an auto scaling group. When the server load is over a certain percentage, the service will automatically add a new instance into the load balancer. When the server load is reduced, the service will terminate the unused instance. This mechanism supports the use of computing power only when needed, and ensures our DL performance remains the same during different situations.

Moreover, with this design, you can use a small instance as a starting point, and add more servers automatically only during heavy loading and thus reduce your overall cost. Take an AWS instance for example: a t2.xlarge on-demand price is $0.188 per hour and a t2.medium on-demand price is $0.047. According to our results in Tables 6.2 and 6.3, you can use two t2.medium instances with lower cost and get better performance than using a single t2.xlarge instance.

6.5.2 Docker-based design

The Docker-based architecture design is used to package each internal service in the DL into a Docker image. Each Docker image runs in a single container in the cloud environment and provides services. Figure 6.3 shows a Docker-based Ensemble DL; we package three Docker images, which are MySQL, Solr, and Ensemble application images. We create an instance in the AWS and install a container engine to run multiple containers, where each container runs a Docker image.

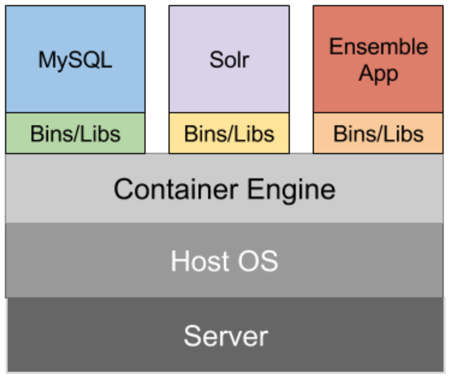


Figure 6.3: A Docker-based Ensemble DL

The major difference in this design is that you can treat each Docker image as a standalone block and specifically assign how much compute resource (such as CPU, memory, or block IO) to run the Docker image. It gives you the flexibility to allocate the compute resource specific to a Docker image that runs a DL internal service and further improve the overall DL performance.

The migration effort of this design is that you need to identify what DL internal components need to be packaged into a Docker image and also assign the appropriate compute resource. Some software such as MySQL database has already an official Docker image provided by MySQL, so you can directly use that in your DL application; otherwise, you need to create your own Docker image. The difficulty of packing an application into a Docker image depends on the complexity of the application.

Cloud advantages This design yields the same cloud advantages as the instance-based design on vertical scaling. The horizontal scaling is the same using ELB in front to distribute user requests; the difference is that cloud providers provide different container management services. Figure 6.4 shows an AWS example: the AWS EC2 Container Service (ECS) runs

Docker containers on a managed cluster of Amazon EC2 instances (ECS cluster).

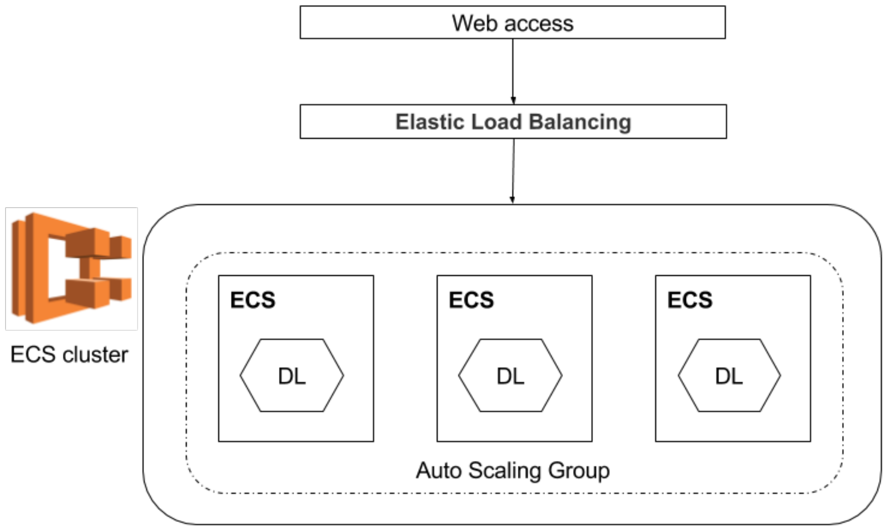


Figure 6.4: An AWS example for horizontal scaling Docker-based DL

6.5.3 Service-based design

The service-based architecture design is either using an existing cloud service or deploying the application into the cloud platform as an endpoint service. For example, use Amazon RDS for MySQL to replace a local MySQL database. Each service has its own instance and maintenance by the cloud providers. The cost model in endpoint services is different; the cost is calculated by how many requests are issued to your service. Taking AWS Lambda for example, the price is $0.20 per 1 million requests and $0.00001667 for every GB-second of compute. It is suitable for lightweight services, such as an OAI-PMH server, or a periodically scheduled job, such as an OAI-PMH harvester or crawler. The cost of this kind of service is cheaper than using an instance to run that service. Assuming that the AWS Lambda daily requests are 10,000 hits and 200 ms of execution time per hit at 1GB memory, the total cost per month is $1.06. A t2.micro (1GB memory) instance costs $8.79.

In service-based design, you can optimize each individual service to improve the overall performance of the DL. Using the Ensemble DL as an example, we can increase the size of the instance for the MySQL database but keep the Ensemble application instance the same. You don't have to provision a large instance to run every service; you manage each service separately. This design lets you have more options and combinations to improve your DL performance.

The migration effort depends on how you deploy your applications in the cloud environment. For the service that the cloud provider is already providing, such as database, the migration task is only data transfer. For a DL specific application, such as an OAI-PMH server, that the cloud provider does not provide, you need to package your application into the corresponding cloud services you chose. With AWS you can deploy the application in the AWS Elastic Beanstalk, AWS Lambda, or an instance that runs only that application.

Cloud advantages The major cloud advantage of this design is that you only need to focus on deploying the applications into the cloud environment and let the cloud service manage the related maintenance tasks for you. For example, AWS RDS for MySQL provides automated backup and path update. AWS Lambda lets you run a service without managing servers. You have more flexibility to optimize each DL service, utilize only the computing power needed, and further reduce the overall cost.

We built each Ensemble DL based on the three types of architecture design described above and did a stress test on these three designs. We also use AWS Simple Monthly Calculator to estimate the cost of each design. We conducted two experiments with different size of instances for each design and estimated the monthly cost. In the first experiment, we use t2.xlarge instance for instance-based and container-based and use two t2.medium and one db.t2.medium instance for service-based. The cost for instance-based and container-based is \$146.40, and the cost for service-based is \$118.60. In the second experiment, we use m4.2xlarge instance for instance-based and container-based and use one t2.xlarge, one

Table 6.4: A stress test performance comparison of three cloud-based designs

	Instance-based	**Service-based**	**Docker-based**
SRSP (s) - 1st	13.27s	12.34s	12.22s
SRSP (s) - 2nd	10.87s	10.26s	10.63s

db.t2.large, and one t2.medium instance for service-based. The cost for instance-based and container-based is \$292.80, and the cost for service-based is \$271.59. The stress test experiment results are shown in Table 6.4. In the first experiment, the docker-based DL performs best. In the second experiment, we increased the instance size of the Solr server and database in the service-based DL; we increased the instance size of both instance-based DL and container-based DL and allocated more compute resource for the Solr server and database in the container-based DL. The result shows that the service-based DL performs best and with the lowest cost.

6.5.4 Ensure efficiency using cloud storage

We measure the cloud computing benefits we can get using cloud storage and CDN (AWS CloudFront). We measure the response time for a static Web page and the download time for binary files. The experiment design is as follows: We create seven clients from seven locations around the world and measure the DL response time, one from Ensemble DL, located in Virginia, one from the cloud storage (AWS S3), and one from CDN.

According to the results shown in Table 6.5 and Table 6.6, we can see that CDN has the best response time, followed by AWS S3, while the local server performs least well. Some locations achieve response times similar to the local server. We can conclude that using CDN achieves better performance. Besides response time, files stored using the cloud storage approaches will receive multiple backups, file integrity, and stable network access – that we are not able to get from the local infrastructure. Moreover, cloud storage such as AWS S3 provides

Table 6.5: Static Web page response time from different locations

Location	**Ensemble (VA)**	**S3 static website**	**CDN**
Asia Pacific (Tokyo)	0.809s	0.551s	0.028s
US East (Virginia)	0.145s	0.045s	0.016s
US West (California)	0.386s	0.319s	0.020s
Canada (Central)	0.188s	0.103s	0.039s
EU (London)	0.408s	0.370s	0.221s
South America (São Paulo)	0.610s	0.506s	0.016s
Texas	0.271s	0.213s	0.055s

Table 6.6: File download time from different locations

Location	**Ensemble (VA)**	**S3 / Transfer Acceleration**	**CDN**
Asia Pacific (Tokyo)	2.674s	2.636s / 1.522s	0.057s
US East (Virginia)	0.132s	0.075s / 0.130s	0.038s
US West (California)	2.104s	1.842s / 0.368s	0.045s
Canada (Central)	0.351s	0.435s / 0.204s	0.094s
EU (London)	3.163s	2.367s / 0.727s	0.524s
South America (São Paulo)	1.499s	1.495s / 0.529s	0.045s
Texas	0.609s	0.795s / 0.486s	0.098s

unlimited storage, and thus one is not concerned about the shortage of storage space and the need to acquire a new disk within a time limit. Another benefit is that AWS S3 integrates with its own service which can archive data into AWS Glacier. Thus, we can store our infrequently accessed files into that service to save costs, and when we need a file, we can simply request it from the service. In AWS S3, you can configure CDN to get your binary files from AWS S3 and distribute them to the CDN edge locations that are near your users. Using this method we can ensure that the user can select retrieval content efficiently.

6.5.5 Cloud-based design to ensure reliability

Cloud computing services provide many features that we can use to ensure the reliability of the DL. We focus on system reliability and content reliability. The system reliability means that the DL is always running and responding to the user request. The content reliability is to ensure that files stored in the DL are always accessible and the content is durable and unchanged.

To avoid DL downtime and provide uninterrupted DL services, a system with high availability is needed. Our cloud-based design uses AWS to achieve high availability using Elastic Load Balancing (ELB) to avoid a single point of failure. Under this design, multiple instances are attached to the ELB and receive the clients' requests. If any of the instances crashes, the ELB will direct the clients' requests to other instances, terminate the crashed instance, provision a new instance, and attach the newly created instance to the ELB. We can assign the instances to be provisioned in different regions for disaster recovery.

We can utilize cloud storage to ensure content reliability in the DL. Cloud storage provides multiple backups, stores files in multiple facilities, and regularly verifies the integrity of the data using checksums; it is ideal for this purpose.

In order to enable a DL with cloud storage capability, our migration approach is to develop a mechanism that can move files from local disk storage to cloud storage. After the DL

Table 6.7: The file upload performance comparison between local and cloud storage

Location	**Upload file to local DL (VA)**	**Upload file to S3**
US West (California)	17.36s	30.16s
Asia Pacific (Tokyo)	58.87s	61.30s
EU (London)	33.00s	46.64s
Canada (Central)	8.77s	18.55s
South America (São Paulo)	33.12s	39.02s

is integrated with the cloud storage, when the user uploads files to the DL, these files will be uploaded to the cloud storage and managed by cloud storage for subsequent operations. The DL only needs to store the file metadata information, such as access URL, file size, content type, etc. In order to verify this approach, we developed an open source library [34] to enable a DL to store the files into the cloud storage and measure the performance of the file upload. Table 6.7 shows that it takes more time to upload the file to the cloud storage. That is because the file is first uploaded to the local storage of the DL, then moved to the cloud storage. However, after the file is uploaded to the cloud, we ensure the efficiency as shown in Table 6.6, as well as reliability.

The cost of the cloud storage is dependent on the size of the files stored in the cloud storage and the data transfer out of the cloud storage. It is suitable for a DL to adopt cloud storage since many DLs do not have high frequency downloading available. Moreover, cloud storage provides more options to store archived files. For example AWS provides Amazon Glacier with extremely low-cost storage service as storage for data archiving.

We conclude with our final design that ensures all of the DL service qualities, in Figure 6.5. This design ensures that there is no single point of failure, is able to scale by DL service level, and uses tailored cloud storage to manage files.

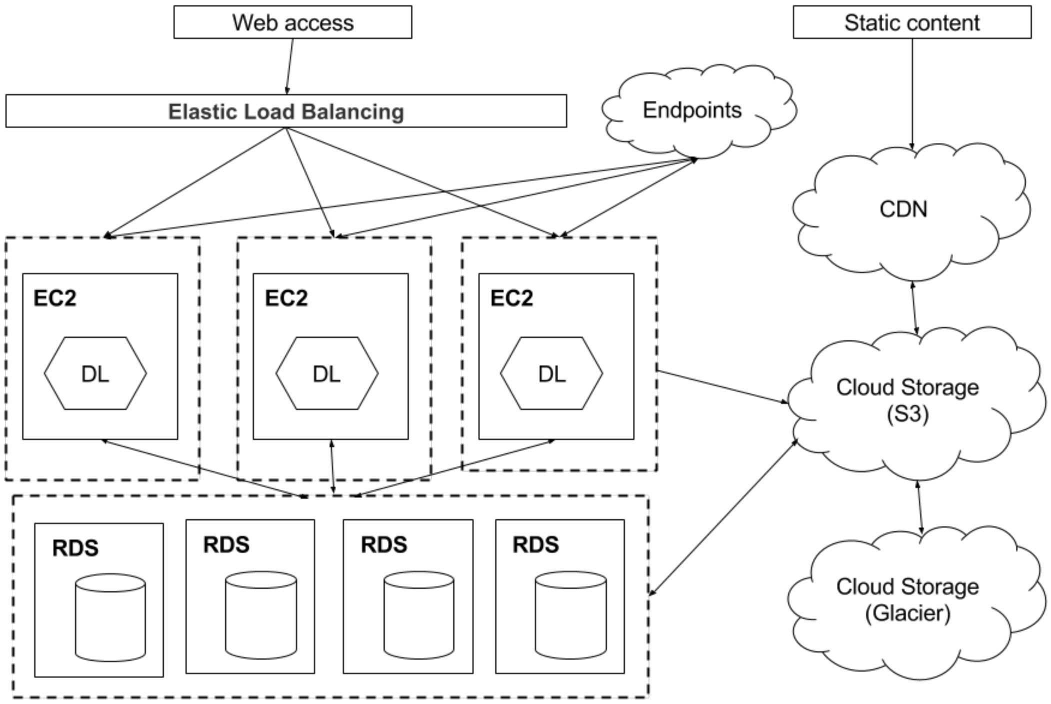


Figure 6.5: The service-based DL with cloud storage design

6.6 Summary

In this chapter, we studied different cloud-based architecture designs to ensure efficiency and reliability in a DL. We consider a large archive, managing only metadata, and long-term operations, to design the cloud-based architecture. We used an on-premise DL – Ensemble portal – as our test bed and migrated it into cloud-based DLs using different design approaches. We compared the performance of these designs and our experiment results show that the performance of the cloud-based DL is improved compared to the on-premise DL. We included advantages from the cloud computing services that help ensure DL efficiency and reliability.

We proposed three types of cloud-based designs, which are instance-based, service-based, and Docker-based. We list the benefits that are brought by the cloud computing services and describe the migration effort to transition an on-premise DL to the cloud-based DL in each design. Among these three different designs, the service-based is the most flexible, has moderate migration effort, and has better performance with the lowest cost. Moreover, we developed an open source library to manage files between the repository and the cloud storage. Our results show that file download time has greatly improved. Finally, we presented a cloud-based design that ensures the efficiency and reliability of the DL.

We propose that this research work can be a starting point for future DL development that ensures the quality of services in a DL. Moreover, a new economic model to build DLs could be developed by adopting cloud computing in the future. This can help solve one of the challenges of building an effective Digital Library as listed by the Library of Congress: Develop economic models to support national digital libraries.

Chapter 7

Conclusion and future directions

In this chapter, we summarize our contributions in this dissertation and conclude by describing future research directions.

7.1 Conclusion

In this dissertation, we presented our work to address both internal and external qualities of an educational DL. The internal qualities include repository and collection completeness and catalog consistency. The external qualities include service efficiency, reliability, and scalability. To achieve repository and collection completeness, we proposed a systematic approach to acquire educational resources in computing from Web 2.0. To achieve catalog consistency, we used the ACM CCS as our classification system and developed a transfer learning application to categorize resources in the DL. Our proposed cloud-based approaches not only ensured efficiency, reliability, and scalability but also brought the cloud computing advantages into a DL.

There are several contributions from this research work:

1. We established a collection aggregation framework that can acquire new educational

video and presentation slide records in computing topics from YouTube and SlideShare into the DL. See Chapter 4.

2. We built two classifier trees for labeling records from YouTube and SlideShare with ACM CCS using our proposed transfer learning method. See Section 5.5.

3. We identified issues in the current ACM DL dataset and described how we addressed these issues to train classifiers with better performance. See Chapter 3.

4. Our MTurk studies showed that we can use MTurk to evaluate the performance of the classifiers with reasonable cost and acquire results efficiently. See Section 5.4.

5. We proposed several cloud-based designs that address key DL service qualities using AWS. The performance of the cloud-based DL outperforms that of the local environment DL. See Chapter 6.

7.2 Future directions

There are several potential research directions that can be derived from the current work.

1. We can report our findings on the ACM dataset to ACM and potentially work with them to classify future articles, as well as current articles which have no CCS labels. We estimate there are over a million articles that might benefit from such an effort.

2. We can put the SlideShare and YouTube record addition process into production operation in Ensemble, adding large numbers of records, using crowdsourcing to enhance the transfer learning, and have incremental updating from those Web 2.0 sites as new resources are added.

3. We can extend our proposed methods to acquire education resources from other Web 2.0 sites, such as Vimeo, the Khan Academy, Google Computer Science Education, etc.

4. We can use our proposed method to classify existing collection records in Ensemble.
5. We can use different “oracles” for active learning; instead of using MTurk, we can implement a rewards system, like reputation and points in StackOverflow or Quora, in the DL, and ask our users to perform similar tasks as what we did in MTurk.
6. We can explore other cloud computing providers and compare different cloud-based designs to ensure the long-term operation of the DL, with high quality services.

Finally, we plan to publish and disseminate our results further, extending the set of conference and journal papers already published that relate to this dissertation.

Bibliography

[1] Amazon CloudWatch. `https://aws.amazon.com/cloudwatch/`. [Online; accessed 20-April-2017].

[2] Amazon Mechanical Turk. `https://www.mturk.com/mturk/`. [Online; accessed 20-April-2017].

[3] SlideShare API. `https://www.slideshare.net/developers`. [Online; accessed 20-April-2017].

[4] YouTube Data API. `https://developers.google.com/youtube/v3/`. [Online; accessed 20-April-2017].

[5] Dublin Core Metadata Editor (DCDOT). `http://www.ukoln.ac.uk/metadata/dcdot`, 2000. [Online; accessed 20-April-2017].

[6] In *HICSS '03 Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, volume 5, page 133, Washington, DC, USA, 2003. IEEE Computer Society. `http://dl.acm.org/citation.cfm?id=821593`.

[7] B Thomas Adler and Luca De Alfaro. A content-driven reputation system for the Wikipedia. In *Proceedings of the 16th international conference on World Wide Web*, pages 261–270. ACM, 2007.

[8] Hisham Al-Mubaid and Syed A Umair. A new text categorization technique using distributional clustering and learning logic. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1156–1165, 2006.

[9] Kathrin Ambrozic, Denis Galvin, and Mang Sun. Avoiding the death zone: choosing and running a library project in the cloud. *Library Hi Tech*, 30(3):418–427, 2012.

[10] Rie Kubota Ando and Tong Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 1–9, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[11] Chidanand Apté, Fred Damerau, and Sholom M Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, 1994.

[12] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

[13] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[14] Jeffrey Beall. Metadata and data quality problems in the digital library. *Journal of Digital Information*, 6(3), 2006.

[15] Justin Bedo, Conrad Sanderson, and Adam Kowalczyk. An efficient alternative to SVM based recursive feature elimination with applications in natural language processing and bioinformatics. *AI 2006: Advances in Artificial Intelligence*, pages 170–180, 2006.

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[17] John Blitzer, Mark Dredze, Fernando Pereira, et al. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.

[18] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[19] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[20] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task Gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008.

[21] C Mic Bowman, Peter B Danzig, Darren R Hardy, Udi Manber, Michael F Schwartz, and Duane P Wessels. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, August 1994.

[22] Roger B Bradford. An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 153–162. ACM, 2008.

[23] Thomas R Bruce and Diane I Hillmann. The continuum of metadata quality: defining, expressing, exploiting. In Diane I. Hillmann & Elaine L. Westbrooks (Eds.), Metadata in practice, pages 238–256. American Library Association, Chicago, US, 2004.

[24] George Buchanan, David Bainbridge, Katherine J Don, and Ian H Witten. A new framework for building digital library collections. In *Proceedings of the 5th ACM-IEEE Joint Conference on Digital Libraries*, pages 23–31, 2005.

[25] Yen Bui and Jung-ran Park. An assessment of metadata quality: A case study of the National Science Digital Library Metadata Repository. In *Proceedings of the Annual Conference of CAIS/Actes du congrès annuel de l'ACSI*, 2013.

[26] L. Candela, D. Castelli, N. Ferro, Y. Ioannidis, G. Koutrika, C. Meghini, P. Pagano, S. Ross, D. Soergel, M. Agosti, Milena Dobreva, V. Katifori, and H. Schuldt. *The DELOS Digital Library Reference Model. Foundations for Digital Libraries.* ISTI-CNR, December 2007. DELOS Network of Excellence on Digital Libraries Project no. 507618.

[27] Asli Celikyilmaz, Dilek Hakkani-Tur, and Gokhan Tur. LDA based similarity modeling for question answering. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*, SS '10, pages 1–9, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[28] Rita Chattopadhyay, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Joint transfer and batch-mode active learning. In *International Conference on Machine Learning*, pages 253–261, 2013.

[29] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in neural information processing systems*, pages 2456–2464, 2011.

[30] Mu-Yen Chen, Edwin David Lughofer, AN Zainab, CY Chong, and LT Chaw. Moving a repository of scholarly content to a cloud. *Library Hi Tech*, 31(2):201–215, 2013.

[31] Yinlin Chen, Paul Logasa Bogen II, Haowei Hsieh, Edward A Fox, and Lillian N Cassel. Categorization of computing education resources with utilization of crowdsourcing. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 121–124. ACM, 2012.

[32] Yinlin Chen and Edward A Fox. Using ACM DL paper metadata as an auxiliary source for building educational collections. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 137–140. IEEE Press, 2014.

[33] Yinlin Chen, Edward A Fox, and Tingting Jiang. Performance of a cloud-based digital library. In *Digital Libraries: Providing Quality Information: 17th International Conference on Asia-Pacific Digital Libraries, ICADL 2015, Seoul, Korea, December 9-12, 2015. Proceedings*, volume 9469, page 298. Springer, 2015.

[34] Yinlin Chen, Zhiwu Xie, and Edward A Fox. A library to manage web archive files in cloud storage. Poster presented at WADL 2016: Third International Workshop on Web Archiving and Digital Libraries, June 22-23, 2016. In connection with ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2016), Rutgers Univ., Newark, NJ, 2016. `http://www.ieee-tcdl.org/mediawiki/TCDL/Bulletin//v13n1/papers/chen.pdf`.

[35] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[36] ACM CCS Update Committee. The 2012 ACM Computing Classification System [2012 Version] http://www.acm.org/about/class/2012, 2012.

[37] Wikimedia Commons. Plate notation of the smoothed LDA model. `https://commons.wikimedia.org/wiki/File:Smoothed_LDA.png`. [Online; accessed 20-April-2017].

[38] Myra Custard and Tamara Sumner. Using machine learning to support quality judgments. *D-Lib Magazine*, 11(10):1082–9873, 2005.

[39] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM, 2007.

[40] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring Naive Bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545, 2007.

[41] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. *Proceedings of the 24th International Conference on Machine Learning (2007)*, 108(2):193–200, 2007.

[42] Sanmay Das, Milton Saier, and Charles Elkan. Finding transport proteins in a general protein database. *Knowledge Discovery in Databases: PKDD 2007*, pages 54–66, 2007.

[43] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156, 1997.

[44] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[45] Hal Daumé III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics, 2010.

[46] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[47] James R Davis and Carl Lagoze. NCSTRL: design and deployment of a globally distributed digital library. *Journal of the American Society for Information Science*, 51(3):273–280, 2000.

[48] Jesse Davis and Pedro Domingos. Deep transfer via second-order Markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224. ACM, 2009.

[49] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[50] Ranjeet Devarakonda, Giri Palanisamy, James M Green, and Bruce E Wilson. Data sharing and retrieval using OAI-PMH. *Earth Science Informatics*, 4(1):1–5, 2010.

[51] ACM DL. The ACM Digital Library. `http://www.acm.org/dl`, 2004. [Online; accessed 20-April-2017].

[52] Naomi Dushay and Diane I. Hillmann. Analyzing metadata for effective use and re-use. In *Proceedings of the 2003 international conference on Dublin Core and metadata applications: supporting communities of discourse and practice—metadata research & applications*, DCMI '03, pages 17:1–17:10. Dublin Core Metadata Initiative, 2003.

[53] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[54] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. Towards systematic traffic annotation. In *Proceedings of the 5th international student workshop on Emerging networking experiments and technologies*, pages 15–16. ACM, 2009. `http://dx.doi.org/10.1145/1658997.1659006`.

[55] Edward A Fox, Marcos Andre Goncalves, and Rao Shen. *Theoretical foundations for Digital Libraries: The 5S (societies, scenarios, spaces, structures, streams) approach.* Morgan & Claypool Publishers, San Francisco, July 2012. 180 pages, ISBN paperback 9781608459100, ebook 9781608459117, `http://dx.doi.org/10.2200/S00434ED1V01Y201207ICR022`, supplementary website `https://sites.google.com/a/morganclaypool.com/dlibrary/`.

[56] Robert Fox. Library in the clouds. *OCLC Systems & Services: International digital library perspectives*, 25(3):156–161, 2009.

[57] Eibe Frank and Remco R Bouckaert. Naive Bayes for text classification with unbalanced classes. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 503–510. Springer, 2006.

[58] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers*, 100(7):750–753, 1975.

[59] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 283–291. ACM, 2008.

[60] Marcos André Gonçcalves. *Streams, structures, spaces, scenarios, and societies (5S): a formal digital library framework and its applications.* PhD thesis, 2004. `http://hdl.handle.net/10919/20034`.

[61] Marcos André Gonçalves, Bárbara Lagoeiro Moreira, Edward A Fox, and Layne T Watson. What is a good digital library? - defining a quality model for digital libraries. *Information Processing & Management*, 43(5):1416–1437, 2007.

[62] Jane Greenberg. Metadata generation: Processes, people and tools. *Bulletin of the American Society for Information Science and Technology*, 29(2):16–19, 2003.

[63] Jane Greenberg. Metadata extraction and harvesting. *Journal of Library Metadata*, 6(4):59–82, 2004.

[64] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[65] Steve R Gunn et al. Support Vector Machines for classification and regression. *ISIS technical report*, 14:85–86, 1998.

[66] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Using kNN model for automatic text categorization. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 10(5):423–430, 2006.

[67] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[68] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[69] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent Dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[70] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in NLP. In *ACL*, volume 7, pages 264–271, 2007.

[71] Thorsten Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142, 1998.

[72] Thorsten Joachims. Transductive inference for text classification using Support Vector Machines. In *ICML*, volume 99, pages 200–209, 1999.

[73] David E Johnson, Frank J Oles, and Tong Zhang. Decision-tree-based symbolic rule induction system for text categorization, February 11 2003. US Patent 6,519,580.

[74] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16(06):1047–1067, 2007.

[75] Ashraf Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial Naive Bayes for text categorization revisited. *AI 2004: Advances in Artificial Intelligence*, pages 235–252, 2005.

[76] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.

[77] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[78] Barbara Lagoeiro, Marcos André Gonçalves, and Edward. A Fox. 5SQual: A quality tool for digital libraries. In *Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries (JCDL '07)*, pages 513–513, New York, NY, USA, 2007. ACM.

[79] Carl Lagoze and Herbert Van de Sompel. The Open Archives Initiative: building a low-barrier interoperability framework. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 54–62, New York, NY, USA, 2001. ACM Press.

[80] Carl Lagoze, Dean Krafft, Tim Cornwell, Naomi Dushay, Dean Eckstrom, and John Saylor. Metadata aggregation and "automated digital libraries": A retrospective on the NSDL experience. *Joint Conference on Digital Libraries*, 06pp:230–239, 2006.

[81] Carl Lagoze and Herbert Van de Sompel. The making of the Open Archives Initiative protocol for metadata harvesting. *Library hi tech*, 21(2):118–128, 2003.

[82] Neil D Lawrence and John C Platt. Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*, page 65. ACM, 2004.

[83] Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, pages 489–496. ACM, 2007.

[84] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the cloud? an architectural map of the cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 23–31. IEEE Computer Society, 2009.

[85] Lydia Leong, Raj Bala, Craig Lowery, and Dennis Smith. Magic quadrant for cloud infrastructure as a service, worldwide. `https://www.gartner.com/doc/reprints?id=1-2G205FC&ct=150519&st=sb`. [Online; accessed 15-June-2017].

[86] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[87] Rachel Tsz-Wai Lo, Ben He, and Iadh Ounis. Automatically building a stopword list for an information retrieval system. In *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, volume 5, pages 17–24, 2005.

[88] Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[89] Julie Beth Lovins. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2):22–31, 1968.

[90] Weiming Lu, Liangju Zheng, Jian Shao, Baogang Wei, and Yueting Zhuang. Digital library engine: Adapting digital library for cloud computing. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 934–941. IEEE, 2013.

[91] Thomas Malloy and Gerard Hanley. MERLOT: A faculty-focused Web site of educational resources. *Behavior Research Methods*, 33(2):274–276, 2001.

[92] Nathan Mantel. Chi-square tests with one degree of freedom; extensions of the Mantel-Haenszel procedure. *Journal of the American Statistical Association*, 58(303):690–700, 1963.

[93] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for Naive Bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI, 1998.

[94] Mark Meredith and Bhuvan Urgaonkar. Towards performance modeling as a service by exploiting resource diversity in the public cloud. In *Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on*, pages 204–211. IEEE, 2016.

[95] Lilyana Mihalkova, Tuyen Huynh, and Raymond J Mooney. Mapping and revising Markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007.

[96] Lilyana Mihalkova and Raymond J Mooney. Transfer learning by mapping with minimal target data. In *Proceedings of the AAAI-08 workshop on transfer learning for complex tasks*, 2008.

[97] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002.

[98] Bárbara L Moreira, Marcos André Gonçalves, Alberto H F Laender, and Edward A Fox. Automatic evaluation of digital libraries with 5SQual. *Journal of Informetrics*, 3(2):102–123, 2009.

[99] Uma Murthy, Douglas Gorton, Ricardo Torres, Marcos Gonalves, Edward Fox, and Lois Delcambre. Extending the 5S digital library (DL) framework: From a minimal DL towards a DL reference model. In *1st Workshop on Digital Library Foundations, ACM/IEEE-CS Joint Conference on Digital Libraries*, Vancouver, British Columbia, Canada, June 2007. `http://si.dlib.vt.edu/publications/2007_DLRL_DLF_v4.pdf`.

[100] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *ACM SIGIR Forum*, volume 31, pages 67–73. ACM, 1997.

[101] David M. Nichols, Chu-Hsiang Chan, David Bainbridge, Dana McKay, and Michael B. Twidale. A lightweight metadata quality tool. In *Proceedings of the 8th ACM/IEEE-*

CS joint conference on Digital libraries, JCDL '08, pages 385–388, New York, NY, USA, 2008. ACM.

[102] Nobal Niraula, Rajendra Banjade, Dan Ştefănescu, and Vasile Rus. *Experiments with Semantic Similarity Measures Based on LDA and LSA*, pages 188–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[103] OAI. The open archives initiative protocol for metadata harvesting – version 2.0. `http://www.openarchives.org/OAI/openarchivesprotocol.html`, October 2004. [Online; accessed 20-April-2017].

[104] Xavier Ochoa and Erik Duval. Automatic evaluation of metadata quality in digital repositories. *International Journal on Digital Libraries*, 10(2-3):67–91, 2009.

[105] Library of Congress. Challenges to building an effective digital library. `https://memory.loc.gov/ammem/dli2/html/cbedl.html`. [Online; accessed 20-April-2017].

[106] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[107] Mark Patton, David Reynolds, G Sayeed Choudhury, and Tim DiLauro. Toward a metadata generation framework. *DLib Magazine*, 10(11):1082–9873, 2004.

[108] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[109] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: Transfer learning from unlabeled data. *Learning*, 24(10-11):759–766, 2007.

[110] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of database systems*, pages 532–538. Springer, 2009.

[111] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[112] Irina Rish. An empirical study of the Naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[113] Martin Röscheisen, Michelle Baldonado, Kevin Chang, Luis Gravano, Steven Ketchpel, and Andreas Paepcke. The Stanford InfoBus and its service layers: Augmenting the Internet with higher-level information management protocols. In *Digital Libraries in Computer Science: The MeDoc Approach*, pages 213–230. Springer, 1998.

[114] David SH Rosenthal and Daniel L Vargas. Distributed digital preservation in the cloud. *International Journal of Digital Curation*, 8(1):107–119, 2013.

[115] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105, 1998.

[116] Kanoksri Sarinnapakorn and Miroslav Kubat. Combining subclassifiers in text categorization: A DST-based solution and a case study. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1638–1651, 2007.

[117] Sam Scott and Stan Matwin. Text classification using WordNet hypernyms. In *Workshop on usage of WordNet in NLP Systems (COLING-ACL '98)*, pages 45–51, August 1998.

[118] Sam Scott and Stan Matwin. Feature engineering for text classification. In *ICML*, volume 99, pages 379–388, 1999.

[119] Aditya Kumar Sehgal, Sanmay Das, Keith Noto, Milton Saier, and Charles Elkan. Identifying relevant data for a biological database: Handcrafted rules versus machine learning. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(3):851–857, 2011.

[120] Burr Settles. Active learning literature survey. *Computer Sciences Technical Report 1648, University of Wisconsin, Madison*, 52(55-66):11, 2010.

[121] Rao Shen, Marcos Andre Goncalves, and Edward A Fox. *Key issues regarding Digital Libraries: Evaluation and integration*. Morgan & Claypool Publishers, San Francisco, Feb 2013. 110 pages, ISBN paperback 9781608459124, ebook 9781608459131, `http://dx.doi.org/10.2200/S00474ED1V01Y201301ICR026`.

[122] Xiaoxiao Shi, Wei Fan, and Jiangtao Ren. Actively transfer domain knowledge. *Machine Learning and Knowledge Discovery in Databases*, pages 342–357, 2008.

[123] Sergej Sizov. Geofolk: latent spatial semantics in Web 2.0 social media. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 281–290. ACM, 2010.

[124] Alexander Sorokin and David Forsyth. Utility data annotation with Amazon Mechanical Turk. *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 51(c):1–8, 2008.

[125] Pascal Soucy and Guy W Mineau. A simple KNN algorithm for text categorization. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 647–648. IEEE, 2001.

[126] Hussein Suleman and Edward Fox. The Open Archives Initiative: realizing simple and effective digital library interoperability. In special issue on “Libraries and Electronic Resources: New Partnerships, New Practices, New Perspectives” of J. Library Automation, 35(1-2):125–145, 2002.

[127] Byung-Chul Tak, Bhuvan Urgaonkar, and Anand Sivasubramaniam. To move or not to move: The economics of cloud computing. In *HotCloud*, 2011.

[128] Pradeep Teregowda and C Lee Giles. Scaling SeerSuite in the cloud. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 146–155. IEEE, 2013.

[129] Pradeep Teregowda, Bhuvan Urgaonkar, and C Lee Giles. Cloud computing: A digital libraries perspective. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 115–122. IEEE, 2010.

[130] Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[131] Chang Wan, Rong Pan, and Jiefei Li. Bi-weighting domain adaptation for cross-language text classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1535, 2011.

[132] Yair Wand and Rachard Y Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, November 1996.

[133] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.

[134] Zhiwu Xie, Yinlin Chen, Julie Speer, and Tyler Walters. Evaluating cost of cloud execution in a data repository. In *Digital Libraries (JCDL), 2016 IEEE/ACM Joint Conference on*, pages 247–248. IEEE, 2016.

[135] Liu Yang, Steve Hanneke, and Jaime Carbonell. A theory of transfer learning with applications to active learning. *Machine learning*, 90(2):161–189, 2013.

[136] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69–90, 1999.

[137] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.

[138] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

[139] Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103, New York, NY, USA, 2003. ACM.

[140] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224, 2004.

[141] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.

[142] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing Twitter and traditional media using topic models. In *European Conference on Information Retrieval*, pages 338–349. Springer, 2011.

[143] Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *ACM SIGKDD Explorations Newsletter*, 6(1):80–89, 2004.

[144] Zhenfeng Zhu, Xingquan Zhu, Yangdong Ye, Yue-Fei Guo, and Xiangyang Xue. Transfer active learning. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2169–2172. ACM, 2011.

[145] Hui Zou and Trevor Hastie. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A

ACM Database Schema

The SQL statements below describe the schema of the ACM database used in this dissertation. I devised this schema based on the metadata format for the data provided by ACM.

Table of ACM CCS

```
CREATE TABLE ccs (
        ccsname varchar(300) NOT NULL,
        ccsid varchar(250) NOT NULL,
        PRIMARY KEY (ccsid)
)
```

Table of ACM Papers

```
CREATE TABLE papers (
        sid int(11) NOT NULL,
        id varchar(250) NOT NULL,
        concept_id varchar(250) NOT NULL,
        concept_desc varchar(500) NOT NULL,
        cite int(11) NOT NULL,
        PRIMARY KEY (sid)
)
```

Appendix B

Crawler Database Schema

The SQL statements below describe the schema of the YouTube and SlideShare databases used in this dissertation. I devised this schema based on the YouTube API JSON-C response [4] and SlideShare API XML response [3].

Table of YouTube

```
CREATE TABLE YouTube (
        yid varchar(150) NOT NULL,
        url varchar(500) NOT NULL,
        embedurl varchar(500) DEFAULT NULL,
        likes int(11) DEFAULT NULL,
        author varchar(100) DEFAULT NULL,
        title varchar(500) NOT NULL,
        duration varchar(100) DEFAULT NULL,
        rating float DEFAULT NULL,
        views int(11) DEFAULT NULL,
        dislikes int(11) DEFAULT NULL,
        category varchar(400) DEFAULT NULL,
        thumbnail varchar(200) DEFAULT NULL,
        published varchar(100) DEFAULT NULL,
        description text,
        keywords varchar(500) DEFAULT NULL,
        length int(11) NOT NULL,
        comments int(11) DEFAULT NULL,
        PRIMARY KEY (yid)
)
```

Table of SlideShare

```
CREATE TABLE slideshare (
        sid varchar(150) NOT NULL,
        title varchar(150) NOT NULL,
        description text NOT NULL,
        username varchar(150) NOT NULL,
        url varchar(150) NOT NULL,
        embedurl text NOT NULL,
        createdate varchar(50) NOT NULL,
        language varchar(50) NOT NULL,
        format varchar(50) NOT NULL,
        SlideshowEmbedUrl varchar(500) NOT NULL,
        tags varchar(500) NOT NULL,
        downloads int(11) NOT NULL,
        views int(11) NOT NULL,
        favorites int(11) NOT NULL,
        pages int(11) NOT NULL,
        relatedslides varchar(200) NOT NULL,
        PRIMARY KEY (sid)
```

)

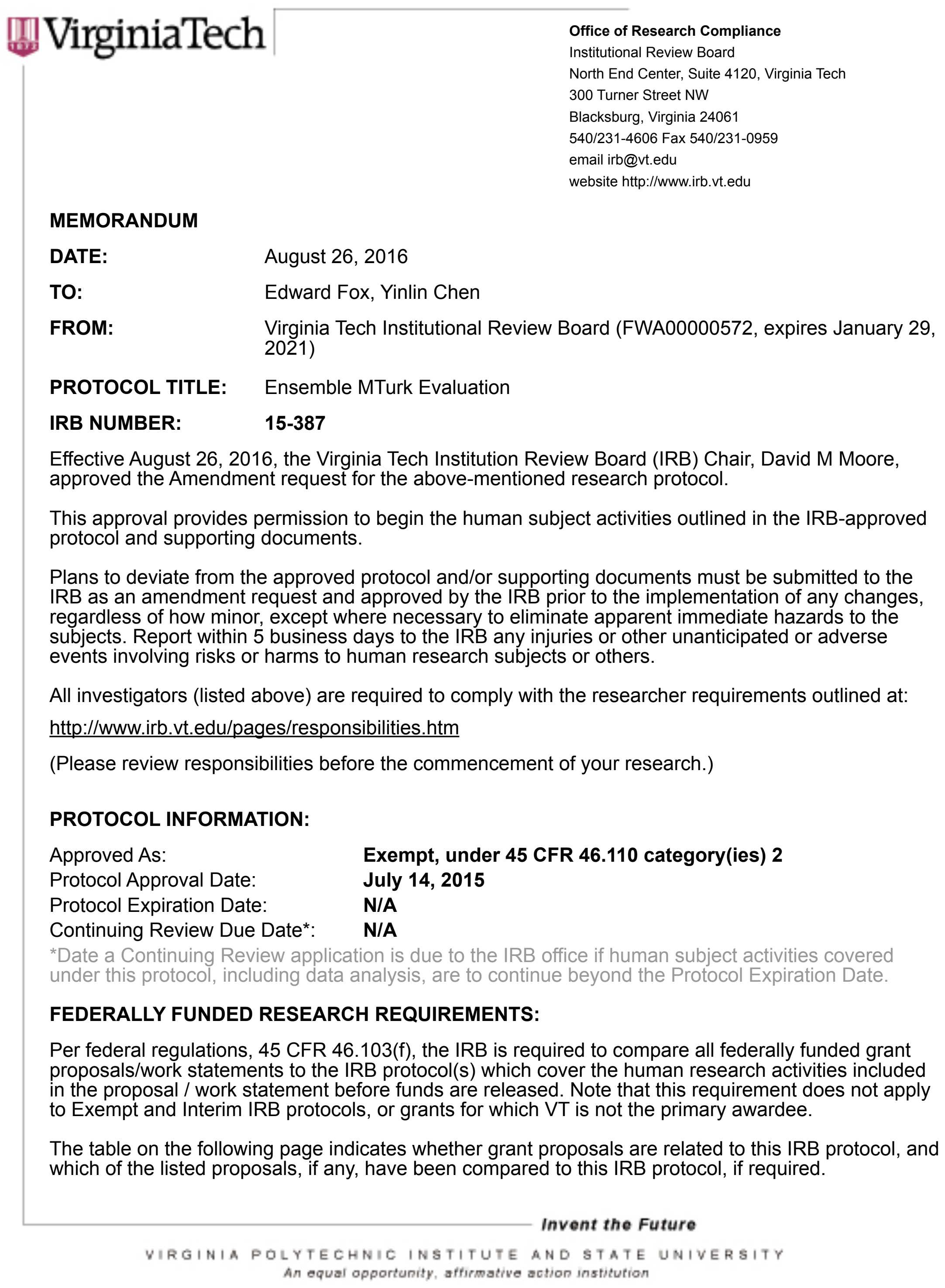
Table of YouTube Playlist

```
CREATE TABLE playlist (
        pid varchar(250) NOT NULL,
        url varchar(500) NOT NULL,
        title varchar(500) NOT NULL,
        author varchar(200) NOT NULL,
        description text NOT NULL,
        checked int(11) NOT NULL DEFAULT 0 COMMENT “video metadata down-
        loads”,
        PRIMARY KEY (pid)
)
```

Appendix C

IRB

We received approval from the Virginia Tech Institutional Review Board for conducting the MTurk experiment on August 26, 2016. See approval letter on the next page.



VirginiaTech

Office of Research Compliance
Institutional Review Board
North End Center, Suite 4120, Virginia Tech
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE:	August 26, 2016
TO:	Edward Fox, Yinlin Chen
FROM:	Virginia Tech Institutional Review Board (FWA00000572, expires January 29, 2021)
PROTOCOL TITLE:	Ensemble MTurk Evaluation
IRB NUMBER:	**15-387**

Effective August 26, 2016, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the Amendment request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Exempt, under 45 CFR 46.110 category(ies) 2**
Protocol Approval Date:	**July 14, 2015**
Protocol Expiration Date:	**N/A**
Continuing Review Due Date*:	**N/A**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Invent the Future

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
An equal opportunity, affirmative action institution