Enabling Automated and Efficient Personalization Systems

Verma, Chetan Kumar

2015

Peer reviewed|Thesis/dissertation
Enabling Automated and Efficient Personalization Systems

A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Chetan Kumar Verma

Committee in charge:

Professor Sujit Dey, Chair
Professor Vikash Gilja
Professor David Kriegman
Professor Truong Nguyen
Professor Lawrence Saul

2015
The Dissertation of Chetan Kumar Verma is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2015
DEDICATION

To mom and dad
# TABLE OF CONTENTS

Signature Page .................................................................................. iii
Dedication ......................................................................................... iv
Table of Contents ............................................................................... v
List of Figures .................................................................................. viii
List of Tables .................................................................................... xi
Acknowledgements .......................................................................... xiii
Vita ................................................................................................. xv
Abstract of the Dissertation .......................................................... xvii

## Chapter 1  Introduction

1.1 Personalization Systems .............................................................. 1

1.2 Contributions and Overview .................................................... 2

1.2.1 Automatically obtaining training data to enable content classification ........................................... 3

1.2.2 Addressing sparsity in annotations of online content ................................................................. 4

1.2.3 Improving time efficiency of personalized recommendations in enterprises .................................... 5

## Chapter 2  Methods to Obtain Training Data for Fully Automated Application-Specific Classification

2.1 Introduction ............................................................................... 7

2.1.1 Related work ........................................................................ 9

2.2 Desired properties of training data ......................................... 13

2.3 Overview of the proposed framework to obtain training videos .......................................................... 16

2.3.1 Overview of proposed framework ........................................ 17

2.3.2 Selection Procedure for Selected Retriever Keywords (SRKs) .................................................. 18

2.4 Approaches to select SRKs ...................................................... 24

2.4.1 Linear combination of proximity and diversity (LCPD) ............................................................... 24

2.4.2 Annealing based alternating optimization (AAO) for proximity and diversity maximization ............. 30

2.5 Experimental Results ............................................................ 46

2.5.1 Retail Product categories: Baby, Clothing, Fitness, Food .............................................................. 47

2.5.2 Genres of Music: Classical, Electronic, Jazz, Rock ................................................................. 52

2.5.3 Genres of Movies: Action, Comedy, Horror, Romantic .............................................................. 53

2.6 Summary ............................................................................... 54
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Construction and Evaluation of Ontological Tag Trees</td>
<td>56</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>57</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Related Work</td>
<td>63</td>
</tr>
<tr>
<td>3.2</td>
<td>Problem Statement</td>
<td>70</td>
</tr>
<tr>
<td>3.3</td>
<td>Construction of Ontological Tag Tree</td>
<td>70</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Constructing WordNet-based Preliminary Tag Tree</td>
<td>70</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Co-occurrence based Tag Tree Refinement</td>
<td>71</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Optimization based on Local Search</td>
<td>75</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Effect of initializing using WordNet</td>
<td>76</td>
</tr>
<tr>
<td>3.4</td>
<td>Evaluation</td>
<td>78</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Datasets</td>
<td>78</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Effect of Local Search based Optimization</td>
<td>79</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Tag Prediction Task</td>
<td>80</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Efficient Classification Task</td>
<td>90</td>
</tr>
<tr>
<td>3.5</td>
<td>Robustness Analysis</td>
<td>95</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Robustness to Label Noise in Training Data</td>
<td>96</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Robustness to Difference between Training and Test Data</td>
<td>98</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Size of Training Data</td>
<td>99</td>
</tr>
<tr>
<td>3.6</td>
<td>Discussion</td>
<td>99</td>
</tr>
<tr>
<td>3.7</td>
<td>Summary</td>
<td>104</td>
</tr>
<tr>
<td>3.8</td>
<td>Acknowledgments</td>
<td>107</td>
</tr>
<tr>
<td>4</td>
<td>Improving Scalability of Personalized Recommendation Systems for</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>Enterprise Knowledge Workers</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>109</td>
</tr>
<tr>
<td>4.2</td>
<td>The File Recommendation System</td>
<td>111</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Metadata Features</td>
<td>113</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Modeling Techniques</td>
<td>113</td>
</tr>
<tr>
<td>4.3</td>
<td>Evaluation Framework</td>
<td>115</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Data</td>
<td>115</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Evaluation Methodology</td>
<td>117</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance Evaluation</td>
<td>119</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Metadata Models</td>
<td>119</td>
</tr>
<tr>
<td>4.4.2</td>
<td>CF Models</td>
<td>121</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Temporal Variation in Performance</td>
<td>121</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Scalability</td>
<td>125</td>
</tr>
<tr>
<td>4.5</td>
<td>Active Features-based Model Selection (AFMS)</td>
<td>127</td>
</tr>
<tr>
<td>4.5.1</td>
<td>AFMS Algorithm</td>
<td>128</td>
</tr>
<tr>
<td>4.5.2</td>
<td>AFMS Evaluation</td>
<td>132</td>
</tr>
<tr>
<td>4.6</td>
<td>Related Work</td>
<td>133</td>
</tr>
<tr>
<td>4.7</td>
<td>Summary</td>
<td>135</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1. (a) Sample training videos for categories: \{Baby, Clothing\}. Circled video is wrongly placed in category Baby, and is hence a mislabeled video. (b) Variety of video topics belonging to category Baby .......... 14

Figure 2.2. Selection criteria for Selected Retriever Keywords .................. 19

Figure 2.3. Most proximate and diverse keywords for the category ‘Baby’ .... 19

Figure 2.4. Overview of the AAO approach for alternately optimizing the average proximity and diversity of selected keywords ...................... 33

Figure 2.5. Variation in average proximity (red) and diversity (blue) of selected keywords for selecting 5 SRKs from 20 valid keywords of category “Baby” using (a) AAO with $\Delta = 0.005$, (b) AAO with $\Delta = 0.05$, and (c) Adapt AAO. ............................................ 45

Figure 2.6. Classification accuracy and Intra-Category Diversity variation with number of training videos for baseline classifier ......................... 48

Figure 2.7. Classification accuracy variation for \{Baby, Clothing, Fitness, Food\} with respect to $\alpha$. Dotted line shows baseline accuracy ... 48

Figure 2.8. Classification accuracy variation for \{Baby, Clothing, Fitness, Food\} with respect to $L$ .................................................. 48

Figure 2.9. Variation of time taken to select $L$ SRKs, and time taken to train linear SVM on obtained training data, with respect to $L$ ............... 49

Figure 2.10. Classification accuracy variation for \{Classical music, Electronic Music, Jazz music, Rock music\} with respect to $\alpha$ ............... 49

Figure 2.11. Classification accuracy for \{Action movies, Comedy movies, Horror movies, Romantic movies\} with respect to $\alpha$ ............... 49

Figure 3.1. Examples of incorrect tags given by users on www.flickr.com. (a) An image of a ‘cat’ tagged as ‘car’, which most likely is a spelling mistake, (b) an image of a ‘mushroom’ also tagged as ‘wedding’ and (c) an image of a ‘goat’ tagged with ‘snow’ ......................... 57

Figure 3.2. Two examples of subgraphs built using the proposed data-driven approach and corresponding sub-graphs obtained using WordNet. 63

Figure 3.3. Building the Similarity Graph for threshold $\tau=0.25$ ............... 73
Figure 3.4. One iteration of the proposed approach. Eq. (3.3) is utilized to calculate the objective function for neighbors of a tree T. ....... 77

Figure 3.5. The variation of the objective function (3.3) with the number of iterations for a sample run on Flickr tag corpus. ............... 80

Figure 3.6. Average Tag Prediction Accuracies marginalized over $N_{Tags}$ for various methods for the tag prediction task on Flickr corpus. .... 85

Figure 3.7. Average Tag Prediction Accuracies marginalized over $N_{Tags}$ for various methods for the tag prediction task on Stock images corpus. 87

Figure 3.8. Tag Recall obtained with respect to number of classifications performed for Flickr corpus. $\theta$ is chosen to be 0.5. ............... 95

Figure 3.9. Tag Recall obtained with respect to number of classifications performed for Flickr corpus. $\theta$ is chosen to be 0.33. ............... 95

Figure 3.10. Tag Recall obtained with respect to number of classifications performed for Stock images corpus. $\theta$ is chosen to be 0.5. ........ 96

Figure 3.11. Robustness analysis using Overall Tag Prediction Score (in %) obtained by proposed approach in Section 3.4.3. ............... 97

Figure 3.12. Example images where LS-SA method gave 100% tag prediction accuracy when the first two tags were seen and the next three were unseen and predicted. ............... 100

Figure 3.13. Example images where LS-SA method gave 0% tag prediction accuracy when the first two tags were seen and the next three were unseen and predicted. ............... 100

Figure 3.14. Variation of number of iterations with $\tau$ ............... 101

Figure 3.15. Variation of number of candidate neighbors with $\tau$ ............... 101

Figure 3.16. Variation of time (hours) taken by local search to converge ...... 102

Figure 3.17. Variation of the Average Tag Prediction Accuracy with $\tau$ ....... 102

Figure 4.1. Approach Overview ........................................ 112

Figure 4.2. Social network graphs for shares show that users collaborate and tend to form communities by accessing common files. ......... 116
Figure 4.3. Splitting dataset into various training and testing periods ........................................... 118
Figure 4.4. AF for metadata models with the fixed testing and varying training periods ............................................................ 122
Figure 4.5. AR@75P for metadata models with the fixed testing and varying training periods ............................................................ 122
Figure 4.6. AF for metadata models with the fixed training and varying test periods ............................................................ 123
Figure 4.7. AR@75P for metadata models with the fixed training and varying test periods ............................................................ 123
Figure 4.8. Distribution of the least number of weeks for each user of share D that gives highest F-score. ............................................................. 124
Figure 4.9. AFMS technique first creates FTM in an offline manner. The classification step applies only those models that are triggered by active features. ............................................................. 130
Figure 4.10. Speed-up vs. $\tau$ for metadata models ............................................................. 132
Figure 4.11. Speed-up vs. $\tau$ for CF models ............................................................. 132
Figure 4.12. Precision vs. $\tau$ for metadata models ............................................................. 137
Figure 4.13. Recall vs. $\tau$ for metadata models ............................................................. 137
Figure 4.14. F-score vs. $\tau$ for metadata models ............................................................. 137
Figure 4.15. Recall@75%Precision vs. $\tau$ for metadata models ............................................................. 137
Figure 4.16. Precision vs. $\tau$ for CF models ............................................................. 138
Figure 4.17. Recall vs. $\tau$ for CF models ............................................................. 138
Figure 4.18. F-score vs. $\tau$ for CF models ............................................................. 138
Figure 4.19. Recall@75%Precision vs. $\tau$ for CF models ............................................................. 138
LIST OF TABLES

Table 2.1. Proximity and diversity scores for a few valid keywords of category Baby, calculated as detailed in Section 2.4.1 26

Table 2.2. Summary of average classification accuracy obtained using Adapt AAO approach 47

Table 3.1. Effect of initializing the proposed local search based approach using WordNet. 30 tags from stock image corpus (described in Section IV) are used 78

Table 3.2. Overall Tag Prediction Accuracy (%) for various methods for tag prediction task on Flickr and Stock images corpora 84

Table 3.3. Average Tag Prediction Accuracies (in %) obtained using Random method on Flickr corpus 85

Table 3.4. Average Tag Prediction Accuracies (in %) obtained using WordNet method on Flickr corpus 85

Table 3.5. Average Tag Prediction Accuracies (in %) obtained using Google Similarity Distance method on Flickr corpus 85

Table 3.6. Average Tag Prediction Accuracies (in %) obtained using LS-WAH method on Flickr corpus 86

Table 3.7. Average Tag Prediction Accuracies (in %) obtained using LS-SA method on Flickr corpus 86

Table 3.8. Average Tag Prediction Accuracies (in %) obtained using Symmetric sum method on Flickr corpus 86

Table 3.9. Average Tag Prediction Accuracies (in %) obtained using Random method on Stock images corpus 87

Table 3.10. Average Tag Prediction Accuracies (in %) obtained using WordNet method on Stock images corpus 88

Table 3.11. Average Tag Prediction Accuracies (in %) obtained using Google Similarity Distance method on Stock images corpus 88

Table 3.12. Average Tag Prediction Accuracies (in %) obtained using LS-WAH method on Stock images corpus 88
Table 3.13.  Average Tag Prediction Accuracies (in %) obtained using LS-SA method on Stock images corpus. ........................................... 89

Table 3.14.  Average Tag Prediction Accuracies (in %) obtained using the Symmetric sum approach on Stock images corpus. ...................... 89

Table 4.1.  Statistics of shares used for evaluation ................................................. 115

Table 4.2.  Performance summary for metadata models. Performance numbers are averaged over 5 iterations with random initialization of the linear SVM model training. Numbers are listed along with the standard deviations. ................................................................. 120

Table 4.3.  Performance summary for CF models. Performance numbers are averaged over 5 iterations with random initialization of the linear SVM model training. Numbers are listed along with the standard deviations. ................................................................. 120

Table 4.4.  Statistics for metadata models, averaged across 30 evaluation users and different training periods (Section 4.3.2), showing how AFMS is effective. ................................................................. 128
ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help and support of my mentors, family, friends and colleagues. I would like to take this opportunity to thank them.

First, I would like to thank Prof. Sujit Dey for his patience to train me to think and write with the clarity that academic research demands. His training has helped me not only for my research related goals, but also in other aspects of my life. I would also like to thank my committee members for their feedback on the work and for useful discussions.

My parents have been the cornerstone of my learning and education through their unconditional and selfless support. I would like to thank them for the patience and love which has never faltered even though it must have been difficult to always understand my work, actions and reasonings. This achievement is as much mine as it is theirs.

I would also like to thank Sandeep Bhatkar, Michael Hart, Vijay Mahadevan, Nikhil Rasiwasia, Gaurav Aggarwal for mentoring me during internships. My internships have facilitated a significant portion of my learning and the contribution of my mentors has been crucial.

Graduate school has been a fun and learning experience for me. Looking back, I often laugh at how my perceptions have changed with time, through discussions with friends, family, labmates, and as a result of changes in my own thinking process. I would like to thank my friends Kritika, Nitesh, Bharathan, Rishi, Manish, Yao, Dipen, Karan, Minu, Ravee, Sumit, Mayank and many others for helping out and for making the journey enjoyable. It would have been difficult if I did not have them to pour my heart out. My labmates especially Yao, Jason, Ranjini, Ali, who I have spent a significant time with, have also been helpful through innumerable discussions over the past years.

Chapter 2, in part, contains material as it appears in the Proceedings of the 13th
International Conference on Web Intelligence (WI’13). “Fully Automated Learning for Application-Specific Web Video Classification”. Chetan Verma, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in part, contains material as it appears in IEEE Access. “Methods to Obtain Training Videos for Fully Automated Application-Specific Classification”. Chetan Verma, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, contains material as it appears in the Proceedings of the companion publication of the 23rd International Conference on World Wide Web (WWW ’14). “Construction of Tag Ontological Graphs by Locally Minimizing Weighted Average Hops”. Chetan Verma, Vijay Mahadevan, Nikhil Rasiwasia, Gaurav Aggarwal, Ravi Kant, Alejandro Jaimes, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, contains material as it appears in Elsevier Expert Systems with Applications. “Construction and Evaluation of Ontological Tag Trees”. Chetan Verma, Vijay Mahadevan, Nikhil Rasiwasia, Gaurav Aggarwal, Ravi Kant, Alejandro Jaimes, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, contains material as it appears in the Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS ’15). “Access Prediction for Knowledge Workers in Enterprise Data Repositories”. Chetan Verma, Sandeep Bhatkar, Michael Hart, Aleatha Parker-Wood, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, contains material as it has been accepted for publication for the IEEE Access. “Improving Scalability of Personalized Recommendation Systems for Enterprise Knowledge Workers”. Chetan Verma, Sandeep Bhatkar, Michael Hart, Aleatha Parker-Wood, Sujit Dey. The dissertation author was the primary investigator
and author of this paper.
VITA

2008 Bachelor of Technology, Electrical Engineering, Indian Institute of Technology, Madras

2011 Master of Science, Electrical Engineering (Computer Engineering), University of California, San Diego

2010–2015 Graduate Student Researcher, Mobile Systems Design Lab, Department of Electrical and Computer Engineering, University of California, San Diego

2011–2015 Doctor of Philosophy, Electrical Engineering (Computer Engineering), University of California, San Diego

PUBLICATIONS

Journals


Conferences


ABSTRACT OF THE DISSERTATION

Enabling Automated and Efficient Personalization Systems

by

Chetan Kumar Verma

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2015

Professor Sujit Dey, Chair

Information explosion and the increasing use of Internet have fueled the growing popularity of personalization systems. Such systems can understand user interests to customize the information served to them, thereby addressing information overload. At the same time, personalization systems can also enable service and content providers to serve targeted advertisements and recommendations to users. In order to operate on the massive scales of the number of users and the amount of data available, personalization systems have a crucial requirement for automation and efficiency. In this dissertation, we identify key challenges faced by personalization systems and provide solutions to
address them such that the above requirements can be achieved.

We first note that content classification is an important component of personalization systems. Approaches to train classification models typically depend on manual collection and labeling of training data, which makes personalization non-scalable. To address this, we develop a completely automated framework that can provide labeled training data for arbitrary set of categories. Experiments using online videos demonstrate the feasibility and effectiveness of our approach.

The second key challenge we address is the sparsity in annotations of popular online content such as Flickr images. Sparse or missing tags hamper the ability of personalization systems to recommend content or to infer the interests of users that access them. Towards this, we show how ontological tag trees can be constructed from corpus based statistics and semantic relationships between tags, to alleviate tag sparsity in a space efficient manner. Through evaluations, we demonstrate the effectiveness and efficiency of ontological tag trees as compared to existing methods.

Lastly, we focus on alleviating information overload in enterprise repositories. We design a file metadata based recommendation system that captures per user access patterns and user collaboration to recommend new files. In order to address scalability concerns of per user modeling, we propose optimizations that significantly reduce the time to serve recommendations to users. Experiments over actual enterprise data show that more than two orders of speed up is obtained as a result of the proposed methods.
Chapter 1

Introduction

The past two decades have witnessed a tremendous growth in the amount of data available to users [31, 59, 40, 91]. Often called information explosion [120], this trend is reflected across multiple ecosystems including content sharing systems, social networks and enterprise systems, as well as across different types of the data including videos, images, documents, audio files etc. For example, the total number of images uploaded on Flickr [35] has risen from 6 billion in 2011 to 10 billion in 2015 [37, 38]. Facebook [33] stores over 260 billion images and grows at an explosive rate of one billion new photos per week [9]. In the enterprise ecosystem, unstructured data has been observed to grow annually at a rate of 40-50% [40]. While the growing size of the available data provides valuable resource to users, its sheer size limits its usability and hence its value. Users face the daunting challenge of obtaining useful or interesting information from the deluge of available data, an issue that is commonly referred to as information overload [31].

1.1 Personalization Systems

In order to assist users navigate the overwhelming amounts of data, there has been a steady increase in the number and popularity of personalization systems that customize the information presented to users [32]. These include systems that can provide content recommendations, targeted advertisements, personalized news feeds,
suggested connections, preference based query ranking and other forms of personalization. For example, Amazon recommends products to users based on their previously made purchases or viewed products [3], Netflix recommends movies accordingly to users’ movie watching patterns [90], LinkedIn can suggest professional connections based on users’ professional social networks and their activities [74], and Google serves personalized advertisements to users based on their activities on certain websites and their inferred profiles [43, 18, 44]. The above are only a few examples of widely used personalization systems that have become household names owing to their popularity. Such systems help users combat information overload by tailoring their experience, and at the same time, enable the content providers and advertisers to understand user interests for better targeting of content, connections, advertisements etc. Indeed the rapid adoption of personalization systems provides compelling evidence to their growing significance [17].

The growing popularity of personalization systems, combined with the enormous quantities of data that they operate on and the large number of users that access them, call for design considerations to make such systems scalable. The first scalability consideration is to reduce the manual effort required to deploy and operate personalization systems. Manual involvement can make the process time consuming, less scalable and subjective. In addition, a high efficiency is desired such that the systems can operate on large volumes of data and users while having low time and space requirements. In the next section, we discuss how our research enables personalization systems meet the above considerations.

1.2 Contributions and Overview

In this dissertation, we focus on several aspects and variations of personalization systems, and demonstrate the proposed techniques through experiments over different
types of the data. The first key contribution of the dissertation addresses automation considerations of personalization systems by focusing on content classification. The second key contribution addresses sparsity of annotations associated with online content such as Flickr images, thus enabling personalization systems leverage the content more effectively and efficiently. The third key contribution addresses efficiency considerations for recommendation systems.

We first provide an overview of our contributions, followed by individual chapters that give detailed treatment to each of the contributions, including the related literature and how our work relates to and differentiates from the existing literature. Our contributions are summarized below.

1.2.1 Automatically obtaining training data to enable content classification

In order to customize user experiences based on their inferred interests, personalization systems often build user interest profiles in terms of a set of categories and sub-categories. For example Google Adsense [44] collects the sites that have been visited by a user and builds an interest profile from them, as a set of categories and sub-categories. If a user visits a football site several times, Google Adsense may associate the category Sport, or more specifically the sub-category Football with the user. For such systems, the ability to classify content accessed by users into categories (and sub-categories) of interest is crucial. Training classification models to do so requires data that is labeled as per the same set of categories. Since different personalization applications may be interested in different sets of categories, such labeled training data for arbitrary categories is not readily available and manual labeling is typically employed to obtain the same. In this work, we develop a framework that can automate the training data collection and labeling process. By leveraging publicly available resources such as Wikipedia [130],
our framework selects keywords that can help retrieve training videos. Towards this, we define the desired objectives for selection of keywords, and formulate novel optimization methods (LCPD and AAO) to obtain the training data. We discuss and evaluate our framework through text based classification of online videos (such as from YouTube [143]); however our approach is applicable to other forms of data as well. Through experiments over real world videos, our framework shows high performance as compared to other methods despite using no manual effort.

1.2.2 Addressing sparsity in annotations of online content

Most web based content sharing systems such as Flickr [35] allow social tagging that helps assign tags or annotations to content. Since users aren’t typically incentivized to annotate content accurately or comprehensively, the content suffers from insufficient and often missing tags, usually referred to as tag sparsity [118, 119]. As a result, personalization systems face difficulty in recommending such content or in inferring user interests based on access to the content. In this work, we show how ontological tag trees can be used to address tag sparsity in an effective and efficient manner. We formulate the construction of ontological tag trees as an optimization problem over the set of tags in a corpus, and leverage tag statistics and semantic relationships (from WordNet [83]) for the construction. The optimization is solved using a novel local search based approach, and experiments on publicly and professionally annotated image corpora show that our approach can achieve as high performance as existing approaches, while offering significant savings in the space requirement. Furthermore, we show how the constructed ontological tag trees can be used to speed up the classification of content. Our work also shows that ontological tag trees are significantly robust to tag noise and variations in tag distributions.
1.2.3 Improving time efficiency of personalized recommendations in enterprises

In this work, we focus on users accessing data from enterprises repositories. Since enterprise users are overwhelmed by the massive growth of the data [57], we develop a file recommendation system that can efficiently connect the users with useful new content. In order to do so, the system models file access patterns for individual users and uses the model to predict if a new file would be accessed by the user. Furthermore, we show how the collaboration among users can be leveraged to improve the system performance. While the evaluations based on actual enterprise data show that our system can recommend content to users with high predictive correctness, the per user modeling raises scalability issues while recommending files to a large number of users accessing a busy repository. In order to address this shortcoming, we develop a novel optimization that can achieve the same predictive correctness of our models, while significantly speeding up the file recommendations. We further provide a novel method that allows trading off marginal amounts of model correctness for several orders of speed up in the system. Experiments on enterprise data show that this can make the system faster by as much as 169 times on average.

For each of the above contributions, we have focused on demonstrating our methods by focusing on one or two data types. However the proposed approaches could potentially be applied to other types of data as well.

This rest of this dissertation is organized as follows. Chapter 2 describes methods that enable automated content classification by using keywords to obtain training data. Chapter 3 presents approaches to construct and use ontological tag trees and Chapter 4 presents the design of the file recommendation system and how its time efficiency can be improved. Chapter 5 concludes the dissertation and provides suggestions for future research.
Chapter 2

Methods to Obtain Training Data for Fully Automated Application-Specific Classification

Personalization approaches seek to estimate user preferences in order to recommend content or social network connections, or to serve personalized advertisements to users. Such approaches are being increasingly adopted by organizations to build customized personalization applications. Leveraging the growing popularity of web videos for such approaches necessitates the ability to classify web videos into application-specific categories since different applications are interested in different aspects of the user preferences. A key requirement of supervised classification models to address this is the availability of training videos labeled to the arbitrary application-specific categories. In order to address this requirement, we propose a completely automated framework to obtain training web videos for arbitrary categories, which does not rely on any manual labeling of videos. This is achieved by utilizing keywords to retrieve training videos, thereby simplifying the problem of obtaining training videos to the problem of selecting keywords to retrieve them. We show that there are two opposing objectives (proximity and diversity) that need to be considered while developing such keyword selection techniques. We propose two efficient approaches (LCPD and AAO) and study
the trade-offs between them, with respect to performance and the human input required to tune parameters of the approach. Through experiments over several sets of categories, we demonstrate the feasibility of the automated framework to select training videos for application-specific categorization. We also show that the proposed approaches lead to a substantial improvement in the performance of classification models, as compared to other automated methods.

2.1 Introduction

Over the past few years, there has been a steady rise in the number and popularity of personalization applications available on the Internet. These include applications based on personalized advertisements, content recommendation systems, social network connection suggestions, and several others, that attempt to understand the preferences of users. Personalization applications have been traditionally based on learning user preferences through queried keywords and viewed articles [85]. The last few years have also witnessed a tremendous increase in viewing and sharing of web videos (such as on YouTube[143]), with significant increases in unique viewers, total streams viewed, number of streams per viewer, and the time per viewer [91]. Given their unique characteristics, web videos offer a tremendous potential for understanding user preferences.

User preferences can be inferred based on the types/categories of web videos seen. Such videos are generally organized at video sharing websites on the basis of labels that the video uploaders choose from among a set of common categories that are used by such websites. Examples of such common categories include Comedy, Music, People, Entertainment, Pets, Science, etc. On the other hand, the categories of interest to personalization applications may be arbitrary, and quite different from the above common categories. Consider a department store (such as Sears or Walmart) that might want to offer promotional coupons to buyers. Knowing whether a person (a buyer) has interest in
product specific categories like fitness equipment, clothing items, or baby products would be of high interest to the department store, as compared to knowing whether he/she is interested in the common categories mentioned above. A movie recommendation system would like to learn if a viewer prefers action, horror, or comedy movies. Categorizing viewed videos and understanding user preferences in terms of the common categories used by video sharing websites might not be useful for different personalization applications. In addition to the above observation, it should be noted that different personalization applications are interested in understanding user preferences with respect to very different sets of categories, as shown by the above examples. It is clearly not sufficient to use a common set of categories for every personalization application, as the categories of interest for one application might be irrelevant and useless for another.

This calls for techniques to classify viewed web videos, and hence estimate user preferences, in terms of any arbitrary set of categories appropriate for a given personalization application. Various modes of information (such as audio, visual, textual and social network) can be employed to assist in the classification of web videos. Classifiers employed for this task have the inherent requirement of training videos labeled to the set of categories as desired by the personalization application. Since the set of categories suitable for a personalization application might be very different from the common categories used by video sharing websites, training videos labeled according to the required set of categories are often unavailable. Our work addresses this requirement of obtaining training videos labeled as per the required set of categories, which are not necessarily the categories commonly associated with web videos. We propose a fully automated framework to obtain training videos with properties that can lead to high performance of trained classification models. To achieve the above, the proposed framework neither relies on labels associated with online videos, nor requires any manual labeling of videos. Instead, we develop approaches to select keywords based on their
suitability to retrieve high quality training videos for a specified set of categories. Such a methodology requires the consideration of two opposing objective, namely proximity and diversity. In order to select the keywords, we first present a tunable approach (LCPD) based on the Linear Combination of Proximity and Diversity. While such an approach can give good classification performance, it requires the tuning of a parameter in order to obtain optimal performance, which may require manual effort. We thus also propose an approach using Annealing based Alternating Optimization (AAO) where we balance between the two objectives such that the final solution is a trade-off between the two. Complexity, convergence and correctness of the proposed algorithms are presented, along with experiments over several sets of categories.

2.1.1 Related work

We discuss below the related work on video classification. Since the keyword selection based approach requires optimizing for two objective functions, we also provide a discussion on the related work on multi-objective optimization.

Video classification:

A significant amount of work has been done to address the problem of video classification. Such work can be looked at on the basis of two dimensions - modalities used for classification, and approaches to obtain labeled training videos. While the focus of our work is on obtaining training videos, we first briefly describe video classification approaches in terms of modalities used, including our approach, and then discuss approaches to obtain training data, contrasting our approach from others.

A characteristic property of web videos is that they have rich information in several modes – audio, visual, textual, and social network being the most common ones. Methods such as [114, 128, 145, 98, 136] present multi-modal techniques for
classification of web videos. Others such as [106, 21] classify videos using only the audio-visual information in the videos, while [23, 133] approach classification of web videos by treating them as text documents. A detailed survey on video classification is provided in [12]. In our work, we classify web videos on the basis of the contextual information surrounding them, such as the title, keywords, and description. This is because text-based classification approaches are computationally much less expensive than multimedia features-based classification approaches, and as shown in existing literature as well as in Section 2.5, offer good classification performance.

With respect to the approaches to obtain labeled training videos, [23, 133, 98] obtain training videos that are labeled according to categories used by YouTube [143]. Hence, such approaches cannot be used for classification of web videos to arbitrary set of categories, which is the focus of this chapter. Approaches such as [106, 136, 114] utilize training videos that are labeled manually. Recently, techniques have been developed [128, 145] which expand the set of training videos starting from a set of manually labeled videos. With the help of social network structure of the video sharing website, co-watched videos, or text-based classifiers, [128, 145] increase the number of training videos in a semi-supervised fashion. However, manual labeling requires human experts to go through at least a part of the video, and come up with a label. The labeling process is prone to human errors and inconsistencies, and more critically, is not scalable to large sizes of training data, especially given the enormous scale of web videos [91]. Contrary to these approaches, we propose a framework that does not require any manual effort to obtain training videos, even for any arbitrary set of categories desired by a personalization application.

**Multi-objective optimization:**

In general, for multi-objective optimization problems, there does not exist a
single solution that optimizes all objectives. [79] surveys the different approaches that are adopted to solve multi-objective optimization problems. Certain approaches require apriori knowledge of the preferences for the different objective functions. In order to utilize these preferences, a global objective function can be defined based on the individual objective functions and by using the preferences as numerical weights, as is shown in [7, 13, 65]. For example, a common approach to multi-objective optimization is to combine the individual objective functions through a weighted sum method as shown in [115].

Other works embed the notion of preferences into their methodology, for example by ranking the objective functions in the order of their relative importance [142, 51] or by modifying the bounds of the individual optimization problems [56, 48]. Such works however require the apriori knowledge of the preferences of the multiple objectives, as obtained by a human decision maker. In the context of our problem of selecting keywords to obtain training videos, such relative preferences of different objective functions is unavailable and hence these approaches cannot be applied.

A different line of works approach multi-objective optimization problems by first obtaining a set or a representation of pareto optimal solutions for the problem, and then employing human judgment to decide the best solution. Pareto optimal solutions are alternative solutions for the multi-objective optimization problem that are optimal in the wider sense that no other solutions in the search space are superior to them when all the objectives are considered [117]. Refer to [79] for the mathematical definition of pareto optimality. Thus, works such as [81, 80, 82] focus on finding the set of pareto optimal solutions or their representation to serve as a palette of solutions for the human decision-maker. Varying the weights given to different objective functions is a common approach to determine the pareto optimal solutions or their subset [79]. Such approaches however can be extremely inefficient as they require solving the weighted formulations
several times [79]. [49] works with a set of solutions searching for Pareto optimal solutions in parallel. In order to find the best candidate in the neighborhood of each current solution, the goodness is taken as the weighted average of the objective functions. The weights are dynamically updated so that unexplored regions of solution space get more preference. Certain works such as [116, 112, 8] approach the search for pareto solutions using simulated annealing framework based on an energy function for states. These energy functions are defined based on the number of solutions that dominate or are dominated by a particular solution. See [8] for a definition of domination. Such approaches require the computation of the objective functions for all or a subset of the possible solutions. While these approaches may work for certain domains [116, 112, 8], they would be infeasible to apply to domains when the discrete solutions are subsets of a larger set and thus the number of solutions is extremely large. As shown in Section 2.4.2, for our problem of selecting keywords, the total number of subsets can easily be of the order of $10^{30}$ and hence such approaches would be infeasible. In addition, in order to use the solution in a practical setting, such approaches would still require a decision maker to choose the best solution based on his/her human judgment. Compared to these, we propose two approaches LCPD and AAO to select keywords to obtain training videos. In the former, we heuristically combine two objectives linearly and select keywords one by one without computing the objective function for all subsets. In the latter, we balance the two objective functions in our formulation (Section 2.4.2) by alternating between algorithms that optimize for each objective individually, in a simulated annealing framework. As a result of this, for AAO it is not required to provide preferences for the individual objective functions and the approach yields a solution that is convenient trade-off between the two objective functions. We empirically demonstrate the effectiveness of the two proposed approaches for selecting keywords to obtain the training videos for a category. Furthermore for AAO, in order to reduce the time taken for the annealing
based approach to converge, we propose an adaptive technique to update the annealing constraint parameter, leading to substantial reduction in the convergence time.

This work is based on [122] which provides an approach to automatically obtain training videos. We refer to the techniques proposed in [122] as LCPD, and in addition we propose an annealing based alternating optimization approach (AAO) that does not depend on parameters that may require tuning based on human input as in [122]. Further, we propose adaptive variation of AAO to make it more efficient, and provide performance comparisons with [122].

For multi-class, single-label classification of web videos, we first discuss the desired properties of training videos that can lead to high performance of trained classification models. This is done in Section 2.2. Section 2.3 provides an overview of our framework of identifying keywords to retrieve training videos with the desired properties. It also provides a discussion on the two objective functions to select keywords. Section 2.4 discusses the two approaches - LCPD and AAO. Section 2.5 details the experimental set-up and presents performance results. Section 4.7 concludes.

### 2.2 Desired properties of training data

For a given classification model, a good training data would be one that has no mislabeled instances, and has high Intra-Category Diversity for each category. We discuss both factors in this section. The goodness of training data is reflected in terms of its performance on a large test set.

In the domain of video classification, a mislabeled instance refers to a video that has the label of category $i$ as per the training data, but in actual, belong to a different category $j(\neq i)$ as per an oracle. For instance, consider the set of categories \{Baby, Clothing, Fitness, Food\} that a retailer may be interested in, to enable personalized promotions of the above product categories. If certain approach for obtaining training
Figure 2.1. (a) Sample training videos for categories: \{Baby, Clothing\}. Circled video is wrongly placed in category Baby, and is hence a mislabeled video. (b) Variety of video topics belonging to category Baby videos includes a video on Shawls (as shown in Fig. 2.1(a)) to the set of training videos for Baby, the video would be a mislabeled video since its true label should be Clothing, but it has the label of Baby in training data. A true label of a video is defined as the label that an oracle would assign to the video. \[14, 147\] discuss techniques to identify (and eliminate) mislabeled instances from training data, for classification tasks. The performance of classification models is shown to have increased considerably after identifying (or eliminating) mislabeled videos, thus supporting that less mislabeled instances is desired in training data. Note that the above works address the cleaning of an existing dataset whereas the problem addressed in this chapter is that of forming a
dataset by obtaining training videos with desired properties for certain given categories. In addition, in works such as [14] and [147], the mislabeled instances are identified as outliers in the given data, when the data itself is representative of the class in concern and captures its constituent topics. While forming a training data from real world videos retrieved using keywords, this condition is not guaranteed to be satisfied and the above techniques may not be applied. These techniques can be considered orthogonal to our work, and may further improve the training data after it is formed using the proposed framework.

By Intra-Category Diversity of training videos $T(i)$ of category $i$, we refer to the extent to which $T(i)$ encompasses the essence of category $i$. Let us denote Intra-Category Diversity of $T(i)$ as $div(T(i))$. In order to first intuitively motivate why high $div(T(i))$ is desired, consider the same set of categories \{Baby, Clothing, Fitness, Food\}. Fig. 2.1(b) shows some of the various topics of videos that one would associate with the category Baby. A set of training videos $T_1$, having videos on Funny kids only has less Intra-Category Diversity than a set $T_2$ of same cardinality as $T_1$ but having videos on Funny kids, Newborn care, Babysitting, and Stroller reviews. A classifier trained over $T_2$ is expected to have higher likelihood of categorizing correctly a test video $v$ belonging to category Baby as compared to a classifier trained over $T_1$. For instance, if a user watches a video related to review of popular strollers for infants, the model trained on $T_2$ will find it more similar to the training videos on Baby than the model trained on $T_1$ will, and hence will have higher likelihood of categorizing it correctly.

As discussed later in Section 2.3, one of the shortcomings of approaches that obtain training videos without manual labeling is low Intra-Category Diversity. In such scenario, the training data of a category is skewed towards certain dominant themes within the category, and encompasses only limited topics of videos within the category. Improved techniques are hence required to obtain training videos with high Intra-Category
Diversity. While it is understandable what $\text{div}(T(i))$ means, calculation of $\text{div}(T(i))$ requires knowledge of a) different topics within category $i$, and b) the extent to which these topics are covered by the training videos of category $i$. For an arbitrary set of categories, it is extremely difficult to obtain (a) and (b) above without the help of an oracle. Thus in our research, we propose the use of different techniques to obtain the degree of variation in $T(i)$ to measure $\text{div}(T(i))$. These techniques are discussed in Section 2.3, and their details are provided with the respective approaches (Sections 2.4.1 and 2.4.2).

The notion of diversity has been used for a variety of problems in related domains. Diversity between classification models has been discussed in works on classification model ensembles [67]. Works on summarization have used the notion of diversity between documents [69, 148], images [63] and videos [109]. In the context of active learning [127] and semi-supervised learning [84], diversity has been used to efficiently determine training instances to label manually. To the best of our knowledge, the use of diversity metrics in the formulation to automatically obtain training videos for classification is completely novel.

2.3 Overview of the proposed framework to obtain training videos

In this section, we provide an overview of our proposed framework and approaches for identifying and selecting keywords to obtain training videos with the two desired properties described in Section 2.2.

As discussed in Section 2.1, obtaining training videos for arbitrary set of categories through manual effort is not scalable to large sizes of training data. Training videos could alternatively be obtained in an automated manner using a video search engine. This approach has also been briefly mentioned in [128], to obtain weakly labeled videos. Let
$RV(K)$ represent the set of retrieved videos obtained by querying keyword $K$ in a video search engine, such as YouTube [143] or Vimeo [125]. The training videos of category $i$, i.e., $T(i)$ can then be obtained simply as $RV(C_i)$, where $C_i$ is the name of category $i$. Though simple, this technique has few shortcomings. It focuses only on occurrence of the name of category and not on its semantic meaning. For example $RV(‘Baby’)$$ mainly retrieves videos on funny kids, and music videos or other popular videos containing the word ‘Baby’ in their title or tags. As a result, there are several mislabeled videos among training videos obtained by this approach. In Section 2.5, we show how this leads to poor performance as more videos are retrieved just by the category name. At the same time, this approach does not cover the many of the semantic topics that we associate with the category of concern. For the category Baby, these include topics such as (in addition to funny kids,) Strollers and Bassinets reviews, Babysitting tutorials, Newborn care, Pregnancy, and several others (Fig. 2.1(b)). The Intra-Category Diversity by such an approach is hence quite low, leading to poor classifier performance. Through the proposed framework, we attempt to address the above shortcomings. We use the training videos obtained by querying name of category, i.e., $T(i) = RV(C_i)$, to train baseline classifiers to compare with our proposed approach.

2.3.1 Overview of proposed framework

In the proposed framework, we first collect several keywords that are related to the name of category ($C_i$). These comprise the Candidate Keywords. The Candidate Keywords (called candidates for brevity) can be obtained on the basis of correlation or co-occurrence with the name of the category from publicly available text documents (such as Wikipedia). Thesauri [87] also provide a good source for semantically (i.e., in terms of meaning) similar keywords, and can be used to obtain candidates. In addition, candidates can be obtained as related concepts or keywords from sources such as [100].
The candidates can be queried in a video search engine, and their retrieved videos can be collected to obtain $T(i)$. However, some candidates would be more useful than others, and some might be outright harmful, if used to retrieve training videos for category $i$. We discuss this in the next section. On the basis of a proposed keyword selection algorithm, we select a subset of keywords from a set of candidates for category $i$. The Selected Retriever Keywords (or SRKs) thus selected are used to retrieve training videos. If $\{K_{i,1}, K_{i,2}, ..., K_{i,L}\}$ are the SRKs for category $i$, then the training data for category $i$, i.e., $T(i)$, can be obtained as:

$$T(i) = \bigcup_{j=1}^{L} RV(K_{i,j}) \cup RV(C_{i}). \quad (2.1)$$

If $T(i)$ were obtained as per (2.1) by selecting any arbitrary candidate keywords of category $i$ as SRKs, then $T(i)$ may not necessarily have the desired properties discussed in Section 2.2, namely low mislabeled videos, and high Intra-Category Diversity. In the next section, we discuss how we can determine the suitability of candidates to retrieve training videos with the desired properties discussed in Section 2.2.

### 2.3.2 Selection Procedure for Selected Retriever Keywords (SRKs)

Before we discuss suitability of candidates, and the selection procedure for SRKs, we provide the following result to help in our discussions. Consider the set of categories $\{i\}_{i=1}^{N_{Cat}}$, where each category $i$ is a multivariate normal distribution with mean $\mu_i$. Assume that the set of videos across all categories (referred to as data) is whitened, i.e., has uncorrelated dimensions of variance unity. Then $\Sigma_i = I$, i.e., the covariance matrices for all categories reduce to an identity matrix. Under assumptions of equiprobable categories,
Figure 2.2. Selection criteria for Selected Retriever Keywords

Figure 2.3. For a set of 20 valid keywords for the category ‘Baby’, (a) shows the 5 most proximate keywords and (b) shows the 5 most diverse keywords. Diversity is measured as outlined in Section 2.4.2. As can be seen, the set of keywords optimizing either objective function are very different from each other. Note that in order to plot the keywords on a graph, the first two dimensions obtained from principal component analysis (PCA) [61] are used.
we derive that a video (represented as $v$) belongs to category $\hat{i}$ iff

$$\hat{i} = \arg\max_i e^{-\frac{(v-\mu_i)^T \Sigma_i (v-\mu_i)}{2}}, \text{ i.e., iff}$$

$$\hat{i} = \arg\min_i e^{\|v-\mu_i\|^2}, \text{ i.e., iff}$$

$$\hat{i} = \arg\min_i \|v-\mu_i\|.$$  \hspace{1cm} (2.2)

Here $\mu_i$ is the mean of category $i$, as determined by an oracle. $\|v-\mu_i\|$ is the Euclidean distance between $v$ and $\mu_i$. Note that for this chapter, $v$ is represented as a bag of words [104] vector based on the contextual information surrounding $v$. On the basis of the above assumptions, and using (2.2), a keyword $K$ retrieves more videos having true label as category $i$ than videos having true label as category $j$ ($\neq i$) if

$$\sum_{v: v \in \text{RV}(K)} \mathcal{I}(\arg\min_l \|v-\mu_l\| = i) > \sum_{v: v \in \text{RV}(K)} \mathcal{I}(\arg\min_l \|v-\mu_l\| = j) \forall j \neq i. \hspace{1cm} (2.3)$$

Here, $\mathcal{I}(.)$ is an indicator function that is 1 if its argument is true, and 0 if its argument is false. $\mu_i$ is the true mean of category $i$. In (2.3), the closest category for each video in $\text{RV}(K)$ is obtained, in terms of Euclidean distance. Checking (2.3) for a keyword $K$ thus has complexity $O(\text{|RV}(K)| \cdot N_{\text{Cat}})$, where $N_{\text{Cat}}$ is the number of categories, and $\text{|RV}(K)|$ is the cardinality of the set $\text{RV}(K)$. In order to reduce the above high complexity, (2.3) can be approximated by obtaining the closest category of centroid of the set $\text{RV}(K)$. This reduces the complexity to $O(N_{\text{Cat}})$. For category $i$, we define **Valid Candidate Keywords** (called valid keywords for brevity) as those Candidate Keywords that retrieve more videos having true label of category $i$ than of any other category. Then for a candidate $K$ of category $i$, $K$ is a valid keyword if

$$\|\mu_K - \mu_i\| < \|\mu_K - \mu_j\| \forall j \neq i. \hspace{1cm} (2.4)$$
Here $\mu_K = \frac{\sum_{v \in RV(K)} v}{|RV(K)|}$. The true mean $\mu_i$ for category $i$ can be approximated as centroid of $RV(C_i)$, i.e., of the set of videos retrieved by name of category $i$. Equation (2.4) is called the **Validity Filter**. Only valid keywords should be considered for being selected as SRK to ensure more number of training videos are added in $T(i)$ that have true label of category $i$ than videos that are mislabeled as category $i$.

Note that we have assumed that the data generating the above distributions is whitened. If the given data is not whitened, its dimensions can be rotated into space of principal components, and each dimension divided by square root of variance in that dimension, in order to whiten the data. Also, if the assumption that categories are equiprobable multivariate normal distributions does not hold, then the exact distributions could be used to derive the Validity Filter as shown in (2.4). As a result, the complexity of checking the validity of a keyword may be more than $O(N_{Cat})$, however our approach of using a Validity Filter, and the following keyword selection procedure would still be valid. We continue our discussion with the above assumption.

Let us consider the scenario shown in Fig. 2.2. Keywords $K_1$, $K_2$, $K_3$ are candidates for category 1 ($C_1$). $K_3$, as can be seen, is closer in terms of Euclidean distance, to category 2 ($C_2$) than to $C_1$, and hence fails the Validity Filter (2.4). For $C_1$ and $C_2$ as Baby and Clothing respectively, example keywords (from actual data) for $K_1$, $K_2$, $K_3$ are ‘newborn’, ‘bathing’, ‘shawl’ respectively. Note that a candidate may have a weak semantic relationship with its corresponding category name. For example, the candidates’ source [100] recommends ‘shawl’ as a candidate keyword for Baby category since they are related perhaps owing to the use of shawls to cover or carry babies with. However querying ‘shawl’ ($K_3$) in a video search engine is much less likely to retrieve Baby related videos than it is to retrieve Clothing related videos. Thus including $RV(\text{‘shawl’})$ in training data of category Baby would add more mislabeled videos in $T(Baby)$ than videos having true label of category Baby. Validity Filter (2.4) ensures that candidates...
such as ‘shawl’ are not valid keywords for category Baby.

Let $N_{\text{Valid},i}$ be the number of valid keywords for category $i$. In order to select keywords from the valid keywords, we determine the suitability of keywords to retrieve training videos for category $i$ based on two components: 1) High Proximity, 2) High Diversity. We discuss below how the above components lead to training videos having the desired properties, less mislabeled videos and high Intra-Category Diversity respectively.

**High Proximity.** Under assumptions of whitened data and multivariate normal distributions as the categories, the likelihood of a video $v$ belonging to a category $i$ is proportional to $e^{-\frac{1}{2}|v - \mu_{i}|^2}$. Assuming equal prior probabilities for all categories, the probability that $v$ has its true label as category $i$ is higher when $|v - \mu_{i}|$ is lower, where $\mu_{i}$ is the true mean of category $i$. Thus, if $v$ is a video in training data of category $i$, then $P(v \text{ is a mislabeled video})$ is lower when $|v - \mu_{i}|$ is lower. Consider a valid keyword $K$ for category $i$. Satisfying (2.4) merely implies that $RV(K)$ contains more videos of true label category $i$ than videos that are mislabeled as category $i$. Preference should be given to valid keywords that lead to lesser mislabeled videos in the resulting set of training videos of category $i$. In order to do so, a valid keyword $K$ should be preferred if the videos in $RV(K)$ are closer to $\mu_{i}$. We thus calculate **Proximity score** for each valid keyword $K$ as

$$\text{Proximity score for } K = \{1 / |\mu_{K} - \mu_{i}| \},$$

(2.5)

where $\mu_{K}$ is the centroid of $RV(K)$. A keyword $K$ should be preferred to be selected as a Selected Retriever Keyword (SRK) if the Proximity Score of $K$ is high.

In Fig. 2.2, ‘newborn’ ($K_1$) and ‘bathing’ ($K_2$) are both Valid Candidate Keywords for category Baby ($C_1$). Since the centroid of $RV(\text{‘newborn’})$ is closer to the mean of $C_1$ than the centroid of $RV(\text{‘bathing’})$, the proximity score of ‘newborn’ is more than that of ‘bathing’. Thus as per the proximity score, the valid keyword ‘newborn’ should be
preferred over ‘bathing’. Since \( P(\text{true label of } v = C_1 \mid v \in RV(\text{`newborn'}) ) \) is more than \( P(\text{true label of } v = C_1 \mid v \in RV(\text{`bathing'}) ) \), preferring ‘newborn’ over ‘bathing’ as SRK reduces mislabeled videos in resulting set of training videos.

Note that the measurement of Proximity score of a valid keyword is the same for both approaches proposed in Section 2.3.2.

**High Diversity.** Intuitively, the diversity of a set of keywords reflects the Intra-Category Diversity of the set of videos that are retrieved using the keywords. For the LCPD approach discussed next in Section 2.4.1, the Intra-Category Diversity of the training videos of category \( i \), i.e., \( \text{div}(T(i)) \) is measured based on variance of the set of videos \( T(i) \). Based on such a measure for \( \text{div}(T(i)) \), a diversity score can be defined for each keyword \( K \) given a subset of the training data of category \( i \), denoted by \( T'(i) \). Details of such a per-keyword measurement of Intra-Category Diversity are provided in Section 2.4.1. As an alternative way, since training videos are retrieved using SRKs, the degree of variation in \( T(i) \) could also be estimated based on the degree of variation in the set of keywords that are used to obtain them. Based on this insight, the second keyword selection approach (AAO approach in Section 2.4.2) uses a diversity measure for a set of keywords, instead of a diversity measure per keyword as used in Section 2.4.1. In order to obtain a diversity measure for a set of keywords, distance or linear independence based measures can be used. Section 2.4.2 discusses the efficiency implications of different diversity measures for set of keywords. For the AAO approach, we utilize linear independence based diversity measures for set of keywords.

Based on the above discussion, SRKs need to be selected such that they have high proximity score, and high diversity. Selecting keywords based on high proximity tends to prefer keywords that retrieve videos similar or close to those retrieved by the name of the category. On the other hand, selecting keywords based on high diversity tends to prefer keywords that are more different or far from each other. Thus the two
objectives tend to be opposing in nature. Figures 2.3(a) and 2.3(b) provide an example of a set of 5 keywords selected based on either objective for a set of 20 valid keywords of the category ‘Baby’. As can be seen, the sets of keywords optimizing either objective are very different from each other and optimizing for one tends to lead to keywords having poor objective score for the other. In the next section, we propose approaches to address keyword selection problem based on such conflicting objectives.

2.4 Approaches to select SRKs

In order to obtain SRKs from a set of valid keywords, we propose two efficient approaches. The first approach (LCPD) is based on defining a diversity score per keyword. By defining a linear combination of the proximity and diversity scores of keywords, we propose an iterative algorithm to select SRKs one by one. Such an approach leads to high accuracy in video classification based on varying a parameter that controls the relative importance given to the two objective functions, namely proximity score and diversity score of a selected keyword. However, since the best parameter value cannot be known beforehand, and is different for different sets of categories, we also propose an annealing based alternating optimization (AAO) approach. In the AAO approach, we utilize two iterative algorithms that optimize for either objective function and we combine them in a simulated annealing framework. In order to facilitate the second approach, we define diversity for a set of keywords. Such an approach doesn’t need preferences to be embedded in form of parameters, however as compared to the first approach, it may suffer some performance loss.

2.4.1 Linear combination of proximity and diversity (LCPD)

In this approach, we define Suitability for Retrieving Training video (SRT) score for a valid keyword $K$ as a score that indicates how suitable the valid keyword is to
retrieve videos for category $i$ such that the resulting training data has the desired properties discussed in Section 2.2. SRT score of a valid keyword is defined based on the Proximity score of the keyword as defined and measured in Section 2.3.2, and the Diversity score as defined below.

For the training data $T(i)$ of category $i$, the Intra Category Diversity $div(T(i))$ can be measured as the average pair-wise distance between the videos in $T(i)$, assuming a distance measure between the videos. The time-complexity of such a measure, however, is $O(N^2)$ where $N$ denotes the cardinality of $T(i)$. For the approach proposed in this section, we choose to estimate $div(T(i))$ by the variance of $T(i)$, primarily because of its low time-complexity $O(N)$.

$$
\text{div}(T(i)) = \sqrt{\frac{\sum_{j=1}^{N} |v_j - \mu_i|^2}{N}},
$$

(2.6)

where $\mu_i = \frac{\sum_{j=1}^{N} v_j}{N}$; $v_j$ is a training video for category $i$, and $N$ is the cardinality of the set $T(i)$, i.e., $\{v_j \in T(i)\}_{j=1}^{N}$.

Assume that $T'(i)$ is the current training data of category $i$, then the set of training videos of category $i$ when $RV(K)$ are added to $T'(i)$ would be $\{T'(i) \cup RV(K)\}$. We hence define Diversity score of a valid keyword $K$ for category $i$, given existing training data $T'(i)$, as

$$
\text{Diversity score for } K \text{ given } T'(i) = div(T'(i) \cup RV(K)),
$$

(2.7)

where $div()$ can be calculated using (2.6).

For the set of categories Baby, Clothing, Fitness, Food, Table I shows the Proximity and Diversity scores for certain valid keywords of category Baby. For the purpose
Table 2.1. Proximity and diversity scores for a few valid keywords of category Baby, calculated as detailed in Section 2.4.1

<table>
<thead>
<tr>
<th>Valid Candidate Keyword</th>
<th>Proximity Score</th>
<th>Diversity score</th>
</tr>
</thead>
<tbody>
<tr>
<td>alert</td>
<td>0.607</td>
<td>0.3737</td>
</tr>
<tr>
<td>baby carriage</td>
<td>3.346</td>
<td>0.2525</td>
</tr>
<tr>
<td>baby sit</td>
<td>2.815</td>
<td>0.2559</td>
</tr>
<tr>
<td>bassinet</td>
<td>3.148</td>
<td>0.3127</td>
</tr>
<tr>
<td>newborn</td>
<td>1.794</td>
<td>0.3495</td>
</tr>
<tr>
<td>swaddle</td>
<td>3.287</td>
<td>0.1899</td>
</tr>
<tr>
<td>tootsy</td>
<td>0.989</td>
<td>0.4769</td>
</tr>
<tr>
<td>bathing</td>
<td>0.756</td>
<td>0.2036</td>
</tr>
</tbody>
</table>

of calculation of these scores, the existing training data $T'(i)$ for category $i$ is taken to be $RV(C_i)$. The valid keyword ‘baby carriage’ has a high Proximity score, indicating it retrieves videos that are very close (in terms of Euclidean distance) to $RV(‘Baby’)$, but has a low Diversity score, indicating low Intra-Category Diversity of the resulting training data $\{RV(‘Baby’) \cup RV(‘Baby Carriage’)\}$. Compared to this, ‘alert’ is a Valid Candidate Keyword for Baby which has a high Diversity score, but has a very low Proximity score and hence very unlikely to retrieve Baby-related training videos. Note that Table I only shows valid keywords so candidate keywords such as ‘shawl’ that fail the Validity Filter are not provided.

In order to combine the Proximity and Diversity scores to obtain $SRT$ score of a valid keyword $K$, we assume $SRT$ to be a simplistic linear combination of the two scores. $SRT(K, T'(i))$ denotes the Suitability for retrieving Training video score of a valid keyword $K$ for category $i$, given that existing training data for category $i$ is $T'(i)$.

$$SRT(K, T'(i)) = \alpha \{ N_1 \frac{1}{|\mu_K - \mu_i|} \} + (1 - \alpha) \{ N_2 . div(T(i) \cup RV(K)) \}.$$  \hspace{1cm} (2.8)

Here, $N_1$ and $N_2$ are normalization factors used to ensure that Proximity and Diversity scores have the same order of magnitude in (2.8). $\alpha \in [0, 1]$ is the moderation
factor, which decides the weight given to the Proximity score relative to the Diversity score. We next discuss an iterative algorithm to obtain (a maximum of) \( L \) keywords as SRK from a given set of candidates for each category, using \( SRT \) as calculated in (2.8).

**SRK Selection Algorithm for LCPD**

The category names \( C_i \), and number of SRKs to be selected (\( L \)) are inputs to the proposed SRK Selection Algorithm (Algorithm 1) shown below. Assume that \( M \) Candidate Keywords are available for each category. Let the set of Candidate Keywords for category \( i \) be \( K_{\text{Candidates},i} \). For each category, a set of valid keywords \( K_{\text{Valid},i} \) is selected as a subset of \( K_{\text{Candidates},i} \) that satisfy (2.4). Starting with \( T'(i) = RV(C_i), SRT(K,T'(i)) \) is calculated for each valid keyword using (2.8), and the top keyword \( K_{\text{top}} \) is selected as an SRK. \( T'(i) \) is then updated to \( T'(i) \cup RV(K_{\text{top}}) \), and the process is repeated until \( L \) SRKs are selected or there are no valid keywords left. The proposed algorithm selects SRKs in an iterative manner, as compared to ranking valid keywords by their \( SRT \) score calculated once and selecting the top \( L \) keywords. While the latter calculates \( SRT \) scores independent of other SRKs selected, the proposed algorithm attempts to increase Intra-Category Diversity of the resulting training data, leading to better performance of trained classification model.

For each category, number of valid keywords may be different and hence, the number of selected SRKs by the proposed algorithm need not be similar across different categories. In order to avoid any class imbalance, we select the first \( L' \) SRKs for each category to obtain training videos, where \( L' \) is the minimum number of SRKs selected by the proposed algorithm across all categories. \( L' \leq L \) since a category may have less than \( L \) valid keywords. The training videos retrieved by SRKs as per (2.1) can be utilized to train a classification model by giving equal weight or unequal weights to training videos, as discussed below.
• **Non-Weighted Instances:** Classification model is trained by giving equal weight to all training instances (videos).

• **Weighted Instances:** Classification model is trained by giving a weight to a training video \( v \) depending on the order in which the proposed algorithm selected the SRK corresponding to \( v \). Consider a SRK \( K \) that is selected for category \( i \) by the proposed algorithm in the \( n^{th} \) iteration. Each video \( v \) in \( RV(K) \) gets weight equal to \( (1 - 1/n) \).

**Algorithm 1. SRK selection algorithm for LCPD approach**

**Inputs:**
- Names of categories: \( C_i \)
- Number of SRKs: \( L \)
- Candidate keywords per category: \( K_{Candidates,i} \)
  (Note that the algorithm is run for each category)

**Initialization:**
- \( K_{SRK,i} \leftarrow [\] \) (empty set)
- \( T'(i) \leftarrow RV(C_i) \)

**Applying Validity Filter:**
- \( K_{Valid,i} \leftarrow \{ K \in K_{Candidates,i} : K \text{ satisfies (2.4)} \} \)

**Iterative Algorithm:**
- For \( n = 1 \) to \( L \) (each iteration)
  - If \( |K_{Valid,i}| = 0 \); STOP
  - Calculate \( SRT(K, T'(i)) \forall K \in K_{Valid,i} \) as per (2.8)
  - \( K_{top} \leftarrow \arg \max_K SRT(K, T'(i)) \)
  - \( K_{SRK,i} \leftarrow K_{SRK,i} \cup K_{top} \)
  - \( T'(i) \leftarrow T'(i) \cup RV(K_{top}) \)
  - \( K_{Valid,i} \leftarrow K_{Valid,i} \setminus K_{top} \)
- EndFor

**Output:** \( K_{SRK,i} \)

**Analysis of Algorithm 1 and proposed methods.** For the two methods described above, if the candidates are distinct, selecting more SRKs is in general expected to increase the Intra-Category Diversity, thus leading to better performance. However, after certain number of SRKs are selected, it is expected that the additional information of the new topics coming into the training data will reduce, and performance might saturate. Also, the SRKs selected in earlier iterations were determined to be more suitable for retrieving training videos than ones selected later by Algorithm 1. The SRKs selected in
last few iterations might not be very suitable to retrieve training videos of the respective category although they cleared the Validity Filter. Such SRKs might add videos of topics beyond the realm of categories of interest, and make the training data too general and less discriminative. Hence, the performance of Non-Weighted Instances method might peak at a certain $L$, and then degrade since it gives equal weight to training videos retrieved by all SRKs. In the Weighted Instances method however, the weight given to videos retrieved by SRKs accepted later is lesser. This makes the trained classifier less influenced by videos retrieved by SRKs that are selected in the last few iterations by the Algorithm 1. Thus, unlike in the case of the Non-Weighted Instances method, the performance of Weighted Instances method may saturate with increasing $L$. The number of SRKs that lead to best classification performance for Non-Weighted Instances method may vary with the set of categories, the number and source of candidate keywords, among other factors. It is hence a better approach to choose as large $L$ as permitted by computational resources, and utilize the training videos for training of classification model, using the Weighted Instances method.

**Complexity analysis for Algorithm 1.** The time-complexity of Algorithm 1 is $O(M \cdot N_{Cat} \cdot L)$, where $M$ is the number of candidates for each category, $N_{Cat}$ is number of categories, and $L$ is the number of SRKs selected. This is because in every iteration, the valid keywords for each category are given an SRT score, and the maximum number of iterations that the proposed algorithm can run for is $L$. For fixed $M$ and $N_{Cat}$, the time-complexity of proposed algorithm varies as $O(L)$. The space-complexity of the proposed algorithm is $O(L)$ when the number of videos retrieved per SRK is kept constant. For learning tasks, when a very large sized training data is available, the size of training data used for training a model is generally constrained based on space and time-complexity of the employed learning algorithms, and available resources. Such constraints can dictate the total number of SRKs, i.e., $L$, utilized to retrieve training
videos. In the following section, we provide observed space requirement, and time taken by the proposed algorithm based on our implementation, as well as its performance.

The moderation factor $\alpha$ controls the relative importance given to proximity and diversity scores of valid keywords, and thus controls the set of SRKs that are finally selected. The value of $\alpha$ that leads to best performance of the video classification model depends on the category in concern, and on its set of candidate keywords and is thus hard to estimate beforehand. As will be shown in Section 2.5, the video classification accuracy varies substantially with the moderation factor $\alpha$. Thus, as is also discussed in Section 2.1, a shortcoming of the LCPD approach is that it requires a pre-determined value for the moderation factor $\alpha$, which may not be always available. So we next propose an approach that balances the objective functions of proximity and diversity of a set of keywords in a simulated annealing framework, thus converging to a unique solution, without the need to articulate preferences or bias towards either objective apriori.

### 2.4.2 Annealing based alternating optimization (AAO) for proximity and diversity maximization

In this section, we describe an approach where two iterative algorithms, each optimizing for a certain objective function, are run alternately in order to obtain a solution that is a mutual trade-off between the two objectives. The two algorithms have the task of selecting a set of $L$ keywords amongst the valid keywords, such that the selected set maximizes average proximity and diversity respectively. The algorithms are discussed in detail in Sections 2.4.2 and 2.4.2 respectively. The methodology adopted for measuring average proximity and diversity of the set of selected keywords for the purpose of this approach is described next.
**Measurement of proximity and diversity scores**

For the AAO approach, proximity of a keyword $K$ is measured as detailed in Section 2.3.2. For the LCPD approach, diversity was measured as the variance of the set of videos retrieved. Based on this, a diversity score per keyword given a training data was defined (2.7). Such a measurement was feasible for LCPD as it does not need to employ an algorithm to maximize the diversity of a set of keywords. In contrast to LCPD, the AAO approach measures diversity for a set of keywords, and not per keyword given certain training data. Using such a diversity measure, AAO employs an efficient algorithm capable of selecting the set of $L$ keywords that has the maximum diversity.

In order to define diversity for a set of keywords, we first represent each keyword $K$ by the centroid of its retrieved videos $RV(K)$. The diversity of a set of keywords could be defined based on certain pair-wise distances measures between the keywords, however an optimization technique based on such a measure would essentially boil down to performing a combinatorial search over all $L$-sized subsets of the set of valid keywords, making it highly inefficient if not infeasible. As an example, for selecting 25 keywords from 200 valid keywords, this means going through $\binom{200}{25}$ i.e., around $4 \times 10^{31}$ subsets. Hence, for AAO we develop a different way of measuring diversity: the diversity of a set of keywords is defined based on the degree of linear independence between the keywords. Doing so leads to highly efficient optimization through Rank-Revealing QR (RR-QR) factorizations, to obtain which efficient algorithms have been proposed in literature [16, 42, 20, 47].

Assume that the set of $L$ keywords for which we would like to calculate the diversity is $x_L = \{K_j\}$ where $j = 1$ to $L$. The matrix representation of $x_L$ is $X_L \in \mathbb{R}^{m \times L}$ where $m$ is the dimensionality of the representation of each video i.e., the size of the bag of words representation of each video, and $m \geq L$. The $j^{th}$ column of $X_L$ is represented
as $\mu_{K_j}$ i.e., as the centroid of the set of videos $RV(K_j)$. The diversity score of the set of keywords $x_L$ is then defined as

$$\text{Diversity score for } x_L = \prod_b \sigma_b(x_L), \quad (2.9)$$

where $\sigma_b(x_L)$ is the $b^{th}$ singular value of the matrix $X_L$.

**Overall Approach for AAO**

The AAO approach optimizes for two objectives (average proximity score, and diversity score of a set of keywords) alternately, under a simulated annealing framework. Average proximity score of a set of keywords, as the name implies, is the average of the proximity scores of the keywords in the set. This is represented as $O_{Proximity}$. Diversity score of a set of keywords is represented as $O_{Diversity}$ and is calculated using (2.9). The two iterative algorithms that optimize for these objective functions individually can converge to the most proximate or the most diverse set of keywords respectively. However, we add a constraint parameter $T$ where $T \in [1, \infty)$ in the framework that controls the extent to which each algorithm is permitted to run. The individual optimization algorithms are such that $T$ can be provided as an input. They also take a set of keywords as input for the initial solution and improve the solution in each iteration. For a lenient (lower) value of $T$, either of the algorithms is permitted to run for more number of iterations and as a result, the solution at the end of the run is better in terms of the corresponding objective function. The constraint parameter $T$ is set to 1 to start with, which allows both algorithms to optimize for their respective objective functions.

Fig. 2.4 gives an overview of the AAO approach. The iterative algorithm to optimize for $O_{Proximity}$ (shown as Max-Proximity in Fig. 2.4) is initially run based on a random set of $L$ keywords and $T = 1$, and selects the set of $L$ keywords that has the
Figure 2.4. Overview of the AAO approach for alternately optimizing the average proximity and diversity of selected keywords.
highest average proximity. This set is provided as input to the iterative algorithm to optimize for $O_{\text{Diversity}}$ (shown as Max-Diversity in Fig. 2.4), along with the same value of $T$. This leads to the selection of a set of $L$ keywords having the maximum diversity score. This completes the first iteration of the AAO framework, also referred to as the first framework iteration. Framework iterations are represented by $n_{fi}$. For 20 valid candidate keywords of the category Baby and for $T = 1$, the sets of 5 keywords obtained at the end of either algorithm are provided in Figures 2.3(a) and 2.3(b) respectively. The output, i.e., the set of keywords selected at the end of the first framework iteration is then routed back to the Max-Proximity algorithm to start the second framework iteration. At this point, the constraint parameter $T$ is increased by a constant amount $\Delta$ to a higher and thus a stricter value. As a result of the stricter value of $T$, the algorithm Max-Proximity is constrained from running completely. While it may select a set of keywords with higher average proximity than the output of first framework iteration, the set of keywords selected may not maximize the average proximity as the algorithm was prevented from running completely. This process continues and the value of $T$ is increased at the end of each framework iteration, thus making it harder for the algorithms to select keywords that maximize their corresponding objective functions. The framework stops when either of the two algorithms cannot proceed.

The sets of selected keywords obtained at the end of the algorithms Max-Proximity and Max-Diversity vary a lot with each other in the initial framework iterations. This can be seen in Fig. 2.5(b) where a higher intensity of colors (red or blue) reflects a higher value of the corresponding objective function (average proximity or diversity respectively). As $T$ increases, the variation in the outputs at the end of the two algorithms reduces. The overall annealing based optimization framework is detailed in Algorithm II. Note that the overall framework is iterative and the individual algorithms that maximize average proximity or diversity are iterative too. In order to avoid confusion, an iteration of either
Max-Proximity algorithm, or of Max-Diversity algorithm is referred to as an algorithm iteration and is represented by \( n_{ai} \). Details on the individual iterative algorithms that optimize for the two objective functions are provided in the next section. Note that the proof of convergence for Algorithm II is given after describing Max-Proximity and Max-Diversity algorithms since the proof requires convergence properties of Max-Proximity.

In Section 2.4.2 we study the effect of \( \Delta \) on the convergence of the AAO framework, specifically on the resulting set of keywords after convergence and on the time taken to converge. Further, in order to reduce the time taken by the optimization framework to converge without inducing bias in the end result, we propose an adaptive technique to update the constraint parameter \( T \) based on the weakest permitted algorithm iteration of the previous framework iteration. Details and the definition of the weakest permitted algorithm iteration are provided in Section 2.4.2.

**Algorithm 2.** Annealing based alternating optimization (AAO) framework for maximizing average proximity and diversity of selected keywords

**Inputs:**
- Number of keywords: \( L \)
- Set of valid keywords: \( K_{valid} \)
- Representation of valid keywords in matrix form: \( X \)
- Proximity score \( \forall K \in K_{valid} \)
- Increment for constraint parameter (\( T \)): \( \Delta \)
  (Note that the subscript \( i \) is dropped since the algorithm is run for each category separately)

**Initialization:**
- \( K_{SRK} \leftarrow \) set of random \( L \) keywords from \( K_{valid} \)
- \( T \leftarrow 1 \)
- \( n_{fi} \leftarrow 0 \)

**Begin framework iterations:**
While (1)

Maximize proximity:
- \( (K_{SRK}, n_{ai}) \leftarrow \text{Max-Proximity}(K_{SRK}, T, \text{Proximity scores}) \)
  If \( (n_{ai} = 0) \): STOP

Maximize diversity:
- \( (K_{SRK}, n_{ai}) \leftarrow \text{Max-Diversity}(K_{SRK}, T, X) \)
  If \( (n_{ai} = 0) \): STOP
  \( T \leftarrow T + \Delta \) (increment constraint parameter)
- \( n_{fi} \leftarrow n_{fi} + 1 \)

End While

**Output:** Selected set of retriever keywords: \( K_{SRK} \)
Algorithm 3. Max-Proximity algorithm

**Inputs:**
- Initial set of selected keywords: $I_{SRK}$
- Constraint parameter: $T$
- Valid keywords $K_{Valid}$ and Proximity score $\forall K \in K_{Valid}$
  (Note that the subscript $i$ is dropped since the algorithm is run for each category separately)

**Initialization:**
- $L \leftarrow |I_{SRK}|$
- $K_{SRK} \leftarrow I_{SRK}$
- $W I_{Proximity} \leftarrow$ an arbitrarily large value
  ($W I_{Proximity}$: Weakest permitted algorithm iteration)
- $n_{ai} \leftarrow 0$

**Begin framework iterations:**
- While (1)
  - $O_{Proximity} \leftarrow$ Average proximity score of $K_{SRK}$
  - Replace least proximate keyword in $K_{SRK}$ with most proximate keyword in $\{K_{Valid} \setminus K_{SRK}\}$ to get $K'_{SRK}$
  - $O'_{Proximity} \leftarrow$ Average proximity score of $K'_{SRK}$
  - If $O'_{Proximity} > T$:
    - $K_{SRK} \leftarrow K'_{SRK}$
  - If $O'_{Proximity} < W I_{Proximity}$:
    - $W I_{Proximity} \leftarrow O'_{Proximity}$
  - Else: STOP
  - $n_{ai} \leftarrow n_{ai} + 1$

**End While**

**Outputs:**
- Selected set of retriever keywords: $K_{SRK}$
- $W I_{Proximity}$ (useful for Algorithm 5)
- Number of algorithm iterations: $n_{ai}$
Average proximity score maximization

The proximity scores of valid keywords for a category are fixed values defined as per (2.5). The easiest way to maximize the average proximity of selected $L$ keywords from the set of valid keywords for category $i$, i.e., from $K_{\text{Valid},i}$, is to select the $L$ keywords from $K_{\text{Valid},i}$ having the highest proximity scores. However such an approach does not permit adding a constraint parameter ($T$) to control the extent to which the proximity maximization algorithm is run. In order to do that, we propose an iterative algorithm $\text{Max-Proximity}$ that maximizes the average proximity of selected $L$ keywords by performing exchanges or swaps between the selected keywords and the non-selected valid keywords. $\text{Max-Proximity}$ is presented in Algorithm 3. Each iteration of the algorithm checks if replacing the selected keyword having the least proximity score (i.e., the least proximate selected keyword) with the most proximate non-selected valid keyword leads to a relative benefit in average proximity above the constraint parameter $T$. The relative benefit is defined as a ratio of the average proximity of the selected keywords after the swap, with the average proximity before the swap. If that is the case, the above mentioned two keywords are swapped and the set of selected keywords $K_{SRK,i}$ is updated. If not, the algorithm terminates since this means no other swap between the keywords would lead to a relative benefit in average proximity above $T$. The algorithm takes as input and starts from a set of $L$ keywords as $K_{SRK,i}$, referred to in Algorithm 3 as $I$. When $T$ is equal to 1, any improvement in the average proximity score is acceptable and $\text{Max-Proximity}$ converges to the set of $L$ keywords having highest proximity scores.

Proof of correctness for Algorithm 3. We prove that $\text{Max-Proximity}$ (Algorithm 3) leads to maximization of the average proximity score when permitted to run completely, i.e., when $T = 1$. Assume that $S_{OptProx}$ is the optimal set of keywords that maximizes the average proximity score amongst a set of valid keywords $K_{\text{Valid}}$. $S_{OptProx}$
then comprises of the \( L \) keywords in \( K_{\text{Valid}} \) which have the highest proximity score. Also assume that \( S_{\text{init}} \) is the set of valid keywords that are given as initialization to Algorithm 3. The cardinality of \( S_{\text{init}} \) is \( L \), the number of keywords that are to be selected by \textit{Max-Proximity}. Then, there are (\( |S_{\text{OptProx}} \cap S_{\text{init}}| \)) keywords in \( S_{\text{init}} \) that are also in \( S_{\text{OptProx}} \), and are also the (\( |S_{\text{OptProx}} \cap S_{\text{init}}| \)) most proximate keywords in \( S_{\text{init}} \). In \( |S_{\text{OptProx}} \setminus S_{\text{init}}| \) iterations of Algorithm 3, the \( |S_{\text{OptProx}} \setminus S_{\text{init}}| \) least proximate keywords from \( S_{\text{init}} \) will be exchanged (swapped) with the keywords in the set \( S_{\text{OptProx}} \setminus S_{\text{init}} \) in the order of decreasing proximity score of the latter. All such exchanges replace a keyword having a lower proximity score with another having a higher proximity score. Thus, the average proximity score of the selected keywords goes up by some amount in each iteration and since \( T = 1 \), any improvement would be acceptable. Thus, at the end of \( |S_{\text{OptProx}} \setminus S_{\text{init}}| \) iterations, the set of keywords that will be selected will be \( S_{\text{OptProx}} \). □

As a corollary, since \textit{Max-Proximity} takes \( |S_{\text{OptProx}} \setminus S_{\text{init}}| \) iterations to select the set of most proximate keywords, and since \( |S_{\text{OptProx}} \setminus S_{\text{init}}| \) can take a maximum value of \( L \), the maximum number of iterations that \textit{Max-Proximity} can run for is \( L \).

\textbf{Complexity analysis for Algorithm 3.} The complexity of Algorithm 3 per keyword exchange is \( O(L(N_{\text{Valid}} - L)) \) where \( L \) is the number of SRKs. Since \( N_{\text{Valid}} \geq L \), and the maximum number of algorithm iterations is \( L \), the worst case complexity of Algorithm 3 is \( O(L^2N_{\text{Valid}}) \).

\textbf{Diversity score maximization}

We propose to use Rank Revealing QR factorization (RRQR) [47] to select keywords having the highest diversity, where diversity of a set of keywords is measured using (2.9), since there exist efficient algorithms to provide RRQR factorization [47, 42]. RRQR factorization has also been used in [110] to obtain diverse video frames for the application of video summarization. Let \( x_i \) be the set of valid candidate keywords for
Algorithm 4. Max-Diversity algorithm

Inputs:
Initial set of selected keywords: $I_{SRK}$
Constraint parameter: $T$
Valid keywords $K_{Valid}$ and their matrix form: $X$
(Note that the subscript $i$ is dropped since the algorithm is run for each category separately)

Initialization:
$L \leftarrow |I_{SRK}|$
$K_{SRK} \leftarrow I_{SRK}$
Obtain $\Pi_{init}$ such that keywords in $K_{SRK}$ are the first $L$ columns of $X$
$X \leftarrow X \Pi_{init}$ (rearrange columns of $X$)
$WIDiversity \leftarrow$ an arbitrarily large value
($WIDiversity$: Weakest permitted algorithm iteration)
$n_{ai} \leftarrow 0$

Begin framework iterations:
While (1)
  Compute QR factorization of $X$ to get $A_L, B_L, C_L$ as per (2.11)
  Calculate $G_{i,j}$ as:
  \[ G_{i,j} = \sqrt{(A_L^{-1} B_L)_{i,j}^2 + \left( \frac{\gamma_j (C_L)_{i,j}}{w_i (A_L)_{i,j}} \right)^2} \] (2.10)
  Obtain the best swap as: $(\hat{i}, \hat{j}) \leftarrow \arg \max_{i,j} G_{i,j}$
  If $G_{\hat{i}\hat{j}} > T$:
    Update $K_{SRK}$: Replace $\hat{j}^{th}$ keyword of $K_{SRK}$ with $\hat{j}^{th}$ keyword of $K_{Valid} \setminus K_{SRK}$
    $X \leftarrow X \Pi(\hat{i} \leftrightarrow \hat{j})$
    If $G_{\hat{i}\hat{j}} < WIDiversity$: $WIDiversity = G_{\hat{i}\hat{j}}$
  Else: STOP
  $n_{ai} \leftarrow n_{ai} + 1$
End While

Outputs:
Selected set of retriever keywords: $K_{SRK}$
$WIDiversity$ (useful for Algorithm 5)
Number of algorithm iterations: $n_{ai}$
category $i$. The discussion in this section is for a fixed category (category $i$) and so the subscript $i$ is dropped for convenience. Thus the set of valid keywords is represented as $X = \{K_j\}$ where $j = 1$ to $N_{\text{Valid}}$. As mentioned in Section 2.4.2, the matrix representation of $x$ is $X \in \mathbb{R}^{m \times N_{\text{Valid}}}$ where $m$ is the number of dimensions used to represent each valid keyword and $m \geq N_{\text{Valid}}$. In order to describe the algorithm used to select the $L$ keywords from $x$ that maximize diversity, we first provide a brief discussion on RRQR factorization [47, 42] and its objectives.

Given a matrix $X \in \mathbb{R}^{a \times b}$ where $a \geq b$, consider the partial QR factorization of the form

$$X\Pi = QR = Q^{(A_L B_L C_L)} \tag{2.11}$$

where $Q \in \mathbb{R}^{a \times a}$ is an orthogonal matrix with nonnegative diagonal elements, $A_L \in \mathbb{R}^{L \times L}$ is an upper triangular matrix, $B_L \in \mathbb{R}^{L \times (b-L)}$ and $C_L \in \mathbb{R}^{(a-L) \times (b-L)}$. $\Pi \in \mathbb{R}^{b \times b}$ is a permutation matrix that is a square binary matrix having exactly one 1 in each row and each column and 0 everywhere else. $\Pi$ represents a specific permutation of the columns of the matrix $X$ and the permuted matrix can be obtained as $X\Pi$. For a given permutation $\Pi$, $X\Pi$ can be written as $(X_L X_{(b-L)})$ where $X_L \in \mathbb{R}^{a \times L}$ is the matrix formed by the first $L$ columns of $X$, and $X_{(b-L)} \in \mathbb{R}^{a \times (b-L)}$ is the matrix formed by the last $(b-L)$ columns of $X$. The degree of linear independence of the first $L$ columns of $X\Pi$ can be obtained as the determinant of the matrix $A_L$, also equal to the product of the singular values of the matrix $X_L$, i.e., $\prod \sigma_j(X_L)$. RRQR factorization aims to find the permutation $\Pi$ such that $X\Pi = (X_L X_{(b-L)})$ reveals the rank of the matrix $X$. This permutation $\Pi$ provides $X_L$ the $L$ most linearly independent columns of $X$. RRQR factorization does so by seeking $\Pi$ which maximizes the product of singular values of $X_L$. Please refer to [16, 42, 20, 47] for further details and for proof of correctness of RRQR factorization.
In order to adopt RRQR for AAO approach, we need to add a constraint on the RRQR algorithm based on the parameter $T$. Hence, in each algorithm iteration of RRQR, the relative benefit is defined as the ratio of new and previous objective functions. The relative benefit is compared with $T$ to determine if the updated set of keywords is acceptable or not (Algorithm 4). This ratio is represented as $G_{i,j}$ in Algorithm 4. Since $T \geq 1$, this ensures that within a framework iteration of AAO, the diversity only increases with each algorithm iterations of RRQR algorithm. Note that in Algorithm 4, for a non-singular matrix $W \in \mathbb{R}^{L \times L}$, $1/w_i(W)$ denotes the 2-norm of the $i^{th}$ row of $W^{-1}$. Also, for a matrix $W'$ with $L$ columns, $\gamma_j(W')$ denotes the 2-norm of the $j^{th}$ column of $W'$.

Complexity analysis for Algorithm 4. Using results from [47], the worst case time taken by Algorithm 4 is $O(mLN_{\text{Valid}})$ where $m$ is the number of dimensions used to represent each valid keyword.

Proof of convergence for Algorithm 2. Algorithm 2 calls Max-Proximity with an input set of $L$ keywords. We will first show that if the constraint parameter $T$ is increased beyond a value, $T_{\text{MaxGainProx}}$, then no iteration of Max-Proximity can take place. Let $K_{\text{LowP}}$ and $K_{\text{HighP}}$ be distinct valid keywords with lowest and highest proximity score respectively. Let $S'$ represent certain set of $L - 1$ distinct valid keywords, and let $S$ represent the set of selected keywords in an iteration of Max-Proximity, such that $S = S' \cup K_{\text{LowP}}$ and $K_{\text{HighP}} \not\in S$. The maximum relative gain in the average proximity score of selected keywords by exchanging one keyword would be when $K_{\text{LowP}}$ is replaced with $K_{\text{HighP}}$. Thus, the maximum relative change in the average proximity score of selected keywords can be $\frac{\text{Pr}(K_{\text{HighP}}) + \text{Pr}(S')}{\text{Pr}(K_{\text{LowP}}) + \text{Pr}(S')}$, i.e., equal to $(1 + \frac{K_{\text{HighP}} - K_{\text{LowP}}}{\text{Pr}(K_{\text{LowP}}) + \text{Pr}(S')})$. Here $\text{Pr}(K)$ and $\text{Pr}(S')$ refer to the proximity score of $K$ and the sum of proximity scores of the keywords in $S'$ respectively. The relative change is maximum when $S'$ corresponds to the $L$ least proximate keywords amongst valid keywords, except $K_{\text{LowP}}$. Let this
value of the maximum relative change in average proximity be $T_{\text{MaxGainProx}}$. If $T > T_{\text{MaxGainProx}}$, then Max-Proximity cannot run for any iteration because the relative gain in the average proximity score of selected keywords cannot exceed $T_{\text{MaxGainProx}}$. Thus, as per Algorithm 2 when $T > T_{\text{MaxGainProx}}$, the AAO framework will stop. If the value of $\Delta$ is kept constant, the number of framework iterations it takes for $T$ to increase from 1 to $T_{\text{MaxGainProx}}$ is $\lceil \frac{T_{\text{MaxGainProx}} - 1}{\Delta} \rceil$. Thus the alternating optimization framework will converge in a maximum of $\lceil \frac{T_{\text{MaxGainProx}} - 1}{\Delta} \rceil$ framework iterations. 

Note that the number of iterations as calculated above is only an upper limit to the actual number of iterations that the proposed alternative optimization framework in Algorithm 2 would take.

**Adaptively varying the constraint parameter $T$**

For the scenario when $T = 1$ and $\Delta = 0$, the AAO framework would not converge as each of the algorithms maximizing average proximity and diversity respectively will be permitted to run completely every time, resulting in selecting its own optimal set of keywords. Figs. 2.5(a) and 2.5(b) show how the alternating approach converges to a set of keywords that is a trade-off between the two objective functions, for $\Delta = 0.005$ and for $\Delta = 0.05$ respectively. The results correspond to the set of 20 keywords of the category “Baby” that has been used in Figures 2.3(a) and 2.3(b). As discussed in the previous section, the outputs of the two optimization algorithms vary a lot for initial iterations. These oscillations gradually reduce as $T$ increases. Thus it is evident that $\Delta > 0$ is necessary to ensure that the alternating optimization framework converges. However, it is not clear what the right value of $\Delta$ should be. Having a large $\Delta$ terminates the alternating optimization framework at a point when the last run algorithm to optimize for the corresponding objective function was almost completely run. As a result, the selected set of keywords are very biased towards that objective function. As an example,
Algorithm 5. Adaptive annealing based alternating optimization (Adapt AAO) framework for maximizing average proximity and diversity of selected keywords

Inputs:
- Number of keywords: $L$
- Set of valid keywords: $K_{valid}$
- Representation of valid keywords in matrix form: $X$
- Proximity score $\forall K \in K_{valid}$
  (Note that the subscript $i$ is dropped since the algorithm is run for each category separately)

Initialization:
- $K_{SRK} \leftarrow$ set of random $L$ keywords from $K_{valid}$
- $T \leftarrow 1$
- PreviousSet $\leftarrow$ empty set (used to track previously selected sets of keywords)
- $n_{ai} \leftarrow 0$

Begin framework iterations:
While (1)
  Maximize proximity:
  $(K_{SRK}, W_{Proximity}, n_{ai}) \leftarrow \text{Max-Proximity}(K_{SRK}, T, \text{Proximity scores})$
  If $(n_{ai} = 0)$ STOP

  Maximize diversity:
  $(K_{SRK}, W_{Diversity}, n_{ai}) \leftarrow \text{Max-Diversity}(K_{SRK}, T, X)$
  If $(n_{ai} = 0)$ STOP

  If $K_{SRK} \in \text{PreviousSet}$:
    (Repetitions detected)
    $T \leftarrow T + \min(W_{Proximity}, W_{Diversity})$
    PreviousSet $\leftarrow$ empty set
  Else:
    PreviousSet $\leftarrow$ PreviousSet $\cup$ $K_{SRK}$
    $n_{ai} \leftarrow n_{ai} + 1$

End while

Output: Selected set of retriever keywords: $K_{SRK}$
for the same example of “Baby” category and 20 valid keywords, $\Delta = 1$ leads to selecting keywords that correspond to the set of keywords that maximize diversity (Fig. 2.3(b)) and have a poor average proximity score. As compared to this, having a very small value for $\Delta$ leads to a balance between the two objective functions (Fig. 2.5(a)) but as can also be seen, the number of iterations taken by the AAO framework to converge is very high. Even for a small example of 20 valid keywords and $L = 5$, the number of iterations that the AAO framework takes to converge is 94 when $\Delta = 0.005$ is used.

Since too large or small values of $\Delta$ lead to issues such as bias between objective functions, and large number of iterations taken for AAO approach to converge, we propose an adaptive approach that eliminates the need to have a fixed $\Delta$. Instead, the proposed approach updates $T$ based on the weakest permitted algorithm iteration in the previous framework iteration whenever it detects repetitions in the selected keywords. The weakest permitted algorithm iteration is the algorithm iteration of either Max-Proximity or Max-Diversity with the least relative benefit that occurred in the previous framework iteration. Relative benefit has been defined for Max-Proximity and Max-Diversity algorithms in Section 2.4.2 and Section 2.4.2 respectively. The adaptive annealing based alternating optimization (Adapt AAO) framework is presented in Algorithm 5. $WI_{\text{Proximity}}$ and $WI_{\text{Diversity}}$ represent the strength of the weakest permitted algorithm iteration permitted in the Max-Proximity and Max-Diversity algorithms respectively. Details on how $WI_{\text{Proximity}}$ and $WI_{\text{Diversity}}$ are computed in their respective algorithms are provided in Algorithms 3 and 4 respectively. Repetitions are detected in Algorithm 5 when a set of selected keywords had been selected in a previous framework iteration.

Based on the Adapt AAO approach, Fig. 2.5(c) shows that the number of iterations taken to converge is much lower than in the case of $\Delta = 0.005$ (Fig. 2.5(a)). The resulting set of keywords is the same as in the case of $\Delta = 0.005$, but with substantial reduction in number of iterations taken to converge (94 to 7). Since the Adapt AAO approach
Figure 2.5. Variation in average proximity (red) and diversity (blue) of selected keywords for selecting 5 SRKs from 20 valid keywords of category “Baby” using (a) AAO with $\Delta = 0.005$, (b) AAO with $\Delta = 0.05$, and (c) Adapt AAO. Higher color intensity corresponds to a higher value of the corresponding objective function. $T$ increases along the X axis. Note that the range of variation in the two colors are different, and for both colors, white is assigned to the lowest value of the corresponding objective function value observed during the optimization.
converges in significantly less framework iterations as compared to AAO, and does not result in any artificial bias towards an objective function, we only provide performance results based on Adapt AAO approach in Section 2.5.

**Complexity analysis for Algorithm 5.** The *Max-Proximity* algorithm (Algorithm 3) and the *Max-Diversity* algorithm (Algorithm 4) have worst-case complexities of $O(L^2N_{valid})$ and $O(mL.N_{valid})$ respectively. In each framework iteration, the two algorithms are run in alternation. Since $m > N_{valid} > L$, the dominating number of computations is due to the term $O(mL.N_{valid})$. Thus the total worst case time complexity of Algorithm 5 is $O(mL.N_{valid}N_{FrameworkIter})$ where $N_{FrameworkIter}$ is the number of framework iterations.

### 2.5 Experimental Results

This section discusses our experimental setup and provides performance evaluation of the proposed framework. We conduct our experiments on YouTube videos using the YouTube API. We have used Wikipedia Thesaurus API [87] and Reverse Dictionary [100] as the sources of candidate keywords given the name of a category. The candidates for a category are made distinct by removing any repetitions. The classifier used is a linear Support Vector Machine (SVM). Performance is compared against baseline classifier, which is a linear SVM trained over videos retrieved by category names. Classification accuracy is taken to be the performance measure. Textual data for a video is obtained from the Title, Keywords, and Description in the corresponding webpage. Each video webpage is represented as a bag of word vector of normalized word counts. Textual vocabulary is created based on collecting unigrams that occur in more than 0.5% of total video webpages in training data, and by removing stop words (such as *it, a, or, was* etc).

We conducted a user study to obtain videos viewed by a set of volunteers. More than 14000 videos viewed by 30 volunteers were collected. The testing videos for our
Table 2.2. Summary of average classification accuracy obtained using Adapt AAO approach

<table>
<thead>
<tr>
<th>Approach to select L SRKs</th>
<th>Retail Product categories: Baby, Clothing, Fitness, Food</th>
<th>Genres of Music: Classical, Electronic, Jazz, Rock</th>
<th>Genres of Movies: Action, Comedy, Horror, Romantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximize average Proximity</td>
<td>91%</td>
<td>91%</td>
<td>58.3%</td>
</tr>
<tr>
<td>Maximize diversity</td>
<td>79.4%</td>
<td>89.7%</td>
<td>48%</td>
</tr>
<tr>
<td>Based on Algorithm V</td>
<td>92.6%</td>
<td>92.4%</td>
<td>61.4%</td>
</tr>
</tbody>
</table>

proposed framework are obtained by manually labeling videos collected by the above user study. Testing videos for a category are also supplemented by videos from publicly available lists of popular (or useful or best) videos of the category.

In the next three sub-sections, we summarize results of applying our proposed approach on three different sets of categories.

2.5.1 Retail Product categories: Baby, Clothing, Fitness, Food

As discussed earlier, for a retail or department store (such as Walmart or Sears), knowledge of user preferences in product categories like the above is very useful. We first discuss results obtained using the baseline classifier followed by results obtained using the proposed approaches.

Fig. 2.6 shows the performance of the baseline classifier for classifying 255 test videos. It is seen that the performance of the baseline classifier improves initially as more videos are retrieved by category name and used for training. As shown in Fig. 2.6, the average (across all categories) Intra-Category Diversity (2.6) of the training data thus obtained, increases with the number of retrieved videos, initially leading to performance improvement. However, as more videos are retrieved using the category name, the quality of retrieved videos by the video search engine begins degrading, and more training videos unrelated to the respective categories are retrieved and selected in their training data. This is reflected in loss in classification performance (around 700-800 videos per category)
**Figure 2.6.** Classification accuracy and Intra-Category Diversity variation with number of training videos for baseline classifier

**Figure 2.7.** Classification accuracy variation for \{Baby, Clothing, Fitness, Food\} with respect to $\alpha$. Dotted line shows baseline accuracy

**Figure 2.8.** Classification accuracy variation for \{Baby, Clothing, Fitness, Food\} with respect to $L$
Figure 2.9. Variation of time taken to select $L$ SRKs, and time taken to train linear SVM on obtained training data, with respect to $L$.

Figure 2.10. Classification accuracy variation for \{Classical music, Electronic Music, Jazz music, Rock music\} with respect to $\alpha$.

Figure 2.11. Classification accuracy for \{Action movies, Comedy movies, Horror movies, Romantic movies\} with respect to $\alpha$. 
as more videos are retrieved using category name. In our experiments, 1000 videos are retrieved per SRK. In order to provide a fair comparison of our approach with the baseline, the number of training videos in both cases should be equal. However, from the trend in Fig. 2.6 it can be seen that the best baseline performance is around 82.3%. Since the YouTube API limits number of retrieved videos per keyword to 1000, we utilize the best classification performance (82.3%) of baseline to compare with our techniques.

In Fig. 2.7, we present performance variation of the proposed approaches with the moderation factor $\alpha$. Keywords from [87] and top 200 keywords from [100] are used as candidates, giving a total of 230 candidates per category. The number of valid keywords found per category are Baby: 81, Clothing: 63, Fitness: 78, Food: 52. The coefficients $N_1$ and $N_2$ in (2.8) are chosen such that $N_1 = \frac{1}{N_2} = \text{div}(RV(C_i))$. Fig. 2.7 shows the performance when $L'$ number of SRKs are selected per category, where $L'$ is the minimum number of valid keywords across all categories (which is 52 in this case). We show the performance using both Weighted Instances and Non-weighted Instances methods for LCPD. Weighted support vector machine [137] is used to give varying weights to the training videos as per the Weighted Instances method. Fig. 2.7 also shows the performance obtained by the Adapt AAO approach. As mentioned earlier, the performance of the Adapt AAO approach is same as that of the AAO approach with a small $\Delta$, and so we have only provided performance results of the former. For LCPD, $\alpha = 0.6$ to 1 performs best for Non-weighted Instances method, and $\alpha = 0.6$ to 0.8 performs best for Weighted Instances method. We observe that the Weighted Instances method in general performs better than the Non-weighted Instances method, as discussed in Section 2.4.1. Moreover, both the methods have significantly better accuracy than the baseline classifier accuracy of 82.3% (shown by the dotted line in Fig. 2.7). For the baseline case, the Intra-Category Diversity values (2.6) of the training data are $\{0.259, 0.247, 0.244, 0.243\}$ corresponding to $\{\text{Baby, Clothing, Fitness, Food}\}$. Compared to this, the Intra-Category Diversity after
all 52 SRKs (for $\alpha = 0.6$) are used to retrieve training videos for above categories are
${0.439, 0.418, 0.395, 0.359}$. The average (taken across all categories) Intra-Category Diversity has increased from 0.248 for baseline to 0.403 with LCPD (for 52 SRKs per category, and $\alpha = 0.6$). Consequently, the classifier performance has also increased from 82.3% (baseline performance) to 91% (Non-Weighted Instances) and 93.7% (Weighted Instances), thus demonstrating that higher Intra-Category Diversity in training videos results in better performance of the trained classification model. The classification accuracy obtained using Adapt AAO is 92.6%. Based on Fig. 2.7 it can also be seen that unlike LCPD, the performance of Adapt AAO does not depend on $\alpha$. While the performance of Adapt AAO is lower than the best performance of LCPD, the difference is marginal. In addition, the performance of Adapt AAO is significantly higher than the baseline performance.

Fig. 2.8 shows how the performance of the proposed approaches varies with respect to $L$, i.e., the number of SRKs. For LCPD, $\alpha$ is kept constant at 0.6 for this experiment. As can be seen for LCPD approach, while the performance of the Non-Weighted Instances method starts decreasing after initially increasing with increasing $L$, the Weighted Instances method performs better, and continues its improving performance with increasing $L$. The Adapt AAO approach is seen to consistently provide high classification accuracy. In addition, Adapt AAO offers the advantage of not requiring tuning of the parameter $\alpha$ which may require human effort or the availability of a labeled validation set. Table 2.2 shows the accuracy obtained as per the proposed Adapt AAO approach as well as the accuracy obtained when all SRKs are selected to maximize only one objective function as discussed in Section 2.4.2. It can be seen that the performance of our approach outperforms selecting SRKs based on only one objective function, namely average proximity (2.5) or diversity (2.9).

Fig. 2.9 shows the time taken by the LCPD and Adapt AAO approaches with
respect to the number of SRKs selected (i.e., $L$). Conforming to the complexity discussion in Section 2.4, the time taken to select $L$ SRKs varies as $O(L)$ when the number of candidates and the categories are fixed. Fig. 2.9 also shows the time taken by a linear SVM (LibLinear implementation of SVM) to train over the collected set of training videos. The time-taken to learn the classifier varies approximately as $O(L)$.

For our MATLAB-based implementation, the system memory usage for LCPD approach was 4.46 GBs for selecting 52 SRKs from 230 candidate keywords for category Baby. Since the Adapt AAO approach does not need to store the representations of individual retrieved videos retrieved per keyword, like in LCPD approach, the memory requirement for Adapt AAO approach is only 411 MBs. Training of SVM using 1000 videos for each of 52 SRKs required 1.4GBs. The empirical results presented demonstrate the feasibility of our proposed approach in terms of space and time complexities.

We next present experimental results for two more sets of categories and focus on the classifier performance.

### 2.5.2 Genres of Music: Classical, Electronic, Jazz, Rock

We have chosen these categories keeping the requirements of a music recommendation system in mind. While there are several, and often subjective, categorizations possible within Music, we have chosen the above four categories since these cover most other categories, and are broad in the sense of ease of labeling by a human expert. Top 100 keywords from [100] are used to obtain candidates for each category. 26 SRKs are selected per category. 290 test videos are used to test the performance of the baseline classifier and the proposed approaches. In Fig. 2.10, we present performance of the proposed approaches. While the performance of LCPD varies with $\alpha$, it is seen that performance of classifier for $\alpha \geq 0.2$ is almost the same. The Weighted Instances method is again seen to outperform the Non-Weighted Instances method for higher $\alpha$ values.
(\(\alpha \geq 0.2\)), and both show significantly better accuracy (92.4% and 91% respectively) than the baseline classifier (77.9%). For these categories, Adapt AAO provides as high classification accuracy (92.4%) as LCPD when 26 SRKs are used. This demonstrates the efficacy of the Adapt AAO approach to achieve high performance without the requirement of \(\alpha\).

### 2.5.3 Genres of Movies: Action, Comedy, Horror, Romantic

We here provide results for a set of categories that might be of interest for a movie recommendation system. [100] is used to obtain Candidate Keyword for each category. A total of 223 test videos are used to assess performance. 18 SRKs are selected per category. Fig. 2.11 supports the observation made with the previous sets of categories. For LCPD, the Weighted Instances is seen to be better performing than Non-Weighted Instances and both methods show significantly improved performance (62.8% and 59.6% respectively) compared to performance of baseline classifier (41.2%). Adapt AAO leads to marginally worse performance than LCPD and achieves 61.4% classification accuracy. Both LCPD and AAO approaches achieve significant improvements in predictive correctness as compared to the baseline approach.

Based on the above experiments, it can be seen that the proposed approaches LCPD and Adapt AAO, both lead to significant improvement in classification performance compared to the baseline classifier. The performance of Adapt AAO is slightly worse than LCPD, however since the former does not depend on \(\alpha\), Adapt AAO offers a convenient approach to obtain high classification accuracy without necessitating manually provided preferences for objective functions or requiring manual effort to label validation videos and tune \(\alpha\).
2.6 Summary

We have proposed a fully-automated framework to obtain high quality training videos for any arbitrary set of categories, without the need for any manual labeling that is needed by most related approaches. We analyze properties of training data that lead to high performance of the trained classifier. Based on the above properties, we propose approaches for selecting keywords to retrieve training videos, on the basis of their proximity to the categories of interest, and the resulting diversity in the training data. The first approach (LCPD) leads to high classification accuracy, although requires a parameter representing preference for defined objective functions. The parameter can be obtained through manually provided preferences for the objective functions, or by using manually labeled validation videos for tuning. In order to avoid the manual effort, we also provide an annealing based alternating optimization framework (AAO) and propose its adaptive variant (Adapt AAO) to select keywords. Such an approach does not require the articulation of preferences in parameterized form, with the trade-off of some loss in classification accuracy as compared to the LCPD approach. Experimental results on several sets of categories show the effectiveness of the training videos obtained by the proposed approaches, hence making classification of videos watched by users to arbitrary set of categories feasible. Consequently, this work may enable new personalization applications by enabling identification of user preferences in a set of categories relevant to the application.

2.7 Acknowledgements

We thank the anonymous WI 2013 and IEEE Access reviewers for their feedback and comments on the work. The authors are thankful to Dr. Nitesh Shroff for insightful discussions. The research discussed in this chapter was supported by the UCSD Center
for Wireless Communication and the UC Discovery Grant program.

Chapter 2, in part, contains material as it appears in the Proceedings of the 13th International Conference on Web Intelligence (WI’13). “Fully Automated Learning for Application-Specific Web Video Classification”. Chetan Verma, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in part, contains material as it appears in IEEE Access. “Methods to Obtain Training Videos for Fully Automated Application-Specific Classification”. Chetan Verma, Sujit Dey. The dissertation author was the primary investigator and author of this paper.
Chapter 3

Construction and Evaluation of Ontological Tag Trees

The previous chapter discussed automated methods to obtain the data to train content based classifiers. For annotated online content (such as Flickr images) however, it may be more important to assign good quality tags with the content. This chapter discusses methods to do that, through addressing sparsity of tags associated with online content.

Several methods have been proposed to address the sparsity of tags associated with online content such as images and videos. However most of such methods either necessitate extracting domain-specific features, or are solely based on tag semantics, or have significant space requirements to store corpus based tag statistics. To address these shortcomings, in this work we show how ontological tag trees can be used to encode information present in a given corpus pertaining to interaction between the tags, in a space efficient manner. An ontological tag tree is defined as an undirected, weighted tree on the set of tags where each possible tag is treated as a node in the tree. We formulate the tag tree construction as an optimization problem over the space of trees on the set of tags and propose a novel local search based approach utilizing the co-occurrence statistics of the tags in the corpus. To make the proposed optimization more efficient, we
initialize using the semantic relationships between the tags. The proposed approach is used to construct tag trees over tags for two large corpora of images, one from Flickr and one from a set of stock images. Extensive data-driven evaluations demonstrate that the constructed tag trees can outperform previous approaches in terms of accuracy in predicting unseen tags using a partially observed set of tags, as well as in efficiency of predicting all applicable tags for a resource.

3.1 Introduction

The consumer electronic revolution and the Internet have led to the availability of vast amounts of data including multimedia data such as images and videos. A significant fraction of such data is user generated content, in the form of pictures and videos uploaded onto sites such as Facebook [33], Flickr [35] and YouTube [143]. Owing to the fact that there are minimal requirements when uploading the content and that mobile uploads are on the rise, users rarely add any extra information such as a textual description to the content. At best, most images and videos are tagged with certain keywords. As these keywords or tags are sometimes applied to entire albums of images or videos at once, or applied in error, the information provided by such tags is quite noisy. Some examples of

![Figure 3.1.](image1.png)  
(a) ‘animal’, ‘car’  (b) ‘wedding’, ‘mushroom’  (c) ‘farms’, ‘snow’

Figure 3.1. Examples of incorrect tags given by users on www.flickr.com. (a) An image of a ‘cat’ tagged as ‘car’, which most likely is a spelling mistake, (b) an image of a ‘mushroom’ also tagged as ‘wedding’ and (c) an image of a ‘goat’ tagged with ‘snow’. The Flickr owner and photo ids of these images are (8656572@N04,4670326818), (35468147887@N01,252474171), (39405339@N00,5835556089) respectively.
images having incorrect tags (as per human experts) are shown in Fig. 3.1. The massive scale of data and the lack of useful metadata makes it difficult for users to access data that may be of interest to them [6],[59],[146].

The social tagging at the above mentioned data sharing websites creates a *Folksonomy* [54],[118],[64] which mitigates the information overload to some extent by creating non-hierarchical categories or indexes for the retrieval of data. Folksonomies make it scalable to assign labels to large volumes of data in a collaborative manner and are hence more appropriate for such data than traditional taxonomies established by expert cataloguers [64]. At the same time, collaboratively produced folksonomies have several issues, particularly related to incorrect tags and their sparsity [118],[119]. While incorrect tags has been discussed earlier, the sparsity in folksonomy arises as a result of lack of incentive for the users to tag the resources comprehensively and completely. As a result, the online resources are typically associated with low number of tags, preventing effective searching and browsing through the available data [119].

In order to address the sparsity in folksonomies, several expert systems have been proposed that recommend or suggest additional tags for a resource based on the tags already associated with the resource [118],[111],[54],[22]. Most of such works depend on the availability of content-based features such as textual features from documents or blogs [54],[22],[118], or visual features from images or videos [135], and thus cannot be applied to other domains. In addition, extracting and utilizing content based features is known to be computationally expensive and for certain domains, even infeasible [55], [114], [144], [141], and so the above works may not be applicable to such domains. Furthermore, expert systems such as [119] utilize purely semantic relationships between tags. While semantic relations as obtained from ontologies such as WordNet [83] or ODP [93] are an important resource for linguistic and machine learning related problems, such relationships fail to capture the information that is characteristic of an
available corpus. Consider for instance a corpus of annotated images from Flickr. The co-occurrence of tags in a given corpus provides interesting insights into the nature of the data. For example, the 2008 Olympics were held in Beijing and as a result, there exist a large number of images in Flickr having ‘2008’ and ‘Beijing’ as their tags. Such a relation between ‘2008’ and ‘Beijing’ cannot be obtained from WordNet or similarly formed hierarchies (such as Open Directory Project, ODP [93]) because the semantic relations in the above hierarchies are pre-defined, and do not account for a connection between the two tags. In addition to the above expert systems, works such as [111] capture tag similarities from a given dataset using tag graphs. Tag graphs usually refer to complete graphs representing pair-wise distances or similarities between tags, which are calculated from a given corpus. For certain applications, a threshold is applied and only the most important pair-wise connections are retained. However, storing similarities using tag graphs has several issues. Firstly, the pre-defined threshold value that is chosen to construct them can be arbitrary and there is no clear understanding to what its value should be. The pair-wise edges that have their similarity above the defined threshold are the only ones that are retained in the graph and this leads to completely losing of information of those pairs of concepts or tags that have their similarity below the threshold. Depending on the threshold value, the space requirement of tag graphs can vary as $O(N^2)$ where $N$ is the number of concepts or tags in the tag graph, which can be significantly high for large number of tags. In order to keep a handle on the space requirement, a strict threshold value can be chosen which would result in losing pair-wise similarity information for several pairs of concepts or tags. Lastly, depending on the threshold, it is possible that some concepts or tags are disconnected from the rest. This again implies losing relationship information of the concept or tag with others. [111] estimates the number of tags in Flickr in 2008 to be 3.7 million. Storing each similarity value as a floating point occupying 4 bytes would require more than 27 terabytes just to
store the pair-wise relationships.

We attempt to address the above shortcomings in this chapter. We use the term ontological tag tree or simply tag tree to denote undirected weighted tree of concepts (or tags) where the relationships between the concept nodes in the tree are defined only in terms of a scalar weight. As compared to tag graphs \[111],[75], ontological tag trees are necessarily trees on the set of tags, i.e., are connected and have no simple cycles. We have chosen a spanning tree to represent the relationships between tags because a spanning tree over the set of tags is necessarily connected and does not lead to losing of information due to possibly disconnected components as in tag graphs. Also, the space requirement of a spanning tree is only \(O(N)\) for \(N\) tags. For 3.7 million tags \[111\], this implies a significant reduction in the space requirement from 27 terabytes \(O(N^2)\) to less than 50 megabytes \(O(N)\). As a result, expert systems can be implemented even on computing devices that do not have a gigantic memory. Ontological tag trees are constructed using the semantic and the data-driven relations between the tags and hence lead to significantly better performance on data-driven tasks than using solely semantic relationships between tags \[119],[83\]. For the constructions of tag trees, we do not utilize content based features, rather we utilize data-driven similarities from tag co-occurrences in the given annotated corpus. As a result, compared to previous expert systems that require extracting and processing content-based (such as visual or textual) features \[54],[22],[118],[135\], tag trees can be used to alleviate sparsity in online folksonomies even in domains where extracting domain specific features may be infeasible or inefficient. This also makes the construction approach not married to a single domain such as annotated text documents/blogs or videos or images.

We illustrate the proposed tag tree construction approach using two large image corpora - one obtained from Flickr, and the other obtained from a set of stock images, with the goal of obtaining a tag tree over the set of tags present in these corpora. For
these corpora, the *co-occurrence count* for a pair of tags is defined as the number of images with which both tags are associated. The normalized co-occurrence counts are a measure of how related two tags are. We assume that the concepts or nodes of the tag tree are the tags, and that the tree construction task is to infer the relations between the tags. The task thus becomes a graph learning problem where the nodes of the graph are the tags, and the relations between tags are represented by undirected edges and their weights in the graph. Unlike the relationships given in ontologies, we do not attempt to give semantic interpretations to the relations between tags. To solve the graph learning problem, we formulate an optimization problem on the space of spanning trees of a suitably constructed similarity graph that is based on semantic relations between tags, as obtained from WordNet, and on the normalized co-occurrence counts of the corpus. We solve the optimization problem using the “local search” paradigm by constructing a simple edge exchange based neighborhood on the space of candidate trees. To make the optimization efficient, we initialize our approach using a preliminary tag tree built purely based on semantics from the WordNet hierarchy. The proposed local search based approach is then used to refine the preliminary tag tree based on the corpus statistics.

The evaluation of structures capturing the relationships between different tags or concepts is a difficult task. In the domain of ontologies, there are often no clear quantitative metrics to compare different ontologies that can be built for the same corpus of data. Certain works compare constructed ontologies to a predefined gold standard ontology [97] which is constructed manually. Tag graphs are usually not evaluated explicitly, rather are used in various applications such as Tag Ranking [75]. Since manual evaluations are subjective and are not scalable, in this work, we also propose a novel fully automatic framework to evaluate ontological tag trees over tags using the tag prediction accuracy, given an incomplete set of tags for a resource. Furthermore, we also demonstrate that the constructed tag trees can be used to efficiently assign tags to
resources in domains where content-based features can be derived. Thus as a second evaluation paradigm, we utilize efficiency: for a given resource with no tags, efficiently predict all the applicable tags.

To summarize, the key contributions of this chapter are as follows:

1. We propose a framework to construct an ontological tag tree over tags in a given corpus. The proposed approach requires constructing a preliminary tag tree using semantics obtained from the WordNet hierarchy. This preliminary tree is then refined to incorporate data specific relations by performing a novel local search operating on local neighborhoods in the space of spanning trees of a defined similarity graph over the tags.

2. We propose a completely automated framework for evaluating ontological tag trees over tags by posing two data-driven tasks. The first task is defined such that it does not require content based features to be extracted from resources, in order to assign tags to them. It can thus be applied even to corpora where deriving features from resources is either infeasible or ineffective. The second task is applicable to corpora where domain specific features can be extracted from the resources and classifiers or concept detectors can be trained that can map the content based features of the resource to concepts or tags.

3. We evaluate the constructed tag trees for two large image corpora using the above evaluation framework and show that it outperforms tag trees built using manually created semantic hierarchies such as WordNet, and commonly used approaches using tag graphs of comparable space requirements, in both prediction accuracy and efficiency. We also demonstrate that by using the constructed tag trees, we can achieve a performance that is very close to or better than that of other techniques, while also achieving several orders reduction in the space requirement.
Fig. 3.2. Two examples of subgraphs built using (left) the proposed data-driven approach and (right) corresponding sub-graphs obtained using WordNet. In example (a), ‘holiday’ and ‘travel’ are directly connected using our approach but are separated by multiple hops in the WordNet hierarchy. In example (b), the proposed approach is able to identify ‘party’ as the central node that connects several other party-related tags. For the proposed approach, objective WAH (3.3) is utilized as described in Section 3.3.

Fig. 3.2 illustrates a couple of examples for which the proposed approach captures the inter-dependencies between tags in a qualitatively better form than the tree obtained using WordNet alone. Details on how these trees are obtained are provided in Section 3.3.

We first start with a brief discussion on the related literature.

### 3.1.1 Related Work

In order to discuss the related literature, we study the prior works in terms of works on ontology building, deriving tag relationships, tag recommendation and efficient resource classification, and works on local search paradigm. While we have focused on providing a brief summary of works in these areas that are relevant to this chapter, some works may belong to multiple areas.
Ontology building

A commonly used strategy to organize a collection of data is to group it into categories and specify the relationships among the various categories. Ontologies [34] are often employed to specify predefined relations between categories. Conventionally, constructing an ontology [46] requires significant manual effort. The concepts or categories of the ontology have to be specified, and the relations between the categories defined, all manually. Furthermore, the ontology has to be updated when data belonging to hitherto unseen categories becomes available. Once the ontology has been specified, data samples must be annotated, again manually, to assign them to one or more categories in the ontology so that rules or classifiers can be learned for that category. Therefore, manual techniques for ontological or taxonomic organization of data become especially challenging and cumbersome when there are large amounts of data. Also, ontologies built for one setting are rarely reusable even in other closely related domains. This necessitates the building of an ontology afresh for each new setting. As data could be from one of an ever increasing pool of knowledge domains, manually constructing ontologies for data in each domain is infeasible. Furthermore, when the data obtained is noisy, as is the case for user generated content on the Internet, the problem is accentuated as more manual effort might be needed to clean up the data followed by ontological organization.

The challenges associated with the manual construction of ontologies has led to efforts that use semi-automatic [58] and fully automatic techniques [15] in domains such as multimedia and text based ontologies respectively. Most existing automatic approaches to ontology construction use text mining techniques to identify the concepts and then define relations between the concepts based on their semantic similarity as obtained from lexical databases such as WordNet [83].

Aside of these, other works on ontology building use some form of clustering
to combine similar terms or keywords together to form concepts. First, a similarity metric is defined between tags, words or concepts, and then a hierarchical clustering algorithm is utilized to form a dendrogram with the concepts as the leaves of the formed tree. The hierarchical clustering algorithm can be either bottom-up (agglomerative) or top-down (divisive). Such a procedure creates auxiliary concepts in the tree representing combinations of multiple concepts of interest. For example [89] uses hierarchical clustering based on a compound similarity measure between words. The similarity score is obtained by using a neural network model on syntactical information and corpus based similarities. However such techniques can only group related concepts together at different hierarchical levels, instead of modeling the inter-dependencies in the form of a graph on the concepts. In [28], given a corpus corresponding to a domain, the relations between important concepts are learned with the help of WordNet or by using search engine. Hierarchical clustering is employed to construct a domain specific dendrogram as mentioned above.

Works such as [50] and [25] utilize natural language based grammar rules to learn hierarchies between text entities. Semi-automatic techniques for ontology construction such as Text2Onto [26] assist the user in constructing ontologies from a given set of text based data. Similar techniques have been attempted in the domain of annotated multimedia content, such as images and videos [58]. Fully automatic techniques such as OntoLearn etc. [121], [88], [78] use natural language processing and machine learning to extract concepts and relations from data. For a good review of ontology learning from text see [15]. These works cannot be applied outside of the domain of natural language, since they depend at least in part upon grammatical speech.
Deriving tag relationships

The organization of tags or concepts obtained from different domains has also been explored. Tag clustering has been employed in systems such as Flickr clusters [36] and studies [10] show that it is helpful as a means to allow users to explore the information space of tags. For annotated images, [107] proposed the application of a co-occurrence based subsumption model from [105], to learn whether a tag subsumes another. [45] uses the category confusion matrix to cluster similar categories together in a hierarchical manner. To construct an ontology for a set of tags, [29] maps the tags to WordNet and leverages WordNet’s hierarchy. Tag graphs have been utilized for various applications such as tag ranking [75] to represent the pair-wise similarities or distances between tags. While several works use tag graphs as complete graphs on the set of tags, others choose set of edges that have their distance lower than a heuristically chosen threshold [52]. In general, tag graphs have $O(N^2)$ edges with correspondingly large storage requirement for large values of $N$. For example, [111] estimates the number of tags in Flickr in 2008 to be 3.7 million. Storing each similarity value as a floating point occupying 4 bytes would require more than 27 terabytes to store the pair-wise relationships, as compared to under 50 megabytes as required by the proposed tag tree. This eliminates the need to have computing devices with gigantic memory in order to operate on a large number of tags or concepts for tasks such as tag prediction, resource annotation, etc.

In the domain of annotated images, there exist works that determine semantic relationships between concepts using visual features [132] and using visual features and tags [62]. The approaches proposed in these works are specific to the domain of images and require visual-feature based representation of images. As mentioned in [55][114][144] and [141], extracting and utilizing content based features can be
computationally expensive and even infeasible in certain domains. In addition to above, the space requirement of these works varies as $O(N^2)$ where $N$ is the number of tags or concepts. Our work is different from the above works since we propose a tag tree construction approach which is not dependent on the availability of content-based features from the resources and has a space requirement of only $O(N)$.

Tag recommendation and efficient resource classification

While several expert systems such as [6], [59], [146], [119], [64] address the broader problem of information overload, others focus on addressing the sparsity of online folksonomies through approaches such as tag recommendation and efficient resource classification. In this section we provide a brief summary of the existing literature in the latter category, since such expert systems are closer to our evaluation tasks.

In our first evaluation task, we attempt to predict certain tags associated with a resource while having visibility to other tags associated with the same resource. There exist works in literature that utilize domain-specific features to associate tags to a resource or to determine the relevance of tags to a given resource. For instance, [71] uses visual similarity to determine neighbors of a test images and then aggregates their tags by voting. [131] learns a distance metric to determine images that are close to a given image based on the visual content, and then determines tag relevance for the image. [54] builds a desktop collaborative tagging system to enable collaborative workers to tag their offline documents. [22] approaches tag recommendation as a translation problem to translate the textual description to tags, while [118] uses language modeling to recommend tags for blogs and documents. These works require extraction of domain-specific, in this case visual and textual features from the resource (images or blogs/documents) and cannot be applied to other domains or corpora where deriving features from resources is either infeasible or ineffective [55][114][144][141]. Aside of these works, works such as [119]
use semantic similarity between tags as obtained from WordNet, to address information overload. As discussed in Section 3.1, purely semantics based systems fail to capture data-specific characteristics, thus leading to poor performance on data-driven tasks. The tag prediction task as proposed in this chapter is defined such that it can even be applied to corpora where deriving content based features from resources is either ineffective or infeasible. The proposed task is somewhat similar to the tag recommendation application in [62] and in [111]. However, the procedure for tag recommendation in [62] and [111] requires manual labeling of tags to measure the performance. Since manual labeling is subjective, irregular and not scalable to large numbers of testing resources, we define a tag prediction task where given an incomplete set of tags associated with a resource, we attempt to predict the rest of the tags. Using such an approach, our evaluation is completely automated and does not require any human assistance or labeling. As mentioned above, [62] requires visual features to obtain the concept similarities. We compare the performance of the constructed tag trees with the symmetric sum based tag recommendation approach as outlined in [111] and show that we achieve almost similar performance with several orders reduction in the space requirement.

In the second data-driven evaluation task, we utilize tag trees to associate tags to resources based on their content. Specifically, for domains where content-based features can be extracted, we show that using the constructed tag trees, it is possible to determine which concept detectors should be tested for the test resource, thereby making the resource classification more efficient. Compared to [70], our approach does not require training of faster and efficient classifiers for this task and can utilize pre-trained binary classifiers as concept detectors for different tags. The image annotation application in [132] associates a test image with tags based on applying all concept detectors on the test image and using the predicted image likelihood under the Dual Cross-Media Relevance model [76]. Compared to [132], our approach does not require applying all concept
detectors corresponding to the tags, rather our objective is to determine the concept detectors to apply on a given test image. In order to demonstrate that the proposed approach for the second data-driven task is not applicable to only a single domain, we provide evaluation results based on two types of modalities - visual, and textual. We have used two large sized image corpora to demonstrate the above evaluation tasks. The tasks in Section 3.4 are defined such that they can be used to evaluate the constructed tag trees. Most of the works as discussed above do not offer a way to do so.

**Local search paradigm**

The use of local search methods in combinatorial optimization has a long history [1]. The paradigm has been extensively studied [60][2] due to its practical success on many NP hard problems and also for the insights it provides on the structure of discrete optimization problems. The use of exchange neighborhoods was introduced by [27] and [72] for solving the Traveling Salesman Problem and has since been successfully applied to a wide variety of problems. See [1] for a comprehensive survey.

We formulate an automated approach for building an ontological tag tree with \((N - 1)\) edges for \(N\) tags using WordNet followed by a data-driven refinement. To our knowledge, the formulation of the ontological tag tree construction as an optimization problem on the space of spanning trees and its solution using the “local search” paradigm is completely novel. We use a variant of the edge-exchange method to construct the neighborhood on the solution space.

[124] presents the preliminary results using the proposed approach. We next discuss the problem statement addressed in this chapter, followed by the proposed tree construction approach.
3.2 Problem Statement

We assume that we are given a corpus $\mathcal{C}$ of annotated resources, where each resource is associated with a variable number of tags. The corpus contains:

- Set of resources: $\mathcal{R} = \{r_l\}$ where $l = 1$ to $|\mathcal{C}|$
- Set of tags in the corpus: $\mathcal{T} = \{t_j\}$ where $j = 1$ to $N$
- A binary tag association matrix $B(r, j) = 1$ if tag $t_j$ is associated with resources $r$, and 0 otherwise.

We define an ontological tag tree as an undirected weighted tree on the set of tags $\mathcal{T}$. This implies that the tag tree is connected and has no simple cycles. The task is to arrange the set of tags $\mathcal{T}$ in an ontological tag tree.

3.3 Construction of Ontological Tag Tree

In order to construct the tag tree, we propose an approach that starts with a preliminary tag tree obtained using the semantics encoded in WordNet hierarchy. We follow this by a corpus statistics based tag tree refinement.

3.3.1 Constructing WordNet-based Preliminary Tag Tree

We follow the approach outlined in [29] to derive the semantic relations between the set of tags $\mathcal{T}$. This is done in two stages. In the first stage, disambiguation for the meaning of the tags is done by selecting the most popular concept (synonym set, or synset) for every tag. For example a tag “turkey” can be mapped to the bird, or the country. In WordNet, since the synset corresponding to Turkey, the bird, has a higher frequency count than the synset corresponding to the Republic of Turkey, the former
synset would be selected to map to the tag “turkey”. Then in the second stage, in order to find the relationships between different tags, all links between the mapped concepts are found through the WordNet hierarchy for semantic relationships “is-a” or “part-of”. Since we are only interested in a tag tree which has undirected edges between the tags, we ignore the directions of the edges in the obtained hierarchy, which could otherwise help distinguish more generic concepts or tags from more specific ones.

The resulting undirected graph in general has cycles and is usually disconnected, forming disjoint clusters of tags. In order to construct a tree from the above undirected graph, we first break the cycles and then connect disjoint segments in a greedy manner using inter-tag semantic distances as obtained from WordNet Library [101]. The semantic distance between two synsets in WordNet is defined as:

\[
\frac{\text{MHP}}{\text{HPR} + \text{MHP}}
\]  

(3.1)

where MHP:= Minimum hops to common parent, and HPR:= Hops from common parent to Root of hierarchy. We define the **semantic distance** between tags \(i\) and \(j\) as the semantic distance between the respective WordNet synsets. The steps of the procedure for obtaining a preliminary ontological tag tree starting from WordNet hierarchy are summarized in Algorithm 6, which returns a tree over the set of tags \(\mathcal{T}\).

Once the preliminary tag tree based on WordNet, \(T_w\), is constructed as described, we refine it using a data driven approach based on the co-occurrence statistics of the tags. We describe this below.

### 3.3.2 Co-occurrence based Tag Tree Refinement

The preliminary tag tree is refined by accounting for those tags that strongly co-occur in the corpus but are not linked in the WordNet based tag tree. To achieve this,
Algorithm 6. Constructing Preliminary Tag Tree using WordNet hierarchy

Input:
- WordNet based hierarchy $H_T$ for given set of tags $T$ representing semantic relationships such as “is-a” or “part-of”. $H_T$ is directed graph and can be disconnected.
- Pair-wise semantic distances for tags in $T$ from (3.1)

Breaking cycles:
Obtain $H_{\text{undirected}}$ as the undirected version of $H_T$
Loop While cycles exist in $H_{\text{undirected}}$
  - Find one cycle in $H_{\text{undirected}}$: $E_{\text{cycle}}$=edges in the obtained cycle
  - Remove the edge $e \in E_{\text{cycle}}$ with largest distance (3.1) from $H_{\text{undirected}}$
EndWhile

Connecting disjoint components in $H_{\text{undirected}}$:
Loop While $H_{\text{undirected}}$ is disconnected
  - Obtain pair-wise distances between tags from (3.1)
  - Set distances between tags in same component, to $\infty$
  - Connect tags with least distance. These will form edges between the disjoint components.
EndWhile

Output: tree $T_W$

we first define the similarity score of two tags using Jaccard Similarity:

Definition 1. [Jaccard Similarity]: The Jaccard Similarity $J(s_1, s_2)$ between two sets $s_1, s_2$ is defined to be equal to

$$
\frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}.
$$

Let $T = \{t_j\}_{j=1}^N$ be the set of tags and $R = \{r_l\}_{l=1}^{|R|}$ be the set of resources. For each tag $t_j$ we construct the set $s_j = \{r : r \in R$ and $B(r, j) = 1\}$. Let $J_T$ be an $N \times N$ matrix such that $J_T(i, j) = J(s_i, s_j)$. We will call $J_T$ the Jaccard Matrix of $T$.

We augment the preliminary tag tree based on WordNet, $T_W$, to construct a Similarity Graph as follows. We construct the graph with vertex set $V = \{v_1, \ldots, v_N\}$ where the $v_j$ corresponds to the tag $t_j$. We start with $T_W$, which is a tree on $V$ constructed using WordNet as described in Algorithm 6. Additionally, given a threshold $\tau$, $0 \leq \tau \leq 1$, we join $v_i, v_j$ with an edge iff $J_T(i, j) \geq \tau$, and the edge weight of edge $(v_i, v_j)$ is set to be $J_T(i, j)$. We call the resulting graph $G_T$, the Similarity Graph of $T$ since it captures the tree based on semantic similarity, and additional edges based on corpus based jaccard
similarity. Fig. 3.3 shows an illustrative example for obtaining Similarity Graph based on a threshold \( \tau \). Given the similarity graph \( G_{\mathcal{S}} \), the objective of the refinement stage is to find a tree in the space of spanning trees of the similarity graph \( G_{\mathcal{S}} \) which minimizes a defined objective function. Below we define and motivate two different objective functions based on corpus statistics, for tag tree construction.

1. **Weighted Average Hops (WAH):**

\[
\sum_i \sum_{j, j < i} J_{\mathcal{S}}(i, j)d_{i,j},
\]

(3.3)

where \( d_{i,j} \) represents the number of hops between tag \( t_i \) and tag \( t_j \) in the tag tree. The motivation for such a score is that it is lower when tags \( i \) and \( j \) with high \( J_{\mathcal{S}}(i, j) \) are separated by fewer hops as compared to tags with low \( J_{\mathcal{S}}(i, j) \). The above objective function is equal to the sum of the pair-wise hops between all pairs of tags weighted by the corresponding Jaccard similarity \( J_{\mathcal{S}}(i, j) \).
the sum by \( \sum_{i,j<i} J_T(i,j) \) would be equal to the weighted average number of pair-wise hops where the weights are normalized Jaccard similarities. Since the value \( \sum_{i,j<i} J_T(i,j) \) is a constant for a given set of tags \( \mathcal{T} \), we have removed the scaling factor from the objective function. For a general graph \( \mathcal{G}_T \), the problem of minimizing the weighted average number of hops has been established to be an NP hard problem [41].

2. **Similarity Approximation (SA):**

\[
\sum_{i} \sum_{j, j<i} w_{i,j} | J_T(i,j) - S_T(i,j) |,
\]

where \( S_T(i,j) \) represents the similarity between tags \( t_i \) and \( t_j \) estimated using tag tree \( T \). A very close problem is that of approximating a given distance matrix through spanning trees, which has been established to be NP hard [30]. The objective function in (3.4) is the weighted L1 norm of the difference between the Jaccard Matrix \( J_T \) and the Estimated Similarity Matrix \( S_T \). The weights \( w_{i,j} \) are taken to be the co-occurrence counts of tags \( t_i \) and \( t_j \) and are useful to establish relative importances between different pairs of tags in the objective function. While \( S_T(i,j) \) can be calculated in several ways for a given tag tree \( T \), we define \( S_T(i,j) \) as

\[
S_T(i,j) = \prod_{e \in \mathcal{P}_{i,j}} S(e),
\]

where \( \mathcal{P}_{i,j} \) is the path in tag tree \( T \) connecting tags \( t_i \) and \( t_j \) and \( S(e) \) is equal to the Jaccard similarity between the tags that edge \( e \) connects. Such a definition for \( S_T(i,j) \) ensures that it lies between 0 and 1 and no rescaling is required in order to compare \( S_T(i,j) \) values with \( J_T(i,j) \).

Note that the trees in (3.3) and (3.4) are constrained to be spanning trees over the
Similarity Graph $G_T$. The local search based approach to minimize either of the above objective functions is described next.

### 3.3.3 Optimization based on Local Search

Given the Similarity Graph $G_T$ for a set of tags $T$, our objective is to construct a spanning tree on $G_T$ such that the defined objective function is minimized. Since finding spanning trees of $G_T$ that optimally minimize either (3.3) or (3.4) is a hard problem, we propose an approach to obtain local optimum through the local search paradigm.

**Local Search** – Local search algorithms provide a local optimum to an optimization problem. This is done by moving from one solution to another, in the search space of candidate solutions.

For the problem of constructing an ontological tag tree, we define a simple edge-exchange based neighborhood on the space of spanning trees of the graph $G_T$ as follows. Given two spanning trees $T_1$, $T_2$ we say that $T_2$ is a neighbour of $T_1$ iff it can be obtained from $T_1$ by the following process:

1. Pick an edge $e_1 \in G_T \setminus T_1$ and add it to $T_1$.

2. In the (unique) cycle thus formed in $T_1$ containing $e_1$ pick the edge, say $e_2$ with minimum weight (i.e., jaccard similarity of the tags $e_2$ connects). Remove $e_2$ from $T_1$.

Starting from a spanning tree $T_0$ of $G_T$ as an initial solution, we explore all neighbors of $T_0$ to determine which neighbor minimizes the defined objective function. The winning neighbor is then considered as the next solution and its neighbors are explored until no further benefit is seen in the objective function. The steps of the local search based ontological tree construction are listed in Algorithm 7. The output is the locally optimal tag tree $T_{opt}$. Note that $T_0$ is taken as $T_W$ as obtained from Algorithm 6. Fig. 3.4 show
one iteration of the proposed local search based approach for the objective function in (3.3).

**Algorithm 7. Ontological Tree Construction Algorithm**

**Input:**
- Similarity graph $\mathcal{G}$ for a given set of tags $\mathcal{T}$
- Initial Solution: $T_0$
- Pair-wise Jaccard similarities between tags, i.e., $J(i, j)$ **Initialization:** $S = T_0$

**Loop:**
- $E_{Candidates}$ := set of edges present in $\mathcal{G}$ and not in $S$
  - Add edge $e$ in $S$ to get graph $G$
  - $E_{Cycles}$ := set of edges in the cycle formed in $G$.
  - Remove edge $e'$ with lowest weight (i.e., Jaccard similarity of connecting tags) from $E_{Cycles}$: $e' \neq e$
  - $S_{Neighbor}$ = spanning tree thus formed
  - Calculate objective function at $S_{Neighbor}$

**EndFor**
- Select neighbor giving best objective function as $S_{Next}$
- If $S_{Next}$ improves objective function over $S$
  - $S = S_{Next}$
- Else Stop iterating

**End Loop. Output:** locally optimal spanning tree $T_{opt} = S$

### 3.3.4 Effect of initializing using WordNet

The benefit of using WordNet to initialize the proposed local search based approach is that as compared to random initializations, the former helps achieve a preferred value of the objective function faster. The typical way to attain a lower value of the objective function for an optimization problem as defined in Section 3.3.2 would be to run the local search using randomly constructed tree on the set of tags as initialization, and picking the best tree across several such runs based on the tag tree that has lowest objective function. However this requires running the local search several times which can be large considering that the number of spanning trees on a set of $N$ tags varies as $N^{N-2}$ [19]. The preliminary tree as constructed using WordNet offers a more meaningful initialization to the local search by capturing certain types of relationships between the tags that are dictated by their semantics. Table 3.1 provides the statistics of the objective
Figure 3.4. One iteration of the proposed approach. Eq. (3.3) is utilized to calculate the objective function for neighbors of a tree T. Ties are broken arbitrarily. Since both T1 and T2 have an objective function (3.3) value of 4, we choose T1 and proceed to next iteration.
function of tag trees constructed by running the proposed local search based approach 20 times with random initializations. As can be seen, a single run using WordNet based preliminary tag tree leads to a much better objective function (3.4). Also, the resulting tag tree using WordNet leads to a better performance than the best across 20 runs with random initializations. Note that for Table 3.1, the performance of tag trees is measured using Average Tag Prediction Accuracy as discussed in Section 3.4.

**Table 3.1.** Effect of initializing the proposed local search based approach using WordNet. 30 tags from stock image corpus (described in Section IV) are used.

<table>
<thead>
<tr>
<th></th>
<th>Random initialization</th>
<th>Using WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function ((4) (\times 10^6))</td>
<td>6.4 13.7 47.7</td>
<td>0.6</td>
</tr>
<tr>
<td>Performance (in %)</td>
<td>24.2 43.8 49.2</td>
<td>50.9</td>
</tr>
</tbody>
</table>

### 3.4 Evaluation

In this section, we describe the experimental setup for the construction and evaluation of ontological tag trees. For the evaluation, we define tag prediction and efficient classification tasks as discussed later in this section. The experiments are conducted on two large corpora of images, the details of which are given below.

#### 3.4.1 Datasets

To test the robustness of our approach to build ontological tag trees for domains with varying degrees of tag noise, we use two different image corpora - one from Flickr, composed primarily of user generated content, and one from a professionally curated stock photo agency.

- **Flickr images**: Flickr [35] is a popular image and video hosting website where users can upload images and associate them with annotations such as titles, tags and
descriptions, among others. As Flickr primarily contains user generated content, tags are often noisy, irrelevant to image content or even completely absent. We utilize 500,000 images for training and 100,000 images for testing. All these images are licensed under Creative Commons copyright licenses.

- **Stock images corpus**: To evaluate the proposed approach on less noisy data, we take a corpus of stock photos that are professionally annotated, and hence are accompanied with a variety of accurate annotations - such as keywords, captions, etc. For this corpus, we use the set of keywords to build the ontological tag tree, and refer to them as "tags". We utilize more than 350,000 images for training and close to 70,000 images for testing. The textual captions are used in the efficient classification task as shown in Section 3.4.4.

Training images are used for adapting the WordNet based preliminary tag tree obtained using Algorithm 6 to the given corpus using the local search based approach described in Algorithm 7. Training images are also used for specific required tasks such as training of classifiers. Testing images are used to evaluate the constructed tag trees. There is no overlap between training and test sets.

### 3.4.2 Effect of Local Search based Optimization

We first demonstrate how the proposed local search based approach helps in improving the objective function as defined in Section 3.3. Fig. 3.5 shows the variation of the objective function in (3.3) with the number of iterations on Flickr tag corpus. The objective function value of the WordNet based preliminary tag tree for Flickr corpus before the proposed refinement is 357.2 that becomes 167.1 using the local search based refinement in 68 iterations. Median of the Jaccard similarity values is taken as the threshold $\tau$ for candidate selection in the proposed refinement algorithm. Similar
improvement is observed for the Stock images corpus, for which the objective function values improves from 224.5 to 99 after 23 iterations. For both corpora, the iterations are terminated once no further improvement is observed in the objective function. We describe below the tasks defined to evaluate the constructed tag trees.

![Image](image-url)

**Figure 3.5.** The variation of the objective function (3.3) with the number of iterations for a sample run on Flickr tag corpus.

### 3.4.3 Tag Prediction Task

The tag prediction task is similar to the tag recommendation task in [62] and in [111]. However as outlined in Section 3.1.1, the proposed task does not require manual labeling for the resources (images) to evaluate the proposed tag tree construction approach. In order to do so, we divide the tags associated with a resource into a seen and an unseen set of tags and use the latter to evaluate the predicted tags. We demonstrate this approach through experiments conducted on image corpora. Let an image $i$ in the corpus be tagged with the set of tags $\mathcal{T}_i$, such that $|\mathcal{T}_i| = N_{Tags}$. Assume that out of these $N_{Tags}$ tags, only a subset $\mathcal{T}_{i,\text{seen}}$ are observed, with $|\mathcal{T}_{i,\text{seen}}| = N_{\text{seen}}$. The objective of the tag prediction task is to predict the remaining $(N_{Tags} - N_{\text{seen}})$ tags, i.e., $\mathcal{T}_i \setminus \mathcal{T}_{i,\text{seen}}$. Let $\mathcal{P}_i$ be the set of $(N_{Tags} - N_{\text{seen}})$ tags predicted for image $i$ assuming that $\mathcal{T}_{i,\text{seen}}$ is observed. Note that the prediction assumes the total number of tags for the image, $N_{Tags}$, to be known. Performance of tag prediction can be measured by the Tag Prediction...
Accuracy, defined as follows:

\[
\text{Tag Prediction Accuracy} = \frac{|\{\mathcal{T} \setminus \mathcal{T}_{i,\text{Seen}} \cap \mathbb{P}_i\}|}{|\{\mathcal{T} \setminus \mathcal{T}_{i,\text{Seen}}\}|}.
\]  (3.6)

We now discuss the approach we follow to obtain the set of predicted tags \(\mathbb{P}_i\) when the set of tags \(\mathcal{T}_{i,\text{Seen}}\) is seen, by utilizing a given ontological tag tree.

**Utilizing Ontological Tag Tree for Tag Prediction**

Consider the tag tree \(T\), built over the set of \(\mathcal{T}\) tags in a corpus. For image \(i\) with \(N_{\text{Seen}}\) number of seen tags, each tag \(t \in \{\mathcal{T} \setminus \mathcal{T}_{i,\text{Seen}}\}\) is given a proximity score \(s_t\) based on its proximity from the seen tags, as per \(T\). Specifically,

\[
s_t = \Sigma_{t' \in \mathcal{T}_{i,\text{Seen}}} \text{dist}(t, t'),
\]  (3.7)

where \(\text{dist}(t, t')\) is the distance between tags \(t\) and \(t'\) in \(T\) calculated as shown in Section 3.4.3. A lower proximity score for a tag \(t\) indicates that it is closer in a cumulative sense to the set of observed tags \(\mathcal{T}_{i,\text{Seen}}\). The tags are ordered in the increasing order of \(s_t\), and the first \((N_{\text{Tags}} - N_{\text{Seen}})\) tags, i.e. those corresponding to the least values of \(s_t\), are chosen as the set of predicted tags \(\mathbb{P}_i\).

**Methods Compared**

We compare the following methods in the tag prediction task:

1. **Random**: As the name suggests, this baseline method randomly picks \((N_{\text{Tags}} - N_{\text{Seen}})\) tags from the set \(\mathcal{T} \setminus \mathcal{T}_{i,\text{Seen}}\).

2. **WordNet**: This baseline approach uses the semantics based ontological tag tree constructed from WordNet hierarchy using the procedure described in Algorithm
1. The edge weights are assigned to be semantic distances as obtained from [101].

3. **Google Similarity Distance**: Google Similarity Distance [24] has been used to construct tag graphs in applications such as tag ranking [75]. As mentioned in Section 3.1.1, a threshold is used to discard certain edges in tag graphs. We choose a threshold such that for a tag graph with \( N \) nodes (or tags), there are exactly \((N - 1)\) edges remaining, so that the tag graph thus formed has same number of edges and space requirement as the tag tree learnt from proposed approach. Edge weights for the tag graph are taken to be the Google Similarity Distance as defined in [24].

4. **LS Weighted Average Hops (LS-WAH)**: Here we construct a tag tree using the proposed local search based approach, to minimize Weighted Average Hops (3.3) in Section 3.3.2. If an edge exists between tags \( t_i \) and \( t_j \) then the weight of the edge connecting them is given to be \((1 - J_{S}(i, j))\).

5. **LS Similarity Approximation (LS-SA)**: This tag tree is constructed using the proposed approach with objective corresponding to Similarity Approximation as outlined in (3.4) in Section 3.3.2. The edge weights are assigned as in Method 4 above.

6. **Symmetric sum based**: In order to compare the performance of the proposed tag tree construction approaches with that of other tag recommendation approaches that do not use tag trees or tag graphs or any visual features, we also provide the performance of the symmetric sum based approach as proposed in [111]. Note that the space required to store the pair-wise similarities in [111] is \( O(N^2) \) while the proposed tag tree construction requires \( O(N) \) space to store the tag tree.
The prediction task is performed using the approach described in section 3.4.3. For methods numbered 2, 3, and 4 above, \( dist(t_i, t_j) \) as required in (3.7) is calculated by adding distances of edges in path connecting tags \( t_i \) and \( t_j \). For LS-SA method, \( dist(t_i, t_j) \) is defined as \( \{1 - S_T(i, j)\} \) where \( S_T(i, j) \) is calculated for an ontological tag tree \( T \) as per (3.5). This is because the tag tree construction approach for LS-SA method utilizes product based similarity approximation (3.5) and so it is appropriate to use same approach to estimate similarities based on a tree, and hence to calculate \( dist(t_i, t_j) \).

Similarly, for Symmetric sum based method [111], \( dist(t_i, t_j) \) is defined as \( \{1 - J_T(i, j)\} \) where \( J_T(i, j) \) is the jaccard similarity between tag \( t_i \) and \( t_j \) as defined in Section 3.3.2. Note that this makes (3.7) similar to the sum based scoring approach in [111].

**Flickr corpus:** We begin by choosing the top 150 most popular tags in a sample of Flickr images. After selecting only those tags that also occur in the WordNet database, we are left with 117 tags. These comprise the set \( \mathcal{T} \). The total number of tags in an image, varies from 0 to 6 for most Flickr images. Since we need at least one tag to be seen and at least one to be predicted, we vary \( N_{Tags} \) from 2 to 6. For each value of \( N_{Tags} \), test images are selected which contain exactly \( N_{Tags} \) tags. For each such image \( i \), all combinations of \( N_{Seen} \) tags are selected to comprise the observed tag set \( \mathcal{T}_{i,Seen} \). Predictions are made as discussed in Section 3.4.3 and performance of tag prediction task is measured using (3.6). \( N_{Seen} \) is varied from 1 through \((N_{Tags} - 1)\). Given values for \( N_{Tags} \) and \( N_{Seen} \), the Average Tag Prediction Accuracy is the Tag Prediction Accuracy using (3.6) averaged across test images. We define Overall Tag Prediction Accuracy as the mean of Average Tag Prediction Accuracy across all combinations of \( N_{Tags} \) and \( N_{Seen} \).

The Overall Tag Prediction Accuracies for the various methods compared are shown in Table 3.2. Tables 3.3 through 3.8 show the Average Tag Prediction Accuracies for various combinations of \( N_{Tags} \) and \( N_{Seen} \) for the individual methods discussed in
Section 3.4.3 for Flickr corpus. For comparison, the Average Tag Prediction Accuracies marginalized over $N_{Tags}$ are shown in Fig. 3.6. Note that Google Distance refers to the method corresponding to Google Similarity Distance as outlined above. It can be seen that the LS-SA method outperforms all others and has its performance very close to that of the Symmetric sum based approach [111]. It should be noted that the latter utilizes pair-wise similarities between tags as derived from the corpus and thus has space requirement of $O(N^2)$ which as discussed in Section 3.1.1 is very high for large number of tags, i.e., for large $N$. Compared to this, the proposed approach has only $O(N)$ space requirement. We will discuss in Section 3.6 the reason why LS-SA leads to construction of a tree that outperforms the LS-WAH method. Google distance based method has performance close to that of LS-WAH while the tag tree based on WordNet hierarchy does not work very well for tag prediction task. As expected, random tag prediction performs the worst.

**Table 3.2.** Overall Tag Prediction Accuracy (%) for various methods for tag prediction task on Flickr and Stock images corpora.

<table>
<thead>
<tr>
<th>Tag Prediction Method</th>
<th>Flickr corpus</th>
<th>Stock images corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random tag prediction</td>
<td>2.29</td>
<td>13.21</td>
</tr>
<tr>
<td>WordNet</td>
<td>7.95</td>
<td>22.67</td>
</tr>
<tr>
<td>Google Similarity Distance</td>
<td>34.02</td>
<td>40.05</td>
</tr>
<tr>
<td>LS - WAH</td>
<td>31.53</td>
<td>48.06</td>
</tr>
<tr>
<td>LS - SA</td>
<td>44.52</td>
<td>50.86</td>
</tr>
<tr>
<td>Symmetric sum based</td>
<td>47.13</td>
<td>54.71</td>
</tr>
</tbody>
</table>

We provide examples of some test images from the Flickr corpus that had $N_{Tags}=5$ and $N_{Seen}=2$. Fig. 3.12 shows a few test images for which LS-SA method made 100% correct predictions. Fig. 3.13 shows test images that gave 0% tag prediction accuracies. The corresponding sets of seen ($\mathcal{T}_i, Seen$), unseen ($\mathcal{T}_i \setminus \mathcal{T}_i, Seen$), and the predicted tags ($\mathcal{P}_i$) are also provided.
Figure 3.6. Average Tag Prediction Accuracies marginalized over $N_{\text{Tags}}$ for various methods for the tag prediction task on Flickr corpus. Note that Google Distance refers to the method corresponding to Google Similarity Distance as outlined in Section 3.4.3.

Table 3.3. Average Tag Prediction Accuracies (in %) obtained using Random method on Flickr corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}} \rightarrow$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{Tags}} \downarrow$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2.67</td>
<td>1.55</td>
<td>0.73</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2.19</td>
<td>1.68</td>
<td>1.01</td>
<td>0.78</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>6.80</td>
<td>5.39</td>
<td>3.36</td>
<td>2.41</td>
<td>0.58</td>
</tr>
</tbody>
</table>

Table 3.4. Average Tag Prediction Accuracies (in %) obtained using WordNet method on Flickr corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}} \rightarrow$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{Tags}} \downarrow$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7.05</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>8.17</td>
<td>2.88</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>11.16</td>
<td>5.27</td>
<td>5.39</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>16.50</td>
<td>10.05</td>
<td>13.10</td>
<td>0.84</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>14.72</td>
<td>10.85</td>
<td>9.46</td>
<td>2.97</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 3.5. Average Tag Prediction Accuracies (in %) obtained using Google Similarity Distance method on Flickr corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}} \rightarrow$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{Tags}} \downarrow$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22.07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>17.01</td>
<td>5.50</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>23.70</td>
<td>16.67</td>
<td>11.66</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>51.51</td>
<td>47.09</td>
<td>45.28</td>
<td>43.15</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>54.55</td>
<td>48.28</td>
<td>44.68</td>
<td>41.85</td>
<td>37.30</td>
</tr>
</tbody>
</table>
Table 3.6. Average Tag Prediction Accuracies (in %) obtained using LS-WAH method on Flickr corpus.

<table>
<thead>
<tr>
<th>$N_{seen}$ → $N_{tags}$ ↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>16.13</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>11.53</td>
<td>5.51</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>14.91</td>
<td>10.96</td>
<td>7.17</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>42.76</td>
<td>42.94</td>
<td>39.69</td>
<td>36.95</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>49.37</td>
<td>51.48</td>
<td>49.00</td>
<td>48.89</td>
<td>45.69</td>
</tr>
</tbody>
</table>

Table 3.7. Average Tag Prediction Accuracies (in %) obtained using LS-SA method on Flickr corpus.

<table>
<thead>
<tr>
<th>$N_{seen}$ → $N_{tags}$ ↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>25.87</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>20.34</td>
<td>21.65</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>26.09</td>
<td>28.96</td>
<td>26.43</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>52.33</td>
<td>54.79</td>
<td>54.82</td>
<td>53.16</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>59.57</td>
<td>61.78</td>
<td>62.10</td>
<td>60.81</td>
<td>59.07</td>
</tr>
</tbody>
</table>

Table 3.8. Average Tag Prediction Accuracies (in %) obtained using Symmetric sum method on Flickr corpus.

<table>
<thead>
<tr>
<th>$N_{seen}$ → $N_{tags}$ ↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>25.62</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>21.19</td>
<td>22.17</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>28.69</td>
<td>30.88</td>
<td>28.12</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>55.6</td>
<td>57.88</td>
<td>57.12</td>
<td>54.27</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>62.77</td>
<td>65.48</td>
<td>67.05</td>
<td>66.35</td>
<td>63.75</td>
</tr>
</tbody>
</table>

Stock images corpus: As in the Flickr corpus, we first select the set of most popular tags from the corpus of stock images that are also in WordNet. This produces a set $\mathcal{T}$, of 30 tags. Tables 3.9 through 3.14 show the Average Tag Prediction Accuracies for various combinations of $N_{tags}$ and $N_{seen}$ for the tag prediction methods discussed in Section 3.4.3. The Average Tag Prediction Accuracies marginalized over $N_{tags}$ are shown in Fig. 3.7, plotted as a function of $N_{seen}$. Note that if $N_{tags}$ is kept constant, increasing $N_{seen}$ reduces the number of unseen tags (i.e., $\mathcal{T}_i \setminus \mathcal{T}_{i,seen}$), thus reducing the
random chance of predicting a correct unseen tag from $\mathcal{T} \setminus \mathcal{T}_{\text{Seen}}$. As a result of this phenomenon, a drop in performance for random prediction and other methods can be seen with increasing $N_{\text{Seen}}$.

Figure 3.7. Average Tag Prediction Accuracies marginalized over $N_{\text{Tags}}$ for various methods for the tag prediction task on Stock images corpus.

Table 3.9. Average Tag Prediction Accuracies (in %) obtained using Random method on Stock images corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{Tags}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>5.28</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>11.3</td>
<td>5.8</td>
<td>4.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>14.9</td>
<td>10.1</td>
<td>6.8</td>
<td>6.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>19.1</td>
<td>12.9</td>
<td>10.1</td>
<td>9.5</td>
<td>3.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>21.5</td>
<td>16.1</td>
<td>14.6</td>
<td>10.1</td>
<td>13.1</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>20.7</td>
<td>17.8</td>
<td>11.8</td>
<td>12.7</td>
<td>6.4</td>
<td>9.6</td>
<td>2.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>28.6</td>
<td>25.5</td>
<td>23</td>
<td>19.8</td>
<td>19.4</td>
<td>12.4</td>
<td>7</td>
<td>5.7</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>28.3</td>
<td>27.6</td>
<td>23.5</td>
<td>23.5</td>
<td>17.7</td>
<td>13.6</td>
<td>16.7</td>
<td>7.6</td>
<td>7.4</td>
</tr>
</tbody>
</table>

The results indicates that the proposed local search paradigm based approach has successfully adapted the ontological tag tree obtained from WordNet to the Flickr or stock images corpus. We can obtain performance better than or close to that of other tag prediction approaches, while having several orders of savings in the space requirement. We next describe the second data-driven task used for evaluation of tag trees.
### Table 3.10. Average Tag Prediction Accuracies (in %) obtained using WordNet method on Stock images corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}} \rightarrow N_{\text{Tags}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>8.9</td>
<td>3.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>14.2</td>
<td>10.9</td>
<td>3.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>21.3</td>
<td>16.9</td>
<td>14.7</td>
<td>8.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>28.4</td>
<td>22.8</td>
<td>22.4</td>
<td>16.7</td>
<td>9.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>32.3</td>
<td>28.4</td>
<td>24.8</td>
<td>22.4</td>
<td>12.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>35.7</td>
<td>33.3</td>
<td>32.7</td>
<td>30.7</td>
<td>27.6</td>
<td>23.8</td>
<td>13</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>36.5</td>
<td>34.5</td>
<td>32.7</td>
<td>31.9</td>
<td>28.7</td>
<td>26.3</td>
<td>23.4</td>
<td>15.5</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>37.4</td>
<td>35.2</td>
<td>32.9</td>
<td>30.8</td>
<td>28</td>
<td>26.1</td>
<td>22.3</td>
<td>18.5</td>
<td>9.9</td>
</tr>
</tbody>
</table>

### Table 3.11. Average Tag Prediction Accuracies (in %) obtained using Google Similarity Distance method on Stock images corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}} \rightarrow N_{\text{Tags}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>11.2</td>
<td>8.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>26.6</td>
<td>21.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>32.3</td>
<td>39.8</td>
<td>39.7</td>
<td>35.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>36.6</td>
<td>43.9</td>
<td>46.5</td>
<td>46.5</td>
<td>44.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>39.4</td>
<td>44.8</td>
<td>47.8</td>
<td>49.5</td>
<td>49.5</td>
<td>48.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>43.6</td>
<td>45.2</td>
<td>47.3</td>
<td>48.5</td>
<td>49.1</td>
<td>49</td>
<td>48.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>41.9</td>
<td>43.5</td>
<td>44.4</td>
<td>45.4</td>
<td>45.9</td>
<td>45.8</td>
<td>45.2</td>
<td>44.9</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>41.6</td>
<td>42.7</td>
<td>43.2</td>
<td>42.8</td>
<td>42.6</td>
<td>42</td>
<td>40.7</td>
<td>39.5</td>
<td>38.7</td>
</tr>
</tbody>
</table>

### Table 3.12. Average Tag Prediction Accuracies (in %) obtained using LS-WAH method on Stock images corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}} \rightarrow N_{\text{Tags}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>14.6</td>
<td>15.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>20.5</td>
<td>22.6</td>
<td>22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>36.2</td>
<td>33.2</td>
<td>32.8</td>
<td>31.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>49</td>
<td>47.6</td>
<td>45.2</td>
<td>43.3</td>
<td>41.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>59.6</td>
<td>59.4</td>
<td>58.7</td>
<td>57.1</td>
<td>54.4</td>
<td>51.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>62.3</td>
<td>63.5</td>
<td>63.2</td>
<td>62.8</td>
<td>61.4</td>
<td>58.2</td>
<td>54.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>56.6</td>
<td>59.7</td>
<td>59.2</td>
<td>58.7</td>
<td>57.9</td>
<td>56.3</td>
<td>52.8</td>
<td>49.5</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>53.5</td>
<td>57.6</td>
<td>57.4</td>
<td>56.8</td>
<td>56.1</td>
<td>54.7</td>
<td>52.9</td>
<td>48.5</td>
<td>44.4</td>
</tr>
</tbody>
</table>
Table 3.13. Average Tag Prediction Accuracies (in %) obtained using LS-SA method on Stock images corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}}$ $\rightarrow$ $N_{\text{Tags}}$ ↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>13.4</td>
<td>11.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>28.4</td>
<td>27</td>
<td>21.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>44.5</td>
<td>41.9</td>
<td>38.5</td>
<td>35</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>53.3</td>
<td>53.3</td>
<td>51.3</td>
<td>49.5</td>
<td>46.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>61.7</td>
<td>63.2</td>
<td>62.6</td>
<td>60.8</td>
<td>58.3</td>
<td>56</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>64.3</td>
<td>64.8</td>
<td>65.8</td>
<td>65.3</td>
<td>63.1</td>
<td>60.2</td>
<td>57.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>59</td>
<td>60.2</td>
<td>60.8</td>
<td>61.1</td>
<td>60.8</td>
<td>58</td>
<td>55.3</td>
<td>52.3</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>55.7</td>
<td>57.7</td>
<td>57.9</td>
<td>57.8</td>
<td>57</td>
<td>53.7</td>
<td>50.6</td>
<td>47.1</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3.14. Average Tag Prediction Accuracies (in %) obtained using the Symmetric sum approach on Stock images corpus.

<table>
<thead>
<tr>
<th>$N_{\text{Seen}}$ $\rightarrow$ $N_{\text{Tags}}$ ↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>16.9</td>
<td>17.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>32.1</td>
<td>32.5</td>
<td>29.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>45.1</td>
<td>47.8</td>
<td>46.4</td>
<td>43.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>52.7</td>
<td>56</td>
<td>56.3</td>
<td>55.5</td>
<td>53.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>62.1</td>
<td>64.6</td>
<td>64.9</td>
<td>63.9</td>
<td>62</td>
<td>59.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>64.5</td>
<td>68</td>
<td>68.1</td>
<td>67.8</td>
<td>65.8</td>
<td>63.1</td>
<td>60.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>60.6</td>
<td>63.9</td>
<td>64.9</td>
<td>64.9</td>
<td>64</td>
<td>61.5</td>
<td>58.5</td>
<td>55.9</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>58</td>
<td>61.5</td>
<td>62.9</td>
<td>63.2</td>
<td>63.1</td>
<td>61.6</td>
<td>58.9</td>
<td>55</td>
<td>52.3</td>
</tr>
</tbody>
</table>
3.4.4 Efficient Classification Task

We consider the problem of efficiently associating tags with resources based on the resource content. For domains where it is feasible to extract features from the resource and to train concept detectors, we show how tag trees can be utilized to determine which concept detectors should be applied on a given test resource. We demonstrate this using annotated image corpora and utilize different modalities to represent the content of a given resource. Given a test image, \( i \) from the corpus without any associated tags or keywords, we predict which of the \( N \) tags are applicable to the image. By letting each of the \( N \) tags correspond to a category or class, this is equivalent to a multi-class, multi-label classification task. Let us assume that one-vs-all binary classifiers are available for each tag class - these take the image instance \( i \) as input and can predict \( P(j \mid i) \) where class \( j \) corresponds to tag \( t_j \). We can use this to predict whether or not a tag \( t_j \) should be associated with image \( i \) based on whether \( P(j \mid i) \) is greater than an appropriate threshold \( \theta \) or not.

The naive approach to predict all tags applicable to \( i \) would be to test each of the \( N \) classifiers on \( i \) and accumulate those tags for which the predictions by the corresponding classifiers exceed \( \theta \). Some works such as [70] propose techniques to make classification of images faster. However in order to do this, these works require re-training of classifiers and cannot utilize existing classifiers for each tag. Contrary to these, in the proposed approach, we can utilize pre-trained classifiers corresponding to different tags. Other works such as the image annotation application in [132] associate a test image with tags based on applying all concept detectors on the test image and then combining the predictions. However such approaches require applying all \( N \) classifiers to a given test image and are hence inefficient for a large number of tags. This process can be made more efficient by using the ontological tag tree on the set of tags. Once certain labels
have been predicted for an image $i$, one can utilize the tree structure to decide which tags are more or less likely to be associated with the image. Thus the choice of the classifiers to test next on the image $i$, can be made in a more efficient manner. Therefore, a tag tree that captures the relations between the tags more effectively, is expected to lead to more efficient performance by reaching the correct number of predicted tags with fewer number of binary classifications performed. The efficient classification task is formulated to measure the classification efficiency in such a setting.

This evaluation task is formulated as follows. Given $N$ binary classifiers, the $j^{th}$ classifier predicts probability $P(j | i)$ of tag $t_j$ being associated with a test image $i$. $K$ binary classifications are performed for image $i$ and the set of tags that are predicted positive among those $K$ classifications, comprise the set of predicted tags $P_{i,K}$ for image $i$ for given $K$. The ground truth set of tags that are associated with image $i$ as per the corpus, is denoted $\mathcal{T}_i$. We define performance of the classification task based on the Tag Recall as defined below.

$$\text{Tag Recall} = \frac{|\mathcal{T}_i \cap P_{i,K}|}{|\mathcal{T}_i|}$$

Based on the above definition, the Tag Recall is monotonically increasing with $K$: as more than $K$ classifications are performed, the cardinality of the set of predicted labels can remain constant or increase, leading to a non-decreasing value for the Tag Recall.

The most naive way of choosing the order in which the $N$ classifications are performed, would be to choose each binary classifier randomly. A more sophisticated approach can be adopted by choosing the classifiers based on the decreasing order of their class priors in the training corpus, i.e., choosing the classifier corresponding to the most frequently occurring tag first, followed by the next popular tag, and so on. We refer to these methods as Random, and Prior-based respectively. Below we discuss how the classifiers can be chosen based on their priors and a given ontological tag tree.
Utilizing Ontological Tag Tree for Classification

Intuitively, the procedure for using an ontological tag tree to decide how to choose the classifiers, is similar to the approach for the tag prediction task. For image $i$, once certain tags are predicted as present, the tags in their proximity (as per the tag tree) have a higher chance of being present too, and the corresponding classifiers should be tested sooner. Similarly, the predicted absence of tags brings down the chance that tags in their proximity will be present. We define below a priority score for the tags that have not been tested for, based on their priors and the predictions for the tags that have been tested for. For image $i$ and tag tree $T$, we define

$$priority(j) = P_{\text{prior}}(j) + \sum_{c \in C_{\text{tested}}} (P(c | i) - \theta) \times (1 - dist'(c, j))$$  (3.9)

where $dist'(c, j)$ is the distance $dist(c, j)$ as defined in Section 3.4.3 for various methods, normalized with respect to its maximum value such that $dist'(c, j)$ varies between 0 and 1. $P_{\text{prior}}(j)$ is the prior probability of tag $t_j$, and $C_{\text{tested}}$ is the set of tags that image $i$ has been tested for. Algorithm 8 utilizes (3.9) and presents the algorithm employed to decide the order of classifications for each image $i$. For a given image $i$ and given $K$, the set of labels predicted as present, i.e., $P_{i,K}$ can be obtained using Algorithm 8.

**Flickr corpus:** We use the same set of 117 Flickr tags as described in Section 3.4.3. In order to train image classifiers 500,000 Flickr training images are used. Each tag $t_j$ is treated as a class. Positive training images are those that associated with the corresponding tag $t_j$ and negative training are those that are not. We use a ratio of 1 : 10 for number of positive instances to number of negative instances for each class. SIFT features [77] are used to train binary SVM classifiers for each class. We provide results for the Random and Prior-based methods as discussed in Section 3.4.4, and for methods 2 through 6 described in Section 3.4.3 using the approach outlined in Section 3.4.4. For $\theta = 0.5$, 

Algorithm 8. Algorithm for Efficient Classification by using Ontological Tag Tree

**Input:**
- $C_j$: classifier for label $j$ corresponding to tag $t_j$ ($\forall j \in \mathcal{T}$), that can provide $P(j \mid i)$ for image $i$; threshold $\theta$
- $P_{\text{prior}}(j)$: prior probability for tag $t_j$
- $\text{dist}'(t, t')$: normalized inter-tag distances calculated based on given tag tree as outlined in Section 3.4.4
- $K$: number of binary classifications to perform

**Initialize:** $L_{\text{tested}} = [\ ]$

**Loop While** $|L_{\text{tested}}| \leq K$
- Calculate priority score for all tags in $(\mathcal{T} \setminus L_{\text{tested}})$ using (3.9)
- Test classifier $C_j$ corresponding to tag $t_j$ with highest calculated priority score to get $P(j \mid i)$.
- $L_{\text{tested}} = L_{\text{tested}} \cup \{j\}$.

**EndWhile. Output:** $P_K = \{ j : P(j \mid i) > \theta, j \in L_{\text{tested}} \}$

the Tag Recall observed with varying number of classifiers tested, i.e., $K$, are shown in Fig. 3.8. Note that [111] does not utilize the symmetric similarities for an application as proposed in Section 3.4.4, however we utilize the symmetric jaccard similarities as per Algorithm (8) and compare against our approach for an understanding of the trade-off between performance and space requirement. It is clear that for a given $K$, our approach outperforms the tag tree constructed from WordNet, the tag graph using Google Distance, and the ones that select classifiers randomly or solely based on prior. Performance of the LS-SA method is very close to that of the Symmetric sum method despite having several orders of savings in the space requirement as discussed in Sections 3.1.1 and 3.4.3. For Tag Recall of around 4.4%, the tag tree based on proposed approach (LS-SA) requires only 11 classifications compared to 21 for LS-WAH, 31 for Google Distance based, 51 for WordNet, 36 Prior based, and 61 when the classifiers are randomly selected. These correspond to 91%, 182%, 363%, 227%, 454% additional classifications used by these methods respectively, compared to the proposed method with Similarity Approximation (LS-SA). The WordNet based method performs worse than the Prior-based method, implying that the using semantics based distance with priors is less efficient than using priors alone. Visual-feature based image classification is a hard problem and the Tag Recall for 117 classes even after all 117 classifiers are tested, is observed to be less than
10%. This is also a result of the individual classifiers and of the chosen threshold \( \theta \). Note that the Tag Recall (3.8) is different from recall of a class. For \( \theta = 0.5 \), when all the classifiers are tested, the average recall over all classes is 10.5\% and the average precision is 15.9\%.

A more lenient (i.e., lower) \( \theta \) would lead to higher Tag Recall as can be seen in Fig. 3.9. \( \theta = 0.3 \) corresponds to an average recall of 37.7\% and average precision of 4.9\%. A lower \( \theta \) also implies less relative weight given to \( P_{\text{prior}}(j) \) in (3.9) and as a result, the WordNet method performs worse than even the Random method.

**Stock images corpus:** We perform experiments on the same corpus that was described in Section 3.4.3. In order to show the applicability of our approach to different modalities that can be used to represent a given resource, we utilize the textual caption accompanying the image \( i \) to represent the image. A TF-IDF based bag-of-words representation is used for each image, followed by binary SVM classifiers for each class. As in the case of Flickr, we present variation of Tag Recall with \( K \) for various methods for \( \theta = 0.5 \) in Fig. 3.10. \( \theta = 0.5 \) corresponds to an average recall of 73.6\% and average precision of 75.6\%. Our approach is seen to be able to better decide which classifiers to test, for a given \( K \) and as a result, leads to higher Tag Recall for a fixed \( K \). For Tag Recall of around 48\%, the tag tree based on the proposed approach (LS-SA) requires only 7 classifications compared to 9 for LS-WAH, 9 for Google Distance based, 15 for WordNet, 11 for only prior based and 21 when the classifiers are randomly selected. These correspond to 29\%, 29\%, 114\%, 57\%, 200\% additional classifications used by these methods respectively, compared to the proposed method with Similarity Approximation (LS-SA).

It should be noted that the goodness of predictions i.e., the precision and recall obtained when all classifiers are tested, is essentially a property of the set of classifiers, which are the same for various methods used for this task. Based on the experiments,
we have demonstrated that using ontological tag trees constructed through the proposed approach makes selection of classifiers much more efficient as compared to using purely semantics based tag tree or tag graphs built using commonly used techniques. We have also shown that we can achieve a performance almost as high as other approaches despite having several orders less space requirement.

![Figure 3.8. Tag Recall obtained with respect to number of classifications performed for Flickr corpus. $\theta$ is chosen to be 0.5.](image)

![Figure 3.9. Tag Recall obtained with respect to number of classifications performed for Flickr corpus. $\theta$ is chosen to be 0.33.](image)

### 3.5 Robustness Analysis

We provide an analysis of the robustness of the proposed approach for constructing ontological tag trees. For the purpose of this section, we refer to the resource tag data using which the tag tree is constructed, as training data. The resource tag data over
Figure 3.10. Tag Recall obtained with respect to number of classifications performed for Stock images corpus. $\theta$ is chosen to be 0.5.

which the constructed tag tree is tested for evaluation purpose, is referred to as the test data. As seen in Section 3.4, the LS-SA method has consistently outperformed LS-WAH across different corpora and for both data-driven tasks. Therefore, in this section we only study the robustness of the LS-SA method, i.e., the proposed local search based approach using Similarity Approximation based objective function (3.4). In order to evaluate the constructed tag trees under various scenarios, we provide evaluation using tag prediction task as detailed in Section 3.4.3. Annotated image corpora are used for this purpose. We study the robustness of proposed approach with respect to label noise, difference between training and test data, and the size of training data. These are discussed in detail below.

3.5.1 Robustness to Label Noise in Training Data

As described in Section 3.1, a vast majority of the user generated data available over the Internet has noisy labels (tags) associated with the resources. We study the effect of different degrees of label noise on the tag tree constructed from such a noisy corpus. Fundamentally, we attempt to understand how robust the proposed tag tree construction approach is, to different degrees of label noise. We also attempt to answer questions such as - how much noise is too much for tag tree construction?

We work with stock image corpus since stock images are professionally curated
Figure 3.11. Overall Tag Prediction Score (in %) obtained by proposed approach in Section 3.4.3. The training set used to construct an ontological tag tree is formed by using certain number of stock images, with A) Flickr images, or B) noisy images, or C) none. Testing of the constructed tag tree is done on images of stock photos only. For comparison, the Overall Tag Prediction Accuracies for Random method, and WordNet method (as outlined in Section 3.4.3) are also provided.

and have little to no label noise, thus providing a better control on the amount of noise in the experimental data. We select top 150 tags (keywords) from this corpus and remove those that do not occur in Flickr or in WordNet. This gives a set $\mathcal{T}$ of 108 tags. Each stock image has on an average 3 tags amongst $\mathcal{T}$. A total of 100,000 stock images are used in training set and 50,000 stock images in the test set. In order to simulate varying degrees of label noise in training set, we replace stock images in the training data with artificially created images with noisy tags. The total number of training images (stock images and noisy images) is kept constant at 100,000. The artificially created noisy images are defined as images having exactly 3 randomly chosen tags from $\mathcal{T}$. The test set is not varied. Fig. 3.11 shows the variation of Overall Tag Prediction Accuracy (%) as defined in Section 3.4.3, with number of images that were from stock images in the training set. It can be seen that even with 87.2% noisy images in the training set, the performance of the constructed tag tree (39%) is very close to the performance when there are no noisy images at all (40%). Even when 99.2% of the images in training set are noisy, the Overall Tag Prediction Accuracy is 31.4%. An explanation for such performance even
at high levels of noise is that the noisy images have uniformly random distribution of tags. The overall effect of adding noisy images to a corpus can be understood as adding certain intensity of white noise to the co-occurrence counts between tags. Unless the noise intensity dominates the corpus statistics, the relative order of pair-wise connections between tags remains fairly unchanged. In other words, since the noisy images have no strong biases towards specific tag-pairs, the performance of tag trees constructed using such a hybrid training set is not severely affected.

3.5.2 Robustness to Difference between Training and Test Data

In several machine learning applications, the data over which a model is trained has certain amount of distributional difference as compared to the data over which it is tested. Looking at the construction of tag tree as model training, the degree to which a tag tree will be effective on a test set is a function of the difference between the test set and the training set using which the tag tree is constructed. Here we study how the performance of a tag tree varies for different degrees of difference between the training and the test sets. We utilize the same stock images corpus as used in Section 3.5.1. In order to vary the difference between training and test sets, we replace varying number of stock images in training set with images from Flickr corpus (Section 3.4.1). The total number of images (stock images and Flickr images) used for training of tag tree is kept constant at 100,000. Fig. 3.11 shows the variation of Overall Tag Prediction Accuracy (%) with number of images that were from stock images in the training set. It is interesting to note that for the same number of stock images, adding completely noisy images leads to better performance than adding Flickr images. For instance, for the case when 99.2% of training images are from Flickr, performance of the constructed tag tree (23.1%) is seen to be substantially worse than when 99.2% of training images are noisy (31.4%). The reason for this is that the tag distribution in Flickr corpus is not
random, unlike in the case of noisy images. As a result, there are strong relationships between tags in Flickr corpus that are absent in stock image corpus. Constructing a tag tree on such a hybrid corpus attempts to capture corpus statistics of varying numbers of Flickr images and stock images, and in the process, is less effective at capturing the relationships that were present in the test data that has only stock images.

3.5.3 Size of Training Data

In this section, we study the effect of size of training set on construction of tag trees. Fig. 3.11 shows the variation of Overall Tag Prediction Accuracy (%) with number of stock images in the training data. Here the size of training set is equal to the number of stock images in it and there are no Flickr images or noisy images. The performance of constructed tag tree improves with size of training data and almost saturates after certain number of training images. This is quite similar to the variation in performance of most machine learning models with the size of data over which they are trained. The performance of constructed tag tree from only 800 stock images is very close to the performance when 100,000 images are used. Even when only 100 stock images are used, the constructed tag tree performs much better than the tag trees using 100 stock images with 99900 Flickr or noisy images. Based on above, one can conclude that for the construction of ontological tag trees, it is better to use fewer training images than a larger set which is noisy or dissimilar to the test set.

3.6 Discussion

In this section, we provide a discussion on the key insights and observations made during the course of study on ontological tag trees.

As outlined in Section 3.3, the motivation for the objective function (3.3) in the proposed approach is that in order to get a lower value for (3.3), it is more important to
Figure 3.12. Example images where LS-SA method gave 100% tag prediction accuracy when the first two tags were seen and the next three were unseen and predicted. The Flickr owner and photo ids of these images are (dougtone@7975042008) , (elchupacabra@7023118527) , (jeffwilcox@159730021) , (daquellamanera@4678084101) , (martinrp@3832812191) respectively.

Figure 3.13. Example images where LS-SA method gave 0% tag prediction accuracy when the first two tags were seen and the next three were unseen and predicted. Also provided are the tags that were predicted by the LS-SA method. The Flickr owner and photo ids of these images are (king-edward@4061393892) , (familymwr@4928996212) , (alejandroerickson@7730525250) , (wwarby@5145467790) , (1968-dodge-charger@5507716438) respectively. Seen: ($T_i,\text{Seen}$); Unseen: ($T_i \setminus T_i,\text{Seen}$); Predicted: ($P_i$) as per Section 3.4.3.
Figure 3.14. Variation of number of iterations with $\tau$

Figure 3.15. Variation of number of candidate neighbors with $\tau$

separate the tags $t_i$ and $t_j$ having high $J_{\mathcal{T}}(i, j)$ with less number of hops, as compared to the tags with low $J_{\mathcal{T}}(i, j)$. However, separating the latter set of tags by fewer hops would also lower the value of (3.3). Since (3.3) attempts to minimize the weighted number of hops between tag pairs, a local search in the space of all possible trees on $\mathcal{T}$ would lead to connecting most tags to a central tag, thus forming a star graph. Such a minimum weighted hops spanning tree problem on an all connected graph can be solved in polynomial time using Gomory-Hu trees [95], however star graphs need not reflect the true relationship between tags in a corpus. In order to ensure that the proposed approach does not induce artificial structures because of bias of the objective function, we constrain the search space to be spanning trees of a suitably constructed Similarity Graph with the help of a threshold $\tau$, as outlined in Section 3.3.2. Experimentally, constraining as above does succeed in ensuring that we do not get a star graph. However upon visual
inspection, the proposed approach yields star subtrees connected through their centers. The objective function (3.4) in comparison has no such bias for shorter trees and as a result, the results for proposed approach using (3.4) outperform those using (3.3). Figures 3.14 through 3.17 show the variation of different aspects of constructed tag trees with $\tau$ for 117 Flickr tags. Fig. 3.15 shows that with increasing $\tau$, the number of tag trees that are eligible candidates in an iteration of the local search, drops drastically for both LS-WAH and LS-SA. As discussed above, a lenient $\tau$ leads to the LS-WAH based local search converging to star graphs which are an artefact of the procedure and not of the true relationships between the tags. As a result, for lower $\tau$, the performance of LS-WAH based approach degrades substantially as can be seen in Fig. 3.17. For large values of $\tau$, the number of candidate neighbors at each iteration of the local search become less. This constrains the local search and prevents it from attaining a lower value for
the objective function (3.4). This leads to a drop in the performance of LS-SA as can be seen in Fig. 3.17. We have chosen a threshold as the median of the pair-wise jaccard similarity values since it offers a convenient trade-off between number of candidates and performance for LS-SA.

At each iteration of the proposed approach, there are $O(N^2)$ neighbors of a tree to explore, where $N = |\mathcal{T}|$. Calculating the pair-wise distances (or hops) and computing (3.3) or (3.4) requires $O(N^2)$ operations per neighbor [39]. Thus, the complexity of the proposed approach, for either objective function, is $O(N^4)$ per iteration. A bottleneck of the proposed tag tree construction approach is the time taken for local search to terminate. Fig. 3.16 shows the variation of the time taken by the proposed local search based approach, with respect to $\tau$. Matlab on a 2.9GHz, 8GB RAM 4 core processor is used. One way to reduce the time taken at each iteration is to reduce the number of neighbors. This is achieved by constraining the solution space to spanning trees of a Similarity Graph. While the order of complexity at each iteration remains at $O(N^4)$, the time taken is much less. For instance, when $\tau$ is chosen as median of the pair-wise jaccard similarity values as in Section 3.4, the time taken is nearly half of the time taken when $\tau = 0$, which corresponds to having search space as all possible trees on $\mathcal{T}$. Thus, for the proposed approach using objective function (3.3), constraining the local search to spanning trees of a Similarity Graph is necessary to ensure that we do not get a trivial solution. Doing so also makes the local search faster. As compared to this, for the proposed approach using objective function (3.4), using a Similarity Graph only makes the optimum search faster as can be seen in the reduced number of candidates in Fig. 3.15 and the reduced number of iterations to converge, as shown in Fig. 3.14.

Intuitively, using (3.4) as the objective function minimizes the weighted L1 norm of the difference between pair-wise jaccard similarity and the similarity estimated using a tag tree, thus trying to approximate the normalized symmetric second order statistics...
of a corpus using a tree on the set of tags. We have chosen weighted L1 norm since it was observed to lead to the best results across different corpora among other variants such as L1, L2 and weighted L2 norm. Implementation of the similarity estimated using a tag tree (3.5) can be conveniently done by using Logarithms and Ford Fulkerson Algorithm [39]. Also, since the LS-SA approximates the pair-wise similarities $J_{\mathcal{T}}(i, j)$ in $O(N)$ as compared to $O(N^2)$, approaches such as [111] that utilize $J_{\mathcal{T}}(i, j)$ lead to equal or only marginally better performance, despite having several orders higher space requirement.

3.7 Summary

In this section we summarize the research contributions of this chapter, key practical insights, and advantages and limitations as compared to existing expert systems. We also discuss limitations of our work, and provide several suggestions for future research directions.

We have proposed ontological tag trees to enable expert systems address the sparsity of folksonomies effectively and in a space efficient manner. Ontological tag trees or tag trees are defined as simple trees on the set of tags in a corpus. The construction of tag trees is formulated as an optimization problem on corpus based statistics, and is solved through a novel local search based approach. We have shown that this approach can be used to build tag trees for tags obtained from two corpora, one composed of noisily annotated Flickr images and the other composed of cleanly annotated stock images. To validate the utility of the constructed ontological tag tree, we proposed two evaluation tasks involving tag prediction in images. We have demonstrated that for the task of predicting the unseen tags of a given image with a partially observed set of tags, the proposed ontological tag trees outperform those constructed using only
semantic relationships, or tag graphs constructed using commonly used techniques that have comparable space requirements. In the second task, we have shown that the tag tree obtained from the proposed approach makes the process of using appropriate classifiers to tag an untagged test image more efficient. Robustness analysis shows that the proposed approach is fairly robust to tag noise and differences between the training and test set distributions.

Compared to previous expert systems, our work offers significant advantages. In addition, several key insights can be derived from our study. Since expert systems such as [119] based on only semantic relationships fail to capture the data-specific relationships between tags, their performance is significantly lower on data-driven tasks than that of ontological tag trees constructed using the proposed approach. This is demonstrated through evaluations in Section 3.4. In addition, we show that even though ontological tag trees have space requirement of only $O(N)$ for $N$ tags, as compared to $O(N^2)$ for tag graphs, tag trees can provide equal or better performance than several existing expert systems on the evaluation tasks. Particularly, compared to [111] which may require 27 terabytes of space to store pair-wise similarities, our approach requires less than 50 megabytes, and still achieves almost equal performance. Ontological tag trees can thus offer a very convenient method for expert systems to capture corpus based relationships for tasks such as tag prediction, and efficient resource classification. The significant savings in space requirement facilitates practical deployment of the expert systems even on computing devices that do not have gigantic memory available. The third key advantage of using ontological tag trees is that their construction does not depend on the availability of content-based features (such as textual or visual features). Thus as compared to previous expert systems such as [71], [131], [54], [22], [118], our approach can help alleviate sparsity of online folksonomies even in domains where extracting content-based features may be inefficient or infeasible [55], [114], [144], [141]. Lastly,
as discussed in Section 3.6, the performance of the LS-SA approach is significantly better than that of the LS-WAH approach. It can thus be recommended that the LS-SA approach be used to construct ontological tag trees.

While the proposed ontological tag trees can help expert systems achieve high performance in a space efficient manner, there are certain limitations of the proposed approach. As discussed in Section 3.6, the time taken per iteration of the local search based tag tree construction approach varies as $O(N^4)$. For large number of tags, the total time taken to construct tag trees can be high. In addition, the tag trees only capture undirected weighted relationships between tags. As a result, any directed relationships that may be present in the corpus (for example subsumption relationships [107]) are lost by the constructed tag trees.

We provide several suggestions for future work directions based on our work. Since the tag tree construction approach has a limitation of high time requirement, one direction of future work could be to make the proposed approach faster by choosing a higher or an adaptive $\tau$, with the possible trade-off being increasing the constraint on the local search, thereby achieving a sub-optimal tag tree. Another direction of future work could be to keep a low $\tau$ but develop distributed and possibly approximated versions of the proposed approach, such that available distributed clusters, for example Amazon EC2 cloud server [4], can be leveraged. In the current work, the LS-SA approach can be thought of as approximating the symmetric second order corpus statistics. An extension of the work could be to construct tag structures that approximate higher order corpus statistics. In addition, future work could utilize the constructed tag trees for other practical applications such as tag recommendation: given set of tags associated with a test images, which additional tags would you associate with the image? Removal of potentially noisy tags can also be studied based on whether a tag appears too far in the tree from other tags associated with a given image. Lastly, a hierarchical taxonomy could be constructed in
future, where the edges between the nodes are directed, by adopting similar techniques and using the co-occurrence data from the corpus.

### 3.8 Acknowledgments

We thank the anonymous WWW 2014 and Elsevier Expert Systems with Applications reviewers for their feedback and comments on the work. This research was partially supported by an internship at Yahoo Labs, and partially by the Center for Wireless Communications at University of California San Diego.

Chapter 3, in part, contains material as it appears in the Proceedings of the companion publication of the 23rd International Conference on World Wide Web (WWW ’14). “Construction of Tag Ontological Graphs by Locally Minimizing Weighted Average Hops”. Chetan Verma, Vijay Mahadevan, Nikhil Rasiwasia, Gaurav Aggarwal, Ravi Kant, Alejandro Jaimes, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, contains material as it appears in Elsevier Expert Systems with Applications. “Construction and Evaluation of Ontological Tag Trees”. Chetan Verma, Vijay Mahadevan, Nikhil Rasiwasia, Gaurav Aggarwal, Ravi Kant, Alejandro Jaimes, Sujit Dey. The dissertation author was the primary investigator and author of this paper.
Chapter 4

Improving Scalability of Personalized Recommendation Systems for Enterprise Knowledge Workers

The previous two chapters discussed methods that enable automatic content classification, and that address tag sparsity of annotated online content. This chapter studies a specific type of personalization system, namely a recommendation system, and shows how it can be designed for enterprise ecosystem, as well as how it can be made efficient and hence scalable.

Enterprise knowledge workers have been overwhelmed by the growing rate of incoming data in recent years. In this chapter, we present a recommendation system with the goal of helping knowledge workers in discovering useful new content. Specifically, our system builds personalized user models based on file activities on enterprise network file servers. Our models use novel features that are derived from file metadata and user collaboration. Through extensive evaluation on real world enterprise data, we demonstrate the effectiveness of our system with high precision and recall values. Unfortunately, our experiments reveal that per-user models are unable to handle heavy workloads. To address this limitation, we propose a novel optimization technique, Active Feature-based Model Selection, that predicts the user models that should be applied on each test file.
Such a technique can reduce the classification time per file by as much as 23 times without sacrificing accuracy.

We also show how this technique can be extended to improve the scalability exponentially at marginal cost of prediction accuracy, e.g., we can gain 169 times faster performance on average across all shares by sacrificing 4% of F-score.

4.1 Introduction

The amount of data created, replicated and consumed every year has been growing exponentially over the years [57]. Between 2010 and 2020, the digital data is expected to grow by 50 times [40]. Following a similar trend, enterprise data was projected to increase by 76% within the last one and half years [40]. The tremendous growth of data has become one of the biggest challenges for enterprises. It places unreasonable burden on enterprise users or knowledge workers. Studies [57] have shown that nearly 65% of knowledge workers report feeling overwhelmed by the incoming data. The situation is further exacerbated by the fact the enterprise data resides across heterogeneous devices and environments, including network file servers, source control repositories, emails and file servers, cloud applications [103, 92], and enterprise social networks [68]. This makes finding relevant content a very challenging task for knowledge workers.

In this chapter, we present a system that assists knowledge workers to discover relevant content by providing personalized recommendations. In order to serve these recommendations, our system monitors user activity over a training period and trains per-user access models. Although our approach currently focuses on files on networked file servers (shares), it could potentially be applied to other types of data as well. For training user models, we extract interesting metadata features based on file path and hierarchical directory structure. In our experiments, we observe that many enterprise users show a high degree of collaboration, which is inferred based on their common
file accesses. To leverage user collaboration, we propose a collaborative filtering aware modeling approach that provides significant improvement over metadata-based models. Our evaluation uses activity logs from eight different shares of an enterprise customer. We measure the effectiveness of our system in providing predictions for files that are new with respect to the training data. Our experiments show that the metadata-based models can achieve an average F-score of nearly 50% at 75% precision across all shares and users. Our collaborative filtering-based approach remarkably improves the same F-score to over 70%.

Although effective, per user models are not at all efficient in terms of the runtime performance of testing. This is because every file that is created or modified in a share needs to be tested against the model of each user in the share. This could be computationally very expensive especially for heavy workloads with a large number of users. For instance, for handling a heavy workload (i.e., activity involving high file edit rate) in our experiments, our system will need 48 64-core machines to test all edited files. The high computational and financial cost poses a severe constraint that could adversely affect the adoption of our system. To address this issue, we propose a novel technique, called Active Features-based Model Selection (AFMS), that automatically selects only those user models that are highly likely to generate positive predictions, while ignoring the models that will definitely or most likely generate negative predictions. Thus using this technique, a test file is tested against only the selected models, thereby speeding up the overall testing time. Our experiments show that AFMS drastically reduces testing time without any loss in the prediction accuracy. Furthermore, we show that AFMS can be used adaptively to trade-off marginal prediction accuracy for significant performance gains during the periods of heavy workload.

Our previous version [123] of this work presented the basic technique for modeling file accesses. This chapter presents a new technique, AFMS, for improving solubility,
along with new experiments and insights. The main contributions of our work are the following:

1. Propose a system to analyze user file accesses in order to train personalized models, and recommend relevant content with a high degree of accuracy

2. Propose a novel model selection technique that can speed up the testing time by more than 6 times on average, and by over 23 times in the best case without any loss in the prediction accuracy

3. Extend the model selection technique to speed up the testing time significantly at the cost of marginal loss in the prediction accuracy

4. Demonstrate practicality of our system using a systematic study and evaluation based on real world enterprise data

**4.2 The File Recommendation System**

Our system uses a binary classification approach for providing file recommendations. For this purpose, it builds a separate classification model for each user of a file share. The dataset of a given user includes all the files that have been accessed by all users of the share. The label of a file in the dataset is 1 if the user accesses it, 0 otherwise. It is important to know that if a user accesses a particular file during the training phase, it is highly likely that the user will access the same file during the testing phase. Recommending such a file to the user is not very useful. Hence the main objective of our system is to discover useful but new files that may have been created or modified by other users of the same share. For this reason, the testing phase includes only those files that are new with respect to the training phase for the user. Our system then applies
112

(a) **Training**: metadata models for each user

(b) **Testing**: apply the trained models for all users in share

(c) **Training**: CF models for each user

**Figure 4.1.** Approach Overview
the model of the user to these files to predict classification labels. The rest of this section
describes the features and the modeling techniques used in the system.

4.2.1 Metadata Features

The metadata features include features extracted from different attributes of a file:

- **Folder**: We create a folder feature corresponding to every folder and its ancestor
  folders observed in the training period\(^1\). For a file \( f \), we capture the folder features
  as a binary vector with the value 1 in the locations corresponding to the folder
  or the ancestor folders of \( f \), and 0 elsewhere. These features help in learning a
  user’s preference for certain subtrees of the file system hierarchy in a share without
  having to define a folder distance metric.

- **Token**: We tokenize the path name of each file using a novel tokenization approach,
  the details of which can be found in [123]. We construct a token vocabulary using
  all the tokens observed during the training phase. For extracting features, we
  capture tokens of a file as bag-of-words representation on the entire vocabulary.

- **Extension**: We construct an extension vocabulary based on the popular extensions
  observed in the training data. We then use this vocabulary to represent a file’s
  extension with one-hot encoding-based features.

4.2.2 Modeling Techniques

We first build basic metadata models based on the metadata features. Using these
models, we then develop an innovative approach to leverage collaboration among users
for building collaborative filtering aware models that provide better performance over the
basic metadata models.

\(^1\)Please note that no features are derived for completely new attributes seen only during the testing
phase.
Metadata-based Modeling

For each user, we train a metadata model using the metadata features. Figure 4.1 (a, b) shows the training and the testing phases of this approach. Considering the trade-off between accuracy and training time efficiency [123], we pick Support Vector Machine (SVM) with a linear kernel as our modeling technique because it provides high accuracy without incurring significant training time. More importantly, a linear SVM provides feature weights that capture the relative significance of individual features. We leverage this aspect in our scalability optimization technique AFMS as described in Section 4.5.

Collaborative Filtering (CF)-based Modeling

Enterprise users often show a high degree of collaboration in terms of accessing common files in a share. We make this observation based on the measurement of the metric, normalized triangle count (Section 4.3.1), that captures the degree of collaboration among users. As an illustration, see Figure 4.2 that shows user communities in terms of social network graphs for a couple of shares.

We leverage user collaborations to build CF models that are significantly more effective than the metadata models. Here are the details of the method used in building a CF model for a user $u$. First, we divide the original training set into a new training set and a validation set. We then use the new training set to train a metadata model for each user. Second, for each file in the validation set, we apply the metadata models of all the users, and generate a vector of the resulting predicted labels. Third, we train the CF model for the user $u$ using the files in the validation set. For this, we construct the feature vector for each validation file by concatenating the metadata features with the vector of the predicted labels (Figure 4.1(c)). For each test file observed during the testing phase, we first apply the metadata models of all the users to generate a vector of the predicted labels. Finally, we append this vector to the metadata features to create a new feature
vector and feed it to the user \( u \)’s CF model to obtain the final predicted label.

An interesting point to be noted here is that our CF-based modeling technique does not require a pre-defined user-user relationship or even a user similarity metric. Rather, the model can automatically learn positive or negative correlation between the activities of users, and accordingly adapt to make better predictions. For instance, if two users have similar access patterns, the CF model for one user could learn that its labels positively correlate with labels predicted by the other user’s metadata model. Thus, greater the degree of collaboration, the better will be the effectiveness of the CF models. Another benefit of our technique is that, unlike pure collaborative filtering-based systems developed in the past, it does not suffer from a cold-start problem. Previous systems recommend an item to a user if other similar users have shown a preference for that item. Therefore, these systems are incapable of recommending a completely new item because no other user has seen it before. In contrast, the novelty of our technique is in using CF features that are based on predicted labels rather than the history of past accesses.

4.3 Evaluation Framework

This section describes the evaluation data and methodology.

4.3.1 Data

Table 4.1. Statistics of shares used for evaluation

<table>
<thead>
<tr>
<th>Share</th>
<th># days</th>
<th># users</th>
<th># files operated on</th>
<th># file operations</th>
<th>Norm. Triangle Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>123</td>
<td>992</td>
<td>36,009</td>
<td>11M</td>
<td>8K</td>
</tr>
<tr>
<td>B</td>
<td>122</td>
<td>464</td>
<td>1,309</td>
<td>136K</td>
<td>3K</td>
</tr>
<tr>
<td>C</td>
<td>122</td>
<td>160</td>
<td>1,044,779</td>
<td>3M</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>D</td>
<td>121</td>
<td>183</td>
<td>746</td>
<td>11K</td>
<td>951</td>
</tr>
<tr>
<td>E</td>
<td>66</td>
<td>1,288</td>
<td>99,733</td>
<td>292K</td>
<td>3K</td>
</tr>
<tr>
<td>F</td>
<td>66</td>
<td>937</td>
<td>6,911</td>
<td>4M</td>
<td>0.4</td>
</tr>
<tr>
<td>G</td>
<td>66</td>
<td>198</td>
<td>334</td>
<td>14K</td>
<td>3K</td>
</tr>
<tr>
<td>H</td>
<td>57</td>
<td>398</td>
<td>133,006</td>
<td>4M</td>
<td>45</td>
</tr>
</tbody>
</table>
Figure 4.2. Social network graphs for shares show that users collaborate and tend to form communities by accessing common files. In these graphs, each node corresponds to a user, and an edge connects two users when the Jaccard index of common files accessed by those users is above 0.5.

We use file activity logs from eight network file shares of an enterprise customer for evaluation. Table 4.1 provides key statistics of this data. Because our recommendation system targets user collaborations, we select shares from 90\textsuperscript{th} percentile in terms of triangle count – a metric to capture degree of collaboration among users. Triangle count for a share is the number of triangles formed, where a triangle edge corresponds to a connection between two users based on accessing at least one common file. The table shows the triangle count values normalized with respect the number of files. For better visualization, see Figure 4.2 that shows users forming different communities based on their collaborations.

**Removal of automated activity:** We observe two types of file activities in our data. The first corresponds to the activities that are manually initiated by users. These activities are characterized by a low number of file operations per hour. The second corresponds to
scripted activities that are performed by automated computer programs or scripts. These activities are generally seen as bursts or exceptionally high number of file operations per hour. Since our goal is to assist knowledge workers and not automated programs, we need to remove the scripted activity from the evaluation data. For this purpose, we remove the activities by a user in an hours if the number of activities are determined to be exceptionally high. Towards this, an appropriate threshold is determined using Tukey’s outlier factor [126].

**Selection of evaluation users:** Since our system helps users discover new or modified content, it is primarily targeted for active users. Therefore, for each share, we sample 30 users from the highest quartile of users in terms of number of activities, and use them in our evaluation.

### 4.3.2 Evaluation Methodology

We experiment with different training and testing periods to evaluate the performance of our system under different settings.

**Varying training and test periods**

We divide the total duration of data from each share into several slices and create training and testing periods such that testing occurs immediately after training. For consistency, each share is divided into seven slices and five training and five testing periods are created as shown in Figure 4.3. We perform experiments with different combinations:

- **Fix testing, vary training.** A longer training window has more number of activities, thereby allowing better model training. However the activities become older as the distance between them and the testing period increases, thus preventing the
model from reflecting the latest user patterns accurately. On the other hand, a small training window has less number of activities, but these activities are relatively fresh. Thus, there exists a trade-off between the size and the recency of the training data. We study this trade-off by varying the training duration, while keeping the testing period fixed.

- **Fix training, vary testing.** As a user’s behavior evolves over time, it is expected that the effectiveness of the user’s model would degrade as the testing period becomes longer. We can measure the robustness of a model by observing the rate of degradation. With this goal, we experiment with varying testing periods, while keeping the training period fixed.

**Evaluation metrics**

Consider an experiment with a certain combination of training and testing periods for a user \( u \). Let \( F_{true+u} \) denote the set of files that \( u \) accessed in the testing period, and \( F_{pred+u} \) denote the set of files predicted by \( u \)’s model. Then the precision and the recall
of the model respectively are $\frac{|F_{true+} \cap F_{pred+} \cap u|}{|F_{pred+} \cap u|}$, and $\frac{|F_{true+} \cap F_{pred+} \cap u|}{|F_{true+} \cap u|}$. We use the following metrics to evaluate the effectiveness of the model.

- **F-score**: The F-score for a model is the harmonic mean of the precision and the recall, and provides a balanced picture of the model’s effectiveness.

- **Recall@75P**: In practice, it is important for a recommendation system to have high precision because a large number of false positives may discourage a user from using the system altogether. Therefore, we additionally measure the recall values when the precision is high – 75%. This is achieved with the help of confidence scores associated with the predicted labels.

The measurements for F-score and Recall@75P are averaged across all the evaluation users to obtain AF and AR@75P respectively. For each share, we report the average and the maximum values of AF and AR@75P across the results over all combinations of training and testing periods. Please note that the maximum values provide realistic performance estimate of a properly tuned system.

### 4.4 Performance Evaluation

We evaluate our system using the Python-based scikit-learn library [108]. We train models on a 32 core, 64GB, 2.6GHz machine, and conduct testing on a separate 8 core, 32GB, 2.7GHz machine, leveraging multiple cores with multiprocessing. We use 3-fold cross-validation to learn the regularization parameter $C$ for the linear SVM models by varying $C$ over \{10^{-2}, 10^{-1}, 1, 10^1, 10^2, 10^3\}.

#### 4.4.1 Metadata Models

Table ?? shows performance results of the metadata models. The numbers in column Max AR@75P are reasonable performance estimate of a well tuned system. The
Table 4.2. Performance summary for metadata models. Performance numbers are averaged over 5 iterations with random initialization of the linear SVM model training. Numbers are listed along with the standard deviations.

<table>
<thead>
<tr>
<th>Share</th>
<th>Avg AF</th>
<th>Max AF</th>
<th>Avg AR@75P</th>
<th>Max AR@75P</th>
<th>% TP by others</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80.7±0.0</td>
<td>91.1±0.0</td>
<td>77.6±0.0</td>
<td>93.9±0.0</td>
<td>20.4±0.0</td>
</tr>
<tr>
<td>B</td>
<td>46.4±0.0</td>
<td>51.2±0.0</td>
<td>34.9±0.0</td>
<td>44.2±0.0</td>
<td>73.1±0.0</td>
</tr>
<tr>
<td>C</td>
<td>23.1±0.0</td>
<td>24.2±0.0</td>
<td>11.4±0.0</td>
<td>12.4±0.0</td>
<td>87.7±0.0</td>
</tr>
<tr>
<td>D</td>
<td>30.7±0.0</td>
<td>36.3±0.0</td>
<td>19.0±0.0</td>
<td>24.7±0.0</td>
<td>82.8±0.0</td>
</tr>
<tr>
<td>E</td>
<td>26.9±0.0</td>
<td>35.0±0.0</td>
<td>17.9±0.0</td>
<td>24.1±0.0</td>
<td>66.0±0.0</td>
</tr>
<tr>
<td>F</td>
<td>81.5±0.0</td>
<td>84.3±0.0</td>
<td>82.9±0.0</td>
<td>84.6±0.0</td>
<td>9.8±0.0</td>
</tr>
<tr>
<td>G</td>
<td>44.8±0.0</td>
<td>51.3±0.0</td>
<td>50.6±0.0</td>
<td>58.3±0.0</td>
<td>99.9±0.0</td>
</tr>
<tr>
<td>H</td>
<td>47.6±0.0</td>
<td>50.2±0.0</td>
<td>49.9±0.0</td>
<td>53.0±0.0</td>
<td>74.4±0.0</td>
</tr>
</tbody>
</table>

Table 4.3. Performance summary for CF models. Performance numbers are averaged over 5 iterations with random initialization of the linear SVM model training. Numbers are listed along with the standard deviations.

<table>
<thead>
<tr>
<th>Share</th>
<th>Avg AF</th>
<th>Max AF</th>
<th>Avg AR@75P</th>
<th>Max AR@75P</th>
<th>% TP by others</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80.7±0.0</td>
<td>100.0±0.0</td>
<td>78.5±0.0</td>
<td>100.0±0.0</td>
<td>21.2±0.0</td>
</tr>
<tr>
<td>B</td>
<td>48.6±0.0</td>
<td>77.1±0.0</td>
<td>32.2±0.6</td>
<td>62.1±0.0</td>
<td>73.1±0.0</td>
</tr>
<tr>
<td>C</td>
<td>23.5±0.0</td>
<td>36.0±0.0</td>
<td>10.1±0.0</td>
<td>16.0±0.0</td>
<td>87.5±0.0</td>
</tr>
<tr>
<td>D</td>
<td>32.3±0.0</td>
<td>47.5±0.0</td>
<td>21.5±0.0</td>
<td>38.3±0.0</td>
<td>83.3±0.0</td>
</tr>
<tr>
<td>E</td>
<td>27.6±0.0</td>
<td>41.1±0.0</td>
<td>25.3±0.0</td>
<td>100.0±0.0</td>
<td>65.7±0.0</td>
</tr>
<tr>
<td>F</td>
<td>81.5±0.0</td>
<td>87.9±0.0</td>
<td>83.3±0.0</td>
<td>89.2±0.0</td>
<td>9.8±0.0</td>
</tr>
<tr>
<td>G</td>
<td>55.1±0.1</td>
<td>76.2±0.0</td>
<td>58.0±0.2</td>
<td>89.9±0.4</td>
<td>96.6±0.1</td>
</tr>
<tr>
<td>H</td>
<td>47.6±0.0</td>
<td>57.2±0.0</td>
<td>49.6±0.0</td>
<td>66.5±0.0</td>
<td>75.4±0.0</td>
</tr>
</tbody>
</table>

average of these numbers across all the shares, 49.4%, shows that our metadata models can capture nearly half of users’ access for new files, while having under 25% wrong file recommendations. This demonstrates the effectiveness of the trained models.

We would like to point out that our dataset includes an undesirable case where during the testing phase, a user accesses a new file that was recently created by the same user. Since recommending such a file to the user is not useful, we measure their proportion in our recommendations. The last column of Table ?? shows that most of the correctly recommended files do not fall in the above category, thereby upholding the validity of our models.
4.4.2 CF Models

We experimented with different strategies to sample validation files for training CF models. We use the strategy that provides the best performance, which is using the training set itself as the validation set. The results in Table ?? clearly highlight the significant gains of CF models over metadata models. The average of Max AR@75P across all shares is 70.2%, which is an improvement of a whopping 20% over the performance of metadata models. Also, it is not very surprising that the top four shares, B, D, E, and G, in terms of gains of the CF model, are found in the top five shares in terms of normalized triangle counts. Share A, which is the remaining share in the top five, doesn’t show substantial improvement primarily because the performance of its metadata model is already too high.

4.4.3 Temporal Variation in Performance

To study the performance variations for different lengths of training and testing periods, we focus only on the metadata models because as compared to CF models, they provide larger variation owing to lower baseline performance.

Varying training period

Figures 4.4 and 4.5 show the performance variation of metadata models with varying training periods. The best performance for most shares is observed for index 1 that corresponds to the longest training period. As expected, the performance degrades as the training window shrinks. Performance can also degrade for a large training window wherein a model gives importance to activities that are outdated with respect to the user’s recent access pattern. However, we don’t observe this behavior because the longest training periods, ranging from 41 to 88 days for the eight shares, are not long enough to cause the degradation.
Figure 4.4. AF for metadata models with the fixed testing and varying training periods

Figure 4.5. AR@75P for metadata models with the fixed testing and varying training periods
Figure 4.6. AF for metadata models with the fixed training and varying test periods

Figure 4.7. AR@75P for metadata models with the fixed training and varying test periods
Varying testing period

Figures 4.6 and 4.7 show the performance variation of metadata models with varying testing periods. Beyond the testing periods with indices 1 and 2 which are the smallest and hence potentially the noisiest, the performance degrades mildly as the testing window widens. This is an excellent indicator of the robustness of our models.

![Histogram showing optimal training window size](image)

**Figure 4.8.** Distribution of the least number of weeks for each user of share D that gives highest F-score.

Next, we perform a thorough study of the effect of training window size on the performance of the models of individual users. For this purpose, we select share D, and divide its data into 17 parts, each corresponding to one week. We remove 4 out of 30 users because they do not have sufficient activity in each week. We then pick the last five weeks for the testing period, and create 12 different training windows, varying from 1 week to 12 weeks. Based on the results, we plot (Figure 4.8) the distribution of the optimal training window size (in terms the least number of weeks) providing the highest F-score for the evaluation users of share D. The median and the mode of this window is 8 weeks even though there are longer training windows up to 12 weeks, thereby confirming our previous assertion that more training data does not necessarily translate into better performance. Additionally, we observe that the optimal window size varies for each user. The most active user (with highest activity count) requires 4 weeks, whereas the least active user requires 11 weeks. This suggests that the optimal window also depends on
workload, with more active users requiring lesser training time.

In a practical deployment, we could use an online evaluation approach to continuously tune the system based on performance and workload characteristics. Lastly, it is important to note that some of the recommended files, which are flagged as false positives in our experiments, could in fact be true positives in an actual deployment because users may genuinely find them useful. Hence a precision reported in this evaluation serves as the lower bound for the precision expected from an actual deployment.

### 4.4.4 Scalability

Unlike training, which is an offline process, classification needs to be performed in real time. Therefore, we analyze time and space complexities of only the testing, i.e., the classification phase.

**Time complexity**

In an actual deployment, whenever a file is created or modified in a share, our system applies personalized models of each user in the share. Therefore, the runtime performance depends on the rate of file edit operations $R_{edit}$ (in files per second) and the number of users $N$.

Let us compute the time taken to classify one file. Let $T_{meta\_model}$ and $T_{cf\_model}$ represent the average times (in seconds) required to apply a metadata model, and a CF model respectively. We need time $T_{meta\_feature}$ seconds to extract metadata features, and $NT_{meta\_model}$ seconds to extract collaborative filtering features by applying the metadata model of each user. We then apply the CF model of each user, which takes $NT_{cf\_model}$ seconds. Thus, the total classification time for one file is: $T_{meta\_feature} + NT_{meta\_model} + NT_{cf\_model}$ seconds.

To make approximate calculations, we replace $T_{meta\_model}$ with $T_{cf\_model}$ and
ignore $T_{meta, \text{feature}}$ because it is negligible as compared to $NT_{cf, \text{model}}$. Thus, it would take total time of $2NT_{cf, \text{model}}R_{edit}$ seconds to process all the files edited in each second. Considering the sample values we observed for some of the high workload shares: $N = 16,000$, $R_{edit} = 5$ files/second, $T_{cf, \text{model}} = 19 \times 10^{-3}$ second, the total time would be 3040 seconds. Hence, in order to process all the files created or modified per second, we would need at least 48 64-core machines, with each core concurrently performing classification.

Undoubtedly, the computational resources required to handle this kind of workload would amount to a significant financial cost to the enterprise. In the next section, we address this problem by presenting an optimization technique that substantially lowers the computational requirements without sacrificing classification accuracy. With this improvement, we need just one 64-core machine to manage the above mentioned heavy workload.

**Space complexity**

Unlike computation, memory requirements of our system are quite modest even for heavy workloads. Considering values: $N = 16,000$, $R_{edit} = 5$ files/second, $F_{meta} = 35,000$ (highest number of metadata features in our datasets), and $S = 4$ bytes (size of value and weight of a feature), we would need maximum $(F_{meta} + N)R_{edit}S$ i.e., nearly 1MB memory for all the features in the worst case. Note that the features of a file remain the same for all the user models. Each metadata model is represented by $F_{meta}$ weights, and each CF model is represented by $F_{meta} + N$ weights. Thus, the total memory to store metadata and CF models of all the users is: $(2F_{meta} + N) \times NS$ i.e., approximately 5 GB. In practice, the required memory would be much smaller because collaborative filtering will be applied to a small subset of users that show high degree of collaboration.
4.5 Active Features-based Model Selection (AFMS)

To speed up the classification time, our optimization technique AFMS leverages properties of a soft-margin linear SVM model. In this model, the predicted label for a file $f$ is determined using the score:

$$ g(f) = w_0 + \sum_{j=1}^{F} w_j * x_j $$  \hspace{1cm} (4.1)

Here, $F$ is the number of features, $w_0$ is the intercept, $w_j$ is the $j^{th}$ feature weight in the model, and $x_j$ is the value of the $j^{th}$ feature of the file. The model predicts label 1 if $g(f) \geq 0$, and 0 otherwise. Given that our features have non-negative values, without applying the model we can predict label 0 when the following conditions are met:

- **condition 1**: The intercept is negative, i.e., $w_0 < 0$, and
- **condition 2**: None of the weights of active (non-zero) features is positive.

Under these conditions, $g(f)$ is always negative, and hence the predicted label is 0. Otherwise, we need to apply the model to obtain the classification label, which could be 0 or 1. The AFMS technique uses this key intuition to save the cost of applying classification models. Of course, its effectiveness depends on the number of times the conditions are matched. Unsurprisingly, as our experiments show, this happens quite often (refer Table ??):

- **condition 1**: Around 96% of the metadata models have negative intercept (i.e., $w_0 < 0$). This is because a typical user accesses only a small fraction of files. Owing to this natural imbalance, the model favors 0 as the prediction label, resulting in a negative intercept.

- **condition 2**: Our data shows sparsity of features with an average of only 2.2% of metadata features being active (i.e., $x_j > 0$) in the files. Also, the average ratio
Table 4.4. Statistics for metadata models, averaged across 30 evaluation users and different training periods (Section 4.3.2), showing how AFMS is effective.

<table>
<thead>
<tr>
<th>Share</th>
<th>% models with negative intercept</th>
<th>% active features</th>
<th>% features with positive weight</th>
<th>% models per positive feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100.0</td>
<td>0.10</td>
<td>2.8</td>
<td>2.8</td>
</tr>
<tr>
<td>B</td>
<td>98.9</td>
<td>1.90</td>
<td>2.5</td>
<td>3.6</td>
</tr>
<tr>
<td>C</td>
<td>100.0</td>
<td>0.10</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>D</td>
<td>100.0</td>
<td>6.60</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>E</td>
<td>96.1</td>
<td>1.40</td>
<td>15.2</td>
<td>18.5</td>
</tr>
<tr>
<td>F</td>
<td>98.3</td>
<td>1.60</td>
<td>4.2</td>
<td>5.8</td>
</tr>
<tr>
<td>G</td>
<td>74.5</td>
<td>6.10</td>
<td>21.7</td>
<td>40.8</td>
</tr>
<tr>
<td>H</td>
<td>100.0</td>
<td>0.01</td>
<td>4.1</td>
<td>4.1</td>
</tr>
<tr>
<td>Avg.</td>
<td>95.9</td>
<td>2.2</td>
<td>6.4</td>
<td>9.6</td>
</tr>
</tbody>
</table>

of the features with positive weights (i.e., \( w_j > 0 \)) per model is only 6.4%, and each of these features on an average has a positive weight in less than 10% of the models. The combination of these three facts significantly lowers the probability of finding active features with positive weights.

The favorable statistics result in matching the conditions quite often, thereby leading to faster testing of files because the cost of checking the conditions is negligible. Note that faster testing of files is referred to as better performance in the discussion below. While we have provided the above statistics based on the metadata models, similar trends are observed for CF models as well. Section 4.5.2 shows the actual gains in the performance for both metadata and CF models. The next section describes the algorithm to implement AFMS.

### 4.5.1 AFMS Algorithm

Our technique extends the basic idea to make the optimization tunable so that it can potentially trade-off model correctness for even better classification efficiency. In order to do so, we define that a feature with index \( j \) triggers a model when \( w_j \geq \tau \) or \( w_0 \geq 0 \) where \( w_0 \) and \( w_j \) are the intercept and the \( j^{th} \) feature weight of the model
Algorithm 9. AFMS Classification

Variables definitions:

\[ N \]: Number of users
\[ F_{\text{meta}} \]: Number of metadata features
\[ F_{\text{CF}} \]: Number of features in CF models = \( F_{\text{meta}} + N \)
\[ w_{\sigma(i,j)} \]: Weight of \( j^{th} \) feature in \( i^{th} \) user model, \( 1 \leq i \leq N, 1 \leq j \leq F_{\sigma}, \sigma \in \{\text{meta}, \text{CF}\} \)

Offline pre-processing to be done right after model training:

\[ FT_{M}(j) = \{i | w_{\sigma(i,j)} \geq \tau, \text{ or } w_{\sigma(i,0)} > 0\} \]

Classification:

Input:
\[ X = [x_1, x_2, \ldots, x_{F_{\text{meta}}}] \]: Metadata feature vector of the test file \( f \)

Output:
\[ Y = [y_1, y_2, \ldots, y_N] \]: Predicted classification label (0 or 1) by metadata models for each user
\[ L = [l_1, l_2, \ldots, l_N] \]: Predicted classification label (0 or 1) by CF models for each user

Procedure:

\[ M = C = \{\} \]
\[ initialize \text{ triggered metadata and CF models} \]

\[ for \ i = 1 \ to \ N: \]
\[ \text{initialize CF features and output labels} \]
\[ Y[i] = L[i] = 0 \]

\[ for \ j = 1 \ to \ F_{\text{meta}}: \]
\[ \text{step1: obtain triggered metadata models} \]
\[ \text{if } (X[j]): \]
\[ M = M \cup FT_{M}(j) \]

\[ for \ i \ in \ M: \]
\[ \text{step2: compute CF features} \]
\[ Y[i] = \text{ApplyMetadataModel}(i, X) \]

\[ X = [X, Y] \]
\[ \text{step3: append CF features to metadata features} \]

\[ for \ j = 1 \ to \ F_{\text{CF}}: \]
\[ \text{step4: obtain triggered CF models} \]
\[ \text{if } (X[j]): \]
\[ C = C \cup FT_{C}(j) \]

\[ for \ i \ in \ C: \]
\[ \text{step5: compute CF model labels} \]
\[ L[i] = \text{ApplyCFModel}(i, X) \]
Figure 4.9. AFMS technique first creates FTM in an offline manner. The classification step applies only those models that are triggered by active features.

respectively. Here, we use the threshold parameter $\tau \geq 0$ to make the technique tunable. Given a test file, a model would be applied on it if any of the active features of the test file trigger the model. For $\tau = 0$, there is no loss of predictive correctness, and yet there is performance gain due to the optimization. However, as $\tau$ increases, fewer models are triggered, and hence there is further improvement in the performance, albeit at the cost of predictive correctness. With fewer models being applied, there will be more number of 0 as the predicted labels, and hence the recall is expected to degrade. However, precision and likewise F-score can degrade or improve depending on the correctness of the recommendations. Section 4.5.2 shows that we can gain significant efficiency at marginal cost of model correctness. Furthermore, the technique can be made adaptive by varying $\tau$ based on the rate of file edit operations at a given time.

Please refer to Algorithm 9 to understand variable definitions, and the details of the algorithm. As part of pre-processing, which is done right after the training phase, for each feature we compute the set $FTM$ (Feature Triggered Models) consisting of models
that are triggered by the feature (Figure 4.9). The algorithm computes separate sets for metadata and CF models.

The input to the classification procedure is the array $X[1 \ldots F_{meta}]$ of metadata features corresponding to the test file. The procedure produces the outputs: an array $L[1 \ldots N]$ consisting of prediction label for each of the $N$ users by CF models, and an array $Y[1 \ldots N]$ consisting of prediction label by metadata models. The procedure initializes the output arrays with 0 values as the default labels. It also initializes sets $M$ and $C$ that are later used to collect triggered metadata and CF models respectively. The first step obtains the set of metadata models that are triggered by the active features. The second step applies these metadata models to compute metadata predicted labels, which are also the CF features $Y$. The third step appends the CF features to the metadata features. The fourth step obtains the set of CF models that are triggered by the active features. Finally, the last step applies these CF models to compute the CF predicted labels.

If a model is not applied, the default label 0 becomes the predicted label, and the algorithm assigns a random negative value as the confidence score for the label. We need the confidence score because it is used to determine the relative ranking of test files, which is then used in the computation of recall at 75%. As a side effect, it is possible that $AR@75P$ for AFMS with $\tau = 0$ may differ from $AR@75P$ computed without AFMS.

**Complexity analysis**

In AFMS classification procedure, steps 2 and 5 are the most expensive ones, requiring $O(N)$ classifications. Thus, with the rate of file edit operation $R_{edit}$, the algorithm needs to perform $O(NR_{edit})$ classifications. The optimization does not change the worst case time complexity because it is possible that all the models are triggered for a particular dataset. However, given the presence of natural class imbalance in user-activity datasets, our optimization significantly improves the average case time complexity. For
instance, with $\tau = 0.5$, testing of a single file on an average requires approximately 0.5 metadata- and 0.2 CF-based classifications out of 30 evaluation users. This reduces the classification time of metadata and CF models by 62 and 169 times respectively. The next section provides detailed performance results.

### 4.5.2 AFMS Evaluation

We measure the performance improvement using a simple metric:

\[
    Speed-up = \frac{\# \text{ models applied without AFMS}}{\# \text{ models applied with AFMS}}. \tag{4.2}
\]

Speed-up provides a direct estimate of reduction in classification time.

Figures 4.10-4.19 show the effect of $\tau$ on speed-up, and various correctness
metrics including average precision, recall, and F-score. The numbers are averaged across all the evaluation users, and all training and testing periods for individual shares.

In general, we observe that speed-up increases exponentially with respect to $\tau$, whereas the correctness metrics degrade almost linearly, with a mild slope, as $\tau$ increases. This clearly shows that AFMS can obtain significant efficiency at marginal cost of accuracy.

As expected, for $\tau = 0$, AFMS does not lose accuracy, e.g., CF model Avg AF values from Table ?? and Figure 4.18 remain the same for all shares. And yet, AFMS provides 4 times average speed-up across all shares for metadata models, and 6 times average speed-up for CF models. The speed-up for $\tau = 0$ is as high as 15 times and 23 times for share A for metadata and CF models respectively. With $\tau = 0.5$, the average speed-up for metadata models is 62 times, and for CF models is 169 times. The most gain is observed for share A as 147 times for metadata models, and 280 times for CF models. However, this threshold affects the correctness, dropping averages across shares. For example, for CF models, the Avg AF changed from 49.6% to 45.8%, Avg AR from 48% to 33.3%, Avg AR@75P from 44.8% to 36.9%, but slightly improving Avg AP from 38.9% to 42.5%. These results confirm the expected effect as discussed in Section 4.5.1.

### 4.6 Related Work

Several previous approaches have addressed the topic of file access prediction, but with the goal of improving the I/O performance of storage systems [5, 134, 66, 140, 138, 139, 129, 96]. They aim at reducing the widening gap between CPU and disk storage performance by prefetching files to the cache memory. There are a couple of key differences between our approach and the previous ones. Such approaches focused on modeling accesses patterns that are generated by automated activities, whereas our focus is on activities that are manually initiated by physical users. This allows us to consider
certain features that could be prohibitively expensive for caching applications. None of the previous approaches make predictions for completely new content. On the other hand, by leveraging innovative features derived from metadata and user collaborations, our approach can make fairly good predictions for new content. In fact, our goal is to be able to discover useful new content, which could be present in new or modified files.

The approach from Song et al.[113] is closest to ours in terms of providing recommendations to knowledge workers. This approach infers abstract tasks and frequently used workflow patterns from historical user activity. It then makes recommendations based on the workflows that match current activities of the user. Although this approach extends beyond simple file matching, it cannot make predictions for new content.

In terms of features, our approach is significantly different from any of the previous approaches. Our features are derived from rich file metadata, including file name, path, extension, and file system hierarchy. These features allow our system to compare new file activities to the learned activities in the past in terms of similarity in the metadata attributes. More importantly, our system leverages metadata features to automatically generate features for collaborative filtering in an interesting way so as to address the cold start problem (i.e., inability to provide recommendation for new content). In that respect, our approach is better than the traditional collaborative filtering approaches [73, 11] because they suffer from the cold-start problem.

Lastly, advanced machine learning models, e.g., factorization machines [99], topic models [86, 94], and deep neural networks [102, 53], could be used to model access patterns. However, these techniques are complementary and can be applied to make our system more effective. Nonetheless, we demonstrate that simple linear SVM model can be effectively used to build a practical system. Most importantly, it allows us to develop the optimization technique AFMS for significantly reducing the classification time and improving the scalability of our system. This technique, in addition to the basic modeling
approach, constitutes a novel contribution of our research.

4.7 Summary

This chapter presents a system to assist knowledge workers in discovering useful new or modified content. The system builds personalized user models using features derived from file metadata and user collaboration. Our experiments showed that the basic modeling approach does not scale well under heavy workloads. To address this problem, we developed a novel optimization technique that improves runtime performance without sacrificing prediction accuracy. We also showed how the technique can be adapted to improve the performance significantly at marginal cost of the predictive correctness.

For future work, we could improve our system along different directions. Leveraging interactions observed between the features, we could reduce dimensionality of the data for better efficiency and effectiveness. We could implement a weighing scheme that gives more importance to the recent activity as it is more reflective of the current preferences of users. Finally, we could incorporate features that capture producer-consumer relationship between users.

4.8 Acknowledgements

We thank the anonymous ICEIS 2015 and IEEE Access reviewers for their feedback and comments on the work. We are thankful to the researchers at Symantec Research Labs for their insightful discussions and comments on the work from an early stage. The research was primarily conducted when the dissertation author was an intern at Symantec Research Labs.

Chapter 4, in part, contains material as it appears in the Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS ’15). “Access Prediction for Knowledge Workers in Enterprise Data Repositories”. Chetan Verma,
Sandeep Bhatkar, Michael Hart, Aleatha Parker-Wood, Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, contains material as it has been accepted for publication for the IEEE Access. “Improving Scalability of Personalized Recommendation Systems for Enterprise Knowledge Workers”. Chetan Verma, Sandeep Bhatkar, Michael Hart, Aleatha Parker-Wood, Sujit Dey. The dissertation author was the primary investigator and author of this paper.
Figure 4.12. Precision vs. $\tau$ for metadata models

Figure 4.13. Recall vs. $\tau$ for metadata models

Figure 4.14. F-score vs. $\tau$ for metadata models

Figure 4.15. Recall@75%Precision vs. $\tau$ for metadata models
Figure 4.16. Precision vs. $\tau$ for CF models

Figure 4.17. Recall vs. $\tau$ for CF models

Figure 4.18. F-score vs. $\tau$ for CF models

Figure 4.19. Recall@75%Precision vs. $\tau$ for CF models
Chapter 5

Conclusion

Personalization services have seen a significant upsurge in the past two decades, thanks to the explosive growth in data available to users, and the proliferation of the Internet. In this dissertation, we identified three automation and efficiency related challenges faced by personalization systems and presented solutions. We summarize these below, along with a few directions of future work.

Content classifiers are an integral part of most personalization systems. The set of categories into which the content needs to be classified, is typically dictated by the specific personalization application. Approaches for classifier training require significant manual effort to obtain and label the training data, making the overall process labor intensive, non scalable, time consuming and subjective. Thus in Chapter 2, we developed a fully automated framework to select and label training data given an arbitrary set of categories. We provided a discussion on the desired properties that make a training data lead to high performing classification models. We then proposed two methods that can select keywords to obtain training data with the desired properties. Our methods (LCPD and AAO) offer a very convenient trade-off between tunability and performance. We focus on text based classification of online videos from YouTube. Experiments showed that the proposed framework can successfully obtain training videos in a completely automated manner, and that the resulting classification models can achieve a high performance. As
further extensions to this work, it would be interesting to understand the benefit obtained by using different types of modalities to select the training data. For example, for the classification of online videos, understanding the trade-off offered by textual, audio and visual features over only textual features, in terms of classification performance, speed of model training and application, and robustness of the keywords based process, would be useful.

In addition to automating content classification, we identified that a significant portion of online content, such as Flickr images, suffers from sparsity of annotations. Sparse or missing tags inhibit a personalization system’s ability to recommend content or to infer user interests in terms of tags. In order to address this challenge in an effective and a space-efficient manner, in Chapter 3 we defined ontological tag trees and showed how they can be constructed based on corpus statistics and semantic relationships. We formulated the construction of tag trees as an optimization problem solved through a novel local search based approach. We validated the utility of the constructed tag trees over two large image corpora – one from Flickr, and one from professionally annotated stock images. The evaluation of tag trees was done through two data driven tasks, involving associating images with tags – one by considering previously applied tags, and one without any knowledge of existing tags. We further showed that the proposed approach is robust to tag noise and differences in distributions of the training and test images. The work can be extended in several important directions, for example by constructing directional trees or trees that capture higher degree statistics as obtained from a given corpus. Furthermore, it would be useful to study how tag trees could be used to predict label noise in annotated content or to make content classification more effective.

Lastly, we studied how file recommendation system could be useful for enterprise users to discover relevant new content. In order to do so, we first designed a system that
leverages file metadata to learn per user access models, and uses these models to determine if a new file would be relevant to the user or not. We showed how user collaboration could be used to improve the effectiveness of our models, without being affected by the cold start problem like traditional collaborative filtering based recommendation systems. Furthermore, we experimented with real world data and noted scalability limitations of per user modeling based recommendation. In order to address this shortcoming, we proposed a novel optimization that can select user models that may be interested in a given test file. Based on the proposed optimization, we show that the recommendation can be made more than 20 times faster without any loss in the correctness. We also provide a method that can trade-off the model correctness for significant speed up. For example we show that the recommendation can be made more than 169 times for marginal loss in the F-score. In future, it may be useful to study how different dimensionality reduction approaches can improve system efficiency as well as effectiveness. For instance, it may be possible that the speed and correctness of the proposed models system can be further improved by capturing the interdependence between different features through dimensionality reduction techniques such as Principal Component Analysis. Furthermore, leveraging the temporal information during model training may lead to more effective models. Lastly, directed producer-consumer relationships between users could potentially be derived from the file access data, which may be further useful for recommending content to users.
Bibliography


[120] Peter Van Der Putten, Joost N Kok, and Amar Gupta. Why the information explosion can be bad for data mining, and how data fusion provides a way out. In Proceedings of the International Conference on Data Mining. SIAM, 2002.


[141] Zhijun Yin, Rui Li, Qiaozhu Mei, and Jiawei Han. Exploring social tagging graph for web object classification. In *Proc. of SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2009.


