COMPARING APPROACHES FOR WEIGHTING APPLICATIONS SPECIFIC DATA IN MULTI-APPLICATION USER INTEREST MODELING

A Thesis

by

ATISH KUMAR PATRA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Frank M. Shipman III
Committee Members,	James Caverlee
	A.L. Narasimha Reddy
Head of Department,	Nancy Amato

August 2014

Major Subject: Computer Engineering

ABSTRACT

This thesis presents a framework known as User Interest Modeling and Personalization (UIMAP) which builds a model by identifying and aggregating an individual user's interest expressed through their interactions with different applications at different times. To do this, we have implemented a content consumer/producer architecture. For this thesis, Microsoft Word and PowerPoint are treated as content producer applications while a web browser is used as a content consumer application. We unobtrusively observe user interactions with these applications as well as the actual content consumed/prepared in them. The challenge is to understand the importance of each application towards the user's real interest. Based on user activity data in these applications, Multilayer Perceptron (MLP), Support Vector Machine (SVM) and Weighted K-Nearest Neighborhood (WKNN) techniques are compared in their ability to combine these kinds of heterogeneous interest indicators into a single model. Thus, each application is weighted differently based on its contributing indicators to predict the relevant content for the specific need of an individual. We found that textual content from content producer applications plays an equally important role as content from consumer applications. Implicit feedbacks from consumer applications also have a major role in user's interest. The results indicated that WKNN is preferred if feature weighting is the primary goal while SVM is the preferred choice if identifying relevant content is the main objective.

DEDICATION

This thesis is dedicated to my loving parents, my brother and Lopa for motivating me, to my wonderful friends for their unconditional help and to Dr. Frank Shipman for constant inspiration and guidance.

ACKNOWLEDGEMENTS

First and foremost I offer my sincerest gratitude to my advisor, Dr. Frank Shipman, who has supported and guided me throughout my research with his patience and knowledge whilst trusting me to work in my own way. I would like to express my gratitude to Dr. James Caverlee and Dr. A.L. Narasimha Reddy for being a part of my thesis committee.

I would also like to thank Sampath Jayarathna and Su Inn Park for providing valuable inputs and helping me throughout my project, and all my friends for believing in me and being there for me whenever I needed them. I am grateful to everyone who has helped me in some way or another throughout the entire duration of my studies.

Finally, thanks to my parents for their encouragement and to Lopa for her patience and love.

NOMENCLATURE

UIMAP	User Interest Model and Personalization
IPM	Interest Profile Manager
UC	User Characteristics
DC	Document Characteristics
TC	Textual Characteristics
SVM	Support Vector Machine
MLP	Multilayer Perceptron
KNN	K Nearest Neighbor
ADM	Average Distance Measure
RLD	Relevance label difference
ML	Machine Learning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Personalized Information Delivery

"Information overload" has been used so widely in the literature in the last decade that it is on the brink of becoming meaningless. Rather than focus on overall quantity of available information, much of the focus has shifted more towards personalized information delivery to get users the right information at the right time. However, a personalization framework requires an understanding of a user and in particular their interests or current information needs. Therefore, it is interesting to first discuss what exactly personalized information delivery means and how user interest models have evolved to support it.

In short, personalized information delivery involves creating user centric systems rather than generic user system. These systems can gradually adapt to a user's behavior by learning from it and providing visualizations of relevant content that are personalized to the user. User interest modeling is the key to the success of these systems. User models can be developed by adapting the content consumed or produced by the user, and their specific task, background, history and information needs [1]. These models can bring user’s attention to more valuable and personalized content of a page rather than the entire web page. There are three general approaches to user interest modeling with different levels of complexities.

1. Hand-authored models [2] - These models are inflexible and cannot be scaled to accommodate new features or applications. They also require user input whenever information needs change.
2. Learned models [3, 4] - These models use machine learning algorithms to understand the behavioral patterns of the user. They require no action on part of the user and are able to adapt to changes in user interest although modeling a new user interest can require observation of a significant amount of user interaction.
3. Collaborative models [5, 6] - These models take care of the cold-start problem, where a new user with virtually no previous interest will be presented with information based on interest shown by similar users. Even though they are scalable, they implement a shallow model due to their broad generalizations that may not hold for specific users.

The interest model presented in this thesis belongs to the second category. The next question that naturally arises is what actually contributes to the user interest.

1.2 Multi-Application Interest Model

Individuals spend considerable time in multiple applications as they move back and forth from consuming content (e.g. in browsing and reading applications) and creating content (e.g. creating reports or presentations). In such a context, interaction with each application provides unique and useful information about the user's interests. A challenge for multi-application user modeling is that the quantity of usage and content information obtained from the applications can vary widely. For example, content creation applications (e.g. Microsoft word or PowerPoint) are likely to include less content than the amount consumed yet it tends to be more indicative of the user’s interests. The proposed user interest model called *User Interest Model and Personalization (UIMAP)* includes both spatial and temporal information presented within the day-to-day applications such as Microsoft Word, Microsoft PowerPoint and Mozilla Firefox browser. It seeks to incorporate the user's activity or preferences in all these applications by understanding the actual value contributed from above mentioned applications.

1.3 Implicit and Explicit Features

In a multi-application environment, user interest can be recorded either explicitly or implicitly. Explicit interest indicators require the user to rate relevant content/page after reading or skimming them. They are the most accurate indicator of user interest but are hard to obtain. Implicit interest indicators are based on user's actions or behaviors rather than explicit relevance judgments. These are collected by unobtrusively observing user's interactions (such as mouse/keyboard events, time spent in various activities) with the everyday applications and extracting the properties of the used documents. Although these are easy to obtain, this information must be interpreted to provide the desired interest information. Past research studies [7-9] have already proved that implicit feedback can provide valuable feedback. Our proposed frame UIMAP provides a unique way of combining these heterogeneous interest indicators to a single model to identify relevant content to the user.

1.4 Feature Weighting

Feature selection assigns binary weights while feature weighting assigns real numbered weight values. Thus, the importance of a feature can be quantified and set relative to other features. Feature selection algorithms perform best when individual features are either highly correlated with the class label or totally irrelevant to it. However, feature weighting is more appropriate in cases where features vary in their relevance [10]. In our multi-application information environment, there is no particular single pattern to the usage of these applications. They vary from one user to another because of many factors such as skill level, educational background, task at hand, timing constraints, etc. The features extracted from these applications may contribute differently towards the interest model from one user to another. Thus, feature weighting is necessary to model the importance of individual features in such a heterogeneous environment. Feature weighting algorithms can be broadly divided into two categories.

1. Performance Bias: This is also known as wrapper methods in the machine learning community. It uses feedback from a performance function during the training phase to include the classifier's bias during weighting. The primary advantage of this approach is that the selection of feature weight values is guided by how well those values perform in classifier evaluation.
2. Preset Bias: This is also known as filter methods. This method includes a pre-determined bias during a pre-process step. It is mainly data driven and weights are assigned based on heuristic measure of the data.

Preset bias weighting methods mainly include statistical methods such as information gain, entropy, gini index and $\chi 2$ statistics. Performance bias weighting methods involve observing classifier performance before computing weights for each feature. As we intend to suggest relevant content and compute weights of each feature in for a variety of user behaviors, performance bias weighting methods are better suited to the problem studied in this thesis. Moreover, research has shown that performance bias methods perform better than preset bias approaches in many situations [11]. There is no single universal weight learning method that can learn optimal weight settings for all learning tasks since each task requires different learning biases for optimal performance [12]. We have implemented Multilayer Perceptron (MLP), Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) classifiers for both identifying relevant and irrelevant content as well as deciding the importance of each feature. The main contributions of the proposed work are:

- To the best of our knowledge, this is the first-of-its-kind research where a user interest model is built from activity in a web browser, word processor and presentation authoring application.
- Collection of ground truth data set that consists of implicit and explicit feedback available in multiple everyday applications during an information-gathering task and the users' post-task relevance assessments.
- It compares three feature weighting methods that can infer the importance of different interest indicators extracted from different applications.

The remainder of the thesis is organized as follows. The research motivation is presented formally in the next chapter. A literature review on interest modeling in multiple and single application environments and weighting methodologies is discussed in Chapter 3. Chapter 4 addresses the prior work on Interest Profile Manager and WebAnnotate plug-in. The current approach and system design are discussed in Chapter 5 while Chapter 6 describes the evaluation methodology and discusses the results. Finally, conclusions and directions for future work are presented in Chapter 7.

CHAPTER II

RESEARCH MOTIVATION

According to a study by Neilsen and Norman Group [13, 14], web users spend 80% of their time looking at the content above the page fold, i.e., portion of the webpage that is visible without scrolling. They tend to read at most 28% of the words during an average web page visit. This usually happens because user tend to skim or stop reading when they feel what they have is good enough [4]. However, most of the existing research in user interest modeling treats documents as a monolithic unit whereas users tend to express more granular levels of interest in the form of sub-topics within one document. This leads to the question whether we can bring user’s attention to the portions of pages of most value by understanding the user's real interest?

Modeling user interests to achieve this goal is challenging because users often cannot or will not express their interests. Additionally, recognizing user interest based on observed user activity is confounded by idiosyncratic work practices. As a result, systems that aggregate evidence of user interest from a wide variety of sources are more likely to build a robust user interest model. A majority of past studies have focused on monitoring only implicit/explicit interest indicators present in a single application, e.g., the user’s web browser. However, real world activity spans across many applications. As different applications are used to achieve different objectives related to a single information seeking task, each application carries its own value and may contribute uniquely towards the user's interest. We prepare presentations when we want to put forward our view in a short and concise way, whereas we write a document in cases where more explanation is required. Thus, user activity in these content authoring applications can help build a better interest model. Explicit and implicit interest feedback from these applications can augment the interest observed through browser use to improve the understanding of the user interest.

However, how do we merge these sources of evidence of user interest? Each application can provide multiple forms of user activity and the system needs to balance their contribution to the final interest model. Thus, it is important to understand the contribution of each of the interest evidence (such as time spent, mouse actions, content similarity, etc.) from each application towards the final model. Because each user and task can result in a unique work activity, it also helps to understand the value of each application with respect to individual users. Because no single set of weights will work across diverse work practices, we have implemented various different machine-learning approaches to build the user model and extract the weights assigned to individual features after classification.

The research goal studied in this thesis can be stated as follows:

To compare alternative machine learning approaches for building a multi-application user interest model that understands the relative importance of user activity in individual applications towards identifying relevant content.

CHAPTER III

LITERATURE REVIEW

The proposed research work explores issues at the intersection of user modeling, personalization, implicit and explicit feedback, multi-application environments and feature weighting. This chapter presents a focused review on relevant aspect of each of these areas.

In past studies, several systems have adopted different machine learning (ML) and knowledge based (KB) approaches to build the user interest model. LaboUr [15] have proposed a hybrid approach that combines both KB and ML to build a user interest model in a web based information system for research funding opportunities. However, they consider only positive examples during the learning phase. Probabilistic approaches have also been used in the form of Simple Bayesian classifiers (SBC) [16] and "Syskill & Webert" [17] to infer the user interest in a corpus of text documents. Both approaches require users to rate documents to generate the user interest model. Lam and Mostafa (2001) [18] implemented a user interest model based on Bayesian framework. This model focused on interest from documents only and tried to detect interest shift over time. Our previous research [19] successfully created a semi-explicit feedback based user model using Latent Dirichlet Allocation (LDA). But, the model only involved interest expressed through the browser and did not consider other applications such as Microsoft Word and PowerPoint.

Previous research in our lab has already shown that combining evidence from a multi-application environment improves the interest model [20]. There are a variety of methods for combining information from various applications. Work in our lab has considered a document organizing application known as Visual Knowledge Builder (VKB) and document reading application Mozilla Firefox. The current work includes additional content producer applications such as Microsoft Word and PowerPoint to understand the user interest better. Moreover, the prior work proposed a static mathematical interest model while we have explored different machine learning algorithms to build an interest model tailored to the individual's work practices. The main purpose here is to understand the user's interests and preferences to provide services catered specifically to their needs. Analyzing user interaction records to improve support is common. For example, UMEA [21] and Task Tracer [22] record information about user's activities with the computer and try to infer task profiles. The goal of these systems is to help users access past records of documents and quickly restore the historical context. However, they do not explore the importance of the individual applications that actually contribute to the user task profile. Closer to our approach here, the Watson project [23] included a word processor and browser in their system to understand the context of each query and modify queries to produce better results. We are not concerned with localized query expansion but modeling user interests to adapt the presentation of documents. Another approach to gathering information across applications is to look at the text near the user's focus. IntelliZap [24] made context more coherent and focused on a specific topic by extracting lexical meaning of user selected text and its vicinity.

User interest modeling may either be based on explicit interest indicators (e.g. ratings) and implicit indicators, from a single application or multiple applications. In [8], the number of hyperlinks and amount of scrolling are used to predict the user interest and the learned model based on three layer artificial neural network. In [9], the authors used the reading time and the number of scroll events in a browser to determine relevant content. The Curious Browser [25] used several features of use such as mouse usage, keyboard usage, and time spent viewing the page to predict user interest. These systems provide insight into how to combine evidence from different types of data (e.g. different features) but do so in the context of a single application, e.g., the web browser.

In a multi-application environment, it is important to understand the value of each type of user interest evidence in each application. Feature weighting addresses this problem by assigning continuous weights to each element in the total feature space. Different performance bias algorithms have been used in different classification algorithms to compute real valued weights for each feature. Locally Weighted Naive Bayes (LWNB) [26] was one of the earliest approaches to understanding feature importance in naive Bayes (NB) classifiers. It used the Euclidian distance between the n-dimensional feature positions as a distance measure. Wang and Zhang (2007) adapted this approach by including a probabilistic metric such as Inter Value Difference Metric (IVDM) or Minimum Risk Metric (MRM) [27]. An information theoretic method using KullBack-Leibler divergence was used in Feature Weighting Naive Bayes (FWNB) [28] to calculate feature weights. All these naive bayes based methods assumed the features are independent and used Fayyad and Irani's discretization [29] approach in cases where conversion from continuous or numerical attributes to nominal features is required. Since our features are interdependent, we believe discretization of the feature space in Naive Bayes approach would not be suitable for our dataset. Recursive Feature Elimination [30] is an excellent method to compute feature ranking in Support Vector Machine (SVM). Since it only works well for linear SVM, it would not be a good choice for our case due to the high non-linearity decision boundary nature of our data set. Salzberg [31] proposed a variant of nearest neighbor algorithm known as EACH to compute feature weights. It used leave-one-out process to increment the matched feature weights and decrement the mismatched features. Wettschereck et al. presented a good summary of other feature weighting methods in his review paper [10]. This work takes advantage of this more general research in feature weighting techniques and applies it to the domain of multi-application interest modeling.

CHAPTER IV

PRIOR WORK

Prior work at Texas A&M University on the Interest Profile Manager (IPM) and the WebAnnotate client [32] provides the basic building blocks for the UIMAP framework proposed in this work. The IPM is designed to act as the central server for aggregating and storing evidence of user interest.

4.1 Interest Profile Manager (IPM)

Each application in the framework acts as a client to the IPM server. This is illustrated in Figure 1. IPM has three main modules: (1) *Request Handler* (RH), (2) *User Profile Handler* (UPF) and (3) *Inference Manager* (IM). The requests from different clients are received and analyzed by the RH. Then the RH invokes the UPF to update the user events or document attributes if required and forward the request to the IM. Upon receiving a request from the RH, IM interacts with the UPF to get the complete user profile and infers various user interests depending on the type of the request. The UPH then collects the partial interests of the user as they appear and aggregates them to form a complete user profile. It also serializes the entire user profile and saves it in an xml format for post processing. Due to the client-server model, IPM can be easily extended to support additional applications as clients. For example, a new viewing/reading application (e.g. Adobe Acrobat) can be added to enable the user to work with a new content type (e.g. PDF). Any new application can become a client in IPM architecture by exchanging data about the user in a predefined format. These client applications can be implemented to support one way communication in which they either provide information to the IPM or receive information from the IPM, or two way information (thus, both providing and receiving information from the IPM). In the prior implementation, client extensions were developed for two applications, i.e., VKB3 [33], a spatial hypertext workspace for collecting, analyzing and organizing documents and Mozilla Firefox, both supporting two-way communication. This work continues to use the Firefox extension, called WebAnnotate, which is described next.

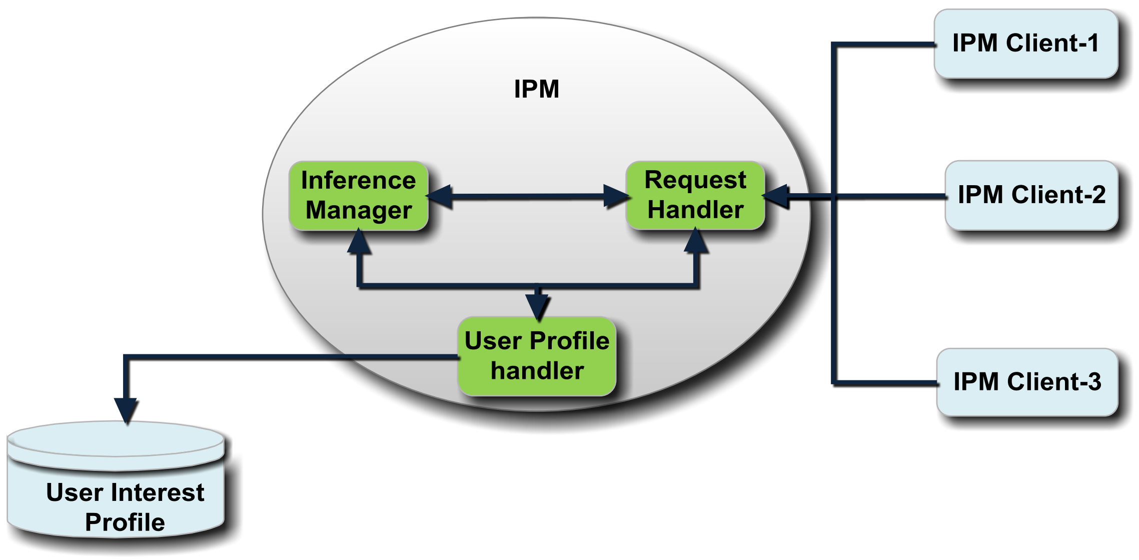
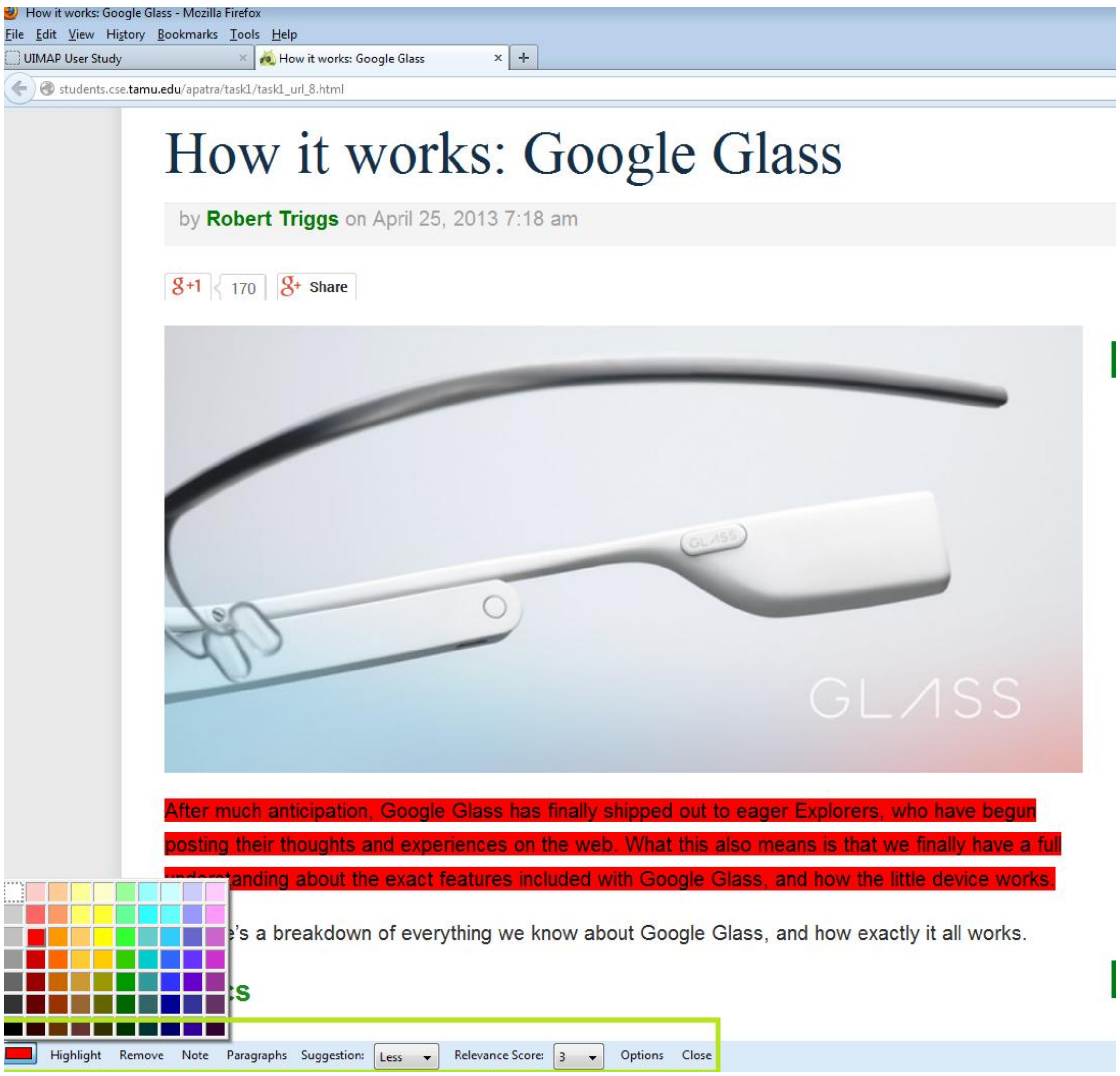


Figure 1: IPM module interaction with clients

4.2 WebAnnotate

For many, the browser is the most commonly used content consumption application in today's world. WebAnnotate is a plug-in developed for Mozilla Firefox that provides basic annotation capabilities and collects data on users' interactions with web pages. It can also generate visualizations to help users quickly identify relevant portion of the documents. However, the work presented here focuses on building a better user interest model rather than visualization. WebAnnotate supports several forms of annotation on HTML documents through interaction with the annotation bar (see bottom


How it works: Google Glass - Mozilla Firefox
File Edit View History Bookmarks Tools Help
UIMAP User Study
How it works: Google Glass
students.cse.tamu.edu/apatra/task1/task1_url_8.html
+1 170
Share
GLASS
Highlight
Remove
Note
Paragraphs
Options
Close

How it works: Google Glass

by Robert Triggs on April 25, 2013 7:18 am

After much anticipation, Google Glass has finally shipped out to eager Explorers, who have begun posting their thoughts and experiences on the web. What this also means is that we finally have a full anding about the exact features included with Google Glass, and how the little device works.

's a breakdown of everything we know about Google Glass, and how exactly it all works.

Suggestion: Less
Relevance Score: 3

Figure 2: WebAnnotate tool

of Figure 2). This includes highlighting text in different colors and attaching notes on top of the HTML document. WebAnnotate stores the reader's annotations separately from the HTML document in the IPM.

4.3 Microsoft Word/Power Point Client

Apart from email, report and presentation preparation are two of the most common content production activities for many users. Microsoft Word and Power Point are commonly used applications for these purposes. Add-ins were developed in C# for both of these applications [34]. These add-ins act as a client for the IPM. They parse the text content in the document or presentation and record a variety of user interactions with the application. At a specified interval, the add-ins send this data to the IPM for interest aggregation. This provides the IPM with information about what the user is working on so that it can be included in the interest model. The add-ins only support one-way communication with the IPM as there is currently no need for the IPM to send information about the user's interests to these applications.

CHAPTER V

APPROACH

The discussion so far explained the motivation for the research and the prior work done on UIMAP. This chapter will focus on specific approaches adopted to solve the problem and extensions to the software infrastructure necessary to explore them.

5.1 System Description

The UIMAP framework presents content consumer-producer architecture for collecting and processing the user activity data. In an information seeking task environment, user interest is distributed in both *content-consumer* (Browser) and *content-producer* (Word/PowerPoint) applications. Here we define *task* as a set of user activities carried out to achieve a specific goal in mind.

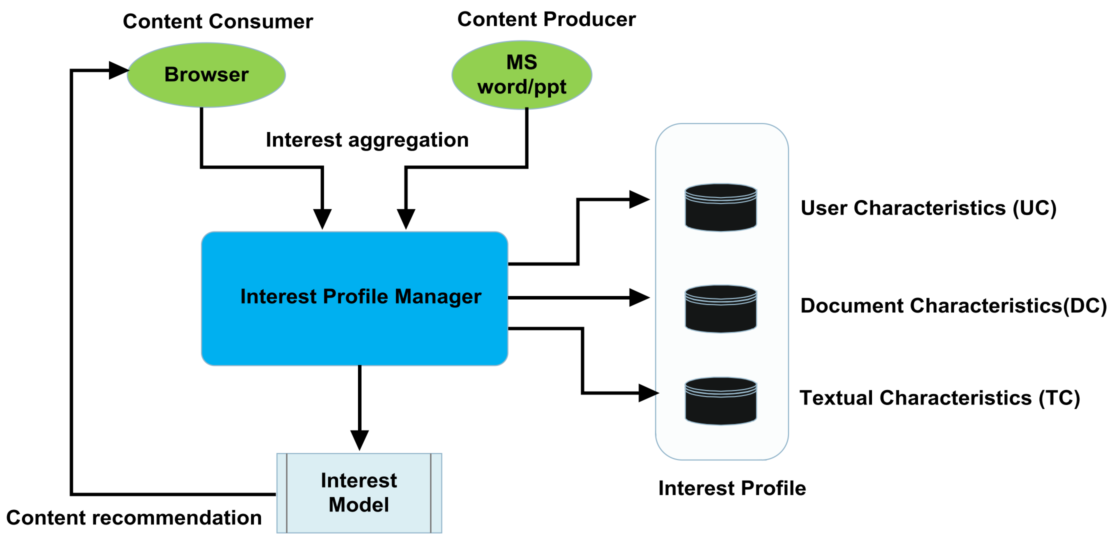


Figure 3. UIMAP architecture

The proposed work includes three applications, i.e., Mozilla Firefox, Microsoft PowerPoint and Microsoft Word. We have considered Microsoft PowerPoint and Word as the *content-producer* applications and Firefox as the *content-consumer* application. Figure 3 shows the architecture of the UIMAP framework. The IPM acts as the core component in UIMAP, which saves different kinds of interest indicators, while the user gathers/searches/creates information in multiple applications. A systematic workflow, as shown in Figure 4, is designed to make the user interest modeling process accurate and scalable.

Step 1. *Preprocess:* This step establishes a socket connection between the IPM clients such as Web Annotate and the Microsoft Word and PowerPoint plug-ins and the IPM server. Individual plug-ins register event handlers for different mouse/keyboard interactions. Any webpage loaded in the web browser is parsed and all the scripts, advertisements, and dynamic content are removed before text extraction.

Step 2. *Extraction:* Each plug-in (WebAnnotate in Firefox and the Microsoft Word and PowerPoint plug-ins) is responsible for extracting the user interest evidence from their respective applications and communicating this to the IPM. Additionally, they send any user-created annotations (for WebAnnotate) and static characteristic of each webpage to IPM.

Step 3. *Aggregation:* The IPM collects the various user interest evidence data, i.e., both implicit and explicit feedback, and converts the data to a common format. It saves all the

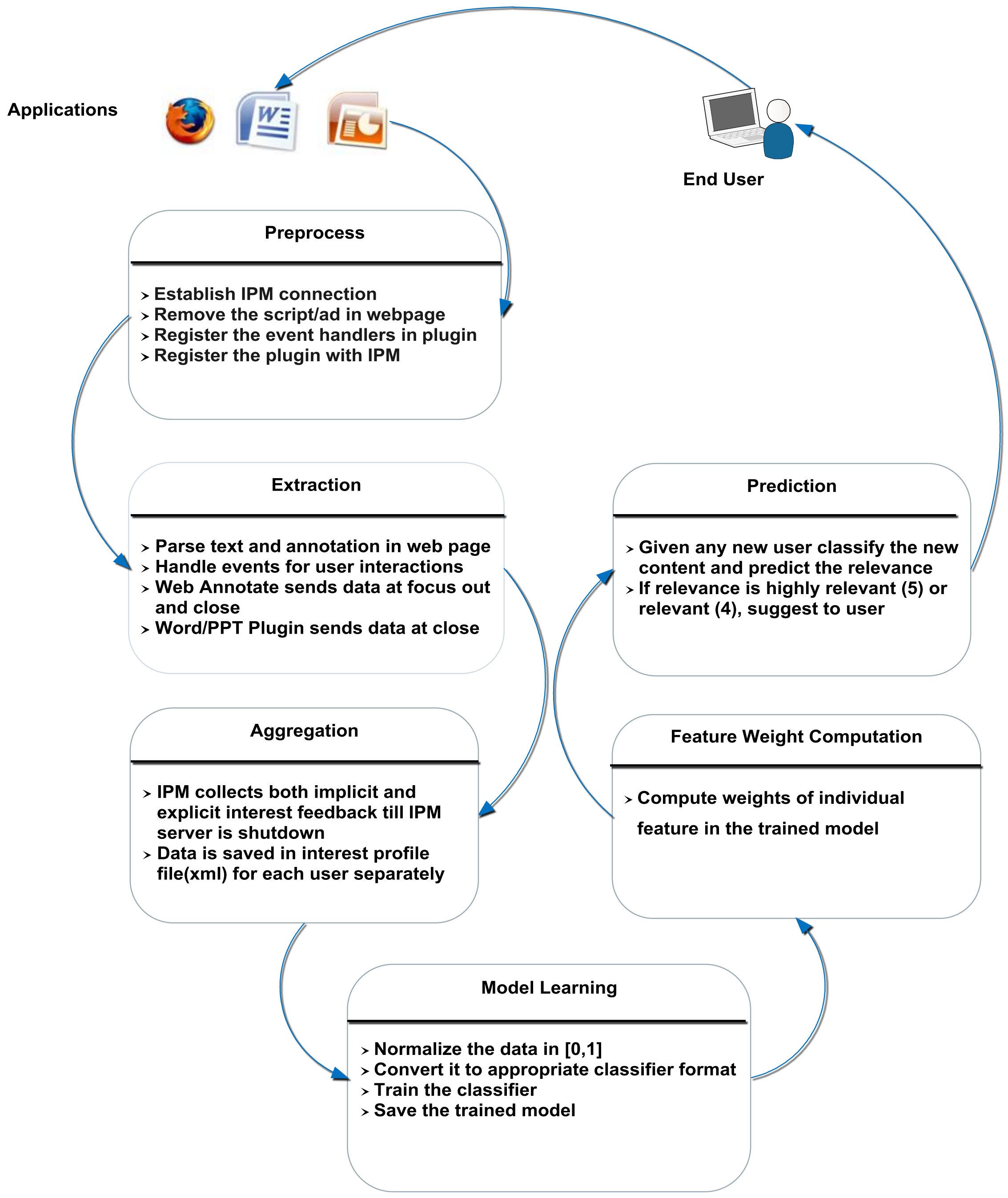

Applications
End User
Preprocess
› Establish IPM connection
› Remove the script/ad in webpage
› Register the event handlers in plugin
› Register the plugin with IPM
Extraction
› Parse text and annotation in web page
› Handle events for user interactions
› Web Annotate sends data at focus out and close
› Word/PPT Plugin sends data at close
Aggregation
› IPM collects both implicit and explicit interest feedback till IPM server is shutdown
› Data is saved in interest profile file(xml) for each user separately
Model Learning
› Normalize the data in [0,1]
› Convert it to appropriate classifier format
› Train the classifier
› Save the trained model
Feature Weight Computation
› Compute weights of individual feature in the trained model
Prediction
› Given any new user classify the new content and predict the relevance
› If relevance is highly relevant (5) or relevant (4), suggest to user

Figure 4. System workflow

data in a *profile.xml* file for later processing. The data is continued to be updated until the IPM server is shut down.

Step 4. *Model learning:* The interest evidence data stored in step 3 is parsed and segmented by the UIMAP module to separate interest evidences from different tasks for a single user. This process also generates the evidence features accepted by the classification algorithms used to identify relevant content. The evidence data is used to train the model, resulting in attribute weights for the features based on the evidence of user interest.

Step 5. *Feature Weight Computation:* As each application contributes towards user interest via features extracted from it, a learned model can express the application's importance in terms of feature weights. Depending on the modeling technique, these weights can be observed after the model is trained or can be expressed as a quantitative function of features in case the feature space is mapped to a higher/lower dimension during classification.

Step 6. *Prediction:* Finally, the learned model is used to predict user interest by recommending relevant content to the user.

5.2 Multi-Grade Relevance

Content relevance values should be continuous rather than dichotomous [36]. Many studies have shown that documents are not always equally relevant to different users ; relevant documents are more relevant to some users while they appear to be less relevant to others, thus relevance has multiple degrees [35, 36]. This conclusion can be naturally extended to the subsections and paragraphs of a document. Thus, user needs and preferences are better identified if multi-graded relevance judgment is considered instead of binary relevance judgment [37]. We have adopted a 5-point scale (very relevant, relevant, somewhat relevant, only slightly relevant, and non-relevant), proposed by Maron and Kuhns (1970) [38], for relevance assessment of the content at both paragraph and document level throughout this thesis. From here onwards we denote ***C*** as the relevance label. Figure 5 shows the annotation toolbar used to assign ratings to each paragraph.

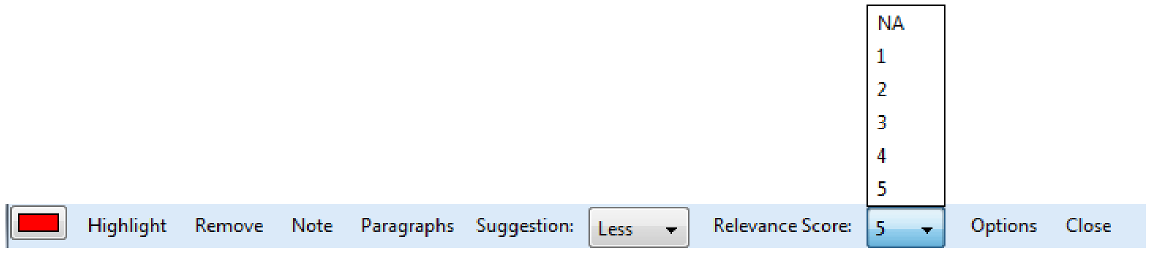


Figure 5. Annotation toolbar for rating paragraphs

5.3 Interest Profile

An interest profile is made up of the aggregated heterogeneous interest evidence collected from the IPM clients. The interest profile is saved in an XML format during the information gathering tasks and is then processed later. It broadly contains three types of interest indicators, which have been summarized in Table 1.

Table 1. Different interest indicators

Interest Category/Application	Microsoft Word/PowerPoint	Browser (Firefox)
User characteristics(UC)	Click, double click, right click, focusIn/Out, total Time, edit time, idle time, away time	Click, double click, right click, focusIn/Out, total Time, reading time, away time, number of scroll, number of scrolling direction changes
Document characteristics(DC)	Size, number of characters, images, links, last access time, last write time, create time, number of Slides, text boxes	Images, links, document relevance and readability score(explicit)
Textual characteristics (TC)	Text edited (explicit)	Text annotated (explicit)

5.3.1 User Characteristics (UC)

These interest indicators are generated by unobtrusively observing the user interactions with both content-consumer and content-producer applications. The features that belong to this category are derived from implicit feedback data. The application plug-ins record different mouse and keyboard interactions such as the number of mouse clicks, double clicks, scroll events and scrolling direction changes. The user's engagement time with the application is also included. Thus, the applications record the total time spent in the application or webpage, the time spent while the application is not in focus and the time spent in creating or consuming the content. All these features vary from one user to another as they heavily depends on individual practices. The nature of the task also leads to different user behavior.

5.3.2 Document Characteristics (DC)

These features depend on the attributes of a specific document rather than user interactions. The web page properties, such as number of links and number of images, are constant across users. Features from content-producer applications include file size, number of characters, last access time, creation time, number of images, links, slides, and text boxes. The above mentioned features are considered implicit feedback as they are gathered from the document silently while the user is performing their task. The IPM can also collect two types of explicit feedback for each web page visited in the browser. Users can volunteer (but in our study the users are asked) to rate the relevance of the webpage to the task and the overall readability of the webpage on a 5-point scale.

5.3.3 Text Characteristics (TC)

This class of features is generated from the user's annotations in consumption applications and from the user's produced content. User annotations reflect the user's interest on a topic in a comparatively univocal manner [39]. A similar argument can also be made for content produced in content-producer applications. It is expected that we can build a better user model by combining these two types of interest indicators. This also provides evidence of more focused interest than the general document features in the previous categories. As most documents include discussion of several subtopics as part of a larger topic, this information allows a content similarity algorithm to identify the individual subparts which may be of more interest to the user. As our goal is to help users identify the specific parts of documents that are expected to be relevant, such focused information is likely to be valuable in this process. Vector space representations of text perform well in computing document-to-document similarity in a large corpus of documents while topic modeling is more capable of finding subtopics within a document. Thus, topic modeling is better suited to identify content for the problem at hand. Our previous research [19] has already supported this hypothesis.

5.4 Interest Representation

The different user interest evidences described above are heterogeneous in nature and differ from one application to another. A common representation is necessary to train our models to learn the idiosyncrasies of the user interest. Before representing in a unified format, we need to understand the format of each of these features as extracted from their individual applications.

5.4.1 Interest Extraction (UC & DC)

Whenever a document is opened in Microsoft Word or PowerPoint, event handlers are registered for different kinds of user events. Event handlers save each interaction and their values locally and send them in XML format to IPM. Additionally, the content of the document and document characteristics are sent to the IPM at the time of closing the document. Similarly, WebAnnotate parses raw text to identify every paragraph when a new web page is opened. It also appends mouse and keyboard events in a buffer and saves the color and relevance score assigned to each annotation until the browser is moved to the background. All the raw information is sent to IPM in an XML format at focus out event or at the web page close event. The buffer is reset once the focus is brought back to the web page.

5.4.2 Interest Extraction (TC)

Content Similarity metrics are used to measure the overlap between the textual content of the user's previous interactions and any future text content. It is computed between the relevant user annotated or produced text from content-consumer/producer applications and all other paragraphs consumed in the browser. The similarity score represents the user's interest expressed through the textual content. In this work, Latent Dirichlet Allocation (LDA) is used to compute content similarity. LDA considers each document as a mixture of various topics instead of a single atomic unit. It uses a hierarchical probabilistic generative model to express collection of documents by number of topics [40]. Given a number of topics K, a document corpus of W distinct words, two smoothing parameters α (Dirichlet prior on the per-topic word distribution) and β (Dirichlet prior on per-document topic distribution), and prior distribution over document corpus, LDA can find the hidden topics as distributions over the words in the vocabulary. Here, topics are considered as latent random variable while the words are modeled as observed random variable.

The LDA parameters such as ∅, θ, K x W matrix of topics and NxK matrix of topic weights for each document need to be learned for a corpus of N documents. The other parameters α, β and K are chosen as per findings of our previous research [19]. Since our corpus is related to specific tasks, the number of topics is kept small. The optimal number of topics is found to be 5 in this kind of environment [19]. We have estimated ∅ and θ by parameter fitting using collapsed Gibbs Sampling [41]. Two additional parameters for the Gibbs sampling are the number of sampling and burn-in iterations, which we set to 1 and 5, respectively. Mallet framework [42] is used for all these operations. The other two Dirichlet prior values are set as per below.

$\alpha = 0.01*K$

$\beta = 0.01$

To compute content similarity using LDA requires a probability distribution of source content and target content. User generated annotations in the browser and the content produced in Microsoft Word and PowerPoint act as the source while any unseen paragraphs are treated as targets. The LDA model is created from the source content and the topic distribution is saved. The topic probability distributions for the target paragraphs are estimated from the already computed LDA model. The similarity between two probability distributions is computed using Hellinger Distance metric [43]. Figure 6 shows the pseudo code for computing similarity score using LDA.

```
computeContentSimilarityScore(TaskInstanceList tList)
    createTopicModel(numTopics,numIterations)
    for i ← 1 to |tList|
        clearTopicModel()
        sourceContent contentSrc starts empty
        DocumentList docList = tList.allDoc()
        for j ← 1 to |docList|
            appendRelevantContent(contentSrc)
    runLdaModel("source",contentSrc)
    srcProbDist = getSourceProbDistribution("source")
    for k ← 1 to |targetDocList|
        targetProbDist = estTargetProbDist(contentTarget)
        simScore = computeSimScore(srcProbDist,targetProbDist)
        updateSimScore()
```

Figure 6. Pseudo code for computing content similarity

5.4.3 Overall Interest Model

Finally, UIMAP stores all the interest evidence in a vector format where each dimension represents a specific interest indicator. All these individual interest evidence values have different ranges. For example, number of clicks, scrolls, scrolling direction changes, number of images, characters, etc. have real integer numbers while the time related features such as total time, idle time, reading time, etc. have much larger values in milliseconds. As a result, all the values need to be scaled down to the same range ([0, 1]) so that they can be represented in a single vector. We have implemented max-min normalization shown in Equation (1)

$$y_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

where $x = (x_1, x_2, ..., x_n)$ and y_i is the i^{th} normalized data. The final user interest profile is formally represented in a feature vector format in the following way:

$$IP = \{D, DC, UC, \mathrm{TC}\} \tag{2}$$

where $D = \{d_w, d_p, d_b^1, d_b^2, .., d_b^n\}$ d_w / d_p - word/power point document prepared by a user,

$\{d_b^1, d_b^2, .., d_b^n\}$ - set of web pages visited by a user in browser,

$DC = \{dc_1, dc_2, ..., dc_n\}$ - document characteristics of the documents consumed/produced by any user,

$UC = \{uc_1, uc_2, ..., uc_n\}$ - user characteristics of individual users,

$TC = \{tc_b, tc_w, tc_p\}$ - textual characteristics represented by content similarity,

tc_b - between relevant annotated text in browser and other unseen paragraphs,

tc_w - between text produced in a report through Microsoft Word application and other unseen paragraphs

tc_p - between text produced in a presentation through Microsoft PowerPoint application and other unseen paragraphs.

5.5 User Model Learning

Having described the feature space in the prior sections, we next move on to describe the mapping of features to user interest. This thesis explores the use of machine learning methods for training the interest model instead of heuristic or rule-based approaches to understand the user specific patterns present in the feature space. We do so because machine learning methods can easily adapt to new user behavior, new applications, and new tasks. Figure 7 shows the interest model architecture. Different classifier algorithms, implemented for the proposed interest model, are discussed here. Since the feature space has a non-linear decision boundary, we only chose non-linear classifiers. Here are the common notations used throughout this section:

- User interest objects are represented as a real number valued feature space *F*. Unless noted otherwise, it will be assumed that F is a d dimensional vector space, i.e. $\boldsymbol{F} = \mathbb{R}^{d.}$. A training set T is a collection of labeled data points o*f* F: $T = \{x^1, x^2, \ldots\ldots, x^n\}$
- Each labeled data point has binary output data $C = \{c^1, c^2, c^3, c^4, c^5\}$ where 1~5 represent multi-grade relevance values. We treat them as five separate classes. C always denotes the set of class labels.
- $l : X \rightarrow C$ is a mapping function which maps each instance of x to its class label *l(x).*
- Since this is a multi-class classification problem, computing weights corresponding to individual classes presents a better picture about the importance of individual features to specific classes. Attribute weight vectors are represented

as $W = \{w_{c1}, w_{c2}... w_{cd}\}$ where w_{ci} is the individual weight of each of the d features corresponding to class $c \in \{c^1, c^2, c^3, c^4, c^5\}$.

We next describe the three machine learning techniques compared in this thesis: multilayer perceptrons, support vector machines, and weighted K-nearest neighbors.

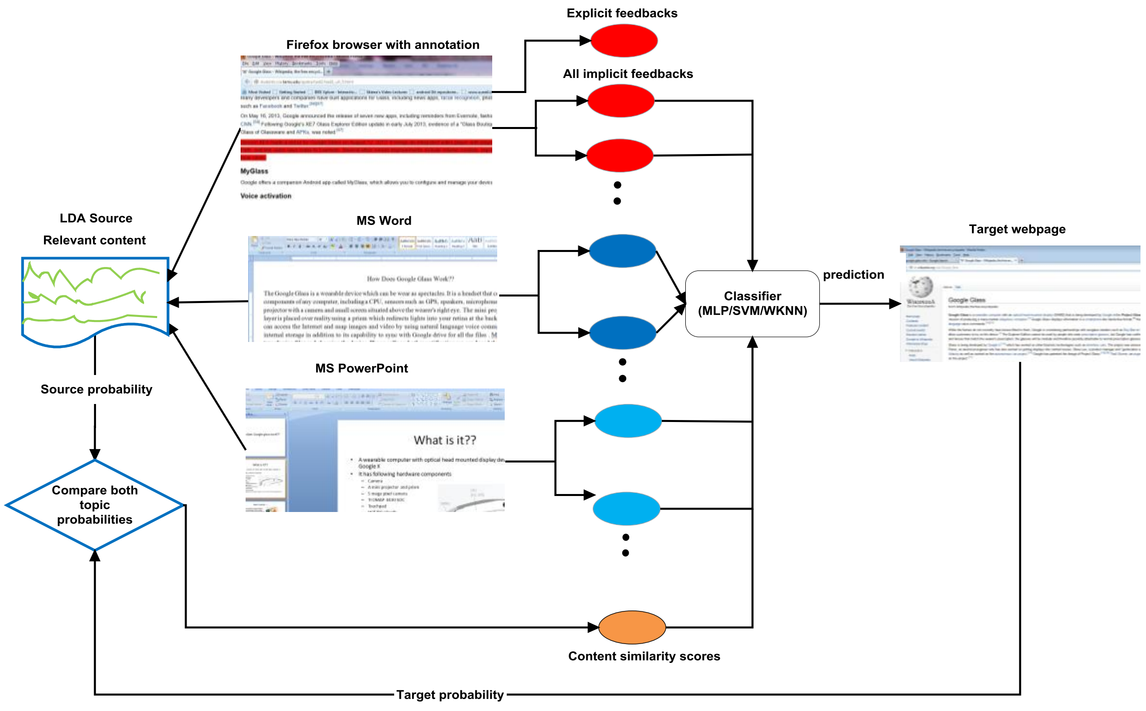


Figure 7. Interest model architecture

5.5.1 Multilayer Perceptron (MLP)

The universal approximation theorem states that a Multilayer Perceptron (MLP) with a single hidden layer and with arbitrary activation function is a universal approximator for any kind of dataset [44]. Moreover, MLP is known to be quite noise tolerant. These two considerations motivate the use of a feed forward neural network as a classifier in our model. The feed forward neural network used in the proposed work is shown in Figure 8. Every Interest indicator from each application appears as a node in the input layer while the relevance labels (1~5) represent the output layer. The nodes in the hidden layer receive input from the previous layer, apply the network connection weights and propagate the value to the next layer. During each epoch, the weights are gradually updated by the gradient descent approach known as "back-propagation". In back-propagation, the error made by the network at the output is calculated and fed backwards to each layer. This error along with a learning rate is used to update the weights. This process is carried out until the stopping criterion is reached. Thus, the network learns to distinguish between different relevance labels. Due to problems of generalization, the number of nodes in the hidden layer and the learning rate are adaptively adjusted to obtain optimal results. Details of these parameter selection approaches are discussed in Chapter VI.

5.5.1.1 Training Stopping Criteria

One of the major issues with MLP classifier is over-fitting of the data. This happens if we continue training based on the total network error computed in the training set only. In this case, training set accuracy increases but at the cost of generalization. Thus, MLP performs badly on the test set. We have implemented early stopping by restraining the training of the network until a minimum error criteria is reached on a separate validation set. The original training set is divided such that 80% is used for training and 20% of the data is treated as validation set. As per the basic early stopping technique, the training is stopped as soon as the error on validation set starts increasing. This method fails for many data sets because real validation error curves usually have more than one local minimum. Therefore, we have implemented one of the early stopping approaches proposed by Prechelt (1998) [45].

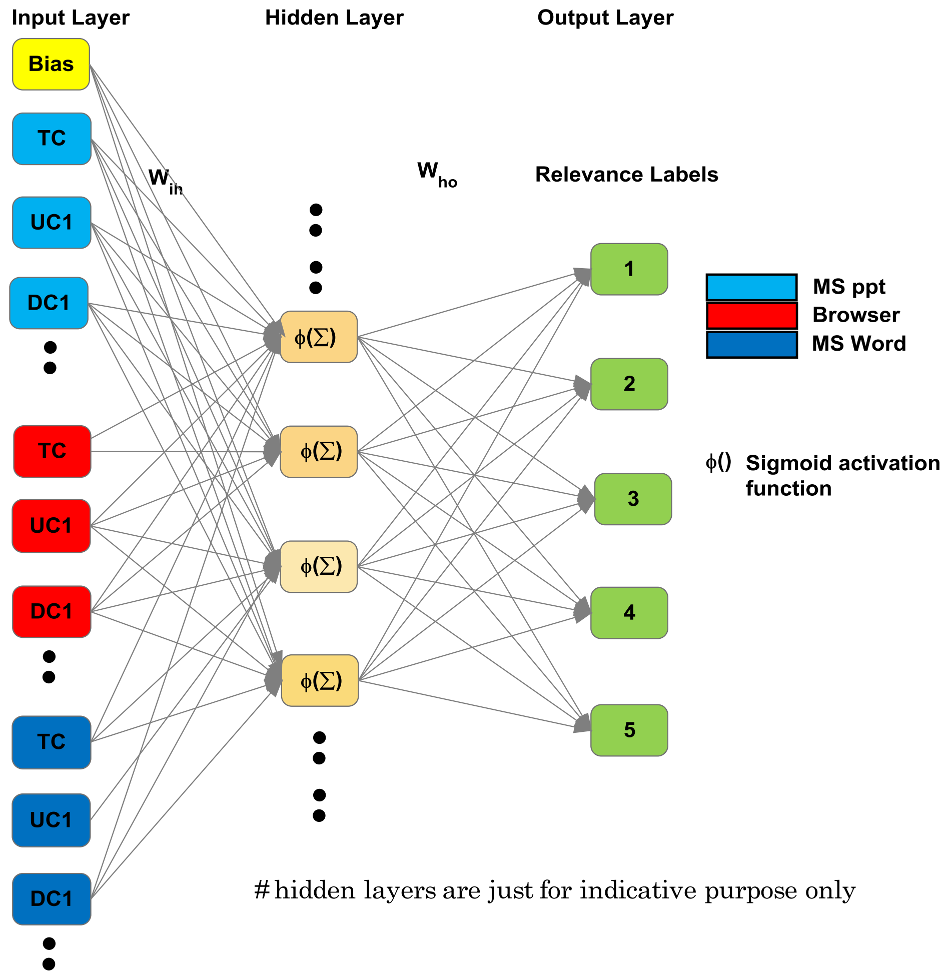


Figure 8. MLP configuration

Let E be the mean squared error (MSE) function of the network. E_{tr} be the training set error and E_{va} be the validation set error. During each 5 epochs we compute these two errors by taking the average of the error over training set and validation set correspondingly. Now, $E_{opt}(t)$ is defined as the lowest validation set error computed in epochs up to t:

$$E_{opt} := \min_{t' < t} E_{va}(t') \qquad \textbf{(3)}$$

A new term known as generalization loss at epoch *t* is defined as the ratio of validation error at *t* to the minimum validation error until *t*:

$$GL(\mathrm{t}) = 100 \times (\frac{E_{va}(t)}{E_{opt}(t)} - 1) \qquad \textbf{(4)}$$

It is evident that the more generalization loss we allow, the more over fitting will happen. Thus, we choose the generalization loss threshold value β after which training is stopped, i.e., we stopped the training of the network as soon as the total generalization loss increases beyond β. The set of network weights of the entire neural network at E_{opt} *(t)* is always saved while computing the minimum error. This saved weight is used later for classification and final feature weight calculation. In this project, β is heuristically chosen as 3.

5.5.1.2 Feature Weight Computation

The MLP approach tested includes a variation of the attribute weighting scheme proposed by Zeng & Martinez (2004) [46]. The difference is that our algorithm computes the weights of each attribute specific to each relevance label rather than one single weight for all the labels, which was proposed originally in [46]. This helps in understanding the importance of individual features towards predicting specific relevance categories. The rationale behind this algorithm is that important features are typically connected to strong links in a MLP and have more effect on the output. After the completion of training, the feature weights are extracted from the trained results as per equation given below.

$$W_{ci} = \sum_{k=1}^{N_H} | CW_{i,k} \times CW_{k,c} | \qquad \textbf{(5)}$$

where W_{ci} is the attribute weight for input node *i* for relevance label c

CWi,k is the connection weight from input node *i* to hidden node *k*

CWk,c is the connection weight from hidden node *k* to output node *c*

N_H is the number of hidden nodes from input node i

After the weight computation from Equation (5), the weights are normalized so that their sum is equal to number of features D.

5.5.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) [47, 48] is a classification algorithm developed from statistical theory. Empirically, it is proven to be a successful classifier model for many domains. The SVM distinguishes between different classes by constructing a decision surface, known as hyperplane, in high dimensional space. Out of many possible separating hyperplanes, an optimal hyperplane is the one that maximizes the margin between different classes. The data points closest to it are called support vectors. In this work, we have implemented Radial Basis Function (RBF) kernel using LIBSVM [49].

Since we deal with a multi-class classification problem, one-against-one is chosen as the preferred method over one-against-all due to its lower training time. The distance of new data points from the optimal hyperplane is computed during the testing phase. The data point is classified to one class or the other based on the sign of the value of this distance. The RBF kernel defines two parameters (C and γ). The parameter C reflects costs associated with the presence of a data point on the wrong side of the hyperplane while gamma (γ) decides the range of the RBF kernel. To find the optimum classifier performance, C and γ need to be searched for using cross validation. The details are explained in Chapter VI. Even though SVM computes feature weights during its computation, these weights represent the importance of attributes projected in high dimension. Thus, real weights of input features are hidden under the weights in the projected feature space and cannot be computed explicitly.

5.5.3 Weighted K Nearest Neighbor (WKNN)

The k nearest neighbor (KNN) algorithm is one of the most simple yet efficient instance learning algorithms used for classification [50]. It is also known as a lazy learning algorithm because it only approximates locally and all computations are deferred until classification. In particular, KNN is easy to implement since the only parameter one needs to choose is k and an appropriate distance metric.

As the name of the approach implies, it is based on a simple idea that data points with similar feature values should be grouped in the same classes. This classical approach does not take into account the feature weighting or distance weighting during the estimation. Given a labeled training set (x^i, c^i) (i = 1,.....,n), where $x^i \in \mathbb{R}^d$ are the feature vectors , and $c^i \in \{c^1,c^2,c^3,c^4,c^5\}$ are binary five class labels, the KNN algorithm first finds the k nearest data points from the test instance x^t. The Euclidian distance is used in this thesis as the distance metric to identify the nearest data points.

$$d(x,y)_{nw} = \sqrt{\sum_{j=1}^{d}(y^j - x^j)^2} \quad \textbf{(6)}$$

where d is the Euclidian distance function between *x* and *y*.

The test instance's label is assigned to the class in which maximum numbers of data points in this neighborhood belong. This majority voting without any kind of weighting mechanism (NW) can be expressed as:

$$y^t_{nw} = \underset{\{c^1,\ldots,c^5\}}{argmax} \sum_{x^i \in d(x^t)} I\left(y^i = c\right) \quad \textbf{(7)}$$

where y^t_{nw} is the estimated class for x^t

$d(x^t)$ set of k training instances closest to x^t

$$I(cond) = \begin{cases} 1 \quad if\ cond = true \\ 0\ if\ cond = false \end{cases}$$

In the proposed work, we have combined two variants of KNN, i.e., attribute weighted and distance weighted KNN to a build our weighted KNN classifier. In the classical KNN algorithm, the distance function does not take into account the importance of features. Attribute weighted KNN uses a modified distance metric to include each of the class-dependent feature weights. The weighted distance function can be expressed mathematically as in Equation (8). By introducing a feature weight component in the distance metric, the quality of the feature is also considered in the distance function in addition to the difference value of the feature. The more useful features will be given more weight while the less useful features will have less weight in the ultimate distance. Thus, useful features will have more impact on the distance function compared to irrelevant features.

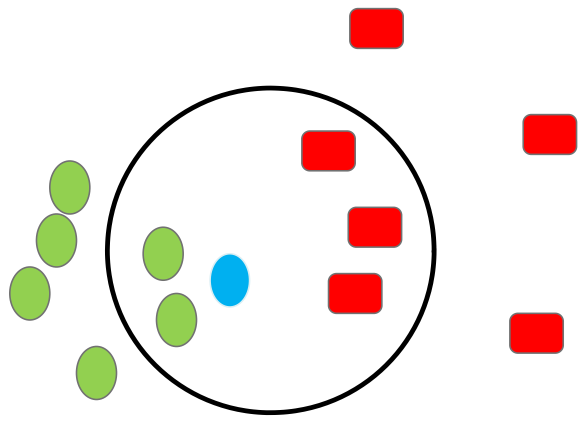
$$d(\mathrm{x},\mathrm{y})_{\mathrm{w}} = \sqrt{\sum_{j=1}^{d} w_{cj}^{\ 2} (y^j - x^j)^2} \qquad \textbf{(8)}$$

where $\mathrm{c} = \mathrm{class}(\mathrm{x}),\ \mathrm{x} \in \mathrm{F}$

$w_{\mathrm{cj}} =$ weight of feature j belonging to class c

Another issue with the classical KNN algorithm is that each of the K neighbors has equal importance in the majority vote. If the k neighbors vary widely in distance, a wrong class may be estimated for the test instance as a result. In this application, the distance between test instance and each of the neighbors is used to assign weights to the K nearest neighbors to avoid problems due to large distance variance. Thus, we give the maximum weight to documents that are most similar to the test instance. Figure 9 explains this scenario. The modified estimation is expressed as follows:

$$y^t{}_w = \underset{\{c^1,\dots,c^5\}}{argmax} \sum_{x^i \in d(x^t)} I\left(y^i = c\right) \times 1 / d(x^t, x^i) \qquad \textbf{(9)}$$



🟦 **Test instance**

🟥 **Training instance from class A**

🟩 **Training instance from class B**

Figure 9. Weighted KNN classifier with K = 5. The test instance most likely belongs to classs B. But, it will be classified as A by majority voting. Distanced weighted KNN can avoid this by dimnishing the vote value of instances from A compared to near placed training instances from B.

5.5.3.1 Feature Weighting Using WKNN

Since we intend to learn the individual importance of each feature corresponding to each class, we have implemented a normalized version of the class dependent RELIEF algorithm proposed in [51] known as NCW-R. Given a training set T, NCW-R generates the feature weight vector W. All the feature weight vector values are initialized to zero and updated iteratively by processing each data point *x* in *X* as per Equation (10).

$$w_c = \sum_{x \in X_c} \left\{ \sum_{z \in WKNN(\mathrm{x,c})} -|x - z| + \sum_{\substack{z \in WKNN(x, c') \\ c' \neq \mathrm{c}}} |x - z| \right\} \Big/ N_c \qquad \textbf{(10)}$$

The term w_c represents the feature weight vector corresponding to class C. The term $|x - z|$indicates the vector of absolute differences between an individual pair of corresponding entries in *x* and *z,* for x, z $\in$F. *WKNN (x, c)* represents the data points in the k nearest neighborhood that belong to the same class as *x*, e.g., C, while *WKNN (x, c')* represents other neighbor data points that do not belong to C. N_c is the total number instances available for class C.

CHAPTER VI

EVALUATION METHEDOLOGY

The primary goal here is to understand the importance of heterogeneous interest evidence from different content consumer and producer applications when building a user interest model in order to recommend new content to users. A key question in testing such user interest models concerns the availability of ground truth data. To the best of our knowledge, there are no publicly available datasets containing implicit and explicit feedback from the browser, Microsoft Word and Microsoft PowerPoint. This chapter outlines the details of collecting a corpus of user activity and document relevance assessments for testing our proposed models, as well as parameter tuning approaches to achieve optimal performance with each model.

6.1 Ground Truth Data Collection

We have conducted a user study to gather data in order to evaluate the effectiveness of the UIMAP framework to achieve the above stated goal. This study collected the user's interaction with and interest expressed in the web browser and Microsoft Word and PowerPoint while performing several information gathering tasks. Thirty-one undergraduate and graduate students (24 male and 7 female) were recruited via email and flyer. The participants were in the age range of 21-40 and studied or worked in various disciplines (Computer Science, Computer Engineering, Electrical Engineering, Biological Engineering and Business Administration). The study was conducted in Harvey Right Bright Building (HRBB) Room No. 232 at Texas A&M

University. The participants were experienced in the tasks we asked them to perform. All participants reported spending at least 1-3 hours daily browsing the Internet and the average self-reported expertise of all participants with Microsoft Word and PowerPoint was four on a five point scale (1 being lowest and 5 being highest).

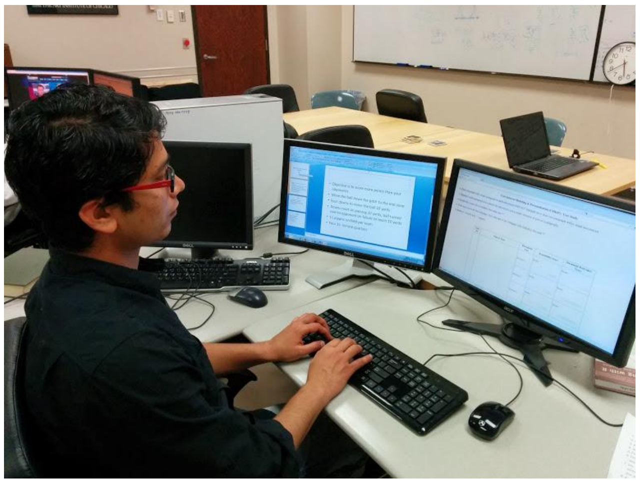


Figure 10. User study in progress

6.1.1 User Study Design

All participants were asked to read eight web pages related to a topic and prepare a one half page report in Microsoft Word and a three slide presentation in Microsoft PowerPoint on a specific task (Figure 10). A session consisted of four such tasks for each participant. During the reading of the selected web pages, participants were asked to annotate the paragraphs of the webpage and assign ratings to them using the WebAnnotate tool. None of the participants had any prior experience with WebAnnotate tool. Once they completed reading each page, they had to provide a readability score and relevance score for that page. The Readability score shows how easy the web page is to read while the relevance score represents the relevance of the web page to the corresponding task. The Readability score signifies the importance of overall appearance and structure of the web page, as well as the language used in the page. On the other hand, the relevance score measures the importance of the overall content of the page. Before performing any tasks, the participants were given a short 5 minute video training session explaining the entire user study procedure along with usage of WebAnnotate tool [51]. They were also provided with a Microsoft Word and PowerPoint template for each task to save time. The instructions suggested that each task would approximately take 30 minutes, but they could continue as long as they wanted.

Four tasks belonging to four different domains (technology, science, finance, sports) were prepared. The task details are given in Table 2. Prior to the user study, we issued task related queries to both Bing and Google. Out of the top twenty results from both of these search engines, eight web pages were chosen. Selection of web pages preferred web pages with more text content rather than pages with image, video or dynamic content. Moreover, question/answer websites such as *ask.com*, *yahooanswers.com*, *stackoverflow.com* and *wikihow.com* were not included as they tend to provide fairly complete summaries of the specific task, removing the need for participants to read the other pages. The selected pages were chosen to include pages with varying degrees of relevance to the task. Table 2 reports the average variance of relevance scores for the web pages assigned by participants per task. It shows that each task contains both relevant and irrelevant web pages in approximately equal proportions.

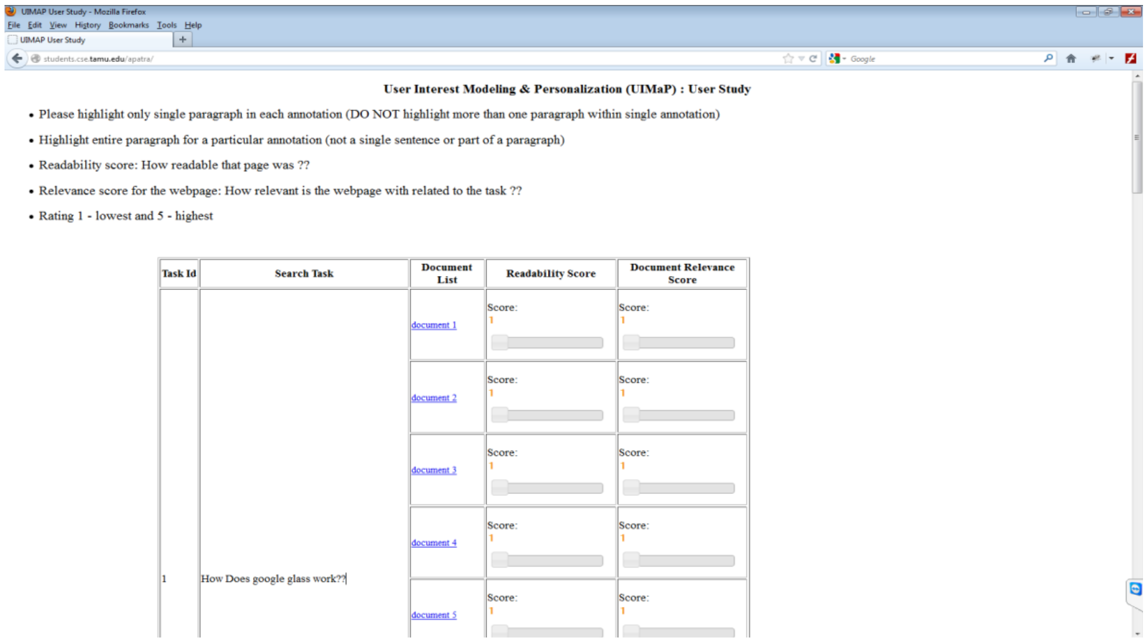
Table 2. Task details

Task No	Task Name	Relevance score Variance
1	How does Google Glass work??	1.9940
2	What is mars one project??	1.1786
3	How to improve your credit score??	2.1417
4	What are the rules of American football??	2.3155

The web pages were saved locally on our own server to ensure consistency of their contents across all the participants. We have also ignored text from dynamic scripts when processing the textual content of the web pages for the same reason. Figure 11 shows the web page used by participants to rate the eight web pages after they had performed each task.

6.1.2 Ground Truth Data Preprocessing

Data collected during the tasks included all the features originally described. Due to experimental setup, this data required preprocessing. For example, as is expected due to the data collection process, document features such as last access time, creation time, and last write time features are not informative because the tasks lasted approximately 30 minutes. Thus, these features are not considered during the evaluation process. In total, the data captured includes 34 potentially useful features.


UIMAP User Study - Mozilla Firefox

File Edit View History Bookmarks Tools Help

UIMAP User Study

students.cse.tamu.edu/apatra/

User Interest Modeling & Personalization (UIMaP) : User Study

- Please highlight only single paragraph in each annotation (DO NOT highlight more than one paragraph within single annotation)
- Highlight entire paragraph for a particular annotation (not a single sentence or part of a paragraph)
- Readability score: How readable that page was ??
- Relevance score for the webpage: How relevant is the webpage with related to the task ??
- Rating 1 - lowest and 5 - highest

Task Id	Search Task	Document List	Readability Score	Document Relevance Score
1	How Does google glass work??	document 1	Score: 1	Score: 1
		document 2	Score: 1	Score: 1
		document 3	Score: 1	Score: 1
		document 4	Score: 1	Score: 1
		document 5	Score: 1	Score: 1

Figure 11. User study home page

The primary artificial aspect of the task was the requirement that participants annotate and rate individual paragraphs of documents. This was requested so that each paragraph in a page could be considered as a unique piece of content with the goal of the interest model learning to identify relevant paragraphs in web pages. Preprocessing of the data assumes any paragraph that was not explicitly annotated and rated by a participant as irrelevant ($C = 1$). Small paragraphs were also removed from consideration; any paragraphs with less than 10 words are ignored from the data set to avoid unwanted noise.

We ignored data collected for task when participants did not generate the requested document or slides and for participants that did not annotate at least fifty paragraphs across all the tasks. Moreover, some users did not wish to continue after performing three tasks. Since each task is independent of other tasks, we have included the data from such participants' completed tasks provided they adhered to the minimum content/annotation rule.

Finally, since the web pages shown to the participants are real web pages, there may be some unwanted paragraphs (comments, page headers) in the content. We removed 6247 such data instances during data filtering stage. Finally, we have 33212 data instances across 104 tasks available for model evaluation, shown in Table 3.

Table 3. Number of tasks and data points considered in the user study

Total tasks performed	Removed tasks	Final tasks included in data	Total data points	Total data points after noise removal
120	16	104	39459	33212

Table 4 shows the relevance distribution of the training data. It is evident from Table 4 that data instances are heavily imbalanced and biased towards irrelevant content (more than 30x). This is because the participants did not annotate all the paragraphs, and we treat any non-annotated paragraph as irrelevant.

Table 4. Relevance label wise training data distribution

Relevance Label	C = 1 (non-relevant)	C = 2 (slightly relevant)	C = 3 (somewhat relevant)	C = 4 (relevant)	C = 5 (very relevant)
No of instances	30300	419	712	925	857

There are multiple explanations for why the participants did not annotate all the relevant paragraphs. Since the web pages are selected from Google/Bing search results, there is overlapping content between them. According to some of the participants, they had already seen similar content in earlier web pages and thus did not consider the new paragraphs as relevant. Moreover, the user study took 2-2:30 hours on average, which is quite tiring. Since the top results of search engines contain wiki pages and other lengthy web pages, these pages were likely to have a lot more relevant content than desired for this activity. Participants did not annotate all potentially relevant paragraphs in those web pages. We observed this common trend in most of the participants.

The kind of severe imbalance in a dataset shown in Table 4 will lead to poor classification results without any data rebalancing [52-54]. Under sampling of the majority class is preferred compared to over sampling of minority classes because over sampling leads to over fitting [55]. However, under sampling has a drawback of under fitting for the majority class (irrelevant class) due to possible loss of valuable information. This is not a serious problem in our case as our priority is to identify the relevant content more accurately. To train the classifiers, random under sampling was used to select a number of data instances of the majority class to balance the dataset.

6.2 Evaluation Metrics

We next describe the evaluation metrics used to assess the quality of the models resulting from the three machine learning approaches.

6.2.1 Precision and Recall

Classification accuracy is not an appropriate evaluation metric in the case of an imbalanced data set because high accuracy can be easily obtained by always predicting the majority class. Typically, it is more important to predict minority classes more correctly than the majority class. Thus, we choose precision, recall and f-measure as one of the evaluation criteria for our work. Prior studies [52, 56] have already proven that these measures are independent of class distributions provided that precision and recall are measured at the same time. Usually, the output of any classifier is presented in a

		Actual	
		YES	NO
Predicted	YES	True Positive(TP)	False Positive(FP)
	NO	False Negative(FN)	True Negative(TN)

Figure 12. Confusion matrix

tabular format known as confusion matrix or contingency table as shown in Figure 12. The confusion matrix consists of four categories. True Positives (TP) are data instances whose relevance labels are correctly predicted by the proposed model. False Negative (FN) refers to the data instances that model falsely predicts to be a wrong label while False Positive (FP) are the cases where the data instances are falsely predicted to be correct label. Finally, True Negative (TN) corresponds to predictions that are correctly predicted not to be the original label. Intuitively, precision measures exactness of the system, i.e., out of all predicted data instances for a specific relevance label how many are predicted correctly, while recall indicates the completeness of the system, i.e., out of all labeled data for a specific a relevance label how many are predicted correctly. Precision and Recall often share an inverse relationship between them. Then the precision and recall are defined as

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Re}call = \frac{TP}{TP + FN} \tag{12}$$

$$F1 = \frac{2 * \text{precision} * \text{recall}}{precision + recall} \tag{13}$$

F1, shown in Equation (13), measures balance between precision and recall in a single value.

Additionally, we also compute precision and recall for every individual relevance label because of the multi-class classification nature of the problem. We report our final precision and recall as a macro-average of precision and recall of each relevance label instead of micro-average. The reason for that is the macro-average gives equal weights to individual classes while micro-average gives equal weights to individual data points.

Thus, large classes dominate the evaluation in macro-averaging if data distribution is skewed.

6.2.2 Average Distance Measure

The average distance measure (ADM) [57] is another evaluation metric that is more suitable for multi-grade relevance judgment. It measures the average distance between the user provided relevance weight (URW) and system predicted relevance weight (SPW). It is based on the same fundamental idea of Mean Absolute Error (MAE). Relevance weight is denoted as the normalized relevance label. In a more formal way, for a given paragraph x_i in data set F we can define ADM as

$$adm = 1 - \frac{\sum_{x_i \in F} | SPW(x_i) - URW(x_i) |}{| F |} \quad \textbf{(14)}$$

where $|F|$ denotes total number data points in the dataset

The ADM score lies between [0...1] range, with 1 being the best performance and 0 being the worst. ADM is more suitable than precision/recall/f1 measures for the proposed work because it provides insight into variation between the user assigned and system predicted relevance values. Understanding this difference is important in the proposed system since predicting a highly relevant content ($C = 5$) as non-relevant ($C = 1$) or as relevant ($C = 4$) should not be considered the same. The former prediction leads to the wrong result while the latter can be considered reasonably successful. While the above metric (Precision/Recall/F1) treats these two cases as the same, ADM distinguishes between them, as the penalty is less in the case of the latter. Since the purpose of the proposed work is to identify the relevance label of any content rather than ranking the documents, other standard methods such as Mean Precision and Weighted Average Precision are not suitable [58]. Another popular evaluation metric is Root Mean Square Error (RMSE). Although, RMSE is very similar to ADM, it punishes the larger error more severely compared to ADM. That is why we choose ADM as the evaluation criteria for parameter and model selection.

6.3 Parameter Selection

It is a common practice to tune the parameters of a classifier to achieve the best performance. We next describe the parameter selection methodology adopted for each of the classifiers used in the proposed work. We employed a grid search mechanism to identify the best parameter combination for optimal result. The entire data set is divided

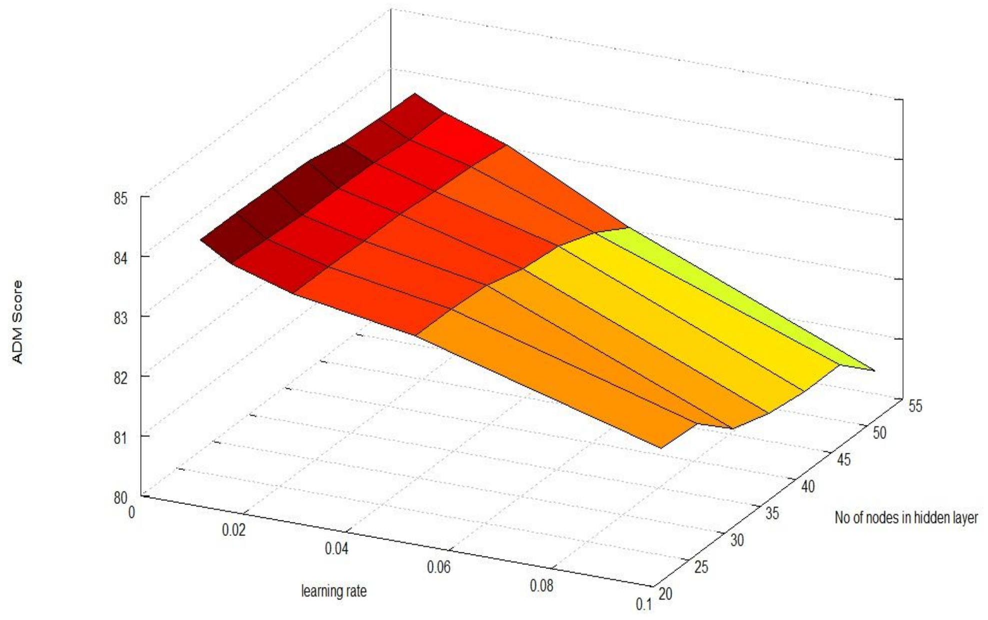


Figure 13. Cross validation result for MLP

into 70:30 ratio as training and testing sets. The performance for each classifier is reported below after a 5-fold cross validation is performed for 40 iterations on the training set only. Figure 13 shows the variation of ADM score for the MLP classifier when the learning rate is varied from 0.003 to 0.1 (2x step) and the number of units in the hidden layer is varied from 24 to 54. The best performance is achieved at 39 (hidden units) and 0.006 (learning rate). Similarly, cross validation results for SVM is presented in Figure 14. Gamma is varied from 0.00003 to eight while cost parameter (C) is varied from 0.03125 to 32768. The optimal result is obtained at 512 (C) and 0.125 (gamma). For WKNN, we have only one parameter to select, i.e., the number of data instances to be allowed in the neighborhood (K). As the value of K increased from 1 to 10, the performance improves until K = 4 and starts to decline afterwards. This trend is shown in Figure 15. Thus, our final evaluation of these classifiers employs the parameters, shown in Table 5, tuned for the optimal performance.

Table 5. Tuned parameter for optimal result

Classifier	Parameter1	Parameter2
SVM	C = 512	Gamma =0.125
MLP	Hidden Units=39	Learning Rate = 0.006
WKNN	K = 4	

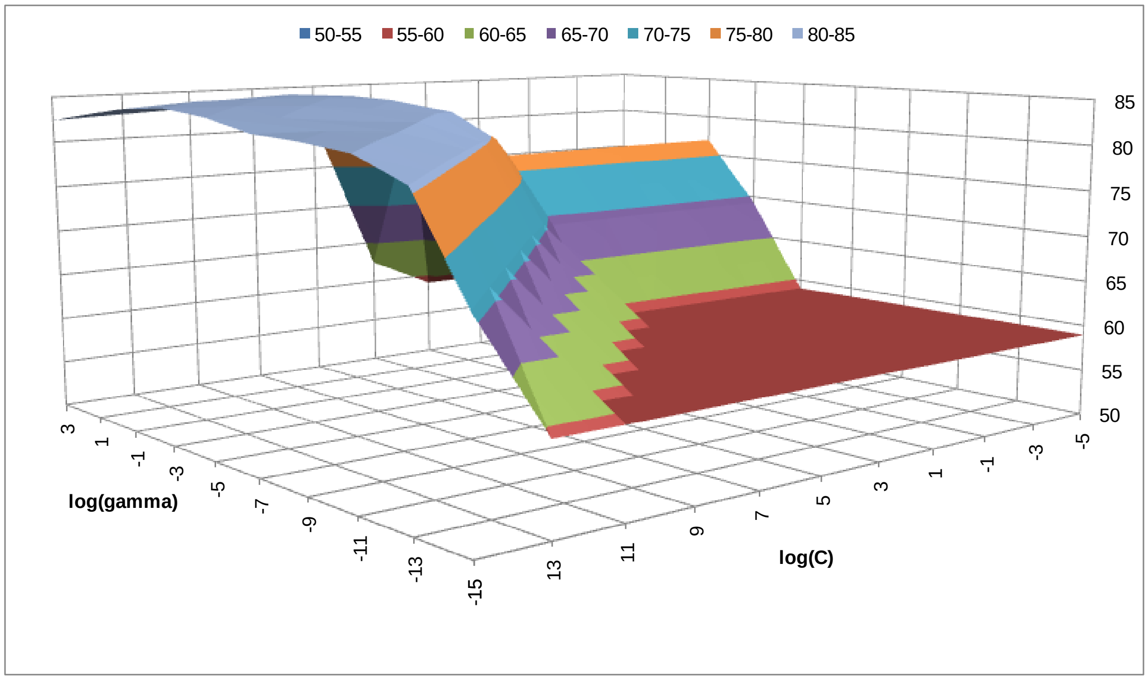


Figure 14. Cross validation result for SVM

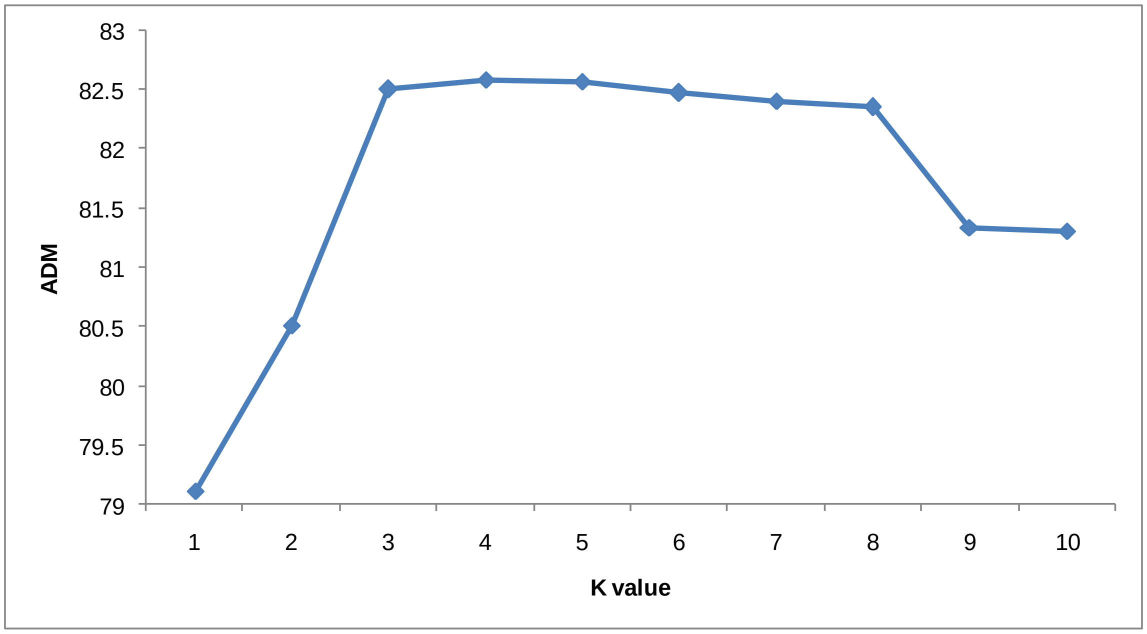


Figure 15. Cross validation result for WKNN

CHAPTER VI

RESULTS AND DISCUSSION

This chapter describes the evaluation results of the three classifiers implemented in the proposed interest model UIMAP. The results are reported after averaging over 200 iterations of training the classifiers using a training data set and testing on the test data set. The training data set is generated by randomly selecting 70% data points from the entire data set and the remainder 30 % is treated as test data set for each iteration. Due to the already described inherent imbalance in the data distribution, we randomly under sample the data instances from the majority class such that each class has approximately the same number of instances in the training data. The minimum sample size is chosen to be 1000 because the relevant labels ($C = 4$ *or* $C = 5$) contain close to 1000 data instances. Reducing the sample size below 1000 will make the relevant classes majority classes, thus leading to a model biased towards relevant classes. We vary the number of sampled data points from the majority class between 1000 and 3000 for each classification method. Figure 16 shows the performance of the classifiers in terms of *overall-ADM* and *relevant-ADM* score. The *overall-ADM* score is defined as the average ADM score for all the relevance labels while *relevant-ADM* score represents the average ADM score for only relevant labels. If both *overall-ADM* and *relevant-ADM* scores are high, then we can conclude that model performed well. As we increase the number of samples, there is a steep decrease in average *relevant-ADM* score while the *overall-ADM* score improved marginally. This is because the classifier becomes biased towards the irrelevant class as the number of samples belonging to that class increases. This bias explains why a majority of the test data points were predicted as irrelevant resulting in a

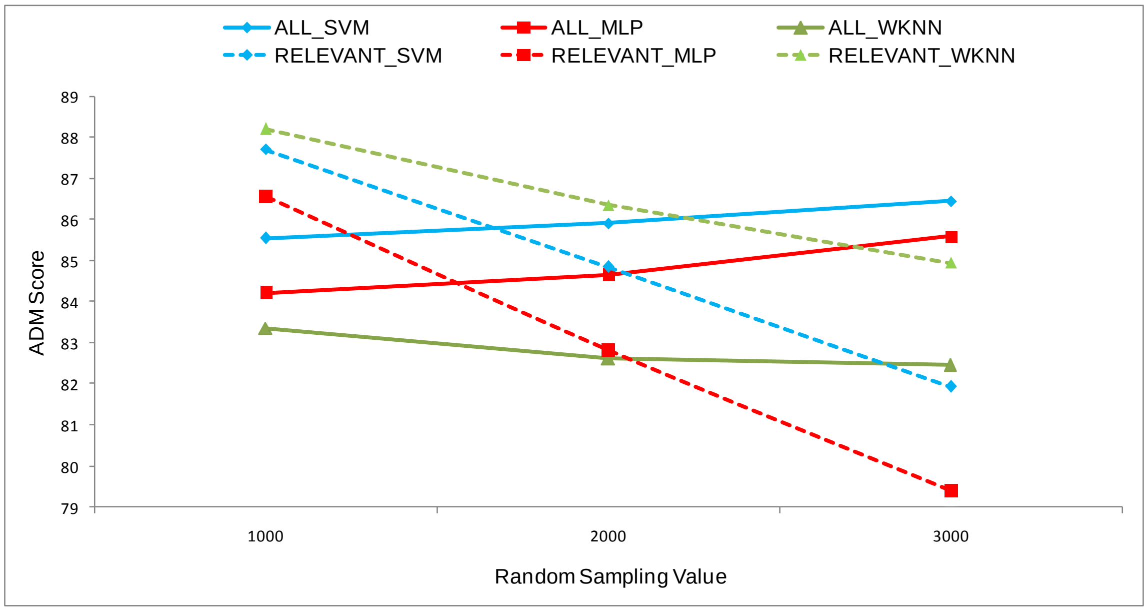


Figure 16. The effect of varying training distribution by under sampling

higher *overall-ADM* score but a lower *relevant-ADM* score. From here onwards, we under sample for 1000 data points from the majority class to alter the original skewed distribution towards a more balanced distribution. Among the three classifiers, SVM performs better than other two classifiers if *overall-ADM* is taken into consideration, but WKNN marginally performs better than SVM if *relevant-ADM score* is considered only. Overall, MLP performed poorly in both cases.

Next we report on the precision, recall, and F1 score for each of the classifiers for each of the five relevance levels. Figure 17 shows the precision value comparison while

Figure 18 and Figure 19 show the recall and F1 score comparison, respectively. The F1 score for *C = {1, 4, 5}* are much higher than *C = {2, 3}* as expected because the number of data points for the latter are quite less compared to the former. Apart from irrelevant data instances, SVM has a higher precision than MLP and WKNN. For relevant labels only, both precision and recall are around 0.6 for all the three classifiers, which indicates

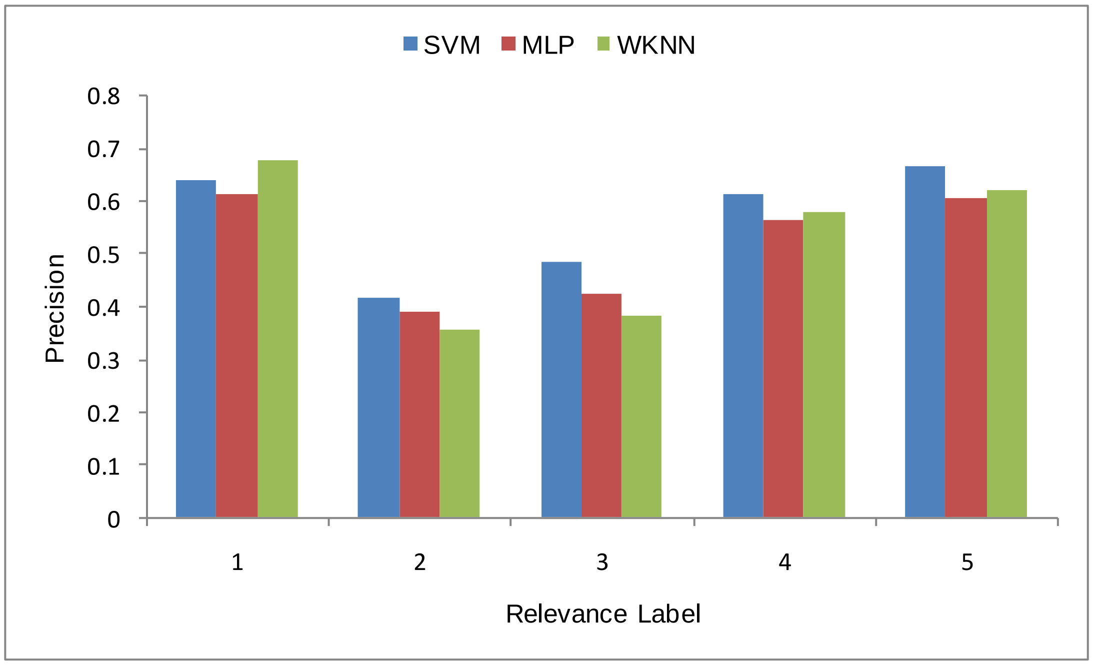


Figure 17. Precision comparison of different classifiers

the robust nature of our model. For irrelevant label (C = 1), WKNN has the highest precision score but the lowest recall score. Overall, MLP did not perform well compared to the SVM or WKNN classifiers using these metrics.

To provide a better illustration to explain the difference between user assigned relevance scores and our model-predicted relevance scores, we also report class-wise

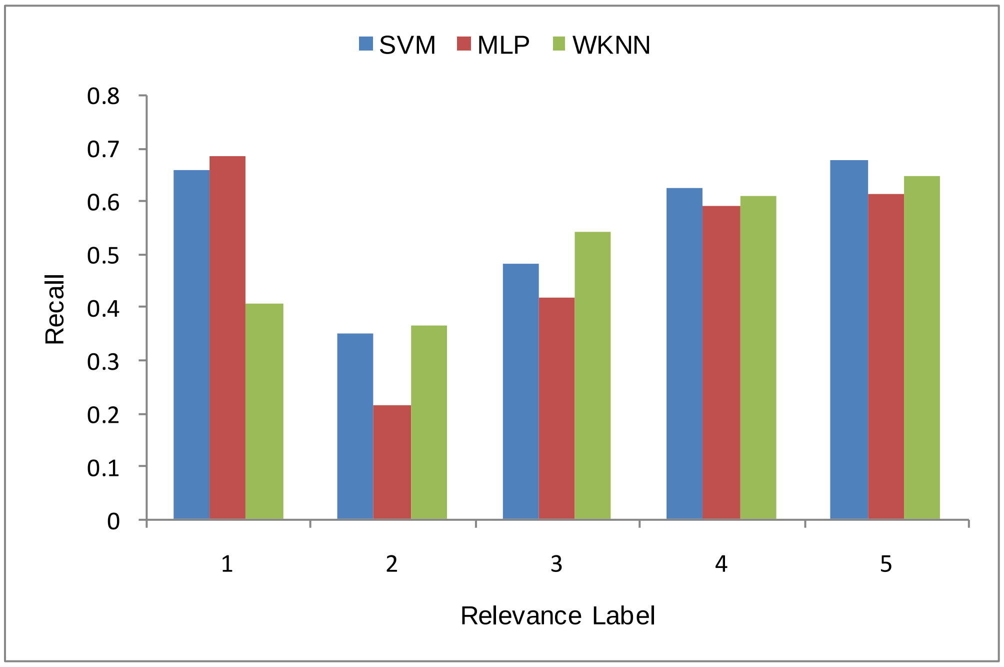


Figure 18. Recall comparison for different classifiers

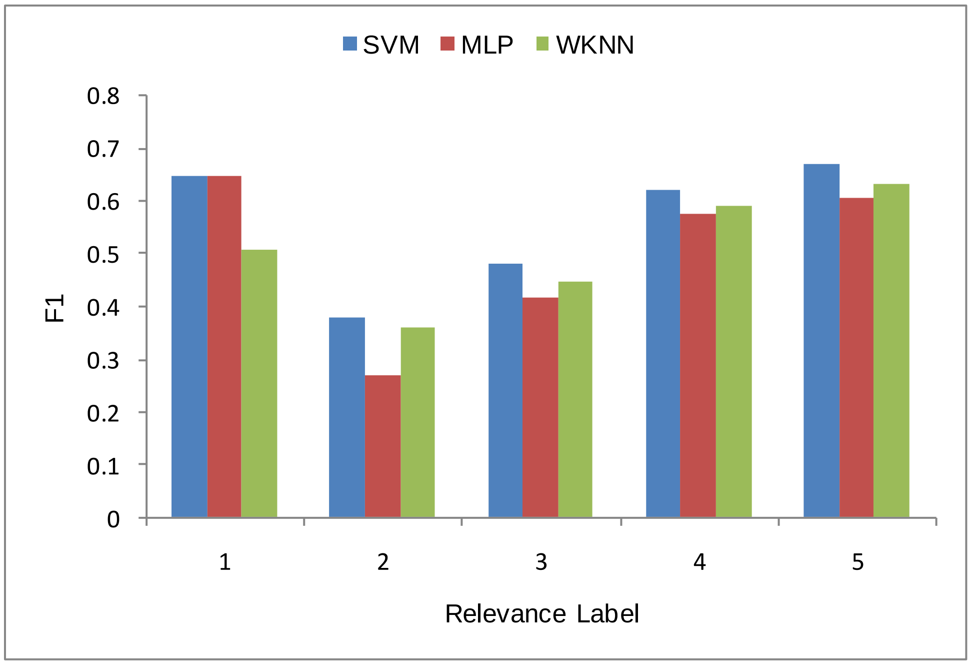


Figure 19. F1 score comparison for different classifiers

distribution of test data points belonging to all possible relevance label difference (RLD) values for each classifier. It is evident from Figure 20 that most of the data points belong to a RLD range = {-1, 0, 1}. It suggests that our model predicted most of the data points almost correctly with a very small margin of error. In addition to that, most of the wrongly predicted data points from highly relevant class (*C = 5*) have a RLD value = -1, while the wrongly predicted data points from relevant class (*C = 4*) have a RLD value = 1. In other words, our model incorrectly predicted relevance label 5 as 4 or 4 as 5 in most of the cases. Since both relevance labels 4 and 5 are relevant, predicting relevance *C = 4* as *5* or vice versa can be considered as good result.

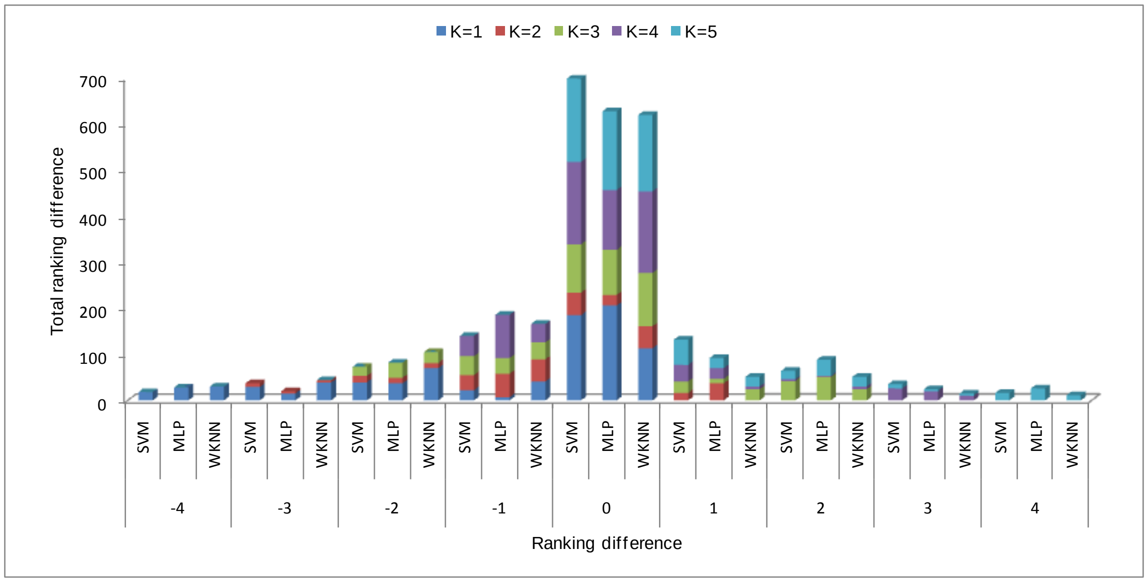


Figure 20. Class wise difference between system predicted and user assigned relevance labels

Another objective of the proposed work is to understand the importance of each interest indicator in the user's final predicted interest. As feature weights cannot be computed in non-linear SVM, we present feature weight comparison only between WKNN and MLP, shown in Figure 21 and Figure 22, respectively. In WKNN, features computed from the content-consumer application clearly have higher weights than the features from the content-producer applications except the content similarity feature. Other two explicit feedback scores, i.e., readability and relevance score, apart from the

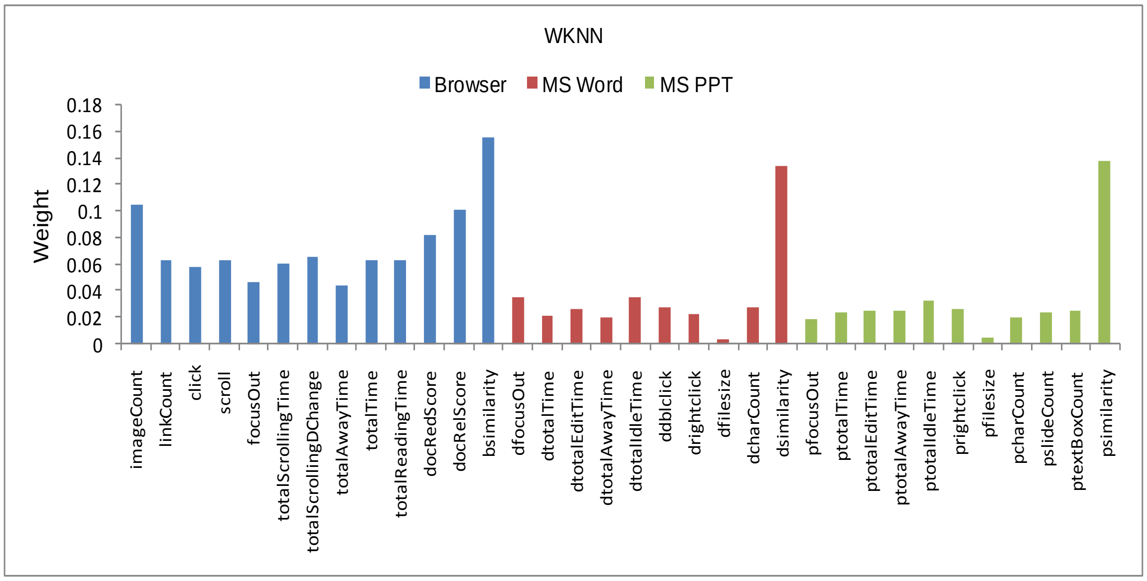


Figure 21. Comparison of feature weight computed from WKNN

content similarity, contribute more towards the predicted user interest. Surprisingly, image count seems to contribute even more than these two explicit scores. We could not reach to any specific hypothesis for such behavior at this point of time. Among user characteristics, all scrolling (number of scroll, scroll time and scrolling direction change) and time related (total time and reading time) features significantly contribute towards the predicted interest. In the browser, the number of focus out events and the total time

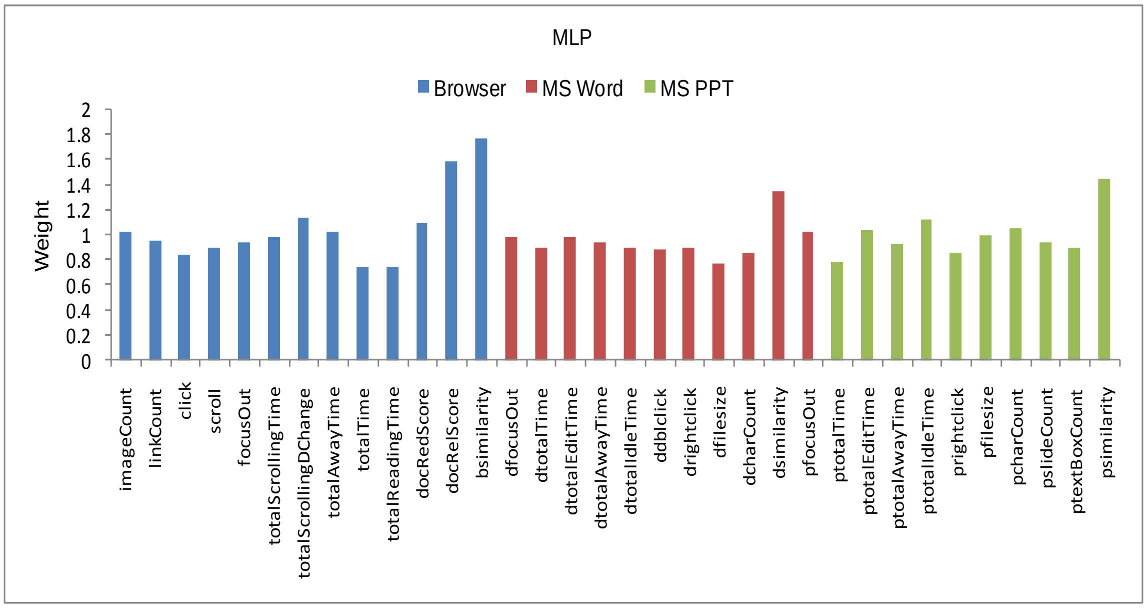


Figure 22. Comparison of feature weight computed from MLP

spent away from browser have the least effect towards the interest. Since both of these features are interrelated, this seems a logical inference. The TC features from all three applications have maximum contribution towards user interest. Even though the amount of content available in content producer applications is low compared to content consumer applications, TC feature weights from the two are similar. This validates our initial hypothesis that content producer applications can have a significant contribution towards the user interest model. However, UC and DC features of content-producer applications contribute the least for both MLP and WKNN classifiers. This is likely due to the fact that the content being authored in our experimental setup is clearly relevant, and using other features to increase or decrease that relevance generates noise.

A similar trend is also observed in feature weights computed in MLP. The top 10 features in both the classifiers are almost the same. However, the results produced by MLP are not very encouraging as the difference between the feature weights are not significant and lie in a very small range. Thus, MLP may not be a good choice for weight computation. Additionally, it also takes longer time to train due to the gradient descent-based back propagation method.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

In summary, the work presented in this thesis addresses one of the less studied approaches to personalizing information delivery. The importance of multiple everyday applications and different explicit and implicit interest feedback features are explored here to build a common user interest model. Lack of research in this domain leads to the unavailability of data sets that represent interest evidence from multiple applications. As a result, this thesis presents a method to collect such data through the design and conduct of a user study involving common applications. The thesis explored the value of three classes of interest feedback from the content consumption and production applications: 1) User Characteristics, 2) Document Characteristics, 3) Textual Characteristics. Some of these are collected implicitly while others are provided by the user explicitly. The IPM infrastructure and its clients for Microsoft Word and PowerPoint have also been extended to support the collection of additional implicit feedback data. Finally, three different classification algorithms, i.e., Multilayer Perceptron, Support Vector Machine and Weighted K-Nearest Neighborhood, are used to build the user interest model. We also investigated the individual contribution of each interest indicator towards the prediction of relevant content by implementing different feature weighting algorithms.

It has been noted that with explicit ratings, users read a lot more than they rate [59]. We observed a similar pattern during our ground truth data collection. Even though the users had spent considerable time reading a specific document, he/she did not rate many of the paragraphs that were relevant. Moreover, relevance is always a subjective judgment and varies from one user to another. Considering all these, we believe UIMAP model performed reasonably well in predicting personalized relevant content. We have evaluated the performance of the three classifiers and compared different feature weighting algorithms. Based on the performance results of F1 measure, we found that SVM is better than WKNN. On the other hand based on the ADM score of relevant classes ($C= \{4, 5\}$), WKNN performed slightly better than SVM while SVM outperformed WKNN if the ADM score for all classes ($C = \{1, 2, 3, 4, 5\}$) is taken into consideration. This is because WKNN is not able to identify the irrelevant class ($C = \{1\}$) very well. MLP did not show promising results in either of the evaluation metrics. Feature weight comparison results indicated that WKNN performed very well in understanding the individual importance of user activities in each application while the feature weights are hidden in SVM. Even though MLP is able to compute feature weights, the relative differences between the weight values are not high and it takes longer time to train the neural network. Thus, it is not scalable in the long run. Overall, content-consumer applications had stronger weights for implicit feedback data mapping to the user's real interest when compared to producer applications. Nevertheless, contribution from content-producer applications cannot be ignored as the textual content from producer applications contributes nearly equally to the content from consumer applications. The nature of the user study likely contributed towards the relative distributions of the various features – there was no time for users to multitask as they would during real world activities. In conclusion, SVM is preferred if the objective is to only identify the relevant content in a multi-application user interest model. However, if feature weighting is equally important as identifying the relevant content then WKNN is the preferred choice.

Certainly, the User Interest Modeling and Personalization (UIMAP) is only the first step towards a personalized information delivery framework which takes into account both implicit and explicit feedback from multiple applications. The performance of the existing framework can be improved by selecting features with higher weights only. Since feature weights provide an insight into which features are more important than others, unwanted noise can be avoided by this. Moreover, we did not distinguish between implicit and explicit feedbacks in our model. Different experiments can be done to use them in a hierarchical manner so that one type of interest indicator can augment the other.

One of the major strengths of UIMAP framework is its extensibility. The framework is designed such that new content consumer/producer applications, e.g., Acrobat PDF, Outlook Mail, etc., can be added very easily. One issue with the UIMAP is that it does not perform with acceptable accuracy until it sees a relatively large amount of data. Thus, it is difficult to predict relevant content for a new user or a new task. Adopting collaborative filtering in our framework may help with this problem. This approach can use similar interest models from prior tasks for predicting a new user/task with insufficient interest information. This method would also be helpful in generating community of users who shares the similar interest area or similar reading/writing patterns.

In the future, we expect to employ interest drift in interest models. Often users move from one interest to another interest seamlessly while doing an information gathering task. This interest drift needs to be identified to discount the interest evidence obtained from the documents that are no longer in use. All the applications involved in this work are only desktop applications. Today, users spend considerable time on mobile devices. Thus, smart phone and tablet application clients also can be added to the UIMAP infrastructure to build a more robust and dynamic user interest models.

REFERENCES

1. Renda, M.E. and Straccia, U., *A personalized collaborative digital library environment: a model and an application.* Information Processing and Management: an International Journal - Special Issue: an Asian Digital Libraries Perspective, 2005. vol.41(1): p. 5-21.
2. *Eclipse from Chrystal Software (GenOmega)*. Accessed: 04/17/2014; Available from: http://www.genomega.com.
3. Pazzani, M. and Billsus, D., *Learning and revising user profiles: The identification of interesting web sites.* Machine learning, 1997. vol.27(3): p. 313-331.
4. Badi, R., Bae, S., Moore, J.M., et al. *Recognizing user interest and document value from reading and organizing activities in document triage*. In *Proceedings of the 11th International Conference on Intelligent User Interfaces(IUI),* 2006. ACM.
5. Moreno-Llorena, J., Roldán, X.A., and Perez, R.C., *Modeling of User Interest Based on Its Interaction with a Collaborative Knowledge Management System*, In *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*. 2009, Springer. p. 330-339.
6. Das, A.S., Datar, M., Garg, A., and Rajaram, S. *Google news personalization: scalable online collaborative filtering*. In *Proceedings of the 16th International Conference on World Wide Web,* 2007. ACM.

7. Shipman, F., Price, M., Marshall, C.C., and Golovchinsky, G., *Identifying useful passages in documents based on annotation patterns*, In *Research and Advanced Technology for Digital Libraries*. 2003, Springer. p. 101-112.

8. Goecks, J. and Shavlik, J., *Learning users' interests by unobtrusively observing their normal behavior*. In *Proceedings of the 5th International Conference on Intelligent User Interfaces (IUI),* 2000. ACM.

9. Kelly, D. and Belkin, N.J., *Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback*. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 2001. ACM.

10. Wettschereck, D., Aha, D.W., and Mohri, T., *A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms.* Artificial Intelligence Review, 1997. vol.11(1-5): p. 273-314.

11. John, G.H., Kohavi, R., and Pfleger, K., *Irrelevant Features and the Subset Selection Problem*. In *Proceedings of Eleventh IEEE International Conference on Machine Learning (ICML),* 1994. Morgan Kaufmann.

12. Mitchell, T.M., *The need for biases in learning generalizations*. (Report CBM-TR-5-110). 1980, New Brunswick,NJ: Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ.

13. Nielsen, J., *How Little Do Users Read?* May 6, 2008 Accessed: 04/17/2014; Available from: http://www.nngroup.com/articles/how-little-do-users-read/.

14. Nielsen, J., *Scrolling and Attention*. March 22, 2010 Accessed: 04/17/2014; Available from: http://www.nngroup.com/articles/scrolling-and-attention/.

15. Schwab, I., Kobsa, A., and Koychev, I., *Learning user interests through positive examples using content analysis and collaborative filtering.* Internal Memo, GMD, St. Augustin, Germany, 2001.

16. Billsus, D. and Pazzani, M., *Learning probabilistic user models*. In *Proceedings of 6th International Conference on User Modeling, Workshop on Machine Learning for User Modeling, Chia Laguna, Sardinia,* 1997.

17. Pazzani, M.J., Muramatsu, J., and Billsus, D. *Syskill & Webert: Identifying interesting web sites*. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI/IAAI, Vol. 1,* 1996. Portland, OR.

18. Lam, W. and Mostafa, J., *Modeling user interest shift using a bayesian approach.* Journal of the American society for Information Science and Technology, 2001. vol.52(5): p. 416-429.

19. Jayarathna, S., Patra, A., and Shipman, F., *Mining user interest from search tasks and annotations*. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM),* 2013. ACM.

20. Bae, S., Hsieh, H., Kim, D., et al., *Supporting document triage via annotation-based visualizations.* Proceedings of the American Society for Information Science and Technology, 2008. vol.45(1): p. 1-16.

21. Kaptelinin, V., *UMEA: translating interaction histories into project contexts*. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 2003. ACM.

22. Dragunov, A.N., Dietterich, T.G., Johnsrude, K., et al. *TaskTracer: a desktop environment to support multi-tasking knowledge workers*. In *Proceedings of the 10th International Conference on Intelligent User Interfaces,* 2005. ACM.

23. Budzik, J. and Hammond, K.J., *User interactions with everyday applications as context for just-in-time information access*. In *Proceedings of the 5th International Conference on Intelligent User Interfaces(IUI),* 2000. ACM.

24. Finkelstein, L., Gabrilovich, E., Matias, Y., et al. *Placing search in context: The concept revisited*. In *Proceedings of the 10th International Conference on World Wide Web,* 2001. ACM.

25. Claypool, M., Le, P., Wased, M., and Brown, D., *Implicit interest indicators*. In *Proceedings of the 6th International Conference on Intelligent User Interfaces (IUI),* 2001. ACM.

26. Frank, E., Hall, M., and Pfahringer, B., *Locally weighted naive bayes*. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence,* 2002. Morgan Kaufmann Publishers Inc.

27. Wang, B. and Zhang, H., *Probability based metrics for locally weighted naive bayes*, In *Advances in Artificial Intelligence*. 2007, Springer. p. 180-191.

28. Lee, C.-H., Gutierrez, F., and Dou, D., *Calculating feature weights in naive bayes with kullback-leibler measure*. In *Proceedings of Eleventh IEEE International Conference on Data Mining Data Mining (ICDM),,* 2011. IEEE.

29. Fayyad, U. and Irani, K., *Multi-interval discretization of continuous-valued attributes for classification learning*. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* 1993. San Francisco, CA: Morgan Kaufmann.

30. Guyon, I., Weston, J., Barnhill, S., and Vapnik, V., *Gene selection for cancer classification using support vector machines.* Machine learning, 2002. vol.46(1-3): p. 389-422.

31. Salzberg, S., *A nearest hyperrectangle learning method.* Journal Machine Learning, 1991. vol.6(3): p. 251-276.

32. Bae, S.I. and Frank, M., *Balancing human and system visualization during document triage*. 2008: Doctoral dissertation, Texas A&M University. Available from: http: / /hdl.handle.net /1969.1/ETD-TAMU-2330.

33. Bae, S., Kim, D., Meintanis, K., et al. *Supporting document triage via annotation-based multi-application visualizations*. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries,* 2010. Gold Coast, Queensland, Australia: ACM.

34. Ganta, P., *A Comparison of Clustering Methods for Developing Models of User Interest*. Master's thesis, Texas A&M University. Vol. 1. 2011. Available from: http://hdl.handle.net/1969.

35. Cox III, E.P., *The optimal number of response alternatives for a scale: A review.* Journal of marketing research, 1980. vol.17(4): p. 407-422.

36. Tang, R., Shaw, W.M., and Vevea, J.L., *Towards the identification of the optimal number of relevance categories.* Journal of the American Society for Information Science, 1999. vol.50(3): p. 254-264.

37. Zhou, B. and Yao, Y., *Evaluating information retrieval system performance based on user preference.* Journal of Intelligent Information Systems, 2010. vol.34(3): p. 227-248.

38. Maron, M.E. and Kuhns, J.L., *On relevance, probabilistic indexing and information retrieval.* Journal of the ACM (JACM), 1960. vol.7(3): p. 216-244.

39. Schilit, B.N., Golovchinsky, G., and Price, M.N. *Beyond paper: supporting active reading with free form digital ink annotations*. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 1998. Los Angeles, California, USA: ACM Press/Addison-Wesley Publishing Co.

40. Blei, D.M., Ng, A.Y., and Jordan, M.I., *Latent dirichlet allocation.* Journal of Machine Learning Research, 2003. vol.3: p. 993-1022.

41. Porteous, I., Newman, D., Ihler, A., et al. *Fast collapsed gibbs sampling for latent dirichlet allocation*. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 2008. ACM.

42. McCallum, A.K., *MALLET: A Machine Learning for Language Toolkit*. 2002 Accessed: 04/17/2014; Available from: http://mallet.cs.umass.edu.

43. Beran, R., *Minimum Hellinger distance estimates for parametric models.* The Annals of Statistics, 1977. vol.5(3): p. 445-463.

44. Csáji, B.C., *Approximation with artificial neural networks.* Faculty of Sciences, Etvs Lornd University, Hungary, 2001: p. 24.

45. Prechelt, L., *Early stopping-but when?*, In *Neural Networks: Tricks of the trade*. 1998, Springer. p. 55-69.

46. Zeng, X. and Martinez, T.R., *Feature weighting using neural networks*. In *Proceedings of IEEE International Joint Conference on Neural Networks(IJCNN),* 2004. IEEE.

47. Vapnik, V.N., *An overview of statistical learning theory.* IEEE Transactions on Neural Networks, 1999. vol.10(5): p. 988-999.

48. Burges, C.J., *A tutorial on support vector machines for pattern recognition.* Data Mining and Knowledge Discovery, 1998. vol.2(2): p. 121-167.

49. Chang, C.-C. and Lin, C.-J., *LIBSVM: a library for support vector machines.* ACM Transactions on Intelligent Systems and Technology (TIST), 2011. vol.2(3): p. 27.

50. Cover, T. and Hart, P., *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, 1967. vol.13(1): p. 21-27.

51. Marchiori, E., *Class dependent feature weighting and k-nearest neighbor classification*, In *Pattern Recognition in Bioinformatics*. 2013, Springer. p. 69-78.

52. Monard, M.C. and Batista, G., *Learning with skewed class distributions.* Advances in Logic, Artificial Intelligence and Robotics, 2002: p. 173-180.

53. He, H. and Garcia, E.A., *Learning from imbalanced data.* IEEE Transactions on Knowledge and Data Engineering, 2009. vol.21(9): p. 1263-1284.

54. Maloof, M.A., *Learning when data sets are imbalanced and when costs are unequal and unknown*. In *ICML-2003 Workshop on Learning from Imbalanced Data Sets II,* 2003.

55. Drummond, C. and Holte, R.C., *Severe class imbalance: Why better algorithms aren't the answer*, in *Proceedings of the 16th European Conference on Machine Learning* 2005, Springer: Porto, Portugal. p. 539-546.

56. Manevitz, L. and Yousef, M., *One-class document classification via neural networks.* Neurocomputing, 2007. vol.70(7): p. 1466-1481.

57. Mizzaro, S., *A new measure of retrieval effectiveness (or: What's wrong with precision and recall).* In *International Workshop on Information Retrieval (IR'2001),* 2001. Citeseer.

58. Sakai, T., *New performance metrics based on multigrade relevance: Their application to question answering*. In *NTCIR Workshop 4 Meeting Working Notes* June 2004.

59. Sarwar, B.M., Konstan, J.A., Borchers, A., et al. *Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system*. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW),* 1998. Seattle, Washington, USA: ACM.