uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Marcel Wirantono
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Automated Text with Categorization with Collaboratively Tagged Data

TITRE DE LA THÈSE / TITLE OF THESIS

Nathalie Japkowicz
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Caroline Barrière
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Diana Inkpen

Franz Oppacher

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Automated Text Categorization with Collaboratively Tagged Data

by

Marcel Wirantono

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the degree of Master of Computer Science (MCS)

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-51865-6
Our file Notre référence
ISBN: 978-0-494-51865-6

Abstract

Recent popularity of collaborative tagging as a component of a retrieval system has lead us to study such a system. Similar to text categorization, albeit in a less centralized fashion, collaborative tagging relies on humans to annotate documents with metadata descriptions, i.e. tags. For that reason, this thesis attempts to extend the tagging process to include a more consistent non-human annotations in the form of automatic text categorization.

In applying automatic text categorization to collaboratively tagged data, we have created two sets of experiment. The first experiment compares two classification methods, Naive Bayes and Support Vector Machine (SVM) in a straightforward 1-vs. all classification. The results of the comparison allow us to make important observations such as the benefit of using a maximum margin classifiers (SVM) in annotating concepts with skewed document distributions as well as establishing a baseline result.

For the second experiment, we have found that the lack of structure in tagging has limited our learning approach to the simple 1-vs. all setting. Inspired by the application of hierarchical categorization in web directories[15], we introduce in our second experiment a categorization approach that automatically builds a hierarchy from the tag space and incorporates it to the training and classification process. Unlike previous hierarchical categorizations that rely on human-generated hierarchies, our hierarchical approach relies on an artificial hierarchy that is created from tag usage analysis. After the method was applied to the dataset, we compared the result of the new methods with the baseline results from the first experiment. Based on that comparison, we observed that our hierarchical approach improves not only on the quality of predictions, but also the efficiency (total training and classification time) of our automatic text categorization system.

Acknowledgements

I would like to thank, first and foremost, my supervisors, Dr. Japkowicz and Dr. Barrière for being very patient and helpful in shaping this thesis. Our discussions on various machine learning and general research topics have been immeasurably valuable to the relevance and coherence of this thesis. I also thanked them for being understanding and for giving me the opportunity to be in the program.

For various people in University of Ottawa I have encountered in my graduate studies, I am thankful for their contribution to my overall graduate experience. Lastly, I would also like to thank my parents, Jo and Yanti, for always supporting and encouraging me to pursue my dreams.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The growth of the World Wide Web (the Web) has resulted in billions of documents which serve different purposes and needs. In order to access these documents, one would need an information retrieval system that can handle these characteristics in an efficient and precise manner. A *web search*, for instance, is a powerful retrieval system in terms of document coverage and scalability. Querying and traversing search results, however, can be a trial-and-error and tedious activity without knowing the right keywords. One way to circumvent such problems is through the introduction of *categories*. In the context of information retrieval and the web, categories are labels that can explicitly and implicitly characterize and group a set of documents. The process of assigning them is called *categorization*.

One of the benefits of categorization is the expansion of the logical view of a document to include implicit knowledge, such as genre[1], which then can be utilized to improve the performance of search engine [14, 24]. For example, searching for humorous news can intuitively be performed using the query keywords "humor" and "news". Most search process, however, would misranked or completely missed popular news humor websites, such as The Onion (http://www.theonion.com), because the keyword "humor" is not explicitly stated within the content of these websites. If the search space is extended to genre categories, which is what the keyword "humor" is, such errors can be avoided.

Another benefit of categorization is the organization or grouping of a large set of documents into a structure that allows a non-query (non-problem specific) retrieval such as *web browsing*. Accessing unread news articles, for example, is easier to perform by navi-

[1] a special nonrestrictive case of categories that is defined by some common communicative purpose or functionality and characterized by formal cues [25] (e.g. scientific paper is a genre whose purpose of conveying a scientific idea through a composition of hypothesis and research)

gating or browsing a group of predefined categories or subjects (e.g. sports, basketball, business, and politics) because it is counter-intuitive for users to come up with specific queries such as names and events. For this reason, newspaper portals and homepages categorize their articles into a taxonomy of generalized subjects.

In this thesis, we consider the realization of web categorization as an important factor in improving web retrievals. We hypothesize that in order to build an effective web categorization system, an approach that combines *automated text categorization* and *collaborative tagging* is needed. Through this approach, a set of training data is derived from collaboratively tagged documents. The data is then used to train a model through automatic categorization method. Finally from the model, unseen documents can now be categorized to their appropriate categories.

For the rest of this chapter we will have the following; Section 1.1 presents the problem of web categorization and the motivation for our hypothesis. Section 1.2 elaborates the contributions we made to the hypothesis and outlines them to the chapters on this thesis.

1.1 Motivations

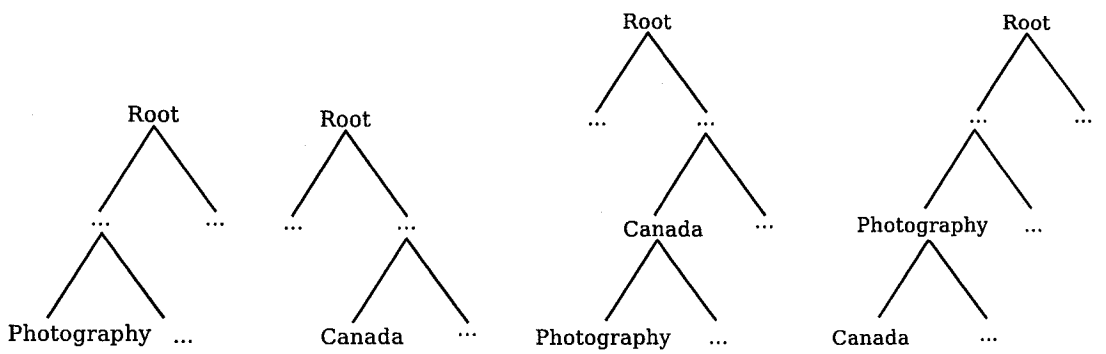
Understanding our motivation for combining automatic text categorization and collaborative tagging is something that has to be looked at in two directions. First, we need to understand why characterizing web categorization through collaborative tagging is beneficial for us. Secondly, we need to see to what extend automatic text categorization can improve the performance of a collaborative tagging system.

1.1.1 Web Categorization and Collaborative Tagging

Using collaborative tagging to categorize the content has been quite popular with recent web applications. In collaborative tagging, we have a system that allows users to annotate a document with their own personal categories and descriptions. This, in contrast to a more traditional approach such as web directories, abandons the need to have expert contribution such as "librarians" [17]. The resulting ontology is also different to that of web directories because instead of a well defined taxonomical hierarchy, collaborative tagging generally assume ontologies as faceted structureless organizations [47, 39].

In the case of web categorization, the advantage of collaborative tagging can best be explained through the following argument. In trying to categorize documents in a heterogeneous setting like the web, many diverse categories can be used to describe

multiple documents. For our example, let us consider **"Photography"** and **"Canada"** as two of these categories.



(a) The hierarchical placement for the category "Photography"

(b) The hierarchical placement for the category "Canada"

(c) The hierarchy with "Photography" as subcategory of "Canada"

(d) The hierarchy with "Canada" as subcategory of "Photography"

Figure 1.1: The possible hierarchical interpretation for categorizing the categories **"Photography"** and **"Canada"**

In traditional taxonomy such as web directory, both categories are generally treated as individual and separate categories in the sense that there are people who wished to browse each category individually. Therefore we generally have a separate hierarchical paths such as the one shown figure 1.1(a) and 1.1(b). When the two categories intersect , i.e. we have documents that are correctly categorized to both categories, however we have the option of placing these documents in either the path shown in figure 1.1(c), 1.1(d), or both. In any case, the placement of intersecting documents in this rigid structure can be confusing to both users and annotators alike.

For users, both paths, while similar at first, may represent the placement of different documents. the path shown in figure 1.1(c), for instance, may refer to documents relating to photographs of Canada, i.e. photographs of Canadian landscape, etc. Meanwhile for the path in figure 1.1(d), users may expect to find document relating to photography in Canada, i.e. famous Canadian photographers, photography clubs in Canada, etc.

For expert annotators, maintaining the relevance and applicability are also non-trivial. First, annotators have to produce a hierarchical structure that can be accepted by all users of their system. With the addition of new documents and new categories, they also need to maintain the relevance of that structure. As argued by Shirky [47], the assumptive nature of hierarchies does not seems to mesh with the heterogeneous nature

of the web.

In collaborative tagging, the annotation categories are simpler because we do not have the need to consider any hierarchical structure. In fact the process is easy enough that users are encouraged to be annotators as well. This in terms of direct applicability and feedback from the users put collaborative tagging as a more collective and democratic approach.

For navigation, user can expect a more intuitive interface. Searching and browsing Tags or categories are akin to the keyword searches that filters document through queried categories. While it is true that the right combination of keywords or keytags would provide a better retrieval result, the proclivity of web user to use search engines makes collaborative tagging to be intuitively used.

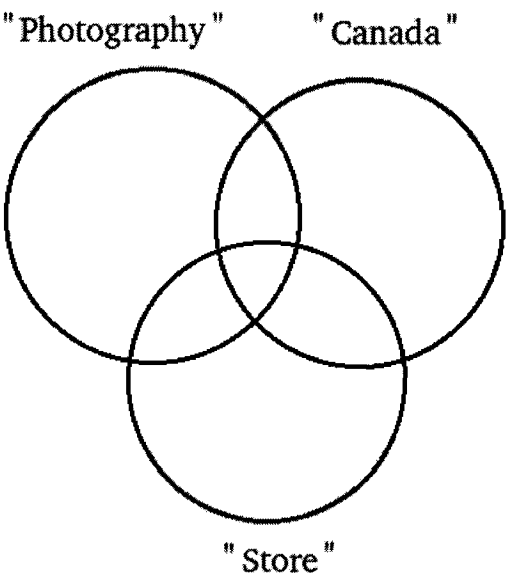


Figure 1.2: Visualizing retrieval in collaborative tagging through the filtering of queried tags.

As illustrated in figure 1.2, we can visualize retrieval and browsing in collaborative tagging as the process of filtering documents through the combination of input categories, i.e. **"Photography"** and **"Store"**; **"Photography"**and **"Canada"**; etc.

If we believe today's web and its users to be heterogeneously spread across different interests and expertize, then it is logical to see collaborative tagging as the better repre-

sentation for web categorization. In this thesis, we assume this to be true. Therefore as the first part of our motivation, we believe collaborative tagging to be the best system to characterize the current state of web categorization.

1.1.2 Improving the Coverage of Web Categorization through Automatic Text Categorization

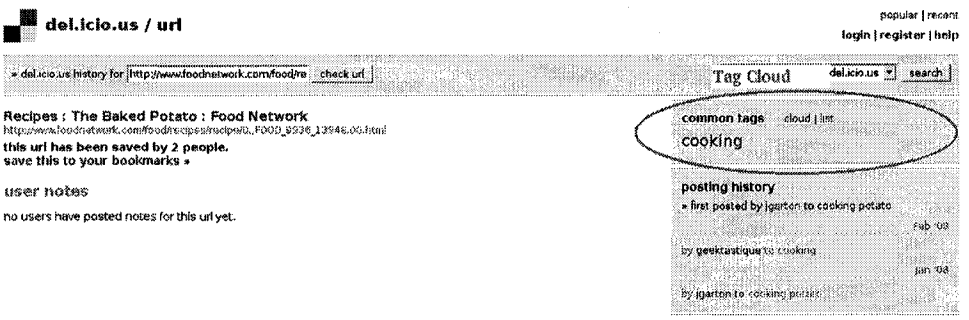
Despite all the advantages collaborative tagging has over hierarchical web directories, the system itself still relies on human annotations. This in return creates the possibility for documents to not be annotated by enough people. As highlighted by Shirky [47], the advantage of collaborative tagging is the evidence of *collective intelligence* on the tags. In other words, the absence of collective intelligence or collaboration may lead to documents of interest not being as reachable (less tags to query at) as documents that have been populated with tags from multiple users, i.e. collaboratively tagged.

To illustrate this argument. we have in figure 1.3, the difference in the number of tags for two recipe pages that have been tagged by different number of people in Delicious [43]. As seen in that example, the recipe document that is tagged by 2 users seems to only have the tag **"cooking"** and **"potato"** attached to it. the recipe document that was tagged by more than 37 people, on the other hand, seems to have a more complete descriptions with tags such as **"recipe"**, **"cooking"**, **"food"**, **"macaroni"** and **"cheese"** attached to it. In practice, it is easier to navigate and browse documents that are sufficiently tagged by enough users because there are many intuitive query tags that allow us to reach them. Therefore for tagging to be useful, there need to be a more consistent source of annotation.

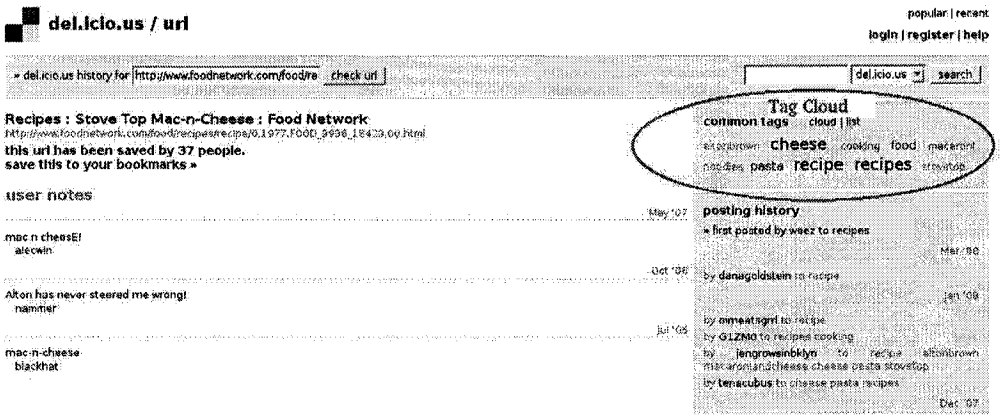
Given the above motivation, we will now focus on methods that offer more consistent and non-human annotations such as automatic text categorization. In the past, automatic text categorization has been researched on as text classification experiments on more traditional text corpora such as newswire documents and web directories.

Mccalum and Nigam [30], for example, shows the feasibility of probabilistic framework such as mixture model Naïve Bayes while study by Dumais and Chen [15] discusses more advance topics like the hierarchical categorization of web directories.

In contrast to these studies, we also have recent studies that tries to apply automatic text categorization on collaborative tagged data. In Brooks and Montanez's [5], we have a study that explores extractive approaches such as keyphrase extraction and in Ohkura et. al. [37], we have an automated tag system that utilize Support Vector Machine



(a) The tagging page for a recipe page that were tagged by 2 users. We can see from the area inside the red ellipse or the tag cloud how sparse the tags assigned to this document



(b) The tagging page for a comparable recipe page that were tagged by 37 users. instead of sparse tag cloud, we can see several tags occupying the space for the tag cloud.

Figure 1.3: A comparison on the the degree of representation or robustness of tags for 2 comparable recipe pages that came from different level of "collaboration".

classifiers. Despite the positive result, however, these studies seemed to be too simplistic to be compared to the ones achieved in traditional corpora literature. Therefore, it seems plausible to improve the classification of collaboratively tagged corpus by retrospectively looking at some of the studies that had been done in traditional corpora.

In summary, from our first motivation, we argue for collaborative tagging to be the best representation for web categorization. This also implies that data that is collaboratively tagged as a better corpus for text categorization than ones that were derived from web directories. Given the infancy of the collaborative tagging literature, our second motivation is to build more definitive and conclusive experiments on the subject of automatic text categorization and collaboratively tagged data. Thus in our contribution, we built a more comprehensive experiment that combines the hierarchical approach from web directory categorization and dataset that had been collaboratively annotated.

1.2 Thesis Contribution

For the thesis contribution, we have laid out an experimental framework on the subject of categorizing collaboratively tagged data. First, we consider collaborative tagging system as a source of categorization dataset. The social bookmarking system Delicious [43], a well known example of collaborative tagging system, is chosen as the source of our dataset because of its coverage and popularity among researchers. Based on the acquired dataset, we made several observations such as Power-Law distribution and robustness of categories to support the argument that collaborative tagged data is comparable to the previously studied web directories, the benchmark corpora for web categorization research.

As contribution to the subject of categorization of collaboratively tagged data, we decided to also adopt a hierarchical text categorization approach and compare them with the normal or "flat" categorization approach. As far as we know, categorization of collaboratively tagged data is considered a new categorization problem. Therefore at its infancy there are few experiments that deals with more structured or advanced approaches such as hierarchical categorization. As prerequisite to applying a hierarchical approach, we also need to consider the creation of a structured taxonomy. For this thesis, we consider a very simple approach to creating an artificial taxonomy by creating a two-level hierarchy using methods that were similar to previous tag clustering researchs[2, 18].

Once all of the parameters of our contributions are explained, we move on by performing two text categorization experiments that compares different classification approaches. For the first experiment, we wished to compare the different learning methods that are

generally used in text categorization problem. The methods chosen are selected to represent two different classification approaches. In Naïve Bayes, we have a method that relies on probabilistic framework to classification. In Support Vector Machine (SVM), we have a maximum margin classifier that aims to reduce generalization error. The result of this experiment clearly shows SVM as the more superior classifier and allows us to establish a baseline to our hierarchical approach. In the second experiment, we apply the hierarchical text categorization approach to the two learning methods we used for our first experiment. Parallel to previous hierarchical text categorization studies[15, 28], we showed some improvements in performance and a significant overall reduction in training and test time.

Based on the result of the two experiments, we have shown that it is plausible for a collaborative tagging system to improve automatic text categorization as an external source of annotations. We have also shown that by applying hierarchical categorization to the problem, we can actually improve the result of such system. This is important because for future of automatic tagging systems we can easily improve the performance of its annotations by tapping into more sophisticated and previously studied text categorization approaches.

1.3 Thesis Outline

This thesis consists of 5 chapters. **Chapter 1** discusses the motivation as well as the introduction to the topic of collaborative tagging and automatic text categorization. **Chapter 2**, formally defines collaborative tagging and automatic text categorization. In this chapter, we will also discuss some of the studies that involve the improvement in navigational and coverage of a categorization system. These studies include: the improvement in navigational elements such as the creation of tag clusters [2], navigational structure such hierarchies [18]; and the aforementioned automation of tags and categories labelling on collaborative tagged data[5, 37]. In **Chapter 3**, we discuss collaborative tagging system as the source of categorization dataset. In Chapter 3, We also discussed the hierarchical text categorization and the methods that creates the hierarchical structure for the said approach. **Chapter 4** deals with the experiments and the results of applying both flat classifications and hierarchical classifications to the collaboratively tagged dataset. Finally, as a conclusion, we have in **chapter 5** the summary of the observations we have made in our experiment; and the discussion about the future work and the overall direction of this research.

Chapter 2

Related Works

As proposed in our introduction, we are interested in merging two research areas: collaborative tagging and automatic text categorization. Since there are many aspects of both areas that were not included in Chapter 1, the following sections will describe the two topics in greater details. More specifically, we will present collaborative tagging as a retrieval system and automatic text categorization as a scalability component for the system.

2.1 Collaborative Tagging

In the first part of our Literature Review, we will look at collaborative tagging both in terms of research and applications. As a formal representation, we will look at Mika's tripartite graph description of a collaborative tagging system[34]. For its applications, we will look at some web services that incorporate collaborative tagging as part of their retrieval system. Finally, for researches related to the subject, we will look at some attempts to improve the ontology[18, 2] and coverage[37, 5] of a collaborative tagging system.

2.1.1 Formalizing Collaborative Tagging through Graph Representation

In [34], Mika describes collaborative tagging as a tripartite graph. In this case, the tripartite graph, T, is a three entities (set of nodes) graph that consists of:

1. A set of actors or users of the system, $A = \{a_1, ..., a_i\}$

2. A set of concepts or tags, $C = \{c_1, ..., c_j\}$

3. A set of resources such as documents, $D = \{d_1, ..., d_k\}$

To complete this graph, we have different sets of edges or annotations that connect the three entities, $T \subseteq A \times C \times D$. For example, if a user annotates the Yahoo! homepage, `http://yahoo.com`, with the tag "***Yahoo***", the graph would represent this activity by connecting an edge from the said user's node to the tag node "***Yahoo***", another edge from the tag node "***Yahoo***" to the resource node that correspond to the `http:///yahoo.com` page and an edge between the user node and the document node.

Mika also proceeds to break down the analysis into smaller subsets or bipartite representations. In one subsection, the actor and concept interactions, $A \times C$, in which annotations of concepts from different users are used as evidence of overlapping patterns or social networks (through clustering). This observation is very important because the social network aspect from the actors node allowed Collaborative Tagging system to form a concept ontology that is more semantically accessible to users.

In another subsection, the concept and resource interactions, $C \times D$ are presented as examples of annotation patterns that are generally associated with text mining tasks. That is, in comparison to the the actor and concept interactions, $A \times C$, the clustering found in concept and resource interactions appears to have stronger concept associations, i.e. a more consistent concept recognition from content, but weaker semantic accessibility to the users, i.e. less intuitive than human generated ontology.

2.1.2 Examples of Collaborative Tagging

In order to understand the importance of collaborative tagging, we need to look at its presence in today's web. The following three examples are web services that have successfully use collaborative tagging to leverage the retrieval needs of their users.

Del.icio.us

Created in 2003 by Joshua Schrater, Delicious is one of the more important collaborative tagging service because its popularity among Internet users as a *social bookmarking* service allows it to be a reliable model or source of data. In Figure 2.1, we can see the homepage of Delicious which offers some recommendation for some web pages (It should be noted that beside the homepage and the tag cloud, Delicious is rarely editorialized and it is completely user generated).

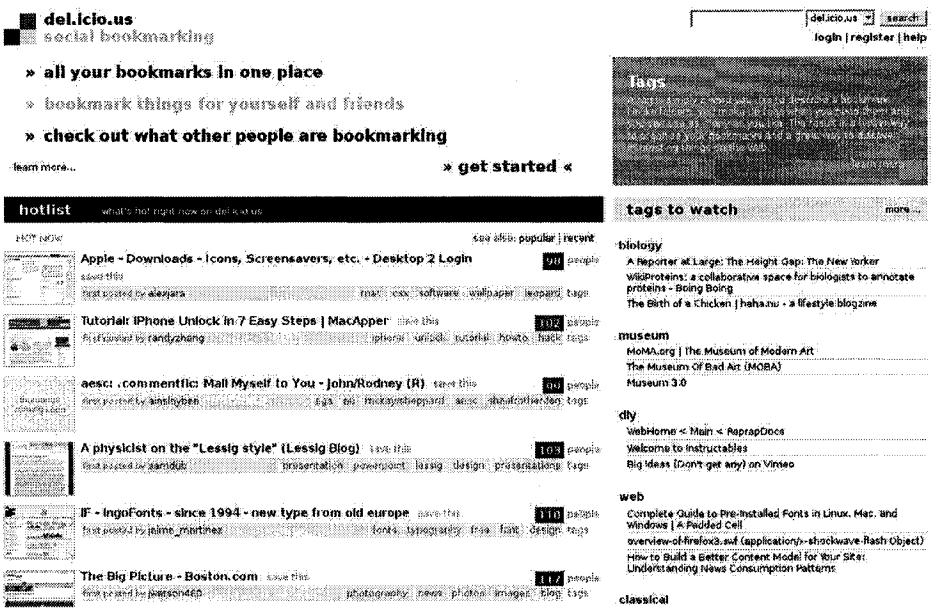


Figure 2.1: The homepage for Del.icio.us which contains recommendations for a few websites or tags that are currently popular within users of the system.

The idea of social bookmarking is to have a collection of websites that have been bookmarked by registered users of the system and share them with other users of the system. The process of bookmarking of a website generally involves user submitting the address of the site and tagged or annotate it with keywords or phrases for retrieval purposes. Meanwhile the sharing process is about making the bookmarked websites accessible to other users to discover and bookmark for their own interest.

Other than Delicious, there are other social bookmarking services. Furl(`http://www.furl.net`), and Digg(`http://digg.com`) for examples are also considered as social bookmarking websites. However, given Delicious' greater emphasis on the tagging aspect of social bookmarking, we believe Delicious would make a better example of collaborative tagging and the tripartite representation mentioned earlier.

In terms of usage, Delicious can be described as the replacement of previous incarnation of web categorization system such as web directories. For visitors of the website, Delicious provides a way to discover new websites and browse them according to topics or categories that users of the system have bookmarked.

As stated in our introduction, a collaborative tagging system differentiate itself from web directories by excluding rigid hierarchical structure. Delicious promotes a non struc-

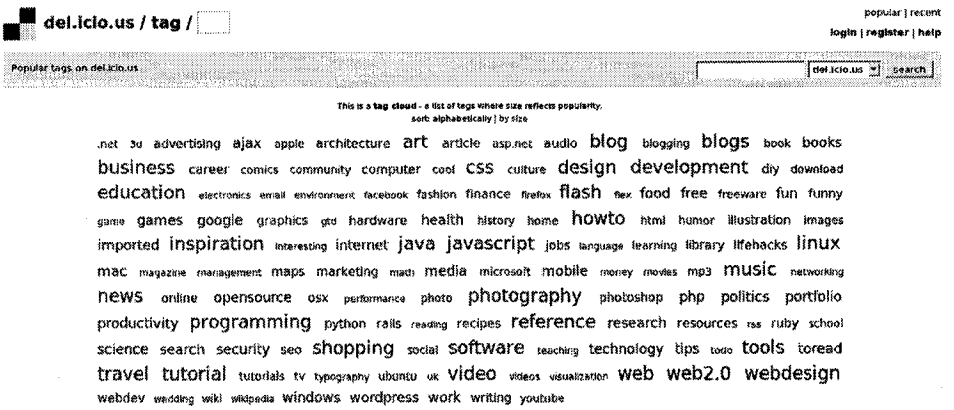


Figure 2.2: A tag cloud structure that allows users to have a starting point to navigate websites within Delicious. Its membership is based on the popularity within the system.

tured navigation that can either be based on keyword searches or tag navigation. In Figure 2.2, we have an example of tag navigation in the form of popular and recent tags called **tag clouds**. Inside a tag cloud, keyphrases or tags are indexed and presented with visual cues that will allow user to retrieve documents.

One of the visual cues, for instance, is the use of font size to represent popularity. A tag with a large font size is generally a tag that is popularly used among the users of the system. The user generated annotation coupled with the ease of use of tag clouds and searches according to some researchers[47, 39] creates a more intuitive and a organic browsing experience than the one provided in web directories.

Flickr

Beyond Social Bookmarking, there are also other types of web services that uses collaborative tagging to organize resources. Flickr (`http://flickr.com`), for instance, is one of the many multimedia sharing/hosting services that uses collaboratively tagging to organize their content.

Inside Flickr, we also have a photo sharing websites that allow users to upload and share pictures. Just like Delicious, Flickr employs tagging as a mean of personal organization and general navigation of topics. However, it should be noted that when we talk about resources on this website we refer to photographs that are uploaded by the user. For example, with the exception of pictorial cues as content, the tag cloud that is used in

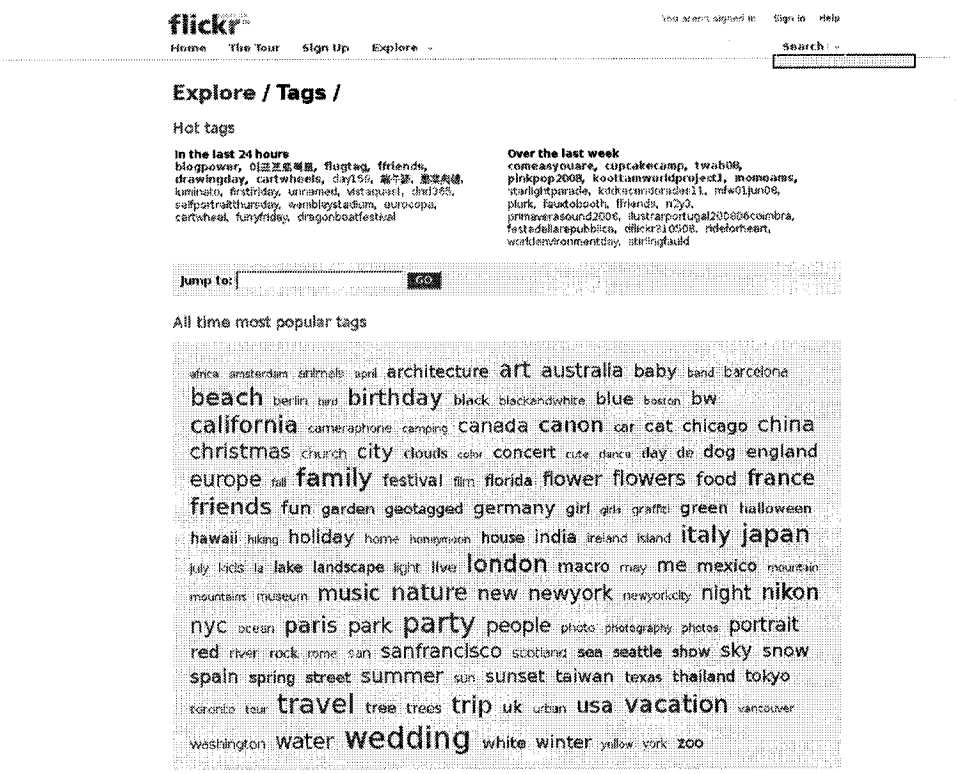


Figure 2.3: A tag cloud from Flickr that generally describes the locations, events, and landmarks of a picture

Flickr (See Figure 2.3) has similar layout and functionality as the one found in Delicious.

In the case of multimedia sharing such as Flickr, collaborative tagging is considered valuable because it allows a semantic indexing of objects that could never be explicitly written in the content of the resource. That is to say if a picture of a beach in Hawaii, would never be specifically retrieved for it with keyphrases ***Hawaii*** and/or ***beach*** if it had not been tagged by the users.

Amazon

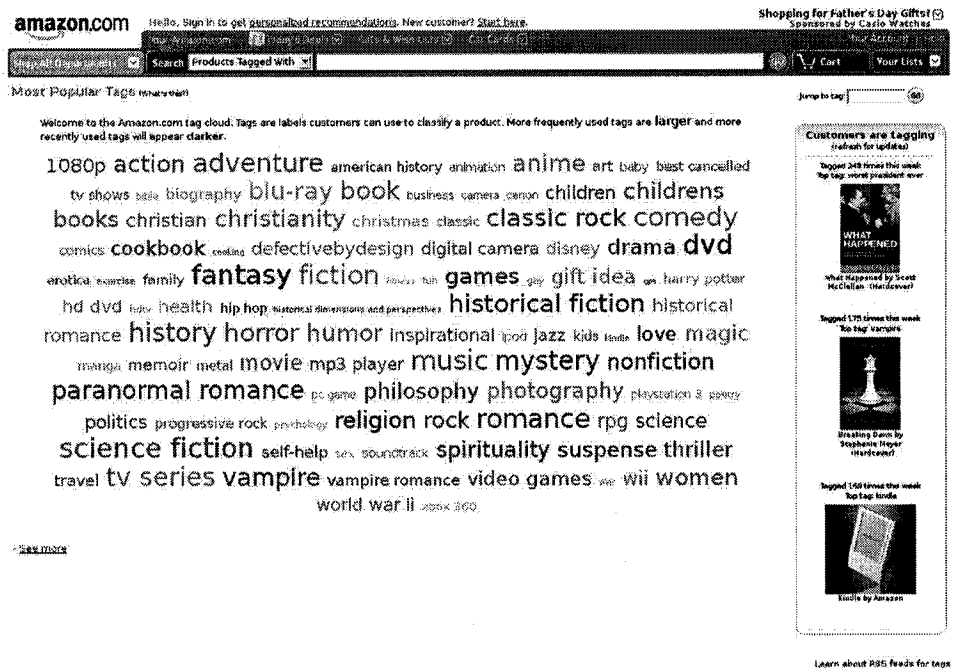


Figure 2.4: A tag cloud from Amazon that allow user to browse popular items sold in the website

In our last example, we can consider the case of having collaborative tagging in an E-commerce environment. Amazon, for example, is a one of the leading E-commerce website that has its start by selling books and multimedia items over the Internet. In recent years, however, the website has diversified its items by allowing medium and large vendors to list their items and sell them under one domain.

With diversified vendors and items as its business model, the challenge for Amazon is to organize and index products with an intuitive and helpful interface for users. By

offering both collaborative tagging and a general top down categorical taxonomy[1], Amazon offerred a compromise answer that allows vendors to organize their items in a large and structured database and user to have the option of using both structured (vendor defined) and non-structured (user generated and socially created) ontologies to browse items.

In Figure 2.4, we have an example of tag cloud that can describe the popular buying habit of users on Amazon without being too similar to the generalized break down that is found in a typical E-commerce store.

2.1.3 Collaborative Tagging as a Research Area

In the study of collaborative tagging, defining and facilitating the actors-concepts-resources interactions are only considered the initial component to the whole retrieval system. Research on improving the navigation in a collaborative tagging system, i.e. tag cloud and queries, is important because the structureless aspect of collaborative tagging prevents a more efficient and structured navigation such as a hierarchy to be built manually. For this thesis (see Section 1.1.2) and other researches, leveraging the coverage and consistency of human annotations are also important due to ratio of human annotations needed to the size of the Web, i.e. the scalability of the system. In the following sections, we will discuss some of the studies that have been done in the two areas.

Improving Navigation in Collaborative Tagging System

Given the raw and structureless value of collaborative tagging, there are many undiscovered semantic and ontological properties that have yet to be mined. In one of the early papers on collaborative tagging, Cattuto et al.[6] observed the shared usage of certain tags and the semantic significance of communal tags (referred as semiotic dynamics) by mining the similarity or co-occurrence statistics of tags in Delicious. The trend of observing the patterns behind tags co-occurrence also continues for the purpose of creating navigational structure for tag browsing.

In [18], Heyman and Garcia-Molina demonstrates a hierarchical taxonomy that was derived from a hierarchical clustering algorithm. Their hierarchical clustering algorithm starts by obtaining a tag co-occurrence matrix and a ranking of tags' centrality. The

[1]It is possible in Amazon to browse both vendor defined hierarchical categories of products and user generated tags to find and purchase objects

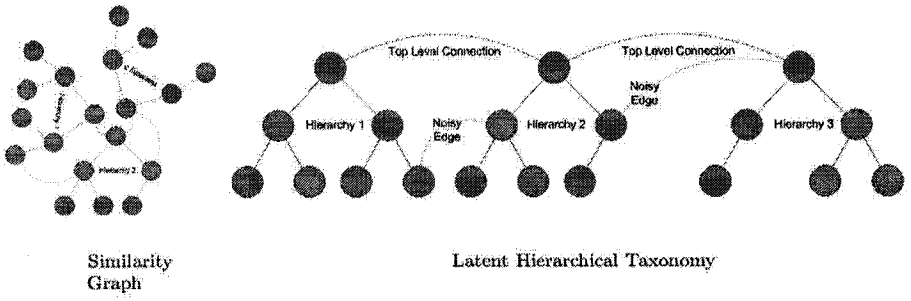


Figure 2.5: An example from [18] that shows a similarity graph and the possible hierarchy that is derived from the hierarchical clustering algorithm.

method would then traverse all candidate tag (starting from the most central) to determine whether it is suitable to be a parent tag or a child tag of a similar node. This, also shown in Figure 2.5, means that there are certain edges that are removed from the similarity graph to induce a hierarchy. As shown in Figure 2.5, a similarity graph can actually be transformed into a top-down hierarchical representation through the aforementioned method.

Similarly, in [2], Begelman et al. proposed a clustering algorithm to deduce a set of groupings for related tags, i.e. a one-level hierarchy, that can be used to navigate a pivot tag or tag space in general. In their more sophisticated algorithm, they propose the adoption a spectral bisection algorithm [54] that uses a modularity function to avoid providing the number of clusters. The algorithm works by essentially removing the edges between two set of nodes that are contained within one cluster recursively, i.e. bisecting a cluster. The modularity function, however, would stop the algorithm to bisect a cluster further if the modularity score of the bisected clusters is smaller than the original one[2]. In figure 2.6, Begelman et al. present an example of the resulting cluster that their clustering method provided.

While it is clear from both of these studies that a more structured taxonomy can automatically be derived, they also contradicted our earlier motivation of having a structureless ontology. To answer this paradox, we need to consider the hierarchical taxonomy

[2]The Modularity Function measures the quality of a set of clusters according to the independence model

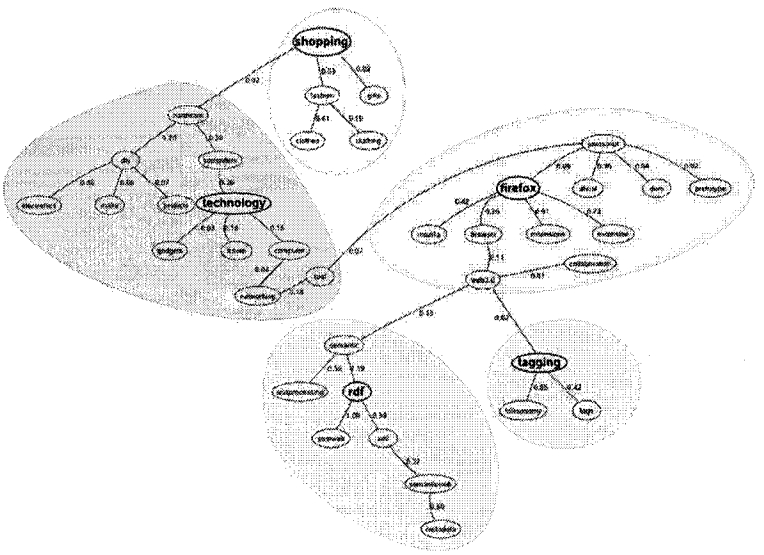


Figure 2.6: A Similarity graph that had been clustered with the spectral bisection clustering. Image taken from [2]

discussed by [47, 39] was predetermined by a central authority and tends to hinder the process of user annotations. The hierarchies or any structured taxonomy proposed on these studies, on the other hand, are derived after the annotation process. They therefore do not negatively affect the complexity of human annotation while providing a more effective way of navigating documents.

Automating Tag Annotation with Text Categorization Approaches

In [5], Brooks and Montanez proposed a study to automate the tagging of blogs and newswire articles. Blogs or weblogs are a sub-genre of web documents in which every day internet users publish articles or journals that can be commented by their readers. Similar to tags for other web documents, tags in blogs are used to organize concepts and help retrieval systems, however for most blogs, annotations of tags are generally done by authors of the blogs themselves. In that study, several datasets such as a set of clustered news items from Google News (`http://news.google.com/`), and a set of blogs from Technorati (`http://technorati.com/`) were used for an unsupervised categorization experiment[3] that involves creating tags from the highest TF-IDF scoring words (top

[3]Unsupervised categorization are a subclass of learning methods that does not require a training phase from a labelled or annotated sets of document.

three words). At the conclusion of that study, Brooks and Montanez showed the cosine similarity or the quality of clusters from the automated tags are found to be significantly higher than the ones labeled by blogs' authors and the external system used in Google News[4]. Thus, despite of the absence categorical measurement such as precision, recall and F-Measure, Brooks and Montanez claims that simple unsupervised methods such as keyphrase extraction can actually produce a grouping or categorization of documents that are comparable to human annotations.

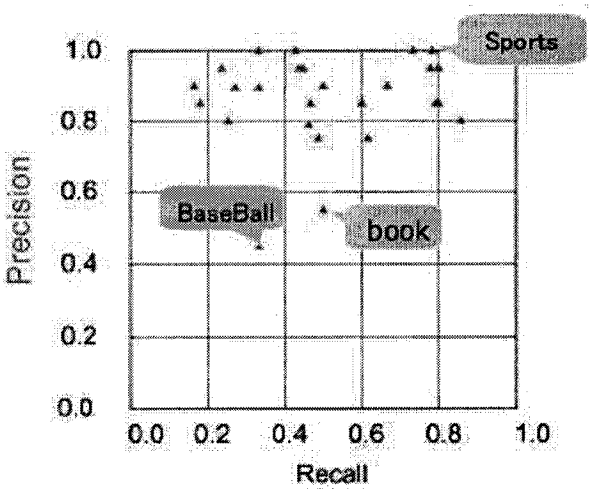


Figure 2.7: A Precision-Recall Graph of several SVM Taggers that were provided by Ohkura et al. in [37]

In another study [37], Ohkura et al. proposes a supervised approach by training SVM (Support Vector Machine) classifiers to annotate documents. In that experiment, a set of Japanese blogs was collected and several tags or concepts were chosen as learning targets based on the their distributions, i.e. popular tags that have more than a certain number of documents annotated to them. The result of this experiment showed that the annotations produced by the SVM classifiers are feasible replacement for human annotations. As seen in Figure 2.7, the SVM classifiers are able to produce high precision (in general around 90%) and inconsistent recall (a fluctuation between 15% and 90%) annotations that are analogous to a highly selective human annotator.

[4]Cosine Similarity measures the quality of a cluster by measuring the vector similarity of each document inside. In the case of tagging, each tag can be perceived as a cluster.

In this thesis, we consider both of the studies done by Ohkura et al.[37] and Brooks and Montanez[5] to be similar in goal. However the method proposed by Ohkura et al. produced a more favorable conclussion because it is a better reflection on human annotations. While unsupervised approaches such as keyphrase extraction are simpler to obtain, they seems to ignore the notion that human generated tags are not necessarily derived explicitly from the content or body of the document. Supervised categorization approaches, such as the one discussed in the next section, are consider the more popular approach because they allow better quantification of limitations and capabilities of learning specific target concepts.

2.2 Automatic Text Categorization

Following our brief discussion of the supervised categorization approach proposed by Ohkura et al.[37]. we decided to have supervised text categorization, sometimes refered to as automatic text categorization[45], as the framework for this thesis.

In formulating our approach as an automatic text categorization problem, there are generally four components that we need to consider. First, we need to determine a dataset or corpus that will be used to train and evaluate our classifiers. Second, we need to consider a representation or a set of features that will transform the documents into a suitable input for the classifiers. Third, we need to consider the classification approaches, as well as the algorithm, that would "learn" to annotate these documents. Finally, for the fourth component, we need to determine methods by which the resulting categorization can be evaluated.

For the rest of this chapter, we will describe automatic text categorization by describing its individual components.

2.2.1 Dataset

One of the initial step in a categorization experiment or project is to find the proper dataset. The choice of dataset is very crucial because it affects the direction of the experiment in terms of goals, challenges, and applicability.

By definition, a dataset is a collection of labelled/categorized documents that will be used for both training and testing a categorization system. More formally, using Sebastiani's description of a categorization task[45], a dataset or a corpus $Co(D, C, L)$ is defined as the following.

First, a dataset can be defined through its elements: a set of Documents $D = \{d_1, ..., d_n\}$; a set of categories or tags $C = \{c_1, ..., c_m\}$; and a set of correct document-category labellings $L = \{l_{1,1}, l_{1,2}, ..., l_{m,n}\}$. The set of documents is the set of training and testing instances which a classifier will learn from, i.e. train and test. In order to use them in this manner, they are generally divided into a training set and a test set. The set of categories is the set of goals for which the classifier will try to classify the documents to. The labellings represent true classifier's value or the assignments of 1 and 0 (or *true* and *false*) for all permutation of document and categories that have been obtained from a *reliable source.*

	training set				test set			
	d_1	...	...	d_j	d_{j+1}	...	...	d_n
c_1	$l_{1,1}$	...	...	$l_{1,j}$	$l_{1,(j+1)}$	...	...	$l_{1,n}$
...	...	...	...	...	...	...	...	...
c_i	$l_{i,1}$	...	...	$l_{i,j}$	$l_{i,(j+1)}$	...	...	$l_{i,n}$
...	...	...	...	...	...	...	...	...
c_m	$l_{m,1}$	...	...	$l_{mj,}$	$l_{m,(j+1)}$	...	...	$l_{m,n}$

Table 2.1: A representation of a Corpus as a labelling matrix

In order to visualize a dataset as a structure, a big labelling matrix such as Table 2.1 can be used to illustrate the different components and their role. In this matrix, the columns represent the different documents $D = \{d_1, ..., d_n\}$; the rows represent categories and tags $C = \{c_1, ..., c_m\}$; and each cell $l_{i,j}$ represents the labelling assignment.

Another description of a dataset involves a binary target function (or true function) $f : D \times C \rightarrow \{0, 1\}$. In this mathematically unknown function, any document and any category defined in their respective set can be taken as inputs while the function produces the value 1 or 0 as the corresponding value of their labeling. This function is called a target function because in a supervised categorization framework, the goal of a classifier is to produce a hypothesis function $f' : D \times C \rightarrow \{0, 1\}$ that best approximates it.

2.2.2 Features

Since automatic text categorization can be represented as a function that takes a document and category as inputs, $f' : D \times C \rightarrow \{0, 1\}$, we still need to consider the appropriate form of document representation $\hat{d}_i$ for each docucments in the collection,

$d_i \in D$. An effective choice of representation would allow the hypothesis function, f' to better approximate the target function or true labeling, f. On the other hand, a bad representation could make a hypothesis function performs worse. The purpose of this chapter is to describe the choices to be made and factors to be concerned when building an effective representation.

In many document categorization systems, reliance on text content and other lexical based features are prefered because they have shown ability to discriminate various genres[48, 14], and topics[15, 20]. For the purpose of this thesis, we will refer to the vector space model [42], a commonly used representation in information retrieval, and discuss its deployment in a categorization framework.

Vector Space Model

The vector space model, in general, deals with any representation that transform a text document into a vector of numerical or binary valued terms, i.e. indexing the content. In this representation, the choice of terms would determine the characteristics of the model. Setting a single-unit or token of word as a term would create a unigram model, a sequence of two words as a term would create a bigram model, and so on. The following are examples of this representation:

Bag-of-Words (BOW) or unigram model. In this representation, documents are tokenized into single-unit of words and indexed to a vector that contains the words' corresponding weights. In many large scale web categorization studies [15, 28, 26], this representation is very common because of its simplicity and comparable effectiveness over more sophisticated features, e.g. ones that consider sequential ordering (N-gram model). For example, from the sentence *"This apple is not as fresh as that orange."* we can tokenize nine words and tabulate them into the vector;

$$\begin{pmatrix} This & :1 \\ apple & :1 \\ is & :1 \\ not & :1 \\ as & :2 \\ fresh & :1 \\ that & :1 \\ orange & :1 \end{pmatrix}$$

As options, there are several refinements to the representation of a Bag-of-Words that can help improve the statistical significance of this representation. First, we remove all

upper case words into lower case because we do not wish to differentiate words by their position in a sentence. Second, we remove potential *stop words*. In the context of lexical analysis, stop words are defined as a list of common English words such as "I", "The", "a", "of", etc. The practice of removing stop words in categorization task is based on the analysis that these common words lack the statistical properties to discriminate between any categories. Third, we *stem* words into their root form before indexing them. In word stemming, we collapse several words into one vector space by removing suffixes. The idea behind stemming is to reduce the size of the vector space with minimal loss of information. One of the most common algorithm for this process is called Porter's stemmer[38].

Bag-of-URLs[26]. As the name suggest, this representation consists of tokenization of URLs that are found in a webpage, i.e. the text between the <a> and tags, and index them like we would for a Bag-of-Words representation. The method in which URLs are tokenized involves the splitting of terms between the symbols "/", "_", ".", "-", "&", "=", and "?". Each symbol mentioned is generally used to delimit terms in URL, however the last three symbols are exclusively used as less secure form of passing variables to scripts that run for a given page. As an example consider the URL `http://www.graphic-design.com/Photoshop/pencil_sketch/index.html`. From tokenizing that address, a Bag-of URL representation is produced in the form of the vector;

$$\begin{pmatrix} www & :1 \\ graphic & :1 \\ design & :1 \\ com & :1 \\ photoshop & :1 \\ pencil & :1 \\ sketch & :1 \\ index & :1 \\ html & :1 \end{pmatrix}$$

In [26], Kolari et al. believe the Bag-of-URL representation is effective in detecting spams because valid weblogs would link to more legitimate domains such as *"Flickr"* and *"Technorati"*. In the case of general categorization, the inclusion of URL tokens as part of our vector space model seems natural because it would at the very least provide features that would discriminate domain specific tags or categories, such as ***Digg***, ***Flickr*** ***Yahoo***, and ***Google***.

Feature Reduction

Despite its simplicity and effectiveness, one disadvantage of using the the vector space or bag-of words model is the resulting vocabulary size. For instance, the Reuters-21578 dataset, a famous benchmark for text categorization, has roughly 16,000 unique word tokens[58]. Despite the application case normalization, stemming, and stop words removal, a vector space model would always produce a high dimensional vectors because they always deal with a collection of documents with multiple topics and robust vocabularies that entails it.

In most text categorization algorithms, high dimensional features are not desirable because they tend to affect the time complexities of these algorithms. In the same study[58], Yang and Pedersen shows that high dimensional features do not necessarily improve but in some cases hinder the performance of a classifier. With that in mind we will look into two methods of reducing the number of features in our bag-of-words representation.

Document frequency. In this method, we are looking at eliminating terms according to the number of documents they have appeared in. The most common method in using this property, is to set a threshold which will act as the minimum number for the frequency to be in the feature set.

Unlike the stop word removal, the idea of reducing a feature set according to document frequency comes from the belief that terms that terms that appear very sporadically offer little in terms of their ability to predict categories and are perceived as noises. This is definitely true for terms that appear in one or two documents (according to Zipf's law [60] there should be many of them) because given their low prior probability, chances for any of them to effectively discriminate any categories for unseen examples are low. However despite this claim, we need to make sure that we do not set a threshold that is too high, because an aggressive threshold might remove terms that are useful to certain target classes, especially minority classes.

Information Gain (IG)[9]. In this second method, we compute for each term the average mutual information of a term for a given target class. Average mutual information is the difference of entropy of class C or $H(C)$ and the entropy of the class given the presence and absence of a term or $H(C|W)$. The formulation for Information

Gain with respect to a given term t is:

$$\begin{aligned} IG(t) &= H(C) - H(C|t) \\ &= -\sum_{c \in C} P(c) \log P(c) \\ &+ \sum_{t \in \{0,1\}} P(t) \sum_{c \in C} P(c|t) \log P(c|t) \\ &= \sum_{c \in C} \sum_{t \in \{0,1\}} P(c,t) \log \frac{P(c,t)}{P(c)P(t)} \end{aligned} \tag{2.1}$$

where $P(c)$ is the the number of documents labeled with the category c in our dataset divided by the total number of documents, $P(t)$ is the the number of documents that contain (do not contain, if $t = 0$) the term t divided by the total number of documents and $P(c,t)$ is the number of documents which is labeled by the category c, and contain (do not contain, if $t = 0$) the term t divided by total number of documents. After IG is computed for each term, we can now rank terms by how much gain they provide (how high their IG scores are) and proceed to select features according to this rank, the top being the priority.

In [58], Yang and Pedersen noted Information Gain for its excellent result in its ability to aggressively remove features on both the Reuters and OHSUMED dataset. They also present a favorable analysis of the complexity of obtaining this score, i.e. $O(N)$ where N is the number of documents.

Term Weighting and Normalization

Fot the final component of the vector space representation, we need to consider how terms are scored inside the vector. A simple weights such as 1 and 0 (1 indicating the appearance of the term in the document and 0 otherwise) might be used effectively in some case, however it ignores the length of documents. A simple tabulation or *term frequency* weighting, on the other hand, might shows how important frequently occurring words are to a document but it does not show the term significance with the rest of the documents and potentially skew the representation towards longer documents. Thus, a more sophisticated weighting scheme and normalization technique are generally applied to this representation.

One weighting scheme that is used in many Information retrieval problems, SMART Retrieval system[42] and Rocchio categorization framework [41] as examples, is the **TF-IDF** scheme. In this weighting scheme, we combine the scoring from the *term fre-*

quency(TF) and *document frequency*, more specifically the inverse of the latter (IDF). The former, as mentioned earlier, scores or weighs the importance of a term to a document according to the number of appearances. The latter, on the other hand, weighs the importance of a term to the whole collection in conveying a specific concept. Given the same analogy of giving lower score to stop words, terms with high document frequency might be considered undesirable. Therefore, in combining the two scores we inverse the document frequency to maintain linearity in our weighting scheme

Through one of the TF-IDF weight scheme (our implementation), we have the weight of a term i for document j expressed as:

$$W_{i,j} = \begin{cases} (1 + \log tf_{i,j}) * \log \frac{N}{df_i} & , \text{if } tf_{i,j} \neq 0, \\ 0 & , \text{if } tf_{i,j} = 0 \end{cases} \tag{2.2}$$

Where $tf_{i,j}$ represents the frequency of term i in document j, N is the number of documents in the dataset, and $\frac{N}{df_i}$ represents the inverse document frequency.

After weights are computed, we still need to consider the issue of document's length. Based on the current formulation of the weighting scheme, there is no difference in weighing a term that appears in a very large document versus one that appears in short documents (term frequency being equal). The ability to discriminate the two cases is important because given the non uniform document length distribution of a text collection such as the Web, longer documents would generally produce inaccurate by higher weights (bias) for terms that are not necessarily central to the documents themselves.

In the case of tackling this problem, document length normalization is required. For normalization, we will be looking at **Cosine normalization** where we will divide the weight of term with the square-root of the sum of the weights squared. That is:

$$normalized\_W_{i,j} = \frac{W_{i,j}}{\sqrt{(W_{1,j})^2 + (W_{2,j})^2 + ... + (Wm, j)^2}} \tag{2.3}$$

As we can see from that equation, our normalization would ensure that documents of uneven lengths will have similar representation because the cosine normalization would make sure each document would have a vector representation that sums to 1.

2.2.3 Classification

Once document representation or features have been built, the next step in a text categorization system is to classify them. In this inductive learning process, we have to

design and deploy a classification algorithm that learns to build a model or hypothesis from a set of training instances and is able to label a set of unseen instances (by proxy of the resulting hypothesis).

In the context of text categorization, the choice of classification algorithm and approaches may depend on the type of features and complexity that one wished to observe. For instance, Support Vector Machines are generally thought to be appropriate because of its ability to take many features and not be prone to overfitting[21].

In this section, we will look at two classification algorithms that correspond to different families of inductive learning.

Naïve Bayes

In discussing probabilistic approaches[35] to text categorization, we present the Naïve Bayes classifier. The goal of a probabilistic classifier is to find the likeliest hypothesis given a set of training data D. Through Bayes theorem the probability of observing a hypothesis is defined as:

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)} \tag{2.4}$$

Through the *maximum a priory* (MAP) and *maximum likelihood* (ML) assumption the likeliest hypothesis can be determined.

in MAP assumption, the most likely hypothesis h_{MAP} is found by finding the most probable hypothesis given the training data. More formally, the MAP hypothesis is defined as:

$$\begin{aligned} h_{MAP} &= \underset{h \in H}{Argmax}\, P(h|D) \\ &= \underset{h \in H}{Argmax} \frac{P(D|h)\,P(h)}{P(D)} \\ &= \underset{h \in H}{Argmax} P(D|h)\,P(h) \end{aligned} \tag{2.5}$$

The denominator $P(D)$ in the simplified form is generally omitted because it is of equal value across all possible h (it does not change the outcome).

Under the ML assumption, the most likely hypothesis is found by finding the most probable hypothesis under the assumption that the probability of each hypothesis is equal. Therefore, from the MAP assumption, we can see that ML derived its maximum hypothesis by only considering the probability of of training data given a hypothesis, i.e. $P(D|h)$. More formally, the ML hypothesis is defined as:

$$h_{ML} = \underset{h \in H}{Argmax}\, P(D|h) \tag{2.6}$$

As an observation, we can see the MAP hypothesis being preferable in many real world observations, because taking into account the prior probability of each hypothesis allows for more realistic prediction on the right hypothesis.

For Naïve Bayes classification, the labellings of unseen documents are similarly defined through the MAP assumption of the probabilistic framework. More specifically, finding the likeliest category or label for a given document that can be expressed as :

$$\begin{aligned} C_{MAP} &= \underset{c \in C}{Argmax}\, P(c|d) \\ &= \underset{c \in C}{Argmax} \frac{P(d|c)\, P(c)}{P(d)} \\ &= \underset{c \in C}{Argmax} P(d|c)\, P(c) \end{aligned} \tag{2.7}$$

As mentioned in the feature section, the representation for each document d, is translated into the observance of features (the selected Bag-of-Words and Bag-of-URLs representations). Therefore when we say "the probability of observing a document $P(d)$" we are actually saying "the probability of observing these features, i.e. $P(f_1,\ f_2,\ ....,\ f_k)$". The new version of the Naïve Bayes classifier is:

$$\begin{aligned} C_{MAP} &= \underset{c \in C}{Argmax}\, P(c|f_1,\ f_2,\ ....,\ f_k) \\ &= \underset{c \in C}{Argmax} \frac{P(f_1,\ f_2,\ ....,\ f_k|c)\, P(c)}{P(f_1,\ f_2,\ ....,\ f_k)} \\ &= \underset{c \in C}{Argmax} P(f_1,\ f_2,\ ....,\ f_k|c)\, P(c) \end{aligned} \tag{2.8}$$

In finalizing the hypothesis made by a Naïve Bayes classifier, we need to assume independence on document features. True Bayesian formulation requires the observation of features as a set, $P(f_1,\ f_2,\ ....,\ f_k|c)$, however the actual chance of learning a concept by observing these features on its entirety requires a very large training data. As a naïve (the origin of the name) assumption, we can assume independence among features by simplifying the equation to $\prod_l P(f_l|c)$. The finalized version of the Naïve Bayes classifier would then be defined as:

$$C_{MAP} = \underset{c \in C}{Argmax}\, P(c) \prod_l P(f_l|c) \tag{2.9}$$

Despite the fact that context dependency among words are ignored, the use of Naïve Bayes in text categorization has been quite prominent because of it is effective[30] and extendable to different learning models such as multi-labeledness[29] and hierarchies[31].

Support Vector Machines

Support Vector Machine (SVM)[52] is a type of classifier that comes from the maximum margin and linear approach. A linear classifier is a two-class or binary classifier whose goal is to construct a hyperplane, $(w \times x) + b = 0$, in an N dimensional space that separates two classes. Using the hyperplane and positive and negative sign as representation of the two classes, a linear classifier can then be formalized as:

$$c(x) = sign((w \times x) + b) \tag{2.10}$$

Based on the equation, the goal of a linear classifier is to have a set of weights w , and a constant b that will ensure every x that is labeled as positive to have $c(x)$ that is also positive, and for every x that is negative to have a $c(x)$ that is negative. Mathematically this is equivalent to saying the product of actual label of x (1 or -1) and the function $c(x)$ is always greater than zero, i.e. $l(x).sign((w \times x) + b) > 0$ for all training instance x. When this condition is met we call this problem a linearly separable problem.

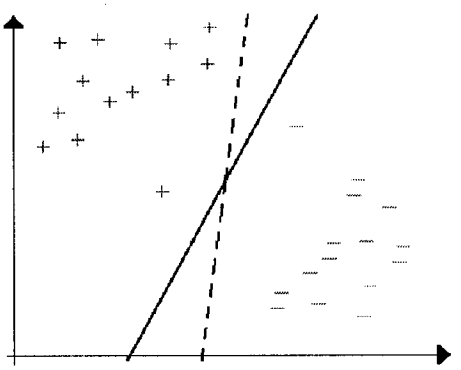


Figure 2.8: An example of two hyperplanes that linearly separate two classes. The solid hyperplane being the better generalization of the class boundary.

The maximum margin linear classifier is a special case of a linear classifier because the hyperplane produced a maximal margin or space between the two classes. As you can see in Figure 2.8 there are many hyperplanes that can optimally reduce the empirical error of the presented linearly separable class however a unique hyperplane such as the one in the solid line can produce the biggest margin or better generalization then other hyperplanes.

Without going into Mathematical details, the optimal (maximum margin) value of w can be expressed as the weighted sum of all the training instances $\sum_i \alpha_i \, . \, x_i$ and be optimized through Quadratic programming. After this computation, the linear classifier shown in Equation 2.10 can also be expressed as:

$$c(x) = sign((\sum_i \alpha_i(x_i.x)) + b) \tag{2.11}$$

For problems that are not linearly separable, SVM classifiers offers two solutions. In the first solution, we can change the kernel to a more appropriate function. The *kernel trick* solution[4] involves adding to the definition of the dot product $(x_i.x)$ a kernel that is non linear, i.e. $k(x_i.x)$. The purpose of this kernel is to map the features into a new high dimensional space F that hopefully will make the problem become linear. Available kernels include polynomial kernels, $k(x_i.x) = (x_i.x)^d$, RBF kernel, $k(x_i.x) = exp(-\gamma||x_i - x||^2)$, and sigmoid kernel, $k(x_i.x) = tanh(\kappa \; x_i.x + \theta)$.

For the second solution, we have the option of retaining a linear kernel through the use a *soft margin*[8]. Soft margin, in general, is the easing of the margin to allow some training instances to be observed inside it. Therefore the preliminary step in the Quadratic optimization problem for finding the maximum margin is reduced to finding the minimum of the absolute value $|w|$ (simplified to $\frac{1}{2}||w||^2$) and b that are subject to the linearly separable constraint. In the original formulation, this is equivalent to $\min_{w,b} \frac{1}{2}||w||^2$ subject to $l(x).sign((w \times x) + b) \geq 1$ for all x.

With a soft margin, the tolerance for error can be added to the aforementioned optimization problem. In general the tolerance is formulated through a constant cost, C, and slack variable ξ_i. The slack variable ξ_i is the counter for misclassification of training instance, x_i. The constant C on the other hand is the desired trade off between training error and model complexity. Thus, the new Quadratic optimization problem can be formulated as:

$$\min_{w,b} \frac{1}{2}||w||^2 + C \sum_i \xi_i \qquad \textbf{subject to: } l(x).sign((w \times x) + b) \geq 1 - \xi \quad \textbf{for all } x \textbf{ in training set.} \tag{2.12}$$

In summary, the SVM classifier has been noted as one of the better classifiers in term of complexity and performance [59]. As noted in [19, 45], SVM is characterized as a robust classifier in dealing with large dimensional features and a suitable classifier in terms of dealing with small concepts because of its emphasis on minimizing generalized error. In web page categorization, Linear kernel SVM has been extensively used as classifier for web directory categorization [15, 28].

2.2.4 Evaluation

In the final part of a text categorization framework, we need to present a conclusion on how good the system is at annotating. Creating a test set that is statistically sound and Choosing the right performance measurements would allow a fair comparison and suitable diagnostics on how different learning parameters effect the outcome of the system. The following sections will discuss both of these issues.

Cross Validation Sets

As mentioned earlier, the use of a dataset in a categorization experiment is to both train and test a learning algorithm. Based on this functionality, there are several ways of dividing a dataset into training and test sets. The Mod-Apte split [1] that was used for the Reuters Dataset, for instance, uses timestamp as a pivot to divide the training and test set. In this scheme, the idea of dividing into training and test set based on certain characteristic of the dataset, i.e. the timestamp, is appropriate because it allows users to observe the quality of a learning model over time. However, in many experiments certain characteristics such as timestamps might not necessarily be available. If that is the case, dividing a dataset into training and test sets randomly can also be considered appropriate.The most common method is to build a K-fold **cross validation** scheme.

In a K-fold cross validation scheme, a dataset is randomly divided into K disjoint sets where K is generally a low number between 2 and 10. After the division is performed, each set is used as a test for a model that is learned from the instances that are not in the corresponding set. Through this process, a classification method can be thoroughly evaluated because the randomness of building the test set and the fact that we tested on all the instances in our dataset means that we have a complete and unbiased way of testing its performance.

A variant of the K-fold cross validation, is the leave-one out cross validation or the N fold cross validation (N is the number of instances in our dataset). In this method, we have an extreme way of dividing training and test set because we have the same number of sets as instances. In other words, in evaluating a classification method through N fold cross validation we need to train or build N hypothesis. While there is no arguing on the reliability of measuring a classification performance through this method, N fold cross validation is rarely used because it consumes significant more time than a normal K-fold cross validation.

Categorical Measurements: Precision, Recall and FMeasure

		Actual	
		0	1
Predicted	0	TN	FN
	1	FP	TP

Table 2.2: A confusion matrix of a binary classification problem

The evaluation of a categorization system generally starts by observing how well the system learns individual categories. In Table 2.2, we have all the possible labeling cases that visualize how well a category is learned. The initial "TP" stands for *"True Positive"* and it corresponds to the number of documents that are related and are correctly labelled (by the system) to the category; The initial "FP" stands for *"False Positive"* and it corresponds to the number of documents that are not related but are incorrectly labelled to the category; The initial "TN" stands for *"True Negative"* and it corresponds to the number of documents that are not related and are correctly not labelled to the category; The initial "FN" stands for *"False Negative"* and it corresponds to the number of documents that are related but are incorrectly not labelled to the category. From these four figures some measurements of a classifier for a given category can now be derived.

One of the simplest measurement that one can calculate from this confusion matrix is **Accuracy**. As a a classifier's measurement, accuracy measures in percentage, the proportion of correct decisions over all the decisions that were made in the system. The formula for Accuracy is:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.13}$$

In general, the measurement produced through Accuracy is a good gauge on how well a classifier performs. However, there are occasions when the result produced through accuracy can be considered deceptive. For problems that has few members for their positive class, accuracy is not very appropriate because it allows the number of correct decisions on the negative class to overshadow the number of correct decisions on the positive class, i.e. a high TN may mask the low TP and still produce high accuracy.

More appropriate measurements for measuring the level learning of a category include: **Precision**, **Recall** and **F-Measure**.

In precision, we measures the ratio of valid documents that are in the pool of documents that are labeled to a category by the hypothesis. More formally, Precision is defined as;

$$Precision = \frac{TP}{TP + FP}. \tag{2.14}$$

To complement precision, recall is used to measure the ratio of documents that are correctly labeled to category by the hypothesis to the number of documents that are actually labeled by that category . More formally, Recall is defined as:

$$Recall = \frac{TP}{TP + FN}. \tag{2.15}$$

Based on the two measurements, we can now correctly gauge the performance of a classifier in terms of recognizing a category (by looking at Recall), and filtering unwanted documents (by looking at Precision). At first, the task of a classifier in recognizing a category and filtering a category seems to pertain to the same goal. However we can see that in majority of casses the two are actually a trade-off. As a simple and unrealistic example consider the deployment of a classifier that always labels a category to a document. In this case, the classifier would be able to achieve perfect recognition of that category, i.e. Recall, at the price of filtering unwanted documents, i.e. Precision. The opposite of this example is a very conservative classifier that labels only a small number of the documents in that category, i.e. has a low Recall value, but does so effectively, i.e. maintains a high precision.

In many classification experiments, the trade-off between precision and recall leads to the need of a unified measurement. This measurement is the F-Measure[50]. In F-Measure, we measure the weighted score of precision and recall. The formula for F-measure is:

$$F = \frac{1}{\alpha \frac{1}{Precision} + (1 - \alpha) \frac{1}{Recall}}. \tag{2.16}$$

As we can see from the formula, the parameter α determines the weight or importance of precision over recall. In general, an α that is equal to 0.5 is used to give equal importance to both precision and recall. In this case, the F-Measure can be simplified to:

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{2.17}$$

Macro Average and Micro Average

Since all the measurements mentioned above are described for one category, we then need to consider how to translate them into measurements that span multiple categories. The

simplest answer is to average the scores of each category.

One way to average these scores is by tabulating all the True positives (TPs), True negatives (TNs), False Positives (FPs) and False Negatives (FNs) and compute a new average precision, recall and F-Measure. The average scores that are obtained from this method is generally called the **Micro-Average** scores. More Formally the Micro Average scores for precision, recall and F-measure are described as:

$$\begin{aligned} Precision_{Micro} &= \frac{\sum_{c \in C} TP_c}{\sum_{c \in C} TP_c + \sum_{c \in C} FP_c}, \\ Recall_{Micro} &= \frac{\sum_{c \in C} TP_c}{\sum_{c \in C} TP_c + \sum_{c \in C} FN_c}, \text{ and} \\ F_{Micro} &= \frac{2 \times Precision_{Micro} \times Recall_{Micro}}{Precision_{Micro} + Recall_{Micro}} \end{aligned} \tag{2.18}$$

It is noted that Micro Average scores tend to behave like accuracy in cases where there are uneven distributions among different categories. Instead of observing the classifiers' performance with equal emphasis on each category in the dataset, Micro-Average scores observed the classifiers' performance in terms of each labeling decision made. Therefore the conclusion made from these scores applies to the overall problem but they do not necessarily apply on categorical level.

Macro-Average scoring ,on the other hand, puts equal emphasis on each category. In Macro Averaging, scores such as precisions, recall and F-Measures are averaged after they are calculated from each category's confusion matrix. For this reason, there are equal emphasis for each category regardless of their distributions. The mathematical definitions of Macro-Average scores for precision, recall and F-Measure are defined as:

$$\begin{aligned} Precision_{Macro} &= \frac{\sum_{c \in C} Precision_c}{|C|}, \\ Recall_{Macro} &= \frac{\sum_{c \in C} Recall_c}{|C|}, \text{ and} \\ F_{Macro} &= \frac{\sum_{c \in C} F_c}{|C|} \end{aligned} \tag{2.19}$$

Given the equal emphasis on each category, the Macro-Average scores in some cases are different than the Micro Averaged ones. For instance for a 3-class single labeled

problem that has the following distribution; class 1 has 500 instances; class 2 has 100 instances; and class 3 has 50 instances. A classifier is able to produce 480 True Positives for class 1, 30 True Positives for class 2, and 10 True Positives from class 3. Using the Micro average formula, we have an average recall of 80% (from $\frac{480+30+10}{500+100+50}$). Through Macro average scoring, however, the average recall is found to be 48.67% (from $(\frac{480}{500}+\frac{30}{100}+\frac{10}{50})/3$). From a first glance, we can see that the two scoring reflect different conclusions, i.e. 80% looks a lot better than 48.67%. The logical answer to why the two scores produce significantly different results is to interpret them differently. For the Micro-Average recall, a score of 80% means that on average for every 100 unseen instances, 80 of them will be correctly labeled to their respective category. For Macro-Average recall, a score of 48.67% means that on average only half of the instances of a category will be labeled correctly.

Chapter 3

Merging Collaborative Tagging and Automatic Text Categorization: a Hierarchical Approach

In our attempt to add automatic text categorization to a collaborative tagging system, we obtain a dataset that would characterize collective annotations on web documents. Based on the characteristics of the dataset, we compare its similarities with datasets from other web categorization studies. From the comparison, we will extend the categorization framework to include hierarchical categorization approach.

In this chapter, we will discuss the categorization dataset that we obtained from Delicious, one of the example of collaborative tagging systems mentioned in Chapter 2, and the framework for hierarchical categorization. More importantly, given the structureless aspect of tagging, we will discuss the method in which hierarchical structures are created from[1].

3.1 Obtaining Datasets from Delicious

The recent emergence of tagging among many new web services have left us with several collaborative tagging systems to consider.

Services such as Flickr (`http://www.flickr.com/`) offer tagging and the sharing of tags as a functionality. They, however, are inappropriate because they deals with image and offers ownership restriction on taggers, i.e. a *narrow folksonomy*[51].Social Book-

[1]Previous hierarchical categorization studies generally assume a human generated hierarchy

marking website such as Delicious (`http://del.icio.us`), on the other hand, are better examples collaborative tagging services because it is not restrictive and is appropriate for its content.

In previous collaborative tagging studies [6, 34], Delicious is noted for its popularity among users. This, indirectly, resulted in a more sound statistical analysis and broad range of domains to be observed in the respective studies.

3.1.1 RSS Feeds in Delicious

```
<channel rdf:about="http://del.icio.us/tag/blogs">
    <title>del.icio.us/tag/blogs</title>
    <link>http://del.icio.us/tag/blogs</link>
     <description></description>
     <items>
     <rdf:Seq>
      <rdf:li rdf:resource="http://regularthoughts.blogspot.com/" />
       <rdf:li rdf:resource="http://www.paigebraddock.com/" />
       ....
       <rdf:li rdf:resource="http://alistapart.com/articles/understandingwebdesign" />
      <rdf:li rdf:resource="http://blog.ted.com/" />

      </rdf:Seq>
      </items>
</channel>
...
<item rdf:about="http://alistapart.com/articles/understandingwebdesign">
<title>Yahoo!</title>
<link>http://www.yahoo.com</link>
<dc:creator>user1</dc:creator>
<dc:date>2007-10-22T08:10:22Z</dc:date>
<dc:subject>searchengine yahoo mail news</dc:subject>
<taxo:topics>
  <rdf:Bag>
    <rdf:li resource="http://del.icio.us/tag/blogs" />
    <rdf:li resource="http://del.icio.us/tag/design" />
    <rdf:li resource="http://del.icio.us/tag/web" />
  </rdf:Bag>
</taxo:topics>
</item>
```

Figure 3.1: A general RSS feed from Delicious showing channel level metadata (the top half) as well as the item level metadata (the bottom half)

In Delicious, retrieval and extraction of tagging information can be performed by accessing many of its RSS (Really Simple Syndication) feeds. An RSS feed is a frequently

updated structured document (similar to XML) that contains light-weight information or a syndication for a topic. Unlike HTML documents, RSS feeds are standardized and built to contain structured information (to be parsed). For this task, we deal with a standardized client and parser that aggregate information from one RSS feed into a database or storage. In many new web services, such as Delicious, RSS feeds are the preferred option for non-browsing type of data collection because they reduce the overheads of webpages and are standardized across different clients.

As an example of an RSS feed, we can see figure 3.1. From the **channel** metadata (the top part of the RSS feed), an RSS client can parse a list of websites from the list of resources to form of the set of documents $D = \{d_1, ..., d_n\}$. The **item** metadata (the bottom part of the RSS feed), on the other hand, contains individual tagging details that are akin to the labellings $L = l_{11}, l_{12}, ..., l_{mn}$ and by proxy categories $C = \{c_1, ..., c_m\}$ needed to complete a categorization dataset (see Section 2.2.1).

Feeds	Parameters	Description
http://del.icio.us/rss/recent	none	most recent tagging activities
http://del.icio.us/rss/recent?min=N	$N \in \{1, 5, 10, 50, 100\}$	most recent tagging activities on documents that have been tagged by at least N users
http:// del.icio.us/rss/popular	none	most recent tagging activities on "popular" documents
http://del.icio.us/rss/X	username X	most recent tagging activities by user X
http://del.icio.us/rss/tag/Y	tag Y	most recent tagging activities involving tag Y
http://del.icio.us/rss/tag/Y+Z	tags Y and Z	most recent tagging activities involving tags Y and Z
http://del.icio.us/rss/X/Y	username X and a tag Y	most recent tagging activities involving tag Y by user X
http:// del.icio.us/rss/X/Y+Z	username X, tags Y and Z	most recent tagging activities involving tags Y and Z by user X
http://del.icio.us/rss/url?url=U	address U	the complete tagging activities for the website U

Table 3.1: A shortlist of the different RSS feeds provided by Delicious

Given specific range of queries, Delicious has provided different types of RSS feeds. For queries about documents that had been tagged with specific keyword, accessing RSS feeds in the form `http://del.icio.us/rss/tag/tagName` would return the list of documents that have been recently tagged by the tag *"tagName"* (replace this with the query tag). For queries about documents that had been tagged by specific user, `http://del.icio.us/rss/userName` (replace the entry userName with the proper user name)

would return the complete tagging history of a user. For queries about specific documents characteristics, user can access the feed `http://del.icio.us/rss/popular` for popular documents or `http://del.icio.us/rss/recent?min=n`. The latter works under the assumption that the documents have specific minimum number of taggers, i.e. n. Lastly, for each document, the complete tagging or annotation history can also be accessed from RSS feeds. Retrieving `http://del.icio.us/rss/url?url=http://example.com` would return the tag history for the document `http://example.com`.

In Summary, Table 3.1 lists the different RSS feeds that are provided in Delicious.

3.1.2 Finalizing Document Annotations through Removal of Weaker and Noisy Tags

No.	Tag	Frequency	No.	Tag	Frequency	No.	Tag	Frequency
1	"recipes"	58	10	"sides"	3	19	"butter"	1
2	"potatoes"	35	11	"home"	2	20	"food_drink"	1
3	"recipe"	29	12	"side-dish"	2	21	"try_me"	1
4	"food"	25	13	"simplyrecipes"	2	22	"roasted"	1
5	"cooking"	18	14	"vegetable"	1	23	"savory"	1
6	"sherry"	17	15	06/10/07	1	24	"starch"	1
7	"vegetables"	17	16	"appetizers"	1	25	"potatos"	1
8	"potato"	10	17	"baked"	1	26	"totry"	1
9	"sidedish"	5	18	"blogs"	1	27	"tocook"	1

Table 3.2: A summary of tags and frequency for a recipe page for cooking potato with sherry

Based on the process described above, a collection of documents and their labels can easily be obtained. At their original state, however, documents can actually be annotated with noisy and personalized tags.

In Table 3.2, we presented the list of tags that are attached to a webpage that contains a recipe for cooking potatoes with cherry (`www.elise.com/recipes/archives/004069sherry_potatoes.php`). For most retrieval cases, the top ranked tags such as "recipe", "potato", "sherry", etc. are sufficient in providing high level description and organizational labels of the document. Low ranked tags such as "potatos" and "try_me" are considered weak and noisy because they symbolize mispellings and personalized goals respectively . In the case of a categorization system, allocating the system's resource to categorize and index weaker tags that have low retrieval utility can be problematic because they create information bottleneck and take the emphasis away from more popular

tags.

In order to avoid this problem, a minimum for the number of taggers of a tag to a document are generally imposed on the annotations that have been collected. For documents that have been annotated by at least 100 users, for instance, a minimum of 5 (small fraction of total number of taggers) annotations for each tags would filter many of the weaker tags. Going back to the list of tags in Table 3.2, the number of tags that are attached to the document is reduced from 27 to a total of 9 tags by imposing the aforementioned rule. Based on the 9 tags that are kept, we can still see that sufficient amount of high level descriptions of the document are retained for retrieval and categorization purposes.

3.2 Characteristics of Delicious as Text Categorization Dataset

Based on the process described above, we compiled a categorization dataset that will be used in automatic categorization experiment. Through a 3-day period in January 2007, a list of websites were collected from RSS feed is `http://del.icio.us/rss/recent?min=100`). The number 100 is chosen for the minimum number of tagging activities because we want each documents to be well tagged and reputable enough, i.e. non spam. The initial number of unique documents aggregated from this method is 8765. However, we have to reduce the number of valid documents to 7,583. The removal of these documents can be attributed to non textual documents such as *movies* and flash files which are all not handled within our text categorization framework.

For all 7,583 documents, we then accessed their tagging history through their respective RSS feeds and compiled a set of labeling aggregates. The resulting tags and annotations, which are aggregated from 4,141,907 tagging activities, includes 187,026 unique tags and an average of 160 $\frac{\text{tags}}{\text{documents}}$. As explained earlier, we also remove weaker tags from each document by setting a minimum of 5 annotations for each tag that are found in a document. The dataset, after this procedure, consists of 7,583 documents, 11,352 tags and 28.5 $\frac{\text{tags}}{\text{documents}}$.

We finalize the dataset by removing rare tags by setting a minimum of 50 documents for each of the 11,352 tags. It is noted that we added the minimum number of documents restriction because we consider the classification of rare tags or category to be a unique problem that requires its own discussion. Future works that utilize semi supervised

classification such as co-training[3] would be an example of how to solve this problem. The finalized dataset would therefore consist of 7,583 documents, 567 tags (or categories) and an average of approximately $22.8\frac{\text{tags}}{\text{documents}}$. In Appendix A.1 we listed all the tags and the number of documents that are related to them.

In the following subsections, we will discuss the characteristics that were found in this dataset and compare them to previous web categorization experiments.

3.2.1 The Presence of Sensemaking

In previous paragraph, we discussed how documents and annotations were compiled. In this section, we would like to analyze the implication of having collaborative annotations have on the relevancy of the dataset.

In past web categorization studies, particularly web genre studies[33, 24], attempts in trying to characterize and enumerate digital categories or genre tend to be centralized and rigid, i.e. having clear cut boundaries. In [24], Kennedy and Shepherd present an argument that the highest level descriptions of web documents would consist of three genres: personalized, corporate and organizational homepages. Similarly in [33], Zu Eissen and Stein survey a small number of university students and produces 8 genres or categories that they believe characterize many web documents.

Despite the fact that each of these studies have their own reason in limiting the scope of categories that they consider, the absence of disagreement or variance in the categories assigned to a document, even on high level view of web categorization, would suggests that these datasets are overfitted to predefined categorical boundaries and lack the generalization to appeal to a larger set of users. As mentioned in Chapter 1, similar argument was also used by early adopters of collaborative tagging to question the relevancy of hierarchical web directories.

The argument to a more decentralized point of view can be illustrated in Crowston and Kwasnick's[12, 10, 11] attempts in trying to characterize digital or web genre. In their earlier paper[10], they managed to assign and to identify 118 web genres across different web pages through a small group of labellers. However, in their most recent attempt at a similar experiment[12], they found that labelling become problematic when it is brought to an unmoderated discussion panel. They observed that in most cases there are lack of agreement on what categories or descriptors are best suited to a document. As a compromise, their recent framework [11] suggests a faceted view of categories that would allow multiple views or facets to be annotated to an object or a document.

From an Organizational Behavior point-of view, the preference towards a decentralized categorization mirrors the motivation in explaining the concept of **sensemaking**. As stated in Weick et al. [53], sensemaking is a retroactive process of building a consensus in large organizational scale. In the case of web ontologies, collaborative tagging has been seen as the reflection of the sensemaking process [17] for a couple of reasons. First, the emergence of popular tags from aggregations of multiple user annotations are analogous to building a general consensus in a sensemaking process. Second, the unrestrictiveness of vocabulary in tagging and the fact that aggregation allows for less rigid boundaries and transitions from popular to less popular tags or consensus to form. Consequently, we believe that our dataset reflects the sensemaking process that is relevant to a large sample of the population, because we only consider documents that have been unrestrictively annotated by many users (greater than 100).

3.2.2 Robustness of Domain and Usage

One aspect of a dataset that reflects on its applicability is the robustness of its categories, or in this case tag space. Representing domains that are limited means the inability for the system to categorize other domains. For instance, a dataset used for classifying spam weblog can not be used as a dataset to classify newswire category.

Despite the big reduction of tag space (from 187,026 to 567 tags or categories), we believe the set of tags at our disposal are also diverse enough to be applicable to general web browsing. In previous studies [17, 55], robustness of tags has been documented in terms of their usages. Similarly for this dataset, we will elaborate our categories (since we formalize our finalized tags as categories) through the context of their usages. Below are some of different types of usage of our categories and examples from our collection of categories (see Appendix A.1).

- **Categories that describe contents and topics.** Like keywords, these are categories that are used to describe the content of a document. When more generalized descriptions are needed , i.e. a superclass, some of these categories formed more topical descriptions such as the ones found in top level hierarchy of web directory. For example: ***Ruby, Ajax , Python, etc.*** are a group of categories that describes documents that discuss these specific programming languages while ***programming, webdev*** are another group of categories that provides a more generalized description on the same document.

- **Categories that describe Forms.** By definition, Form refers to presentation of a document. In the context of non digital communications, forms translate to the description of medium, materials and writing style. For example: ***magazines***,***newspapers***, ***books*** and ***radio*** are all categories that describe certain forms. In the context of digital documents, Forms can be used to describe all of the above but they also include new technologies that are incorporated to publication styles of certain webpages. For example, ***blog***, ***podcast***, ***chat*** are all categories that describes new forms of communication that exist in the Internet today.

- **Categories that describe document functionalities.** Categories that describe document functionalities can sometimes be confused with categories that describe forms, because, in many cases, different functionality may require different writing styles. However, we try to distinguish the two because the two categories may label the same document without making any ambiguous descriptions. For example, a document that is labeled with categories ***tutorials*** and ***blog*** is not considered ambiguous because the former describe a functionality aspect while the latter describes a form aspect of the document. ***Howto***, ***recipe***, and ***tips*** are some examples of these categories that we found in our collection

- **Categories that describe regionality** Categories that describe regionality and language of the documents. For example; the tags ***German*** and ***uk*** may indicate the countries of origin of the websites. Similarly the tags ***programacion*** and ***tecnologia*** may indicate the regionality of the tagger without revealing the origin of the document. For example a Spanish tagger may tag an English document with Spanish tag as a form of personal categorization.

- **Categories that describe opinions.** Categories that describe opinions are categories that described a user's subjective feeling or sentiment about a document. Given the high degree of freedom, more opinions are passed through as high frequency categories in collaborative tagging systems than web directories. For example, we have ***funny***, ***humor*** as some of the common subjective categories that are found in web directories, as well as ***weird***, ***inspiration***, ***useful*** as some of the less popular subjective categories.

- **Categories that describe Personal Organizational tasks.** One of the purpose of collaborative tagging is to help build a personalized document organization. As

a result, organization of documents according to a personal goal can sometimes pass as categories. In our set of categories there are a small group of categories that can be used in this manner. Some categories like ***toread*** and ***todo*** are best described in this manner because they described the organization of future task involving the documents.

3.2.3 Power Law Distribution

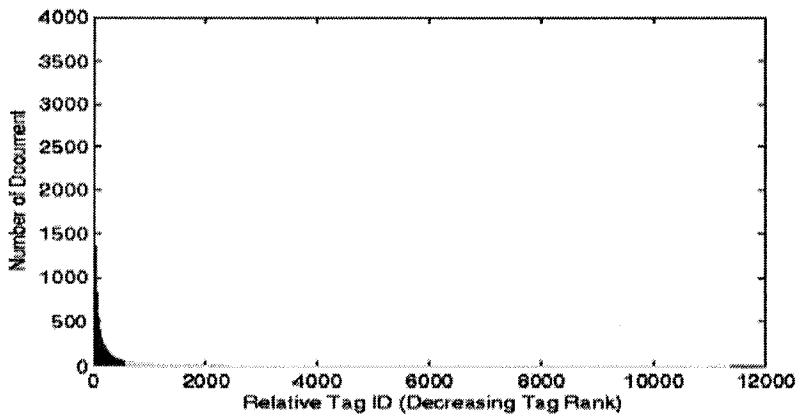
Another observation in our dataset that is worth mentioning, is how the tags are distributed among the documents. In many applications, knowing the right tag distribution for any categorization problem can make a difference in the behavior of the system. In many social sciences, Power-law distributions had been perceived as an important naturally occurring patterns[36]. For instance, Zipf's law [60] observed the pattern of words usage through Power-Law distribution to conclude that words in the English language are not equally used among a large population. This conclusion shows an insight toward human behavior and the study of statistical Natural Language Processing because it demonstrates a bias or tendency to use of certain words (popular words) despite the potential of using other similar in meaning, but different words.

In its mathematical form, the Power Law distribution is defined as:

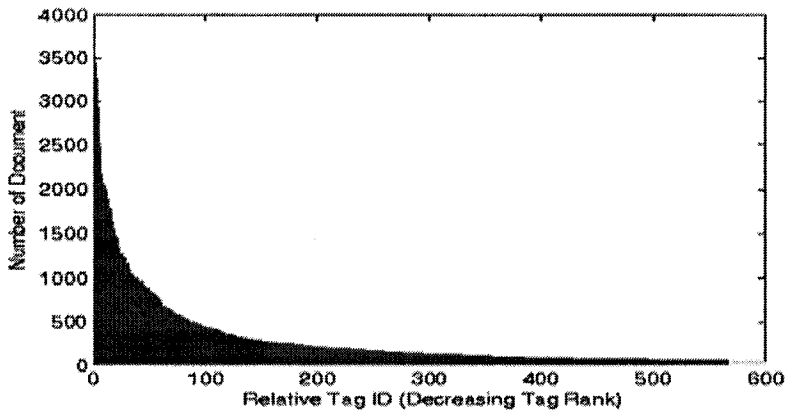
$$p(x) = C\,x^{-\alpha} \tag{3.1}$$

In that equation, we have $p(x)$ as the probability of observing an event that had the rank x or in our case the normalized number of documents that are labeled with the tag that had the rank x. The variables c and α on the other hand are constant parameters that is determined by the starting point and steepness of the distribution curve. For our dataset, we can informally use the rank-frequency graphs shown in figure 3.2 to determine a Power Law distribution. Based on the somewhat consistent exponential decrease in the rank frequency graph or the straightness of the log-log plot, we believe that a Power Law distribution was observed within the 11,352 tags, and indirectly for the 567 tags as well.

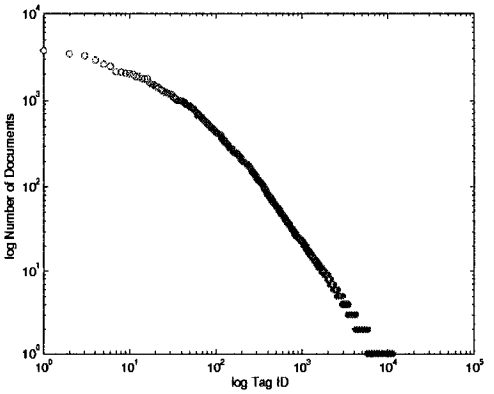
Given the observance of Power Law Distribution, we justify the loss of information from excluding the classification of rare tags. As mentioned earlier, we prioritized our categorization to tags with more than 50 documents, i.e. leaving more than 10,000 other categories unclassifiable. On the surface, this decision seems to undermine the coverage of the resulting categorization system. However given the relatively small loss



(a) Rank Frequency Graph For all Delicious Tags



(b) A closer look at the Rank-Frequency Graph for the top 600 tags



(c) Log-log plot of the Rank-Frequency Graph along (all tags)

Figure 3.2: Rank-frequency histogram of all Delicious tags (after removing potential noises) and the number of documents that are labeled to them

of labeling information and the equivalent alternative of a dataset, we considered our prioritization as justifiable. In the rank-frequency plot of our Power Law distribution, the loss of information can be computed by looking at Figure 3.2(a) and comparing the area under the curve after the Tag ID 567 with the area under the whole curve. In numerical measurement we can try subtracting the average number of tags per document before this prioritization, $28.5\frac{\text{tags}}{\text{documents}}$, with the average after , $22.8\frac{\text{tags}}{\text{documents}}$, and divide it with the former. The loss of labeling information based on this computation equals to $\frac{5.7 \text{ tags}}{28.5 \text{ documents}}$ or 20% of document tag annotations. This in terms of classification complexity is a nice trade-off because given a set of target concepts, the categorization time and performance are parametrized to the size of the set of target concepts.

Therefore, had we observed a uniform or normal distribution for the tags (both do not have the constant exponential decrease in their rank frequency plot), any text categorization approach that ignores a significant number of concepts would be deemed unusable. Power law Distribution helps unsupervised categorization or classification approach to prioritize target concepts.

3.2.4 Comparison with Other Web Categorization Works

Given the three characteristics we had described, we can now compare the problem of categorizing collaboratively tagged data with other web categorization experiments.

	Number of instances	Number of Categories	Categorization Approach	Results
Dumais and Chen[15]	60,100 (training: 50,078 and testing: 10,024)	163categories (13 top level and 150 second level categories)	Hierarchical	Micro Average F-Measure 57.2%(top level), 49.7%(second level)
Liu et al.[28]	767,981(training: 492,617 and testing: 275,364)	132,199 (14 top level, 307 second level, etc.)	Hierarchical	F-Measure:23.7% (Micro Average), 11.6% (Macro Average)
Kennedy and Shepherd[24]	324 (10-fold cross validation)	3 (personal, cooperate, and organizational homepages)	flat	F-Measure 71% (personal), 67%(cooperate) and 55% (organizational)
Zu Eissen and Stein[33]	800 (10-fold cross validation)	8 (link collection, help, shop, portrayal non-priv., portrayal priv., articles, download, discussion)	flat	average classification performance 70%

Table 3.3: A comparison of four web categorization datasets and the result of their categorization

In the previously mentioned genre categorization studies [33, 24], we have attempts on classifying web documents on a small number of target concepts. In [24], Kennedy and Shepherd tried to categorize 3 genres of homepage using Neural Network classifiers. With 321 instances in their corpus, they manage to reach F-Measure scores of 71%, 67% and 55% for personal, cooperate and organizational homepage respectively. In [33], Zu Eissen and Stein attempt to categorize 800 instances from their collection of 8 genres categories with linear discriminant analysis. The resulting classification produces an average of 70% accuracy. It should be noted, we assume similar results for F-Measure because the documents are universally distributed among the 8 categories and the classifier's performance for each category does not fluctuate enough to change both Macro and Micro Average F-Measures.

Figure 3.3: An example of top level categories of an existing web directory(Open Directory Project)

From the two studies, we can see optimistic results that yield a significantly better than 50% F-Measure for each of the categories that are classified. However, we believe such results are misleading because despite an unbiased evaluation procedures, the complexities of their datasets are not reflective of a real world categorization dataset.

Therefore for more appropriate comparison, we refer to two larger web categorization studies that use web directories as their corpus [15, 28].

In [15], Dumais and Chen proposes a text categorization approach that are based on the LookSmart web directory. Their dataset consists of 60,100 web documents ,a significantly larger set than either genre categorization and the data we compiled for this thesis, and a total of 163 categories (13 top level categories and 150 second level categories). Similarly in [28], Liu et al. look at hierarchical categorization on web directories using a larger corpus, the Yahoo! web directory. The Yahoo! dataset is a collection 767,981 web documents that are labelled to 132,199 set of categories. The top level categories the Yahoo! corpus consist of 14 categories, while the second level categories consist of 307 categories. For both studies[15, 28], the attempt to categorize their respective datasets were done with flat and hierarchical approaches. Both also conclude that the hierarchical approach offers better performance and efficiency.

In terms of complexity, both LookSmart and Yahoo! datasets present similar challenges to the collaboratively tagged data that we have gathered. While the number of instances in our dataset is significantly lower (we only have 7604 web documents), the number of categories is similar to the number of categories that both corpora have for the first two level of the hierarchies. we also see the diverse range of the tags such as ***Health***, ***Politics***,***Economy***, ***Entertainment***, ***Education***, ***Shopping***, and ***Art***(see Appendix A.1) as a similarity with the diversity and robustness represented in web directories (see Figure 3.3). Furthermore , for our dataset, we have a comparably similar Power Law distribution (see Section3.2.3) as the one observed by Liu et al.[28]. Dumais and Chen[15] did not provide a complete report on the distribution of their categories. However, their first level categorical distribution also showed a non-uniform distribution that stated that some categories are more densely populated than others.

Since the collaboratively tagged data we have gathered possesses similar characteristics with these web directories, we believe that the categorization of our data would also benefit from the hierarchical categorization approach. In the next section, we will elaborate the difference between hierarchical and flat categorization, as well as the derivation of a hierarchical structure from collaboratively tagged data.

3.3 Hierarchical Text Categorization

Hierarchical text categorization is defined as any learning process that removes the assumption that categories or tags are independent and tries to incorporate the interre-

lationship among different categories with a multi-level structure, i.e. a hierarchy. In a normal or "flat" text categorization approach, it is generally assumed that categories works as independent and a non-overlapping descriptors. In theory, this means classification of individual category is performed with the knowledge that all other categories are considered as unrelated. In practice, however, categories tend to overlap and not independent of each other.

Hierarchical categories such as the one mentioned in [15, 28] are examples how categorical relationships overlaps in a top down generalization that allows documents to be labeled to leaf and parent categories (also grandparents, etc.). Dumais and Chen[15], for example , reported that on average documents belong to 1.2 second level categories and 1.07 first level categories, for a total of 2.27 categories. Liu et al.[28] also reports an average number of categories that belongs to a document to 2.23 categories although it is not clear whether parent categories in the hierarchy are considered. Similarly for our collaboratively tagged dataset, the degree of multi-labeledness is quite prominent at $22.8\frac{\text{tags}}{\text{documents}}$. It should be noted that the collaboratively tagged data do not have a prearranged hierarchical structure attached to them (More on this in Section3.4).

From these observations, it is clear that any categorization approach should consider the possibility of documents attached to multiple categories or tags. In this section, we will first discuss how single model and multiple models classifications are used to classify a multi-label dataset. We will also present the hierarchical versions of both models and explain why the hierarchical versions are considered an improvement.

3.3.1 Single Model and Multiple Models Categorizations

Single model classification is an approach to a multi-class and/or multi-labeled problem that builds a single model or hypothesis to classify documents to any of the of the possible categories. A multi-class problem, is a classification problem in which multiple target classes are being considered as candidate for classification. Subtypes of of multi-class problem are single-label and multi-label problem. In single-label, we restrict the classification or labelling of an instance to only one of the target class (the likeliest). In multi-label classification, we do not have restriction on the number of target class that can be assigned or labeled to a document.

In this approach, a classifier learns a single hypothesis or learned model automatically that handles all possible documents and categories as inputs by assigning scores to these permutations. However, we need to remember the formulation of text categorization, f' :

$D \times C \rightarrow \{0,1\}$, the output for each permutation of document and category permutation is a binary decision, a true or false value on the membership of a document to a category. For single model classification to translate into a comparable learned function $f' : D \times C \rightarrow \{0,1\}$, we need to create a threshold value on the output of the hypothesis. In [56], Yang summarize some of the strategies for setting the right threshold cuts with a validation set.

In contrast to single model classification, **multiple models classification** approaches the problem of text categorization by building multiple classifiers. Through this scheme, each category is assigned its own binary classifier. The binary classifier is a classifier that learns to differentiate documents that belong a category from the ones that do not. Therefore, this approach can sometimes be referred to as the **One-Vs-All** method.

More formally, each model in this case has the task of labeling an unseen document to their corresponding category, i.e. a sub function $f'_{c_i} : D \rightarrow \{0,1\}$. The finalized hypothesis function can then be seen as unified collection of binary hypotheses such as:[2]

$$f'(d,c) = \begin{cases} f'_{c_1} : D \rightarrow \{0,1\} & \text{,if } c = c_1 \text{ ,} \\ f'_{c_2} : D \rightarrow \{0,1\} & \text{,if } c = c_2 \text{ ,} \\ .... & \\ f'_{c_m} : D \rightarrow \{0,1\} & \text{,if } c = c_m \text{ ,} \end{cases} \tag{3.2}$$

where m is the number of categories considered.

It should be noted from the description of both models, there are advantages and disadvantages in choosing one approach over the other. For single model classification, the advantage of having a training and testing time that are unaffected by the number of categories seems appealing at first. However it is generally noted that extra steps in ensuring a multi-label annotations, i.e. threshold cuts and larger feature sets, tend to make this approach as time consuming as the multiple model classifications.

For multiple model classification, the disadvantage is obviously related to the time and resources needed to built a classifier for each single category. The advantage of this model is the luxury of having a dynamic classification approach that allows category to be added and removed without affecting other categories' classification.

In the case of web categorization , we know that large number of categories would likely be a requirement. However, the multiple model classification is preferred because

[2]The change in notation of f' from "$f' : D \times C$" to "$f(d,c)$" does not reflect the interpretation of the hypothesis function, i.e. a function that takes all possible documents and categories and assigned a label of 1 or 0 to the pairing

as noted in [45, 46], web categorization, is a dynamic and technology driven problem. This means that new concepts or categories are created continuously and the learning part of the classification process is considered an ongoing activity.

3.3.2 Applying Hierarchical Text Categorization to Web Categorization

Having solved the multi-labeled aspect of text categorization, we can now extend the approach to include the hierarchical structures of the categories or tags. In both single and multiple model classifications, decisions regarding individual categories still fall under the flat categorization approach. In multiple model it is easy to see this fact because we have individual classifiers for each category.

For single model classification, the decisions on categorical annotation, i.e. the threshold cuts, are generally made without acknowledging inter-categorical relationship. A threshold cut that is based on the average number of categories per document (also known as R-Cut[56]), for instance, does not differentiate on related or unrelated categories ranked to a document. Similarly a threshold that is determined by searching the optimal score on a validation set (also known as S-Cut[56]) works by tuning the threshold of each category individually.

In one of the first hierarchical categorization study[27], the knowledge of hierarchical structure is generally deployed as a divide and conquer strategy that allows classification to be a sequential process. In that study, Koller and Sahami[27] apply the top down sequential categorization to show that by making certain target concept as a sub-class of a bigger concept, the efficiency and performance of learning such concept can actually be improved upon.

Consider as an example the categorization of "Sports" and "Basketball" categories where documents that are categorized in the "Basketball" categories,are also categorized in the "Sports" category, i.e."Basketball" is a subset or children of "Sports". In flat categorization, learning to annotate "Basketball" category means learning to differentiate basketball related documents with documents that are assigned to other categories. Other categories would include categories that are as distant as "Finance" and as close as "Baseball". In hierarchical categories, learning to annotate "Basketball" category means learning to differentiate basketball related documents with documents that are assigned to other "Sports" categories. This is so because in hierarchical categorization there is a previous sequence that attempt to categorize document to category of Sports.

Given the larger scope of the flat categorization, common sport terms such as "scores" and "player" would probably be selected as discriminating features for the "Basketball" category. For hierarchical classification however, common sports terms such the ones mentioned are disregarded because of their diminished discriminating value. A set of discriminating features in hierarchical categorization needs to focus more on basketball related terminology such as "slam dunk" in order to differentiate the category from other "Sports" categories. The difference in the scope of the problem would allow hierarchical categorization to have an advantage because by having smaller scope of categories, the resulting hypothesis are able to better characterize a concept than ones that have a large scope, i.e. a flat categorization. Furthermore, a smaller scope would imply smaller training sets because for hierarchical approach the training set used to classify the "Basketball" category would only concern instances that belong to the "Sports" category (parent category). Therefore with smaller and hierarchically partitioned training sets, the complexity of training and/or running a classifier would decrease and the overall efficiency would improve.

As mentioned in Section 3.2.4, the application of hierarchical categorization on a web categorization problem can be seen in [15, 28]. In [15]. Dumais and Chen perform a flat vs. hierarchical categorization comparison using multiple model classification or multiple binary classifiers. For the hierarchical approach, they constructed 13 SVM classifiers to classify each of the top level category in the LookSmart directory, and 150 SVM classifiers to classify the second level categories that are trained using a hierarchically partitioned training set. The flat approach, on the other hand, consists of 150 SVM classifiers that are trained from the full training set. During classification time of the hierarchical approach, they would first classify document using the top level classifiers and use the output from these classifiers to determine the output of the second level classifiers, i.e. a top down approach. The method in which the parent category scores is used to determine the second level classifiers include the **multiplicative scoring**, i.e. multiplying the score of the first and second level classifiers' output and the **binary scoring**method, i.e. disregard the possibility of classifying the document to any second level category if the first level classifier output a score that is below a threshold.

In contrast to Dumais and Chen's, experiment Liu et al.[28] compares flat and hierarchical categorization using single model classification. For the hierarchical approach, they started by a single SVM classifier that are responsible to classification of the 14 top level categories in the Yahoo! web directory. They then proceed by constructing 14 SVM classifiers for classifying the child categories of the 14 top level categories using

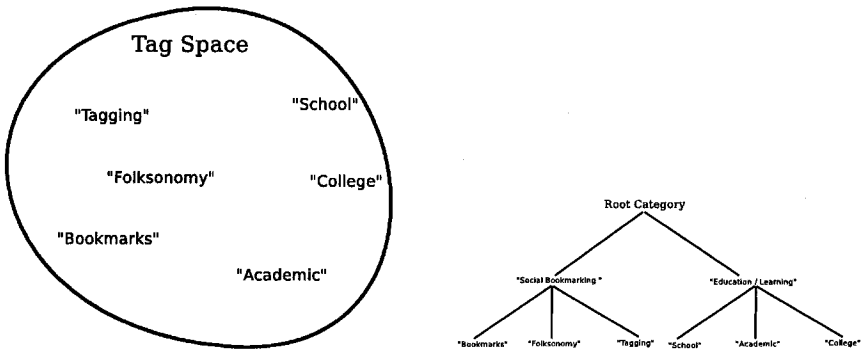
a hierarchically partitioned training set and recursively continue this process until they reach a leaf category. For the flat approach, they employ a single classifier with 4000 features and are responsible for scoring documents to the whole 132,199 categories. During the classification process of the hierarchical approach, binary scoring was used to classify unseen instances.

In either studies, it is showed that hierarchical categorization improves not only the performance but also the overall efficiency of the leaf category classifiers. In terms of performance, Dumais and Chen[15]reported a Micro Average F-Measure of 49.7% which represents an improvement of 2% over the flat categorization of the second level categories (top level categories are not included because they are the parent categories). Liu et al.[28] reported that their hierarchical categorization produced a Macro Average F-Measure of 11.6% and a Micro Average F-Measure of 23.7%. both scores represent improvements of 5% and 0.5% respectively. In terms of efficiency, Dumais and Chen[15] reported a reduction in training time from a total of 768 CPU seconds to 128 CPU seconds. For Liu et al.[28],[28], the improvement over efficiency is reported as the reduction of training time from 310.9 to 2.1 hours and the reduction of testing time from 53.6 to 0.12 hours.

3.4 Creating Hierarchical Structure from Collaboratively Tagged Data

In a collaboratively tagged data, such as ours, there is currently no hierarchical relationship that represents tags' interaction. Therefore, before we can apply hierarchical categorization to our collaborative tagged dataset, we need to derive a hierarchical structure from the dataset. For the purpose of this thesis, we define a hierarchy as the addition of one abstraction level of categories that represent the grouping of similar categories. For instance, for flat categorization, we consider the tags ***bookmarks***, ***folksonomy***, ***tagging***, ***academic***, ***college***, and ***school*** as unrelated categories. In our definition of hierarchical categories each of these tags are clustered to a parent category that serve as a more generalized abstraction that precedes the root category. In Figure 3.4(b), we visualize the cluster of the first three tags as the parent category "social bookmarking" (a self-assigned description) while the cluster of last three tags as the parent category "education/learning".

Similar to the approach mentioned in Section 2.1.3, the procedure for creating a



(a) The lack of interrelationship between tags that also illustrate the approach taken in flat classification

(b) The hierarchical structure of the same tag space that can be utilized in hierarchical classification

Figure 3.4: A comparison between a flat tag space and a hierarchical tag space

hierarchy consists of the creation of a similarity graph and the deployment of a graph partitioning algorithm to cluster tags into a two level hierarchy. In this section, we will elaborate both of these topics

3.4.1 Creating Similarity graph Through Co-occurrence Matrix

The first step in defining a similarity graph is to determine the context and representation of a tag. In the area of computational linguistics, similarity of words are generally quantified through existing lexical analysis and resources such as Wordnet or a thesaurus. Since tags or categories are essentially words, one might assume that a similarity graph for tags is generated from the aforementioned lexical resource. For our similarity graph, however, we believe the existing lexical resources might be problematic. In analyzing informal tags such as: abbreviated words, e.g. ***p2p*** (stands for "peer to peer"), and ***seo*** (stands for "search engine optimization"); proper nouns and technological jargon, e.g. ***Google***, ***Digg***, ***xhtml***; and combination words , e.g. ***weblog***, ***wikipedia***, ***webdevs***, existing lexical resources may not be appropriate because their vocabulary only index formal and well defined words.

Using a statistical approach, we look to *category/tag co-occurrence* as a more simple but effective method in quantifying similarity. In tag co-occurrence we measure similarity through usage, i.e. how often a document is labeled by the same tags. For example, if the tags ***wikipedia*** and ***reference*** are used concurrently to label many of the documents, then we can consider the two very similar or at the very least related. Through this ap-

proach, we are looking at the labeling statistics, which are not some predefined linguistic relations, as the source of measurement. Therefore, the informality and nonrestrictive nature of collaborative tagging would not be a discernible factor to this approach.

In one of the earliest tag clustering experiments, Begelman et al. [2] created a similarity graph that was obtained from co-occurrence matrices. In the following paragraphs, we have some of the steps they have used to build a co-occurrence matrix.

First, given a set of categories $C = \{c_1, ..., c_m\}$, a co-occurrence matrix, R, is an $m \times m$ matrix that lined up each category for both its rows and columns and it is filled with a symmetrical similarity values between two tags, i.e. for each cell $R_{k,l}$ in the co-occurrence matrix, a symmetrical similarity function is chosen to quantify the similarity between category c_k and c_m. Some of the choices we have for symmetrical similarity functions are:

- **Matching**; In this function, a similarity between two categories is measured as the number of documents that are labeled with both categories or more formally:

$$matching(i, j) = |D_{c_i} \cap D_{c_j}| \ ^3$$

- **Jaccard**; In this function, a similarity between two categories is measured as the number of documents that are labeled with both categories divided by the number of documents that are labeled with at least one of the tags or more formally:

$$jaccard(i, j) = \frac{|D_{c_i} \cap D_{c_j}|}{|D_{c_i} \cup D_{c_j}|}$$

- **Dice**; In this function, a similarity between two categories is measured as the number of documents that are labeled by both categories divided by the sum of the number of documents that are labeled with at least one tag (overlapped documents are counted twice) or more formally:

$$dice(i, j) = \frac{2 \times |D_{c_i} \cap D_{c_j}|}{|D_{c_i}| + |D_{c_j}|}$$

After the weights are computed, it is suggested that we remove weak tag similarity weights. As stated in [2], weak tag similarity weights, i.e. tags that co-occur in less than *30* documents, tend to act as noise during the clusterring or partitioning procedure.

[3] D_{c_x} is a set of documents that are labeled by category c_x

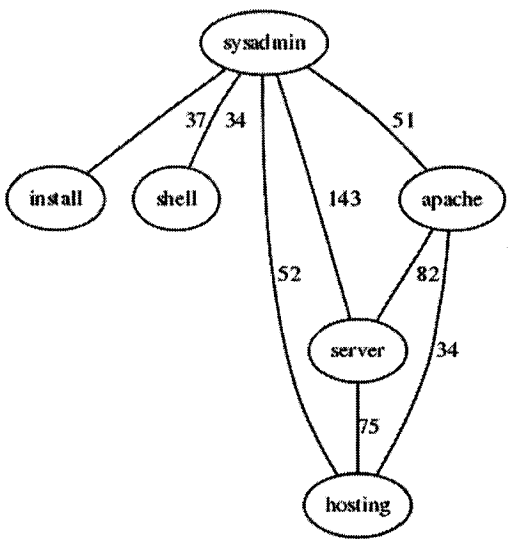
Therefore it is necessary to assume that there is no relation between these tags. Extending this rule to the co-occurrence matrix R, the cell $r(i,j)$ is now populated with the function[4] :

$$r(i,j) = \begin{cases} sim(i,j) & \text{if } |D_{c_i} \cap D_{c_j}| \geq 30 \text{ and } i \neq j \ , \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

As an example, we have in Figure 3.5(b) a similarity graph $G(V,E,W)$ that represents the co-occurrence matrix shown in Table 3.5(a). The set of vertices $V = \{v_1, ..., v_m\}$, in

	apache	hosting	install	server	shell	sysadmin
apache	0	34	0	82	0	51
hosting	34	0	0	75	0	52
install	0	0	0	0	0	37
server	82	75	0	0	0	143
shell	0	0	0	0	0	34
sysadmin	51	52	37	143	34	0

(a) a sample of a co-occurrence matrix that are filled with number of co-occurring documents



(b) A weighted similarity graph that corresponds to Table 3.5(a)

Figure 3.5: A sample of co-occurrence matrix and its similarity graph

this case, represents the set of categories $C = \{c_1, ..., c_m\}$, while the set of edges E and weights W represents the cells of the co-occurrence matrix.

[4] $sim(i,j)$ corresponds a similarity function chosen, i.e. $matching(i,j)$ or $jaccard(i,j)$, or $dice(i,j)$

3.4.2 Creating a Hierarchy through Graph Partitioning

Based on the co-occurrence matrix we mentioned in the previous section, we consider the problem of building a hierarchy or clustering of tags as a **graph partitioning** problem. Given an undirected graph $G(V, E, W)$, the main objective of a graph partitioning problem is to divide a set of vertices $V = \{v_1, ..., v_m\}$ into smaller disjointed sets of vertices $V_1, V_2, ..., V_k$ where k is the desired number of partitions or clusters. The process to build these partitions involves an optimal removal of some of the edges around the graph. These edges are called **edge-cuts**. As an illustration, we refer to Figure 3.6 as an example of how graph partitioning is applied to the tag clustering problem (The graph representation of the tags are derived from the co-occurrence matrix).

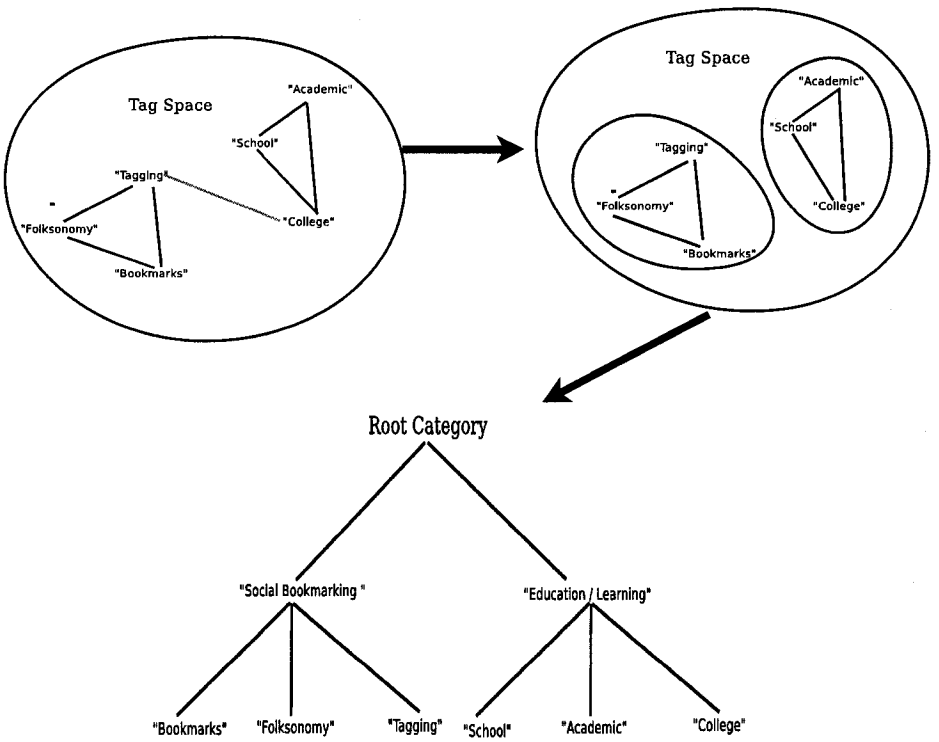


Figure 3.6: An illustration of a graph partitioning problem that produces clusters of closely related tags by removing less than optimal co-occurrences (the red edges).

For the implementation of this task, we consider the graph partitioning package called Metis[22] to partition our co-occurrence matrix because of its popularity and effective-

ness[5]. The Metis package consists of a family of partitioning algorithms that were developed by George Karypis and Vipin Kumar in University of Minnesota. In their algorithms, the optimal objectives of partitioning a graph are to minimize edge-cuts and produce a balanced set of partitions or clusters.

The algorithm itself takes a co-occurrence matrix such as the one described from the previous section and an integer, k as inputs. For the purpose of this thesis, we will use only one of the graph partitioning implementation called **pmetis**[6]. Given the aforementioned input, the pmetis implementation produces a set of k clusters, $C_1, C_2, ..., C_k$ through multilevel partitioning scheme[23].

In a multilevel partitioning scheme, k sets of partitions are obtained by recursively bisecting the graph. This means that at the first recursion the graph is divided into 2 sub-graphs, and for the subsequent recursions sub-graphs are bisected to obtain two more sub-sub-graphs. Despite the potential for being sub-optimal, this greedy approach generally produces k high quality partitions after $log_2 k$ recursions in an otherwise NP-hard problem.

In **pmetis**, the implementation of the multilevel partitioning scheme includes the following steps :

1. Coarsening phase
2. Initial partitioning phase
3. Uncoarsening or refinement phase

In Figure 3.7, we illustrate each of these phases.

During the coarsening phase, the graph is iteratively reduced into a coarser or smaller graph by combining several vertices into one new vertex throughout the graph. the coarsening phase preserves connectivity, by combining all the edges that come from the joined vertices for the new vertex. In the case of overlapping edges, the edges ' weights are summed to form a new edge weight to preserve the property of the original graph during the partitioning phase. In this process, the algorithm relies on the concept of *matching*. A matching of a graph is a set of edges, in which no two edges are incident to the same vertex. Given a matching of the current graph, a coarser version of this

[5]In addendum to [2], Begelman et al. mentioned the Metis package as an alternative to their spectral bisection algorithm

[6]For a more complete description of the different implementations in the package, we need to refer to the official manual [22]

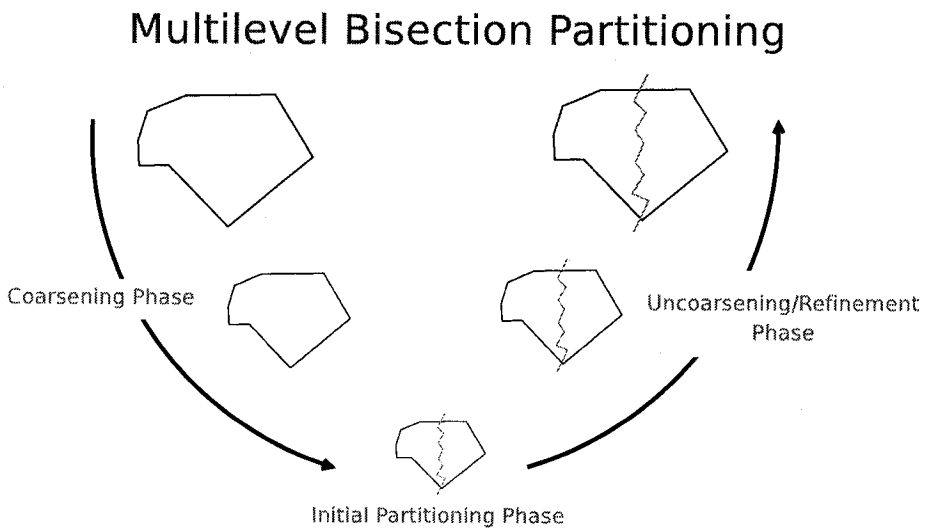


Figure 3.7: The sequence of Phases that a graph is processed through the multilevel bisection partitioning[22].

graph is obtained by collapsing the vertices that are in the matching sets. The Heuristic objective in which edges and vertices are collapsed for is based on the principle of *maximal matching*. A maximal matching of graph is a condition in which an edges that is not in the matching set has one of its vertices matched or collapsed to a new multinode vertex. In *pmetis*' implementation, they used a greedy method named Heavy Edge Matching (HEM) as their heuristic in finding the maximal matching. This method also implements a stopping condition of not coarsening a graph that has less than 100 vertices.

For the initial partitioning phase, the previously obtained coarse graph is partitioned into 2 partitions. In *pmetis*, the implementation they used is *Greedy Graph Growing Partitioning Algorithm* (GGGP). In GGGP, a vertex is chosen at start and is grown in a breadth first fashion, a characteristic taken from Graph Growing Partitioning(GGP) algorithm, to a region that cover half of the vertices, thus bisecting the graph. For each vertex, the gain in edge-cut for insertion to the region is computed and ordered, a characteristic taken from Kernighan-Lin algorithm (KL). The growing process then greedily selected the vertex with the least edge-cut to be inserted. In their experiment, they implemented this method because they found this method is faster and performs better than other partitioning techniques such as the 2 algorithms it is similar to, i.e. Kernighan-Lin's (KL) and Graph Growing Partitioning(GGP); and Spectral Bisection

(SBP) partitioning.

Finally, for the uncoarsening phase, the coarse and partitioned graphs are projected back to the original graphs. The characteristics of this phase ensure *edge-cuts* and *balance* of the resulting partition of a coarse graph are preserved when transitioning back to more finer graphs. For *pmetis*, Karypis and Kumar implemented the uncoarsening phase through modifications of the Kernighan-Lin's refinement algorithm. In the original Kernighan-Lin (KL) refinement algorithm the focus is to iteratively find internal vertices of a partition that can be moved to the other partition to reduce edge cut. Given its stopping condition i.e. unchanged edge-cuts after 50 swaps, and a good partitioning from the coarse phase this method generally yields a good time complexity. In its' modification of KL refinement, *pmetis* introduces an optimization on the complexity by considering only boundary vertices as potential swaps. Since this reduction is shown empirically to be as good as the original, the modified refinement algorithm , Boundary KL refinement (BKL) is the chosen implementation of the system.

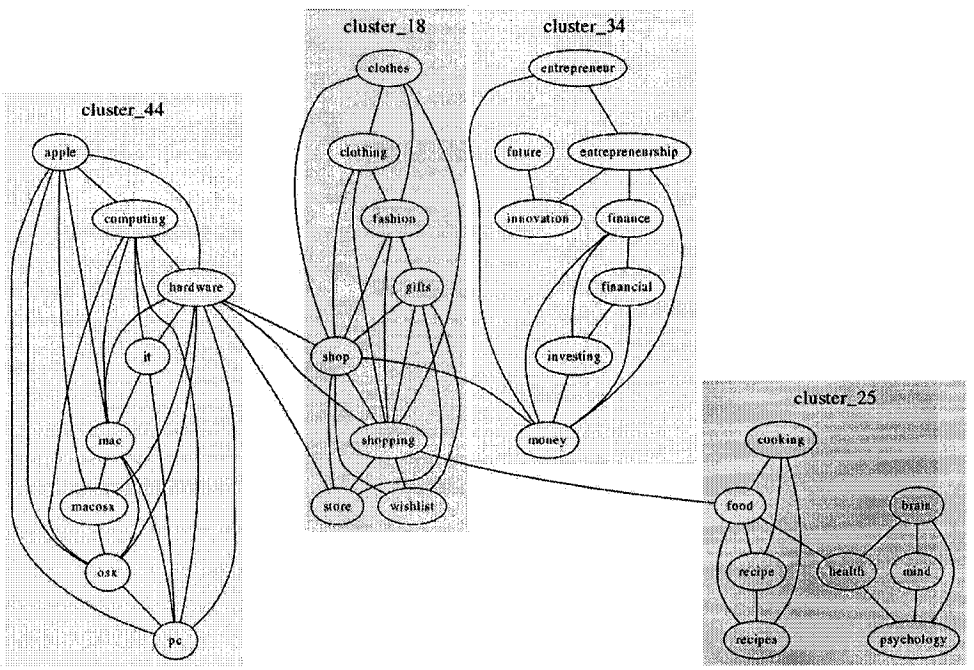


Figure 3.8: Four clusters that were produced from the Pmetis partition algorithm.

As an example of the product of this algorithm, we have in Figure 3.8 4 of the 75

clusters that Pmetis produced from our dataset. The complete list of the 75 clusters is available in Appendix A.2.

Chapter 4

Experiments and Results

On this thesis, we have set out a goal of solving the coverage and consistency of a collaborative tagging system with automatic text categorization. in Chapter 3, we had established that for a large scale web directory categorization problem, hierarchical categorization may outperform flat categorization. Based on the similarity of the two problems, we hypothesize that hierarchical categorization would also outperform flat categorization on a collaboratively tagged data.

In this chapter, we set out to prove this hypothesis by presenting an experiment that deals with the following two comparisons. The first comparison examined the different learning methods in flat categorization. More specifically, we will discuss the characteristics of classification produced from two different learning methods, i.e. Naïve Bayes and Support Vector Machine. The second comparison examined the hierarchical aspect of categorizing collaboratively tagged data. That is, using the hierarchy we obtained from the graph partitioning algorithm, we compare the performance and efficiency of the hierarchical version of the two learning methods with its flat categorization counterpart. Before we discuss the experiment, we will also outline other parameters of the experiment, i.e. dataset, features, and evaluation methods.

4.1 Dataset

In Section3.1, we have explained the steps to obtain a text categorization dataset from documents that have been bookmarked and tagged in Delicious web service. From January 24 to January 27, 2007, we obtained a collection of **7,583** text documents that had been tagged and bookmarked by at least 100 users. The collection of tags that are

considered for this experiment consist of **567** of the most popular tags (Tags that have at least 50 documents). In terms of multi-labeledness, there are an average of $22.8\frac{\text{tags}}{\text{documents}}$.

As mentioned in Section3.2, the dataset are similar in terms of robustness and categorical distribution as the first two-level of categories of web directory dataset that had been used in [15, 28]. In Appendix A.1 we listed all the tags and the number of documents that are related to them.

4.1.1 Hierarchical Structure of The Dataset

In order to incorporate some form of a hierarchy to our dataset, we have added an intermediary level of pseudo-categories. These added categories serve as abstract clustering between the root category and the 567 original tags (now considered leaf categories of a hierarchical categorization).

The extra layer of extraction consist of 75 abstract categories or clusters were obtained through the partition of the co-occurrence matrix of the 567 original categories. The partition were made using the Pmetis partitioning algorithm[22] and the Dice weighted co-occurrence matrix(see Section 3.4.1and Section 3.4.2 for details on this process). The list of clusters, i.e. the list of first level abstract categories, is available in the Appendix A.2.

As a side note, we acknowledge that the number of clusters,i.e. 75, is somewhat chosen arbitrarily. Given the infancy of this research, we wished to first show the deployment of hierarchical categorization as an improvement over the flat categorization before moving into other hierarchical structures. For future work, we will consider different forms of hierarchical ontology such as a deeper levels of hierarchy and dendograms that are produced from agglomerative clustering.

4.2 Features

In terms of features, we have 567 feature-sets[1], each set corresponds to the vector space model of a respective category, that consist of 1000 statistically selected words and URL tokens.

For each category feature-set, we started with 1,508,716 unique word and URL tokens. The Bag-of-Words tokens were obtained through the Libbow Toolkit[32] which includes

[1]The reason we have multiple feature-sets is the one-vs.-all or multi-model classification approach that we have adopted.

the process of tokenizing, removal of stop words, stemming, and conversion to lower case terms. The URL tokens on the other hand were obtained from our own tokenization algorithm.

We then remove low occurring features by setting Document Frequency threshold to 3. As the result of this reduction, we still have 55,467 unique word and URL tokens as our feature set. As a final reduction step, we computed the information gain(IG) score for each of these features and take the top **1000** features as the representation of the corresponding category.

Advice	Business	Dotnet	Government	Tutorial	Digg
advic	busi	asp	(URL) gov	tutor	digg
spend	compani	net	govern	html	bookmark
tip	economi	wpf	congress	method	(URL) digg
import	monei	framework	american	simple	sport
monei	ceo	(URL) msdn	national	css	social
habit	market	microsoft	military	basic	newsvine
people	sale	(URL) aspx	politic	create	(URL) furl
great	custom	studio	official	tip	reddit
thing	entrepeneur	(URL) microsoft	feder	technique	blog
succ	industri	sql	iraq	step	awesom

Table 4.1: A sample of top Information Gain ranked word stems across the tags ***Advice***, ***Business***, ***Dotnet***, ***Government***, ***Digg***, and ***Tutorial***. Terms that have URL bracket on the left side are tokens that are derived from the Bag-of-URL representation

In Table 4.1, we have a short summary of the the top features selected for some of the tags in our dataset. From this list, we can see that IG scoring effectively rank words and URLs that are semantically close to a given category. We also see the inclusion of some URL tokens in tags like ***Dotnet***, ***Digg***, and ***Government*** as indication that the bag-of-URL representations can also be used to discriminate certain concepts.

Finally, after feature reduction is performed we performed the TF-IDF weighting and cosine normalizations [2].

[2]The TF-IDF and cosine normalization were only used in one of the two classifiers that are used in our experiment, i.e. Support Vector Machine. The Multinomial Naïve Bayes does not used the TF-IDF and normalization because it has its own probabilistic weighting

4.2.1 Hierarchical Features

For the Hierarchical features, we have created 75 + 567 new feature sets that consist of 1000 selected word and URL tokens. The first 75 feature sets represent the vector space model of the 75 clusters or pseudo-category we have for the hierarchical version of our dataset. They essentially were gathered using the same process as the 567 feature sets that were used for the non hierarchical version of our dataset. For each of the 567 leaf tags' feature sets, we created a different feature set than the non-hierarchical version due to the divide and conquer nature of our approach.

Given the hierarchical relationship that each of the 567 tags, the vector space representation of a leaf or child category is theoretically used to classify a smaller problem than flat classification, i.e. discriminating documents that relate to itself and documents that relate to its sibling categories. Therefore in the process of gathering and selecting the leaf tags feature sets, we have different feature ranks because we only compute Information Gain on instances that have the same parent category. It should be noted that the process of selecting both hierarchical and flat features are similar to the one performed by Dumais and Chen[15]; and Koller and Sahami[27].

4.3 Classification

For the classification component of our experiment we have selected two learning methods – Naïve Bayes and Support Vector Machine– that will be used in both hierarchical and non-hierarchical classifications. The deployment of each methods also follows the one-vs. all or multiple model classification mentioned in Section 3.3.1. Our decision in deploying multiple classifiers for each tags are similar to Sebastiani's characterization of web categorization??. That is, in order for us to assume the addition of new tags and the changing of ontology, we need to avoid the need to retrain our whole classification system (this is the case for a single multi-class classifier) by using a separate classifier for each category or tag. In the following subsections, we will discuss the implementation of each learning methods as well as the details of our hierarchical classifications.

4.3.1 Naïve Bayes classification

For Naïve Bayes classification, we have chosen the multinomial Bag-of-Words event model[30] that is part of the LibbOW implementation[32]. Through this learning method,

we do not weight and normalize features through TF-IDF and cosine normalization because the probabilistic framework for this method models features as multinomial events that includes properties such term frequency and document length. The method instead computed the posterior probability of a document to category assignment, a yes and no case due to our one-vs. all approach, according to the event of observing a sequence of word tokens in the document. In [30] Mccallum and Nigam claimed a reduction of error by an average of 27% over the Bernoulli model of Naïve Bayes, i.e. a model that only considers terms' appearance. The datasets that were used for that study include: Yahoo! directories, newsgroup and webKB datasets. This claim has also been confirmed through a comparative work by Yang et al. [59].

4.3.2 Support Vector Machine Classification

For SVM classification, we used the implementation of SVM^{Light}[19]. In SVM^{Light}, we have an optimized implementation of SVM because of its ability to reduce the quadratic optimization problem through a general decomposition strategy. Through this strategy, Joachim has reduce the complexity of training time into one that is in the order of $O(N^c)$ where N is the number of training examples and c is a number between 1.2 and 1.5[59].

For the parameters of SVM^{Light} we have selected the default parameters that were provided in the software. However we would like to note that the choice of kernel, i.e. linear kernel, was decided based on selected runs of the algorithm. In these preliminary experiments, we found the linear kernel to outperform other kernels, i.e. RBF, sigmoid, polynomial, and Gaussian kernels, classifying a selected number of tags.

4.3.3 Hierarchical Classification

As mentioned in Section 3.3.2, the training and classification of a hierarchical classifier is structurally different than the ones generally employed in the flat categorization. At the top-level classification, we train 75 binary classifiers (corresponding to the 75 parent categories or clusters) using the whole training set. At the second-level classifications, we trained each of the 567 classifiers (the number of tags in our dataset) using only training instances and features that relate to the parent category of that tag. This, in other words, reaffirms the notion that the hierarchical structure decompose the classification process of our tags by learning a smaller hypothesis[27].

For the classification of a hierarchical test set, we have used the standard approach[15, 28, 27] of decomposing the classification process (for each of the three aforementioned

learning methods) into a two-step sequence. In the first step, test documents are classified to parent categories.In the second step, we proceed to the classification of test documents only for the leaf categories or tags whose parent category is assigned to the documents. It is noted that in [15], Dumais and Chen referred to this technique as the Boolean scoring method.

In general, the Boolean scoring method is performed by recursively classifying unseen documents to subsequently lower hierarchical categories. In our case, we only have a two level recursion because of our hierarchical structure. The stopping condition for these recursions is determined by either the classification result of its previous level or parent categories, i.e. not proceeding to child category classification because it is classified as a parent category, or the fact that the current category is a leaf category, i.e. categories with no children or subtopics.

Based on this description there are both advantages and disadvantages of applying the Boolean scoring method. The advantage of this scoring method is the reduction in classification time. For instance, in flat classification, unclassified documents have to be classified to all of the categories. However, in hierarchical classification, documents that are not classified to a parent category would not be subjected to classification by the corresponding children classifiers. The disadvantage of the Boolean scoring method is the possibility of documents incorrectly classified by the parent classifiers. If a parent classifier does not classify documents that are actually labelled with one of its leaf category, then these documents would not be classified to any of the leaf category. Ultimately, for a document to be correctly classified to a tag, the hierarchical classification needs to produce two correct classifications, i.e. one for the parent category and one for the tag itself. Due to this disadvantage, we need to consider the behavior of each classifier. A lax classifier that produces high recall values may help avoid disadvantage of the Boolean scoring method but it may also decreases the precision or performance of the system.

In the future, we can also consider a less explicit approach to hierarchical classification that allows documents to be classified into every categories in the system and combine the output of these classifiers according to the hierarchical structure. In [15], for instance, Dumais and Chen used the multiplication of parent and child classification outputs to build one of their hierarchical classifications. This method conversely has the advantage of allowing each unclassified instance to be classified to each category and the disadvantage of not potentially reducing the classification of new instances, i.e. a reversal of Binary scoring's advantages and disadvantages.

4.4 Evaluation

In order to evaluate our experiment, we need to first organize the partition into training and test sets. With that in mind, we have selected the **4-fold cross validation** technique because of its ability to add a degree of randomness on our evaluation. In terms of the number of cross validation, we set this number to 4 instead of the normal value of 10 because we are limited to the time and space constraint of training multiple binary classifiers. However, in comparison to the method of using disjoint sets for creating training and test sets in web directory categorizations[15, 28], the 4-Fold cross validation should produce a more statistically sound result[3].

For the choice of performance evaluation, we consider measurements that describe how well categories are learnt. These categorical measurements include: Precision, Recall, F-Measure (see Section 2.2.4), and Receiver Operating Characteristic (ROC) analysis. In **Precision**, we have a measurement of the filtering capability of a classifier or purity of the positive hypotheses. In **Recall**, we have a measurement of the concept recognition capability of a classifier or the completeness of the positive hypothesis. In **F-Measure**, also known as F_1-Measure, we have a measurement that combines both the level of completeness and purity of a classifier or, in other words, the harmonic mean of both Precision and Recall. In ROC analysis, we are looking aat the performance of the learning algorithm by plotting the sensitivity and specificity across different threshold (More on ROC analyses in the next subsection). For the first three measurements, we also computed the **Micro-Average** and **Macro-Average** scores. For the last one, we relied on the average of Area under the ROC curves for each different categories.

Finally, to measure the statistical significance of the flat versus hierarchical comparison, we applied the Wilcoxon signed-rank test and paired t-test to the resulting F-Measures and Area under the ROC curves. In a significance test, the goal is to see whether the observed improvements or decline in performance is produced not by coincidence. The assumption that the performance of two classification systems are similar enough is called the *null hypothesis* and the probability of obtaining our result given that the said hypothesis is true is called the *p-value*. In both Wilcoxon and paired t-test, the p-value are generated using different types of scoring. The former relies on ranking the performance while the latter relies on simply analyzing the actual performance. In [13],

[3]In other web categorizations experiments[15, 28], the use of disjointed training and test sets are justified because the size of the dataset would make any K-fold cross validation a time consuming experiment

Demzar argues that for evaluating two classifiers, the Wilcoxon methods of ranking performance over different dataset (in our case, categories) is appropriate because it is less prone to outliers (extremely good and bad performance). Nonetheless, for this thesis, we however, apply the strategy proposed by Yang and Liu[57] that uses multiple significance tests to a more complete and objective analyses.

4.4.1 ROC Curve Analysis

In various machine learning studies, Receiver Operating Characteristic (ROC) curves have been used to compare the performance of different learning methods. Schapire and Singer[44], for instance, used the ROC curves to compare the performance of Boostexter classifications with some baseline classifiers. Meanwhile, Chawla et al.[7] used ROC curves to show improvements of their sampling technique, SMOTE, over other sampling techniques such as under-sampling.

In general, The ROC curve is a non decreasing function that represents the trade-off between the sensitivity and specificity of a binary classifier. The sensitivity of a classifier is essentially the **true positive rate** or the ratio between the number of correctly classified positive instances and the total number of positive instances. This in terms of the confusion matrix we have in Table 2.2 can be formulated as:

$$\begin{aligned} Sensitivity = TPRate &= \frac{CorrectlyClassifiedPositiveInstances}{TotalNumberofPositiveInstances} \\ &= \frac{TP}{TP+FN} \end{aligned} \tag{4.1}$$

Specificity of classifier, on the other hand, is equivalent to the ratio between the number of correctly classified negative instances and the total number of negative instances.

$$\begin{aligned} Specificity &= \frac{CorrectlyClassifiedNegativeInstances}{TotalNumberofNegativeInstances} \\ &= \frac{TN}{TN+FP} \end{aligned} \tag{4.2}$$

This formulation can also be expressed in terms of **false positive rate** as:

$$\begin{gathered} Specificity = 1 - FPRate \\ \text{Where} \\ FPRate = \frac{FP}{TN+FP} \end{gathered} \tag{4.3}$$

Based on the two measurements, there are several observations that we could made about the points along the ROC curve. First, we noticed that a 100% sensitivity rate would indicate the ability of classifier to correctly classify all positive instances (0% would indicate the opposite). This in other word is the same measurement as Recall. Second, we notice that a 100% specificity rate would indicate the ability of classifier to correctly all negative instances (0% would indicate the opposite). This is different than precision because specificity rates the ability of a classifier in recognizing negative instances while precision rates the purity of a classifier's positive predictions (the rate of True Positives over all positive prediction). Third, we notice that both sensitivity and specificity were derived only from one side of confusion matrix column (see Table 2.2). This according to Fawcett[16], would make both measurements –unlike Precision, Accuracy, and F-Measures– to be insensitive to class skew.

The last observation is particularly important to our task because not only we had different degree of class skews (distribution of positive instance), we also expect class skew to potentially change as we use a bigger and newer collection of webpages. With that in mind, we shall look into the different analysis we can make by using the ROC curve.

The ROC curve represents the trade-off between specificity and sensitivity by plotting the values of false positive rate (the X-axis) and true positive rate (the Y-axis) on an ROC space[4]. It should be noted that, in order to interpolate an ROC curve we need to remember that the discrete and absolute outputs of different classifiers (when applied in blackbox fashion) are actually a product of thresholds being applied to a relative and continuous classification scoring. For example, in a Binary Naïve Bayes classifications, we have a default policy of labeling a document to category,i.e. a discrete decision of "Yes" and "No", based the on the most probable hypothesis. This policy is equivalent to the posterior for the document relating to category, i.e. the probabilistic value of $P(d|c)\,P(c)$, passing the threshold value of 0.5 (since $P(d|c)\,P(c)$ and $P(d|\not c)\,P(\not c)$ add to 1). If we want to see a higher True Positive rate, we can try lowering the threshold to value smaller than 0.5. Since this also opens the chance of obtaining a higher or if possible an equivalent [5] FP rate, we are able to plot and interpolate different points that

[4]Since it is a trade-off analysis, it is convenient to use false positive rate instead of the specificity because the former is the inverse of the later which also meant a positively sloped linear relation with the true positive rate or sensitivity of a classifier.

[5]Based on the formulation of FP rate, we can not have a lower FP rate by allowing more positive predictions.

makes an ROC curve of a classifier.

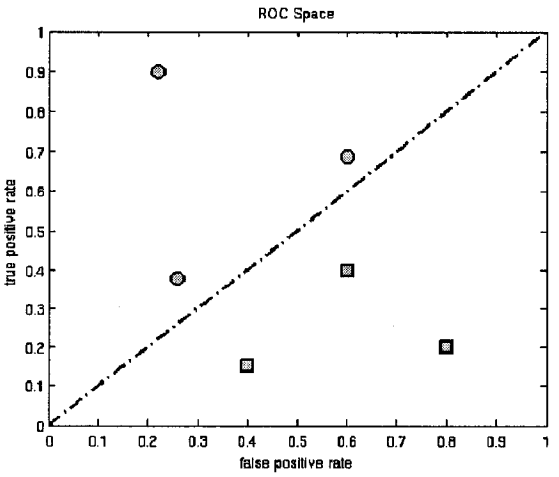
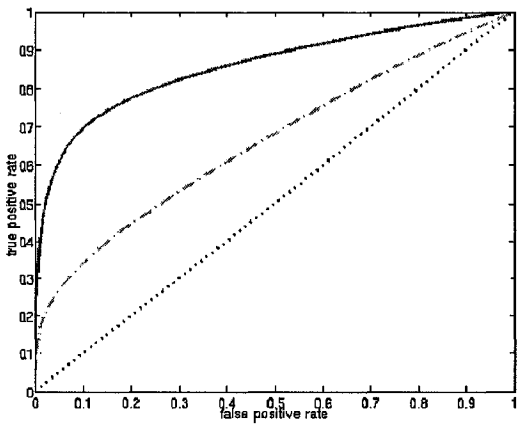


Figure 4.1: An example of ROC space with good and bad classifiers trade-off points

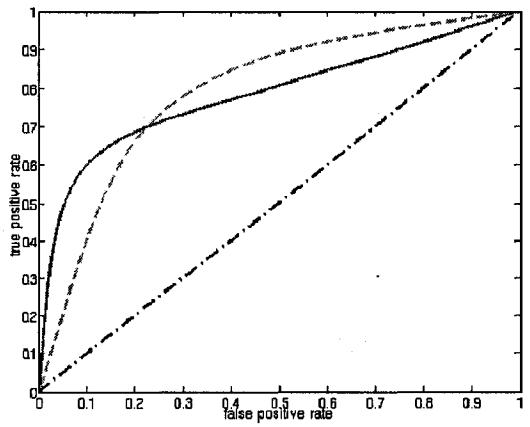
As we can see in Figure 4.1, we have exampled of an ROC space that plots a classifier's false positive rate to its true positive rate. In Figure 4.1, we can see there is a diagonal line that correspond to an identity function (TP rate = FP rate). That line represents the trade-off of having a random binary classifier that produce equal rates of true positive and false positive values[6]. Points such as the square marks below the random prediction line would indicate a classifier that actually performs worse than random because those points would indicate for a two class problem, that in their positive predictions there are a lot more false positives than the desired true positives. Theoretically this is not possible because we can always interpret the inverse of the result of that classifier. Points such as the circular marks that falls above the random prediction line would indicate a better than average performance because of they are opposites of the aforementioned worse than random points.

In terms of comparing two classifiers performance, there are 2 possible scenarios that one can observe from their ROC curves. First, we can have a simple case in which an ROC curve that envelopes another ROC curve (see Figure 4.2(a)). If this is observed, we can conclude that the classifier that produce the enveloping ROC curve to completely dominate the other classifier because for every threshold that produce the same FP rate,

[6]we can see for a random binary classifier that predicts positively for $N\%$ of the time, its composition of positive prediction will on average produce a rate of $N\%$ for its true positive and $N\%$ rate for its false positives.



(a) An example of two classifiers ROC curves in which one curve envelopes the other one (one classifier is more superior in every decision level).



(b) An example of two classifiers ROC curves that intersects (one classifier dominates the other under certain conditions).

Figure 4.2: Two examples of ROC curve comparisons in which one allows a classifier's dominance to observed graphically while the other requires a singular value such as AUC to determine classifier's dominance

i.e. same level of specificity, we have a higher TP rate, i.e. better sensitivity, for the former classifier. In the second and more general case, we can have two ROC curves that intersect one another (see Figure 4.2(b)). If this is observed, we can not say which classifier is more superior except that we can know that under certain conditions , i.e. pivoted around the value of FP rate in which the curves intersect, that one classifier would out perform the other.

Since the second observation is a common one, we will now refer to Area under an ROC curve (AUC) as a singular, more decisive and more generalized observation that we can make from an ROC curve. As the name conveyed, AUC is the geometric area of all the ROC space that are enveloped in between the ROC curve and the X-axis. Statistically speaking, the AUC of a classifier indicates the probability of that classifier's ability to rank a randomly chosen positive instance higher than a randomly chosen negative instance (the former has a higher score)[16].The AUC is also significant in ROC analyses because in the case of two intersecting ROC curves we can have a more generalized comparison by comparing the two classifiers' AUC. If one classifier has a bigger AUC than the other, then that classifier is a better classifier because it has a higher probability of correctly ranking a randomly chosen positive instance over a

randomly chosen negative instance. It should be noted that the diagonal line that is a random classifier's performance would have an AUC of 0.5 and a perfect classifier would have an AUC of 1.

Since, we have used multiple binary classifiers, the aforementioned ROC information is sufficient for our evaluation. However in [16], there are discussions about more advanced issues such as the application of ROC curve in a multi-class classifier or an ensemble of classifiers.

4.5 Results

In an effort to organize the comparison between hierarchical and flat categorization, we have divided the result and observations of our experiment into two subsections. The first subsection, Section 4.5.1, will deal with the result of the flat categorization from Naïve Bayes and SVM classifiers. These results would serve as a baseline comparison for the hierarchical categorization results which are discussed in Section 4.5.2.

4.5.1 Flat Categorizations Comparison: Naïve Bayes and SVM Classifiers

In comparing the categorical performance of Naïve Bayes and SVM Categorizations, we have summarized the Macro and Micro-Average scores of each classifier in the following table. In terms of both Macro and Micro-averaged Precision, we can see that SVM

Micro Average Scores

	Precision	Recall	F-Measure
Naïve Bayes	25.6%	60.4%	35.9%
SVM^{Light}	71.2%	29.9%	42.1%

Macro Average Scores

	Precision	Recall	F-Measure
Naïve Bayes	20.2%	59.6%	28.7%
SVM^{Light}	62.7%	25.2%	35.0%

Table 4.2: The Macro and Micro Average summary of Naïve Bayes and SVM performance

outperform Naïve Bayes classifiers. However, the trade-off of having a very high precision seems to result in a significantly lower Recall. It shoud benoted that Rennie et al. [40] have explained the cause for higher recall in Naïve Bayes classifiers as the assumption of term or feature independence. More specifically, the term independence assumption combined with term selection allowed for redundant unigram models such as having separate entries for bigram words like "Los Angeles" to be redundantly counted during the calculation the posterior probabilities. This explanation is plausible for our result because our vector space model is also derived from the unigram model. Ultimately, from these categorical evaluations, we see SVM as the better classifier of the two because of it scored higher on both Macro and Micro Averaged F-Measure.

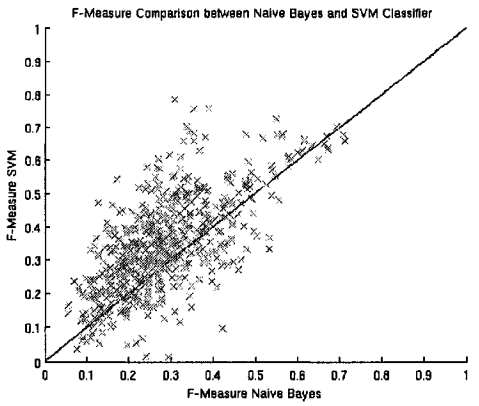
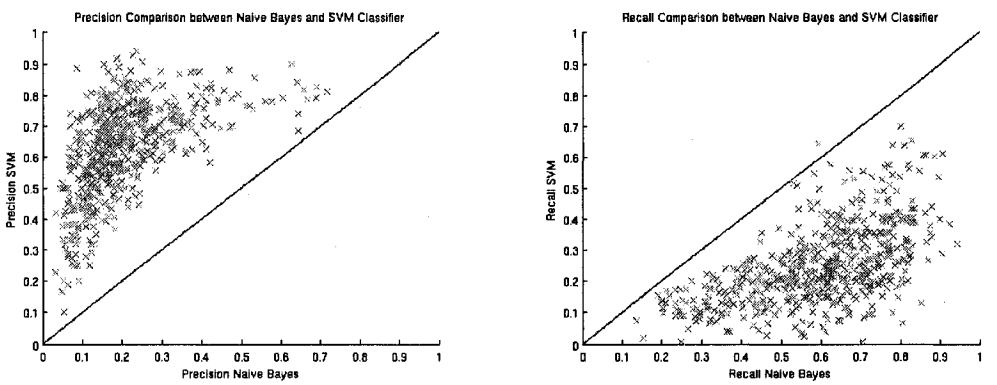


Figure 4.3: The scatter plot comparison between the F-Measure of Naïve Bayes and SVM classifiers.



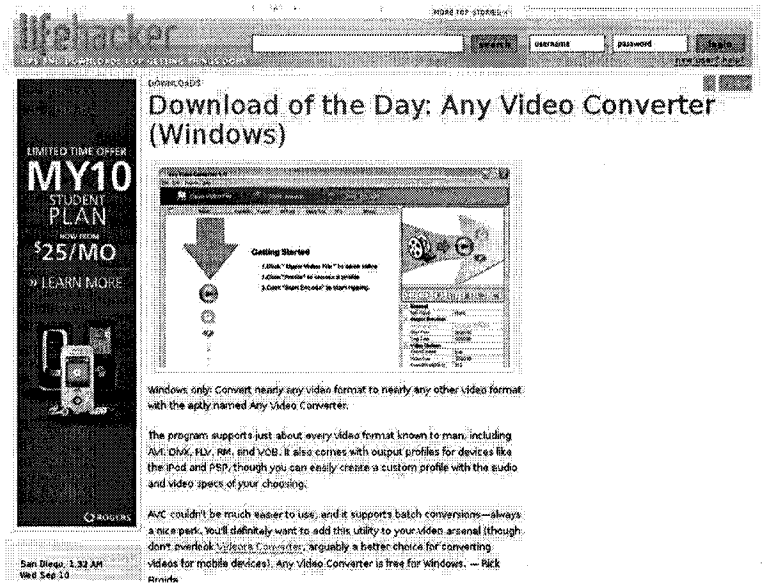
(a) The scatter plot comparison between the Precision of Naïve Bayes and SVM classifiers.

(b) The scatter plot comparison between the Recall of Naïve Bayes and SVM classifiers.

Figure 4.4: The scatter plot comaprison (for each tag) of Naïve Bayes and SVM classifiers' Precision, and Recall

To visualize the performance advantage of SVM and Naïve Bayes in Macro-Average F-Measure, we also have in Figure 4.3 and 4.4, the scatter plot comparison of each tag's Precision, Recall, and F-Measures. The X-axis in these charts represents the respective performance of Naïve Bayes classifiers, the Y-axis represents the respective performance of SVM classifiers and the points (there are 567 of them) plots the performance of the two classifiers for a given tag. More importantly, we can interpret points that are below the diagonal lines as categories or tags in which Naïve Bayes classifier (X-axis) outperforms its SVM (Y-axis) counterpart. Conversely points that are above the diagonal line represents tags in which SVM outperforms Naïve Bayes. Based on these figures, we can clearly see that for most of the tags, SVM does outperform Naïve Bayes in overall F-Measure (points in Figure 4.3 that are above the diagonal line). We can also see a clear trade-off between precision and recall of the two classifiers. That is, for precision, SVM seems to produce higher result on a consistent basis (points above the diagonal line in Figure 4.4(a)) while for recall, Naïve Bayes seems to consistently produce a higher result (points below the diagonal line in Figure 4.4(b)).

In Figure 4.6, 4.7, and 4.5, we have some annotations that were produced by both Naïve Bayes and SVM classifiers. In all of those examples we can see that the number of proper tags (True Positives) are generally higher for Naïve Bayes classifiers. However it should be noted that Naïve Bayes' lower precision have produced a large number of false positives that seems to over-represent the resulting annotations . This according to Ohkura et al.[37] may not be desirable for the complexity of a retrieval system.



(a) A webpage that contains an article about a software that convert videos (http://www.lifehacker.com/software/downloads/download-of-the-day-any-video-converter-windows-218253.php).

Flat Naïve Bayes tags:

application, apps, audio, computer, conversion, convert, converter, download, downloads, dvd, flash, free, freeware, lifehacker, lifehacks, media, movie, movies, software, tech, tools, utilities, video, videos, windows, youtube, advice, applications, backup, career, communication, consumer, daily, diy, entertainment, file, files, firefox, firefoxtoolbar, fun, geek, gtd, guide, hack, hacks, help, howto, imported, info, life, lifehack, list, lists, management, multimedia, music, opensource, optimization, organization, osx, pc, personal, productivity, safari_export, sharing, source, stock, storage, streaming, tips, todo, toread, tricks, tv, useful, weblog, work

Flat SVM^{Light} Tags:

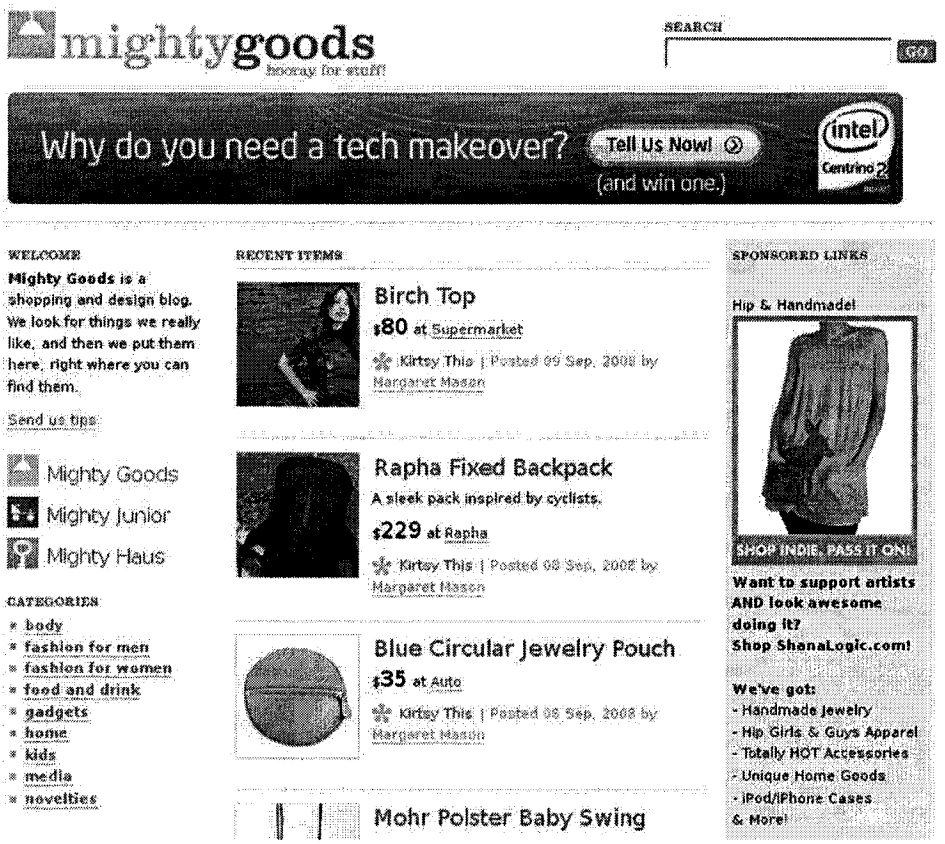
conversion, convert, download, downloads, free, freeware, ipod, lifehacker, lifehacks, software, tools, utilities, video, videos, windows, youtube, opensource

Delicious or human annotated Tags:

application, apps, audio, computer, conversion, convert, converter, download, downloads, dvd, flash, free, freeware, ipod, lifehacker, lifehacks, media, movie, movies, software, tech, tool, tools, utilities, utility, video, videos, windows, youtube

(b) The predictions or labels that were assigned by the classifiers to the website above (The tags in bold are the correct labels)

Figure 4.5: Third example of the tags produced by the SVM and Naïve Bayes Classifiers along with the originally assigned tags



(a) A blog that articles fashion and shopping topics (http://mightygoods.com/).

Flat Naïve Bayes tags:

blog, buy, clothing, cool, fashion, fun, gadget, gadgets, gifts, ideas, inspiration, products, shop, shopping, stuff, wishlist, architecture, art, audio, bargains, clothes, community, craft, crafts, culture, daily, deals, ecommerce, electronics, entertainment, funny, furniture, green, hci, health, home, humor, humour, imported, kids, lifestyle, macintosh, magazine, magazines, music, musica, nyc, personal, reviews, safari_export, store, style, sustainability, technology, toys, tutorials, web

Flat SVM^{Light} Tags:

blog, blogs, buy, clothing, fashion, gadget, gadgets, gifts, inspiration, reference, shop, shopping, stuff, wishlist, bargains, consumer, daily, deals, home, lifestyle, reviews, store, weblogs

Delicious or human annotated Tags:

blog, blogs, buy, clothing, cool, design, fashion, fun, gadget, gadgets, gifts, ideas, inspiration, interesting, online, products, reference, shop, shopping, stuff, web_2.0, wishlist

(b) The predictions or labels that were assigned by the classifiers to the website above (The tags in bold are the correct labels)

Figure 4.6: First example of the tags produced by the SVM and Naïve Bayes Classifiers along with the originally assigned tags.

(a) The homepage of the comic strip Dilbert (http://www.unitedmedia.com/comics/dilbert/).

Flat Naïve Bayes tags:

cartoon, cartoons, comic, comics, daily, entertainment, fun, funny, geek, humor, humour, imported, webcomic, webcomics, art, comedy, community, cool, culture, ecommerce, english, feeds, magazine, maps, online, portal, research, satire, search, service, share, sharing, shop, shopping, social, store, work,

Flat SVM^{Light} Tags:

cartoon, cartoons, comic, comics, fun, funny, geek, humor, humour, webcomic, webcomics

Delicious or human annotated Tags:

cartoon, cartoons, comic, comics, daily, entertainment, fun, funny, geek, humor, humour, imported, safari_export, webcomic, webcomics

(b) The predictions or labels that were assigned by the classifiers to the website above (The tags in bold are the correct labels)

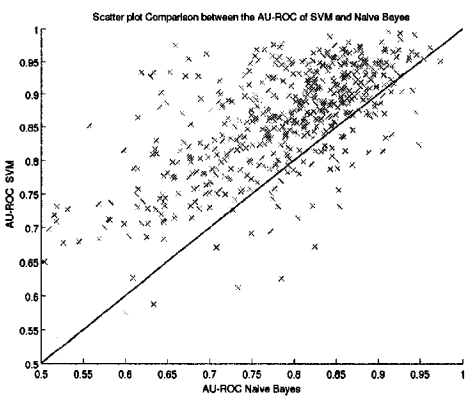
Figure 4.7: Second example of the tags produced by the SVM and Naïve Bayes Classifiers along with the originally assigned tags

ROC Analysis

Based on the result of Micro and Macro Averaged F-Measures, we have established SVM as a better classifier than Naïve Bayes. In this thesis, however, we have decided to include a second evaluation that would confirm this conclusion, i.e the ROC curves. In our ROC analysis, we have plotted the 567 ROC curves for each of the tag of a classifier using the method described in [16]. Since It is impossible to discuss each individual curves[7], we will first report on the summary of area under the ROC curve (AU-ROC) calculations.

	AU-ROC
Naïve Bayes	0.7885
SVM^{Light}	0.8606

(a) The Averaged area under the ROC curve (AU-ROC)



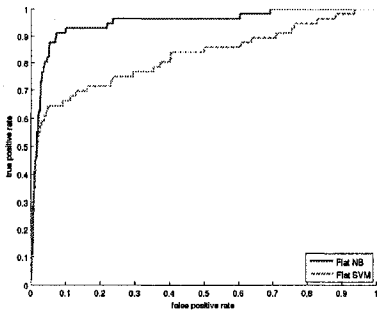
(b) The tag-by-tag AU-ROC comparison between Naïve Bayes and SVM classifiers.

Figure 4.8: The Summary of AU-ROC results for both Naïve Bayes and SVM classifiers

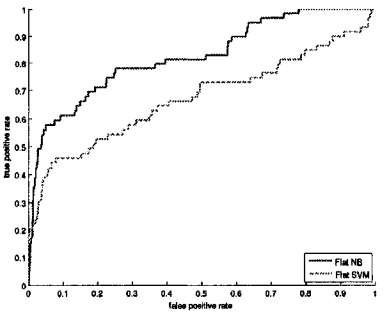
In Figure 4.8, we have averaged the AU-ROC of the 567 tags for both SVM and Naïve Bayes classifiers. Similar to the previous F-Measure comparison, we have SVM to significantly outperforming Naïve Bayes (the average AU-ROC for SVM is 0.8606 and the average AU-ROC for Naïve Bayes is 0.7885). In Figure 4.8(b), we also have the tag-by-tag comparison of the AU-ROC. Overall, we believe that for both classifiers, an average AU-ROC of around 0.8 is a good indication of sufficient performance. That is, given the probability of that classifier's ability to rank a randomly chosen positive instance higher

[7] we choose not to averaged the curves since they came from different binary classifiers

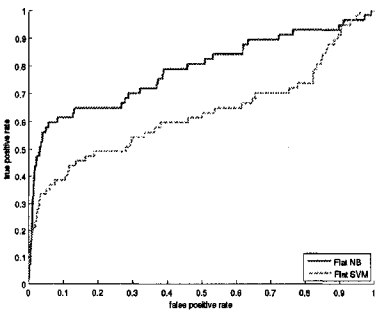
than a randomly chosen negative instance as the definition of an AU-ROC value, either classifier seems to be capable of producing quality annotations.



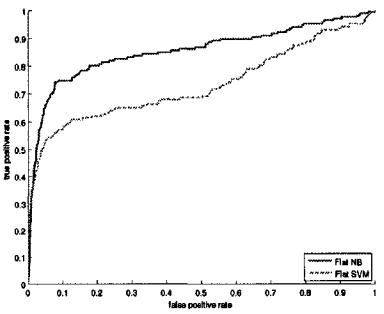
(a) The ROC curve comparison for the tag ***"cs"*** (the AU-ROC for Naïve Bayes =0.9484 and for SVM = 0.8232)



(b) The ROC curve comparison for the tag ***"ai"*** (the AU-ROC for Naïve Bayes =0.8248 and for SVM = 0.6719)



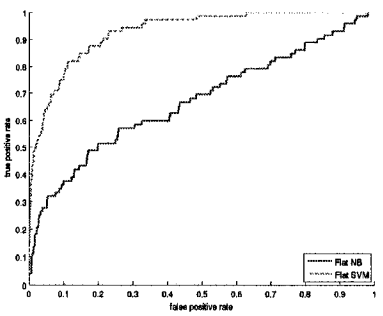
(c) The ROC curve comparison for the tag ***"programs"*** (the AU-ROC for Naïve Bayes =0.7850 and for SVM = 0.6251)



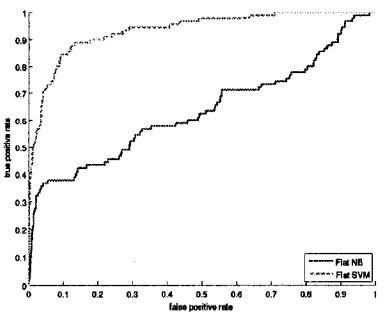
(d) The ROC curve comparison for the tag ***"economy"*** (the AU-ROC for Naïve Bayes =0.8539 and for SVM = 0.7308)

Figure 4.9: Four ROC curve comparisons that show Naïve Bayes outperforming SVM

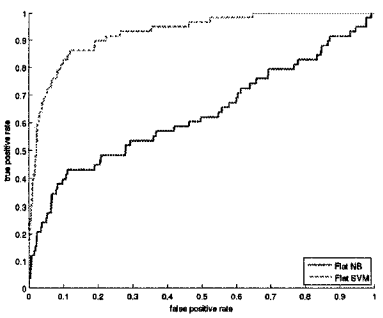
To highlight some of the interesting ROC curves, In Figure 4.9, we have four ROC curves comparisons in which the difference between Naïve Bayes and SVM's AU-ROC favors Naïve Bayes the most, i.e. top four tags that favor Naïve Bayes. Conversely, in Figure 4.10, we have four ROC curves comparisons in which the difference between Naïve Bayes and SVM's AU-ROC is the largest, i.e. top four tags that favor SVM. Based on the two figures, we see that in the event that SVM outperforms Naïve Bayes, the difference in AU-ROC seems to be significantly higher (28%) than the converse (14%).



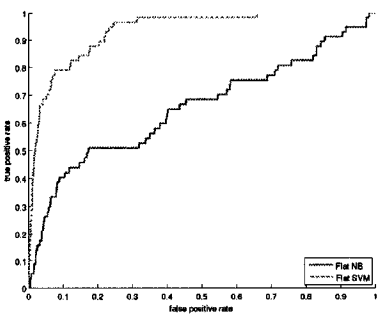
(a) The ROC curve comparison for the tag ***"visual"*** (the AU-ROC for Naïve Bayes =0.6715 and for SVM = 0.9277)



(b) The ROC curve comparison for the tag ***"training"*** (the AU-ROC for Naïve Bayes =0.6336 and for SVM = 0.9328)



(c) The ROC curve comparison for the tag ***"collaborative"*** (the AU-ROC for Naïve Bayes =0.6299 and for SVM = 0.9271)



(d) The ROC curve comparison for the tag ***"beta"*** (the AU-ROC for Naïve Bayes =0.6531 and for SVM = 0.9338)

Figure 4.10: Four ROC curve comparisons that show SVM outperforming Naïve Bayes

4.5.2 Flat and Hierarchical Categorizations

For the comparison between Flat and Hierarchical categorizations, we have in the following table the comparison between the Micro and Macro-average of the 567 tags or category performance.

Micro Average Scores

	Type	Precision	Recall	F-Measure
Naïve Bayes	Flat	25.6%	60.4%	35.9%
	Hier.	32.7%	47.9%	38.8%
SVM^{Light}	Flat	71.2%	29.9%	42.1%
	Hier.	60.1%	38.1.0%	46.6%

Macro Average Scores

		Precision	Recall	F-Measure
Naïve Bayes	Flat	20.2%	59.6%	28.7%
	Hier.	25.7%	44.1%	31.0%
SVM^{Light}	Flat	62.7%	25.2%	35.0%
	Hier.	55.7%	27.7%	35.9%

Table 4.3: The Micro and Macro Average scores comparison between hierarchical and flat classifications

Overall, we can see from the result shown in table 4.3, that for both flat and hierarchical categorization, SVM is still the better classifier than Naïve Bayes. More importantly, however, we can see for Support Vector Machine and Naïve Bayes that hierarchical classification has resulted in better performance than their respective flat classifications. For Support Vector Machine, we see an improvement of 0.9%(35.9% – 35.0%) for the Macro Average F-Measure and 4.5%(46.6% – 42.1%) for Micro Average F-Measure. Relatively speaking this is also equivalent to 2.6%($\frac{0.9\%}{35\%}$) and 10.7%($\frac{4.5\%}{42.1\%}$) improvements respectively. it should be noted that The Micro Average improvement is significantly higher than the 2% (4% relative improvement)[8] reported by Dumais and Chen [15]. For Naïve Bayes classification, we see a bigger improvement of 2.3%(31.0% – 28.7%) for the Macro Average F-Measure and a smaller improvement of 2.9%(38.8% – 35.9%) for Micro Average F-Measure. This relatively speaking, represents a Macro-average improvement of 8.01%($\frac{2.3\%}{28.7\%}$) and a Micro-Average improvement of 8.08%($\frac{2.9\%}{35.9\%}$).

[8]This was achieved through SVM classifiers as well.

Micro Average Scores

	Precision		Recall		F-Measure	
	Wilcoxon	t-test	Wilcoxon	t-test	Wilcoxon	t-test
Naïve $\text{Bayes}_{Flat-Hier.}$	$\ll$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$
$\text{SVM}_{Flat-Hier.}$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$

Macro Average Scores

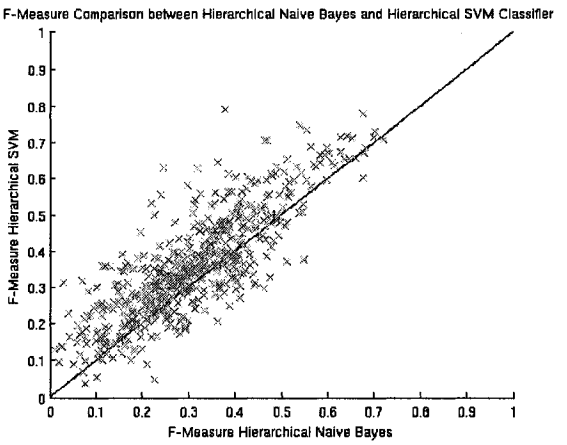
	Precision		Recall		F-Measure	
	Wilcoxon	t-test	Wilcoxon	t-test	Wilcoxon	t-test
Naïve $\text{Bayes}_{Flat-Hier.}$	$\ll$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$
$\text{SVM}_{Flat-Hier.}$	$\gg$	$\gg$	$<$	$\ll$	$\sim$	$\sim$

Table 4.4: The summary of significance test (Wilcoxon signed-rank and paired t-test) result on flat and hierarchical classifier comparison.

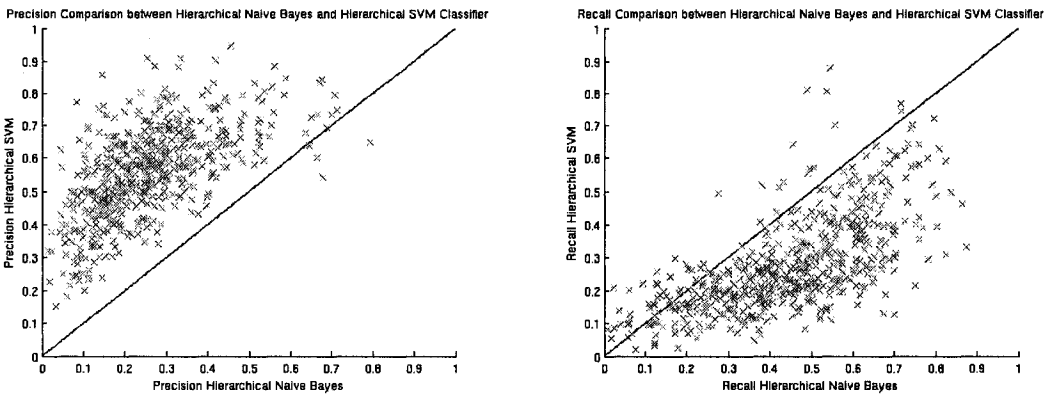
In Table 4.4, we have the statistical significance tests of our flat and hierarchical classifier comparison. The notations "$\ll$" and "$<$" indicate the performance of the hierarchical classifiers to be better and statistically significant. Similarly, the notations "$\gg$" and "$>$" indicate the performance of the flat classifiers to be better and statistically significant. For both "$\ll$" and "$\gg$" the p-values are less than 0.01, i.e. statistically significant for $\alpha \leq 0.01$;For both "$<$" and "$>$" the p-values are greater than 0.01 but less than 0.05, i.e. statistically significant for $\alpha \leq 0.05$; Finally, for "$\sim$" the p-value is greater than 0.05, i.e. the difference in scores of the two systems are not statistically significant. Based on our significance test, we can say with a 99% certainty that the improvements and declines in Micro Average scores are not coincidentally generated. For the Macro average scores, we have less certainty because for the F-Measure scores of Flat and Hierarchical SVM classifiers the difference is not consider large enough for the p-value to be less than 0.05.

We also have in the following table the performance of both classifiers on classifying the 75 parent clusters. As we can see from the result in Table 4.5, the performance of both classifiers in trying to classify parent categories or the cluster of tags seems to meet with a better overall result. In the binary scoring method, a much better performance on classifying higher level concepts (broader and more generalized target concepts) are generally expected and observed[15, 28] because it would subsequently effect the quality of classification on the lower hierarchical levels.

In Figure 4.11, we have included the scatter plot comparison of the F-Measure, Pre-



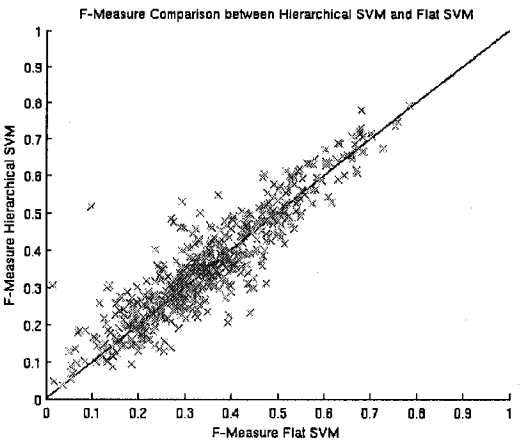
(a) The scatter plot comparison between the F-Measure of hierarchical Naïve Bayes and SVM classifiers.



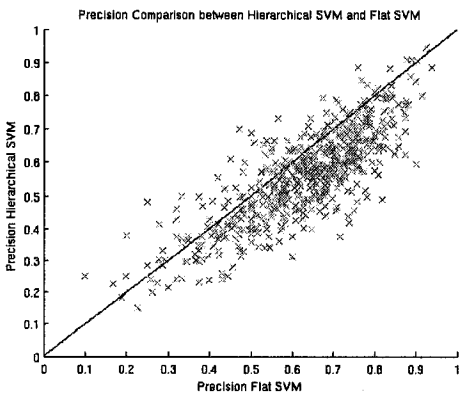
(b) The scatter plot comparison between the Precision of hierarchical Naïve Bayes and SVM classifiers.

(c) The scatter plot comparison between the Recall of hierarchical Naïve Bayes and hierarchical SVM classifiers.

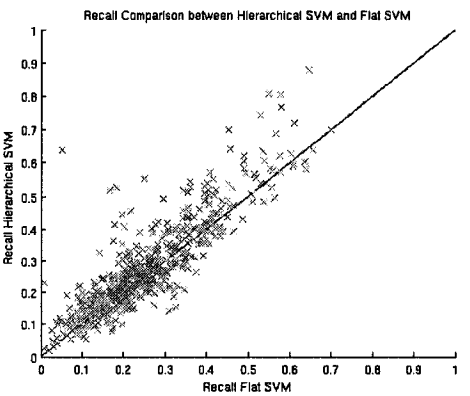
Figure 4.11: The scatter plot comparison(for each tag) of Hierarchical Naïve Bayes (X-axis) and Hierarchical SVM classifiers (Y-Axis)



(a) The scatter plot comparison between the F-Measure of hierarchical and flat SVM.



(b) The scatter plot comparison between the Precision of hierarchical and flat SVM.



(c) The scatter plot comparison between the Recall of hierarchical and flat SVM.

Figure 4.12: The scatter plot comparison(for each tag) of hierarchical (Y-Axis) and flat SVM (X-Axis)

Micro Average Scores

	Precision	Recall	F-Measure
Naïve Bayes	43.5%	63.3%	51.6%
SVM^{Light}	65.9%	57.4%	61.3%

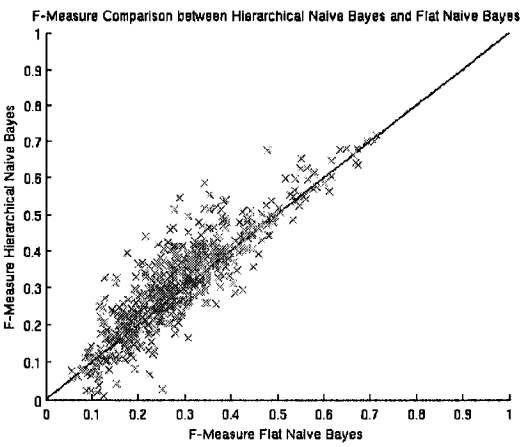
Macro Average Scores

	Precision	Recall	F-Measure
Naïve Bayes	36.2%	59.6%	43.4%
SVM^{Light}	62.4%	41.7%	48.5%

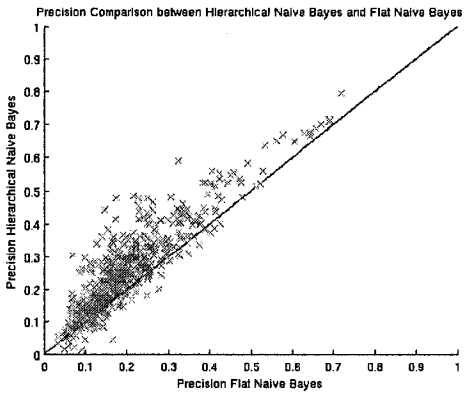
Table 4.5: The Macro and Micro Average summary of Hierarchical Naïve Bayes and SVM performance on the categorization of parent categories or the pmetis derived clusters of tags.

cision and Recall of the two hierarchical classifiers. Similar to the flat categorization result, we still have Naïve Bayes dominance over SVM on Recall and SVM dominance on Precision. We also have in Figure 4.12(for SVM classification) and Figure 4.13 (for Naïve Bayes) a more detailed comparison of both flat and hierarchical classifications. The flat and hierarchical comparisons (Figure 4.13(a) and Figure 4.12(a)) clearly show for both Support Vector Machine and Naïve Bayes classification that there is an overall improvement to applying the hierarchical classification, i.e. majority of points residing above the identity function on the F-Measure graph. It is also important to note that the improvement on the Naïve Bayes classification is actually caused by lower recalls and higher precisions, i.e. majority of points residing above the identity function on the Precision graph(Figure 4.13(b)) and below for the Recall graph(Figure 4.13(c)). Opposite to that observation, the improvement in SVM is actually caused by higher recalls and lower precisions, i.e. majority of points residing above the identity function on the Recall graph(Figure 4.12(c)) and below for the Precision graph(Figure 4.12(b)). Based on these two observations, Hierarchical categorization seems to improve a traditionally conservative classifier, i.e. SVM and a traditionally lax classifier, i.e. Naïve Bayes, by transforming them to a more balanced classifier.

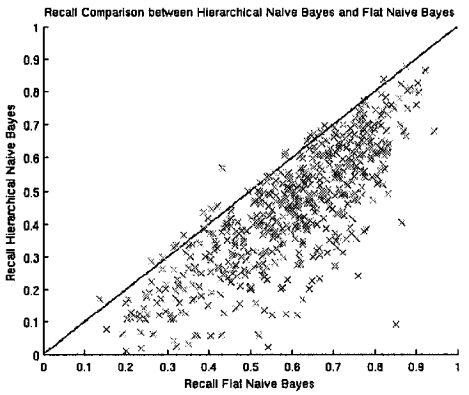
Finally, to put the performance of hierarchical Naïve Bayes and SVM, we have in Figure 4.15, 4.16, and 4.14, the example annotations of all the classifiers we have experimented on. It should be noted that the three webpages that are mentioned in these figures were also used in the discussion of flat categorization (See Section 4.5.1). based on the three figures, we can see for SVM that the hierarchical classification have allowed



(a) The scatter plot comparison between the F-Measure of hierarchical and flat Naïve Bayes.



(b) The scatter plot comparison between the Precision of hierarchical and flat Naïve Bayes.



(c) The scatter plot comparison between the Recall of hierarchical and flat Naïve Bayes.

Figure 4.13: The scatter plot comparison(for each tag) of hierarchical (Y-Axis) and flat Naïve Bayes (X-Axis)

more true positive and false positive tags to be added. Figure 4.15, for instance, shows an increase of 4 true positives (from 14 to 18) and an increase of 3 false positives (from 9 to 12) from the annotation produced by hierarchical SVM classifiers. For Naïve Bayes, the hierarchical classification seems to produce the opposite effect of decreasing the true positives as well as the false positives in documents' annotation. Figure 4.14, for example, shows a decrease of 4 true positives (from 26 to 22) and a more significant decrease of 28 false positives (from 51 to 23).

Flat Naïve Bayes tags:
application, apps, audio, computer, conversion, convert, converter, download, downloads, dvd, flash, free, freeware, lifehacker, lifehacks, media, movie, movies, software, tech, tools, utilities, video, videos, windows, youtube, advice, applications, backup, career, communication, consumer, daily, diy, entertainment, file, files, firefox, firefoxtoolbar, fun, geek, gtd, guide, hack, hacks, help, howto, imported, info, life, lifehack, list, lists, management, multimedia, music, opensource, optimization, organization, osx, pc, personal, productivity, safari_export, sharing, source, stock, storage, streaming, tips, todo, toread, tricks, tv, useful, weblog, work

Hierarchical Naïve Bayes tags:
apps, audio, computer, conversion, convert, download, downloads, dvd, free, freeware, ipod, lifehacker, lifehacks, media, movies, software, tech, tools, utilities, video, videos, windows, backup, computers, digg, files, geek, hack, hacks, howto, lifehack, links, mac, mobile, mp3, multimedia, music, organization, pc, productivity, safari_export, tips, useful, weblog, xp

Flat SVM^{Light} Tags:
conversion, convert, download, downloads, free, freeware, ipod, lifehacker, lifehacks, software, tools, utilities, video, videos, windows, youtube, opensource

Hierarchical SVM^{Light} Tags:
apps, audio, computer, conversion, convert, converter, download, downloads, dvd, free, freeware, ipod, lifehacker, lifehacks, media, movies, software, tools, utilities, video, videos, windows, youtube, howto, mac, opensource,

Delicious or human annotated Tags:
application, apps, audio, computer, conversion, convert, converter, download, downloads, dvd, flash, free, freeware, ipod, lifehacker, lifehacks, media, movie, movies, software, tech, tool, tools, utilities, utility, video, videos, windows, youtube

Figure 4.14: A comparison of the tags produced by the flat and hierarchical classifiers for the webpage http://www.lifehacker.com/software/downloads/download-of-the-day-any-video-converter-windows-218253.php (see Figure 4.5). The tags in bold are the correct labels.

Flat Naïve Bayes tags:
blog, buy, clothing, cool, fashion, fun, gadget, gadgets, gifts, ideas, inspiration, products, shop, shopping, stuff, wishlist, architecture, art, audio, bargains, clothes, community, craft, crafts, culture, daily, deals, ecommerce, electronics, entertainment, funny, furniture, green, hci, health, home, humor, humour, imported, kids, lifestyle, macintosh, magazine, magazines, music, musica, nyc, personal, reviews, safari_export, store, style, sustainability, technology, toys, tutorials, web

Hierarchical Naïve Bayes tags:
blog, blogs, buy, clothing, fun, gadget, gadgets, gifts, inspiration, products, shop, shopping, stuff, wishlist, bargains, blogging, clothes, community, consumer, culture, daily, deals, ecommerce, food, funny, furniture, home, humor, humour, lifestyle, magazine, media, music, news, people, resources, social, store, toys, trends, weblog, weblogs, weird

Flat SVM^{Light} Tags:
blog, blogs, buy, clothing, fashion, gadget, gadgets, gifts, inspiration, reference, shop, shopping, stuff, wishlist, bargains, consumer, daily, deals, home, lifestyle, reviews, store, weblogs

Hierarchical SVM^{Light} Tags:
blog, blogs, buy, clothing, cool, fashion, fun, gadget, gadgets, gifts, ideas, inspiration, products, reference, shop, shopping, stuff, wishlist, bargains, clothes, consumer, culture, daily, deals, ecommerce, home, lifestyle, reviews, store, toys

Delicious or human annotated Tags:
blog, blogs, buy, clothing, cool, design, fashion, fun, gadget, gadgets, gifts, ideas, inspiration, interesting, online, products, reference, shop, shopping, stuff, web_2.0, wishlist

Figure 4.15: A comparison of the tags produced by the flat and hierarchical classifiers for the webpage http://mightygoods.com/ (see Figure 4.6). The tags in bold are the correct labels.

Flat Naïve Bayes tags:
cartoon, cartoons, comic, comics, daily, entertainment, fun, funny, geek, humor, humour, imported, webcomic, webcomics, art, comedy, community, cool, culture, ecommerce, english, feeds, magazine, maps, online, portal, research, satire, search, service, share, sharing, shop, shopping, social, store, work,

Hierarchical Naïve Bayes tags:
cartoon, cartoons, comic, comics, daily, entertainment, fun, funny, geek, humor, humour, imported, safari_export, webcomic, webcomics, comedy, community, map, maps, news, newspaper, newspapers, satire, shopping

Flat SVM^{Light} Tags:
cartoon, cartoons, comic, comics, fun, funny, geek, humor, humour, webcomic, webcomics

Hierarchical SVM^{Light} Tags:
cartoon, cartoons, comic, comics, fun, funny, geek, humor, humour, webcomic, webcomics, reference, tools, web

Delicious or human annotated Tags:
cartoon, cartoons, comic, comics, daily, entertainment, fun, funny, geek, humor, humour, imported, safari_export, webcomic, webcomics

Figure 4.16: A comparison of the tags produced by the flat and hierarchical classifiers for the webpage http://www.unitedmedia.com/comics/dilbert/ (see Figure 4.7). The tags in bold are the correct labels.

ROC Analysis

For the ROC analysis of the hierarchical classifiers, we ran all the test instances to all the leaf or tag classifiers[9] and use the result of the classification of these instances to build an ROC curve that is comparable to its flat counterpart. In the following table, we have the average (for each of the 567 tags) of area under the ROC curve (AU-ROC) from the hierarchical and flat classification of Naïve Bayes and SVM.

	Type	AU-ROC	AU-ROCCH
Naïve Bayes	Flat	0.7885	0.8095
	Hier.	0.7490	0.8139
SVM^{Light}	Flat	0.8606	0.8749
	Hier.	0.8645	0.8869

Table 4.6: The Area under the ROC curve comparison between hierarchical and flat classifications

Based on the result on the normal area under the ROC curve(AU-ROC), we can see that there is a small but insignificant improvement (0.0039) for SVM's tag or leaf classifications. For Naïve Bayes, however we saw a significant drop of 0.0395. Based on the two results, we believe that the divide and conquer nature of hierarchical learning seems to only be effective in improving an instance based and maximum margin classifiers. For probabilistic and Bayesian learning method such as Naïve Bayes, the strategy seems to disadvantageous because the reduction in training data seems to reduce the reliability of the prior and conditional probability of words , i.e. the model hypothesis. That being said, there is a more appropriate method of utilizing hierarchical structure on Naïve Bayes classifier that does not require the reduction of training data. In [31], Mccallum and Nigam proposes a better estimation of word probability in a hierarchical problem through the concept of shrinkage. For this thesis, we did not attempt this method.

Despite the negative result from comparing the AU-ROC of flat and hierarchical Naïve Bayes, we believe there is an alternative interpretation to the ROC curve that would allow us to see the advantage of our hierarchical classification. In the AU-ROC computation, we have treated for the hierarchical leaf classifiers the assumption that all test instances to be positively classified by their parent categories. In practice, this is not

[9]In practice, the hierarchical classification approach or the binary scoring method would use the result of parent category classification to filter a certain amount of the documents.

the case, because the parent category has the job of filtering certain number of instances. If we consider the incorporation of parent categories' classification output, we would have multiple ROC curves that represents the permutation of various threshold of both parent and leaf classifiers. Through this assumption, we can incorporate the concept of a convex hull[16] to make a single representation of these ROC curve permutations. An ROC convex hull is usually generated when there are multiple ROC curves that indicate superior performance over certain conditions, e.g. certain values of FP rate. In the case of hierarchical classification, we believe a threshold on the parent category (as opposed to the no threshold curve) would initially allow for a higher TP rate for lower FP rate but ultimately cost the curve a higher TP Rate by misclassifying positive instances.

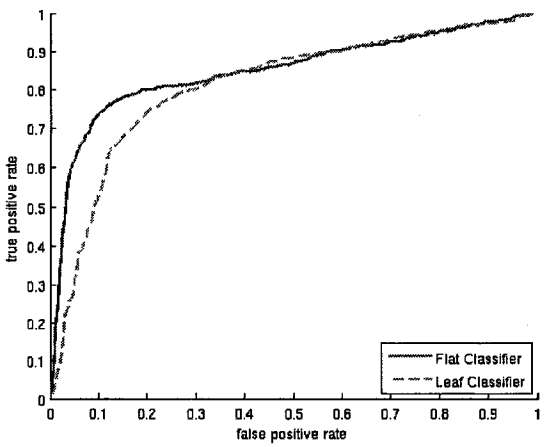
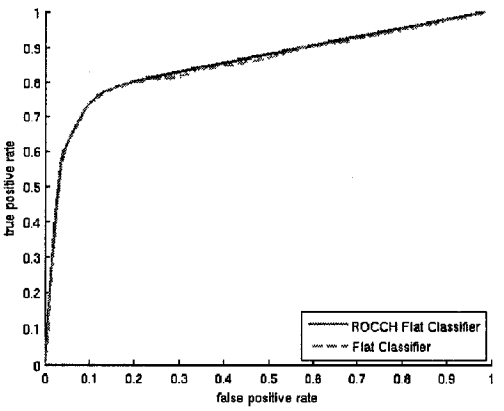
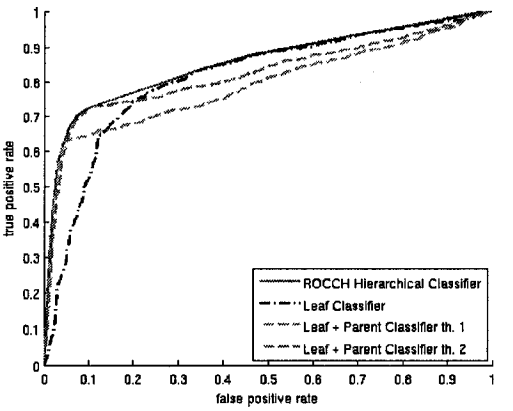


Figure 4.17: an example of an ROC Curve comparison between a hierarchical and flat Naïve Bayes classifier.

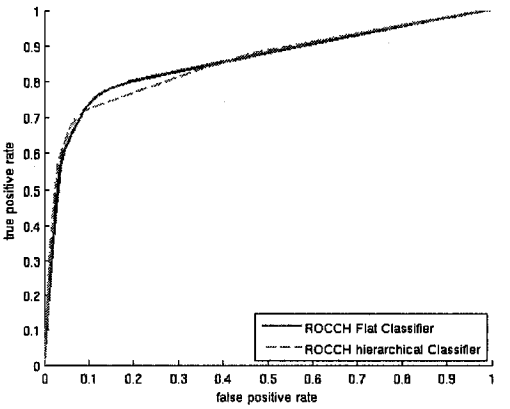
In Figure 4.17 and Figure 4.18, we illustrated this argument by presenting the transformations of the normal ROC comparison to the proposed ROC convex hull (ROCCH) comparison. First, in Figure 4.17, we have a normal ROC graph that shows the performance dominance of flat classifier over the hierarchical classifier (with no threshold from parent classifier). Figure 4.18(b), shows the convex hull of the hierarchical classifiers' ROC curves. The broken lined curves under the convex hull represents the classifiers performance if different thresholds were applied to the parent classifiers. As we can see from that figure, the curves that are the result of thresholding both parent and leaf classifiers have achieved (in comparison to the ROC curve of just the leaf classifier) a higher TP rate at lower FP rate but fall below the achievable TP rate at higher FP rate. To make the ROC convex hull comparison we also have created in Figure 4.18(a) the same



(a) The creation of ROC convex hull(ROCCH) from the flat classifier's ROC (see Figure4.17).



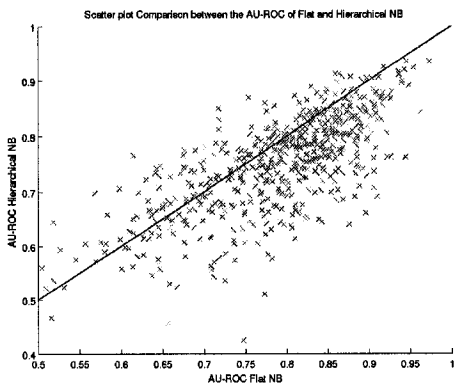
(b) The creation of ROC convex hull(ROCCH) from hierarchical classifier's output (see Figure4.17).



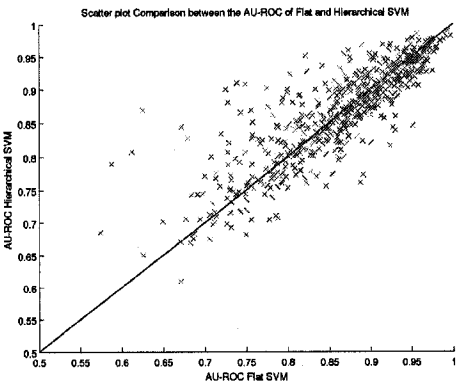
(c) The ROCCH Curve comparison between flat and hierarchical Naïve Bayes classifiers.

Figure 4.18: The derivation the ROC convex hull curves of the flat and hierarchical classifiers from Figure4.17) and the new ROCCH comparison.

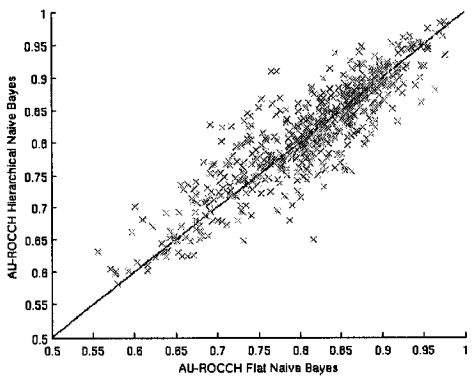
convex hull for the single classifier. Finally in Figure 4.18(c), we have a new comparison that shows that the convex hull comparison between the hierarchical and flat classifiers. Compared to the normal ROC curve comparison (Figure 4.17), The ROCCH comparison (Figure 4.18(c)) seems to put a more favorable opinion on the hierarchical classifier and this we believe reflects the way we have applied our hierarchical classification more accurately.



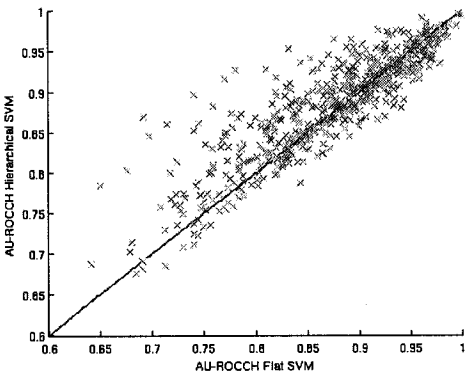
(a) The scatter plot comparison between the AU-ROC of hierarchical and flat Naïve Bayes.



(b) The scatter plot comparison between the AU-ROC of hierarchical and flat SVM.



(c) The scatter plot comparison between the AU-ROCCH of hierarchical and flat Naïve Bayes.



(d) The scatter plot comparison between the AU-ROCCH of hierarchical and flat SVM.

Figure 4.19: The AUC comparison(for each tag) of hierarchical (Y-axis) and flat classifiers (X-Axis).

Going back to Table 4.6, we have averaged the area under ROCCH (AU-ROCCH) and shows for both SVM and Naïve Bayes classifiers that hierarchical classifier does offer

a small improvement (0.0044 for Naïve Bayes and 0.012 for SVM). This is important because it allows, in the case Naïve Bayes classifier,us to observed an improvement that is also found in the F-Measure computed earlier. In terms of statistical significance (See

	AU-ROCCH	
	Wilcoxon	t-test
Naïve $\text{Bayes}_{Flat-Hier.}$	$\sim$	$\sim$
$\text{SVM}_{Flat-Hier.}$	$\sim$	$<$

Table 4.7: The result of the statistical significance test for Area under the ROC curve comparison.

Table 4.7),with the exception of the t-test result for SVM, we find that the result of averaging AU-ROCCH to not be statistically significant.

Lastly, in Figure 4.19, we have illustrated the scatter plot comparison (flat classifications as the X-axis and hierarchical classifications as) for both AU-ROC and AU-ROCCH. As we can see from the difference in the points' location from Figure 4.19(a) and Figure 4.19(c), the AU-ROCCH plots seems to provide a more positive conclussion on hierarchical Naïve Bayes classifiers.

Classifiers' Efficiency

For the comparison of efficiency in flat and hierarchical classifications, we have decided to time both the training and the classification process of the experiments. For all the experiment that are mentioned in this thesis, we ran them through a Computer with 2.8 GHz Pentium 4 CPU and 1GB of RAM. The result of the computation is as follows:

As we can see from Table 4.8, both hierarchical Naïve Bayes and SVM observed a much improved efficiency than their flat counterpart . For the training and classification of Naïve Bayes classifiers, we see a reduction of 2,343 seconds (68.9% of total time in flat categorization). It should be noted, that the total training and classification time of Naïve Bayes are not separable because the implementation of Libbow includes the computation of prior and conditional probabilities in the classification call for a set of data. It should be noted that the pre-processing steps, such as tokenization and indexing, are not included in the classification process. For the training and classification time of SVM classifiers, we see a reduction of 10,612 second (80.6% of total training time) and 1,325 seconds (64.6% of original classification time), respectively. Based on these results,

Naïve Bayes

Type	Training and Classification Time
Flat	3,402
Hierarchical	1,059

SVM

Type	Training Time	Classification Time
Flat	13,160	2,046
Hierarchical	2,548	721

Table 4.8: The total training and classification time for both flat and hierarchical Naïve Bayes and SVM categorizations (in seconds).

we have the following observations on the efficiency of the two classifiers.

First, the total training complexity of SVM classifiers can be formulated as $O(|C| \times |D|^k)$ [59]. In that formula, the symbol $|C|$ represents the number of categories, the symbol $|D|$ represents the number of documents, and the symbol k represents a constant number between 1.2 and 1.5. Since that time complexity seems to depend on the number of categories, $|C|$, it seems contradictory that the total training time for hierarchical SVM classification to be a lot smaller than its flat counterpart because for the hierarchical categorization we have 75 more categories to train. We can, however, explain this contradiction by considering the fact that the classification of the leaf categories, i.e. the 567 categories in hierarchical categorization, were done on smaller set of documents (see Section 3.3.2). If we consider this fact, we would then need to differentiate the complexity of training the classifiers for the top categories or clusters classification and the complexity of training the classifiers for the leafs or tags classification.

For hierarchical SVM, the total time complexity of training a hierarchical model can be formulated as $O((|C_{cl}| \times |D_{total}|^k) + (|C_{tags}| \times |D_{part}|^k))$. The first part of that big-Oh notation represents the complexity of training the parent categories, C_{cl} with the whole training set, $|D_{total}|$ while the second part represents the time complexity of training the leaf categories, C_{tags}, with the smaller and partitioned training set, D_{part}. Using the same notation, we can also reformulate the training complexity of flat SVM as $O((|C_{tags}| \times |D_{total}|^k)$, i.e. the complexity of training all tags using all the training data. With these definitions in place ,we believe the time complexity of training a hierarchical model is a faster than its flat counterpart because the added complexity of training the parent clusters, i.e. $O(|C_{cl}| \times |D_{total}|^k)$, is actually smaller than the reduction of

training set size ($|D_{total}| > |D_{part}|$) for the classification the leafs or tags classification, i.e. $O((|C_{tags}| \times |D_{total}|^k) > O(|C_{tags}| \times |D_{part}|^k)$[10].

For Naïve Bayes classifiers the training time are generally formulated as $O(|C| \times |F|))$, i.e. the pre-computation of the prior probabilities of positive and negative classes and the conditional probabilities for every feature selected (the size of $|F|$) that we have to perform for every single binary classification of a category (C). Since this formulation are not constrained by the size of the training set, it is likely that the reduction in training and classification time of Naïve Bayes classifiers are not the result of the reduction in training time alone (in fact, we should see increase in training time). Therefore instead of just looking at the reduction in training time, we also look at the reduction in classification time.

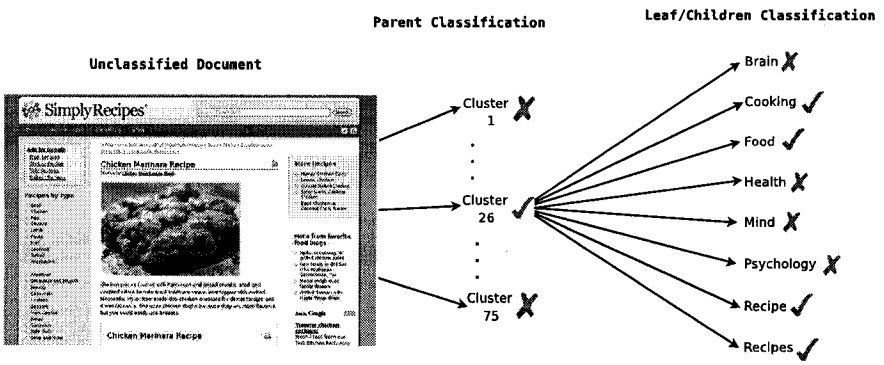


Figure 4.20: A visualization of the hierarchical classification process for a recipe document. During the parent classification, the document is only classified into the parent category "Cluster 26". The leaf classification would therefore omits any tags whose parents are not "Cluster 26".

For the classification time of both Hierarchical SVM and Naïve Bayes classification, we can see there are a lot less attempts on categorizing leaf categories because of the decision made on its parent categories (the binary scoring method). As mentioned in Section4.3.3, a document that is classified to one parent category would only be classified to the child or leaf categories of the one parent category. For instance if a document is classified to the parent topic of ***Food and Health*** or Cluster 26 in Appendix A.2 , we can then try to further classify this document to the categories ***Cooking***, ***Recipe***, etc. If it is otherwise, the document would not be further classified to the child categories. Figure

[10]This may be a motivation for finding a hierarchical tree that produce the optimum cardinality

4.20, for instance, illustrates the classification attempt of an instance that is classified to only 1 parent category. We believe based on the result in Table4.8 that the occurrences of such case lead to the reduction in classification time of the both hierarchical models.

Chapter 5

Conclusion

In this thesis, we have showed hierarchical categorization as a potential improvement to the automatic text categorization of collaboratively tagged data. Despite the fact that collaborative tagging system strives on pooling the collective intelligence of Internet users, i.e. human annotations, we believe that the potential of non-human annotation would offer benefit in cases where inconsistencies and lack of coverage are rampant, i.e. human limitations. From our automatic text categorization experiment, more importantly the hierarchical categorization experiment, we wished to show that it is feasible for non-human annotations to address the shortcomings of current collaborative tagging systems.

5.1 Summary

In Chapter 4 and 3, we have proposed an automatic text categorization experiment from a collaboratively tagged dataset. The following are the findings of that experiment.

Delicious is a Good Source of Text Categorization Dataset

As our choice of collaborative tagging system, we have selected a popular social bookmarking system called Delicious . Unlike other online repositories, Delicious is also popular among collaborative tagging researchers because it uses a very intuitive RSS feeds to provide full access to its database. This, in return, has allowed us to create a text categorization dataset with ease.

Some characteristics of the said dataset include multi-facetted labels and collaborative annotations. The presence of these characteristics seems to address some of the issues that were raised by web genre studies. That is to say the participation of many users

and the ability to facilitate these interactions have afforded us to build a more realistic categorization dataset than previous genre categorization datasets. Other characteristics such as the domain coverage and the exponentially decreasing distribution of the tag space draw comparisons to web directories. Ultimately, the similarities to web directories serves as motivation to look into hierarchical categorization, a common approach to automatically categorize web directories.

Support Vector Machine is a Better Classifier than Naïve Bayes

As part of the experiment, we have decided to compare the performance of SVM and Naïve Bayes classifiers. In our overall assessment of the performance, we conclude that SVM is the better classifier because it manage to outperform Naïve Bayes in both Micro and Macro Average F-Measures and area under the ROC curve (AU-ROC) average. It should be noted that despite the improvement both classifiers have in hierarchical categorizations, SVM still outperforms Naïve Bayes.

One important observation we have about the Naïve Bayes classifiers is the fact that they produced better sensitivities, i.e. larger Recall values, than the SVM classifiers. This technically creates a potential trade-off between the ability of recognizing all the documents that belong to a target concept and the ability to make precise annotations for a target concept. Naïve Bayes thrives on the former while SVM thrives on the latter. Nonetheless, we believe that the ability to make precise annotations as a more redeeming quality because we believe the added complexity of indexing and browsing among false annotations would make any retrieval system unusable.

Automatically Generated Hierarchy Improves to Automatic Text Categorization

In what we believe is an important improvement to the categorization of collaborative tagging system, we have proposed a unique and counterintuitive approach of hierarchical categorization. We believe that this approach is counterintuitive because as a strength, collaborative tagging relies on being free of structures and hierarchy. The approach is also unique because in the absence of a hierarchical structure we automatically generate a hierarchy (a set of clusters) that allows for a more efficient categorization.

In order to clarify our stance in hierarchy, we still support the argument that a rigid hierarchy may not be suitable for retrieval and browsing purposes, i.e. the collaborative tagging argument. However, in the case of automatic text categorization (a process that

are independent of the retrieval and browsing activities) we believe that hierarchical structure may offered a better and more efficient method of classifying web documents.

Based on the Micro Averaged F-Measure performance, we saw a small improvements for both hierarchical Naïve Bayes and SVM classifiers (8% and 10% relative improvement respectively). We also devised an ROC analysis that reflects the improved area under the curve, i.e. the capability of ranking positive examples over negative ones, of our hierarchical classifier. We believe by combining the ROC curves from various parent and leaf category thresholds, we have captured a more accurate depiction of the binary scoring hierarchical classification.

Finally, the most obvious benefit of adopting the hierarchical categorization is the reduced training and testing time of a hierarchical system. For efficiency, we observed not only a significant reduction in training time (67.8% for Naïve Bayes and 80% for SVM) but also a significant reduction classification time (64.6% for Naïve Bayes and 68.9% for SVM).

5.2 Future Works

In terms of growth, the result of our automatic text categorization on collaborative tagging experiment suggests several directions that could be implemented in future research. The following are these suggestions.

Improving The Performance of Hierarchical Text Categorization

First, we would like to note that our use of the graph partitioning algorithm, i.e. pmetis[22], is quite rudimentary compared to other methods of building a hierarchy. In [2], Begelman et. al adapts a more advanced algorithm that relies on spectral bisection and modularity function[54] as their clustering method. This method differs from our graph partitioning method because it partitions or clusters concepts without knowing the number of clusters beforehand. Similarly, Heymann and Garcia-Molina [18] also proposed an non supervised algorithm (in terms of the number of clusters, etc.) that can generates a hierarchical tree (deeper than 2 levels). Since the quality of clusters and the shape (cardinality and depth) of a hierarchy effect the performance of a hierarchical text categorization, we believe for future work that there are classification improvements to be gained from deploying a better hierarchical structure.

Secondly, in our experiment, we have only considered a straight forward hierarchical

approach that relies on dividing the learning and classification process into smaller categorization problems (based on the parent categories). In the machine learning literature, this approach only represents one of the many ways of facilitating hierarchical learning. In [31], for instance, Shrinkage estimation from hierarchical structures was used to improve the conditional probabilities of a Bayesian classifier. Meanwhile, Sun and Lim [49] have also proposed a different method of classifying hierarchical structure by considering non leaf category as a stopping condition. They also remarked on more appropriate measurements such as hierarchical distances between misclassified categories. On the whole, we acknowledge that there are other methods to hierarchical classification. Therefore, despite the improvement of our current hierarchical classification, we believe it is important to evaluate each of these methods and find a potentially better preforming hierarchical method.

Incorporating Non-Human Annotations to Collaborative Tagging System

Given the guarantee of document coverage, it is important to consider the most appropriate way of intergrating non-human annotations, i.e. an automatic tagging system. In [55], for instance, Xu et al. highlighted the need for an automatic tagging for recommending tags. By providing a set of recommended and appropriate tags, they believe that users would avoid or would be less exposed to the hassles of annotating a new webpage of interest, i.e. mispellings and limited vocabulary.

For us, the most obvious benefit of an extended coverage is the potential for personalized tagging system. For social bookmarking system such as Delicious, it is easy to see that each user's interest and sense of document organization. With automatic tagging we can perhaps train a text classifier on the documents that have been bookmarked by the user and apply the learned hypothesis to other potential documents. Unlike the popular tags that we have in this thesis, the personalized dataset would apply only to the corresponding user. Therefore such a dataset would be smaller and in need of better document coverage, i.e. an automated tagging. If we decided to explore this option, we need to consider a machine learning approach that is able to deliver different classifiers to millions of users.

Bibliography

[1] C. Apte, F. Damerau, and S.M. Weiss. Text mining with decision trees and decision rules.

[2] Grigory Begelman, Philipp Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, 2006.

[3] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, New York, NY, USA, 1998. ACM Press.

[4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992.

[5] Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM Press.

[6] Ciro Cattuto, Vittorio Loreto, and Luciano Pietronero. Collaborative tagging and semiotic dynamics. *PNAS*, 104:1461, 2007.

[7] Nitesh V. Chawla, Kevin W. Bowyer, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[9] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[10] K. Crowston. Reproduced and emergent genres of communication on the world-wide web. In *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*, volume 6, pages 30–39, January 1997.

[11] K. Crowston and B. Kwasnik. A framework for creating a facetted classification for genres: Addressing issues of multidimensionality, 2004.

[12] K. Crowston and B. Kwasnik. Challenges in creating a taxonomy for genres of digital documents. 2005.

[13] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Res.*, 7:1–30, 2006.

[14] Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. The form is the substance: classification of genres in text. In *Proceedings of the workshop on Human Language Technology and Knowledge Management*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[15] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 256–263, Athens, Greece, July 2000.

[16] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[17] Scott Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, April 2006.

[18] Paul Heymann and Hector Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Computer Science Department, April 2006.

[19] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Scholkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[20] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th*

International Conference on Machine Learning, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[21] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[22] George Karypis and Vipin Kumar. *MeTis: Unstrctured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*, 1995.

[23] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical report, 1998.

[24] Alistair Kennedy and Michael Shepherd. Automatic identification of home pages on the web. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4*, page 99.3, Washington, DC, USA, 2005. IEEE Computer Society.

[25] Brett Kessler, Geoffrey Numberg, and Hinrich Schutze. Automatic detection of text genre. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 32–38, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

[26] Pranam Kolari, Akshay Java, Tim Finin, Tim Oates, and Anupam Joshi. Detecting Spam Blogs: A Machine Learning Approach. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006).* Computer Science and Electrical Engineering, University of Maryland, Baltimore County, July 2006.

[27] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[28] Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43, 2005.

[29] A. McCallum. Multi-label text classification with a mixture model trained by em, 1999.

[30] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[31] Andrew K. McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In Jude W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[32] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[33] Sven Meyer Zu Eissen and Benno Stein. Genre classification of web pages. In Susanne Biundo, Thom Frühwirth, and Günther Palm, editors, *Proceedings of KI-04, 27th German Conference on Artificial Intelligence*, Ulm, DE, 2004. Published in the "Lecture Notes in Computer Science" series, number 3238.

[34] Peter Mika. Ontologies are us: A unified model of social networks and semantics. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *The Semantic Web - ISWC 2005, Proceedings of the 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10*, volume 3729 of *Lecture Notes in Computer Science*, pages 522–536, Heidelberg, 2005. Springer.

[35] Thomas M. Mitchell. *Machine Learning.* McGraw-Hill Higher Education, 1997.

[36] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323, 2005.

[37] Tsutomu Ohkura, Yoji Kiyota, and Hiroshi Nakagawa. Browsing system for weblog articles based on automated folksonomy. In *WWW2006 Workshop on the Weblogging Ecosystem. Edinburgh, Scotland*, 2006.

[38] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[39] Emanuele Quintarelli. Folksonomies: power to the people, June 2005. `http://www-dimat.unipv.it/biblio/isko/doc/folksonomies.htm`.

[40] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers, 2003.

[41] J. Rocchio. *The SMART Retrieval System—Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[42] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[43] Joshua Schachter. http://del.icio.us.

[44] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[45] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[46] M. Shepherd and C. Watters. The evolution of cybergenres. In *HICSS '98: Proceedings of the Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, volume 2, pages 97–109, January 1998.

[47] Clay Shirky. *Clay Shirky's Writings About the Internet*, chapter Ontology is Overrated: Categories, Links, and Tags. 2005.

[48] Efstathios Stamatatos, George Kokkinakis, and Nikos Fakotakis. Automatic text categorization in terms of genre and author. *Comput. Linguist.*, 26(4):471–495, 2000.

[49] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *ICDM*, pages 521–528, 2001.

[50] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[51] Thomas Vanderwall. Explaining and showing broad and narrow folksonomies :: Personal infocloud, 2005.

[52] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[53] K. Weick, K. Sutcliffe, and D. Obstfeld. Organizing and the process of sensemaking. *Organizational Science*, 16(4):409–421, 2005.

[54] Scott White and Padhraic Smyth. A spectral clustering approach to finding communities in graphs, 2005.

[55] Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*, 2006.

[56] Yiming Yang. A study of thresholding strategies for text categorization. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–145, New York, NY, USA, 2001. ACM Press.

[57] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM.

[58] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

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

[60] George K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley (Reading MA), 1949.

Appendix A

List of Categories and Clusters

A.1 List of Tags and Their Distribution (Sorted by Frequency)

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
1	reference	3719	2	web	3437
3	tools	3259	4	software	2913
5	design	2630	6	blog	2471
7	programming	2153	8	free	2121
9	howto	2054	10	development	2030
11	internet	1981	12	cool	1892
13	webdesign	1890	14	web_2.0	1781
15	technology	1780	16	tutorial	1776
17	tech	1631	18	webdev	1556
19	blogs	1548	20	tips	1477
21	resources	1453	22	computer	1384
23	fun	1343	24	opensource	1283
25	tool	1280	26	news	1241
27	toread	1233	28	tutorials	1223
29	online	1177	30	imported	1169
31	art	1166	32	media	1102
33	community	1074	34	freeware	1043
35	business	1037	36	interesting	1004
37	download	1001	38	search	999
39	useful	998	40	code	991
41	geek	982	42	css	968

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
43	culture	957	44	inspiration	951
45	article	909	46	resource	894
47	ajax	886	48	graphics	880
49	research	879	50	html	850
51	linux	834	52	daily	833
53	utilities	819	54	education	810
55	windows	804	56	social	799
57	articles	775	58	work	752
59	computers	749	60	video	726
61	javascript	682	62	guide	675
63	information	671	64	hacks	669
65	mac	659	66	productivity	647
67	images	644	68	links	641
69	lifehacks	626	70	photography	624
71	music	608	72	blogging	592
73	humor	592	74	downloads	582
75	funny	580	76	library	573
77	learning	571	78	entertainment	555
79	shopping	547	80	photo	544
81	list	538	82	books	522
83	science	522	84	photos	515
85	apple	509	86	osx	508
87	dev	496	88	network	494
89	writing	492	90	ideas	490
91	service	478	92	hack	476
93	apps	475	94	database	464
95	diy	463	96	flash	451
97	security	449	98	marketing	442
99	audio	441	100	networking	432
101	games	430	102	xhtml	427
103	php	426	104	hardware	420
105	safariexport	420	106	usability	417
107	java	413	108	rss	412
109	politics	411	110	website	411
111	language	402	112	portal	399
113	collaboration	398	114	sysadmin	394
115	documentation	390	116	coding	380
117	gallery	371	118	application	362
119	image	359	120	google	358
121	reviews	358	122	management	356

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
123	magazine	354	124	standards	354
125	browser	353	126	lists	343
127	directory	338	128	socialsoftware	331
129	book	328	130	history	328
131	ui	324	132	utility	323
133	applications	322	134	mp3	319
135	layout	313	136	advice	312
137	graphic	310	138	movies	307
139	macosx	304	140	xml	304
141	unix	301	142	interface	297
143	todo	294	144	travel	290
145	hacking	286	146	creativity	283
147	wiki	283	148	reading	281
149	server	281	150	personal	280
151	gtd	279	152	photoshop	279
153	sharing	278	154	organization	276
155	pictures	276	156	illustration	264
157	tv	262	158	visualization	261
159	webapp	261	160	api	260
161	life	260	162	archive	258
163	game	254	164	dhtml	251
165	gadgets	250	166	advertising	249
167	aggregator	249	168	framework	248
169	humour	247	170	architecture	246
171	digital	246	172	english	244
173	ruby	244	174	webdevelopment	244
175	firefox	243	176	2.0	242
177	forum	242	178	shop	241
179	tagging	241	180	services	238
181	computing	236	182	money	233
183	home	230	184	rails	229
185	webtools	228	186	microsoft	227
187	maps	226	188	accessibility	224
189	hosting	224	190	podcast	224
191	tags	223	192	videos	222
193	info	219	194	bookmarks	217
195	review	214	196	gui	209
197	film	206	198	style	206
199	pc	205	200	macintosh	204
201	trends	204	202	philosophy	203

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
203	templates	203	204	diseo	200
205	startup	200	206	projects	199
207	creative	198	208	people	198
209	communication	197	210	os	197
211	animation	195	212	webstandards	195
213	desktop	194	214	generator	193
215	plugin	193	216	literature	192
217	it	191	218	cms	190
219	mobile	190	220	source	190
221	weblog	189	222	misc	188
223	activism	186	224	scripting	185
225	tricks	185	226	gaming	184
227	rubyonrails	184	228	world	184
229	finance	183	230	typography	182
231	storage	180	232	ebooks	179
233	help	179	234	open	179
235	statistics	179	236	project	178
237	map	177	238	lifehack	175
239	wordpress	175	240	webapps	174
241	python	172	242	health	171
243	psychology	171	244	webservices	170
245	folksonomy	169	246	seo	168
247	career	167	248	ipod	167
249	economics	166	250	movie	166
251	searchengine	166	252	scripts	165
253	portfolio	164	254	delicious	163
255	c	162	256	multimedia	161
257	weird	160	258	text	159
259	email	158	260	flickr	158
261	p2p	156	262	editor	154
263	webdesigns	153	264	electronics	152
265	school	152	266	backup	151
267	food	151	268	languages	151
269	net	151	270	color	149
271	analysis	148	272	webservice	147
273	data	146	274	innovation	145
275	mysql	145	276	socialnetworking	145
277	math	143	278	teaching	143
279	ebook	139	280	entrepreneurship	139
281	mashup	137	282	radio	137

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
283	streaming	137	284	js	136
285	plugins	136	286	xp	136
287	magazines	135	288	comparison	134
289	future	134	290	knowledge	133
291	privacy	133	292	read	133
293	webmaster	131	294	publishing	130
295	collection	129	296	academic	128
297	office	128	298	3d	127
299	comics	126	300	examples	126
301	environment	125	302	fashion	125
303	interactive	125	304	share	125
305	theory	125	306	stock	123
307	admin	122	308	icons	122
309	television	122	310	apache	121
311	developer	121	312	dom	121
313	extension	121	314	oss	121
315	society	121	316	toys	121
317	content	120	318	engine	118
319	pdf	118	320	fonts	117
321	store	117	322	ubuntu	117
323	browsers	116	324	jobs	116
325	podcasting	116	326	uk	116
327	script	115	328	tag	115
329	government	114	330	mapping	114
331	journalism	113	332	dictionary	112
333	drawing	112	334	usa	112
335	crafts	111	336	forums	111
337	kids	111	338	patterns	111
339	bookmark	109	340	make	109
341	system	109	342	testing	109
343	dvd	108	344	feeds	107
345	template	107	346	test	106
347	weblogs	106	348	cheatsheet	104
349	extensions	104	350	mozilla	104
351	performance	104	352	app	103
353	file	103	354	camera	102
355	fotografia	102	356	site	100
357	alternative	99	358	learn	99
359	phone	99	360	study	99
361	comic	98	362	gifts	98

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
363	videogames	98	364	www	98
365	gadget	97	366	foto	95
367	fotos	95	368	web_2	95
369	entrepreneur	94	370	podcasts	94
371	print	94	372	files	93
373	calendar	92	374	stuff	92
375	lifestyle	91	376	mail	91
377	comedy	90	378	international	90
379	training	90	380	optimization	89
381	random	89	382	artist	88
383	libraries	88	384	cooking	87
385	deals	87	386	presentation	87
387	physics	86	388	guides	85
389	recursos	85	390	awesome	84
391	best	84	392	craft	84
393	websites	84	394	bittorrent	83
395	feed	83	396	icon	83
397	job	83	398	themes	83
399	font	82	400	recipes	82
401	clothing	81	402	geography	81
403	opinion	81	404	paper	81
405	support	81	406	wireless	81
407	artists	80	408	engineering	80
409	japan	80	410	yahoo	80
411	bookmarking	79	412	ia	79
413	mathematics	79	414	sql	79
415	time	79	416	wikipedia	79
417	retro	78	418	ecommerce	77
419	itunes	77	420	sound	77
421	wallpaper	76	422	stats	75
423	strategy	75	424	w3c	75
425	adobe	74	426	filesharing	74
427	indie	74	428	designer	73
429	elearning	73	430	encyclopedia	73
431	googlemaps	73	432	http	73
433	newspaper	73	434	buy	72
435	essay	72	436	picture	72
437	visual	72	438	youtube	72
439	algorithms	71	440	lifehacker	71
441	planning	71	442	religion	71

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
443	widgets	71	444	actionscript	70
445	dotnet	70	446	editing	70
447	events	70	448	law	70
449	cheap	69	450	torrent	69
451	cartoon	68	452	green	68
453	local	68	454	manual	68
455	pics	68	456	political	68
457	torrents	68	458	words	68
459	graphicdesign	67	460	nyc	67
461	projectmanagement	67	462	brain	66
463	branding	66	464	financial	66
465	geo	66	466	administration	65
467	conversion	65	468	creativecommons	65
469	install	65	470	interaction	65
471	investing	65	472	mind	65
473	musica	65	474	prototype	65
475	upload	65	476	utilidades	65
477	japanese	64	478	space	64
479	syndication	64	480	tracking	64
481	digg	63	482	perl	63
483	theme	63	484	wishlist	63
485	consumer	62	486	shell	62
487	socialnetwork	62	488	type	62
489	urban	62	490	copyright	61
491	example	61	492	house	61
493	portable	61	494	products	61
495	j2ee	60	496	metadata	60
497	navigation	60	498	ads	59
499	agile	59	500	ai	59
501	forms	59	502	hci	59
503	linguistics	59	504	sex	59
505	bargains	58	506	collaborative	58
507	firefoxtoolbar	58	508	freelance	58
509	networks	58	510	beta	57
511	blogger	57	512	distro	57
513	furniture	57	514	programacion	57
515	programs	57	516	aspnet	56
517	clothes	56	518	company	56
519	cs	56	520	effects	56
521	gis	56	522	identity	56

ID.	Tag Name	No. of Documents	ID.	Tag Name	No. of Documents
523	ie	56	524	journal	56
525	recipe	56	526	wifi	56
527	gratis	55	528	illustrator	55
529	sns	55	530	webcomic	55
531	widget	55	532	xmlhttprequest	55
533	essays	54	534	faq	54
535	german	54	536	gnu	54
537	juegos	54	538	showcase	54
539	traffic	54	540	convert	53
541	converter	53	542	nature	53
543	noticias	53	544	open_source	53
545	ror	53	546	tecnologia	53
547	webcomics	53	548	biology	52
549	chat	52	550	commentary	52
551	virtual	52	552	cd	51
553	college	51	554	debian	51
555	newspapers	51	556	pda	51
557	weekly	51	558	arte	50
559	cartoons	50	560	cc	50
561	colour	50	562	fiction	50
563	lessons	50	564	myspace	50
565	satire	50	566	sustainability	50
567	utils	50			

A.2 List of Clusters and Their Tags

Cluster ID.	Tags
1	camera, foto, fotografia, fotos, pics, picture, recursos, stock
2	icon, icons, sns, socialnetwork, space, tecnologia
3	biology, encyclopedia, engineering, events, feed, feeds, syndication
4	ads, conversion, convert, converter, identity, nature, test, testing
5	cartoon, cartoons, comic, comics, uk, webcomic, webcomics
6	comedy, cs, humour, random, satire, sex, wallpaper, weird
7	editing, editor, font, fonts, graphicdesign, text, type, typography
8	arte, artist, artists, designer, illustration, portfolio, showcase

Cluster ID.	Tags
9	3d, animation, drawing, illustrator, math, mathematics, physics, print
10	film,movie, movies, multimedia, streaming, videos, youtube
11	bittorrent, dvd, p2p, television, torrent, torrents, tracking, tv
12	adobe, branding, copyright, itunes, musica, myspace, sound
13	audio, indie, ipod, mp3, podcast, podcasting, podcasts, radio
14	aspnet, c, dotnet, microsoft, net, system, xp
15	admin, administration, app, debian, distro, macintosh, os, ubuntu
16	backup, file, files, filesharing, open_source, upload, weblogs
17	dictionary, german, languages, linguistics, patterns, python, words
18	academic, college, elearning, learn, school, strategy, study, teaching, training
19	clothes, clothing, fashion, gifts, shop, shopping, store, wishlist
20	bargains, buy, cheap, comparison, consumer, deals, ecommerce, products
21	electronics, gadget, gadgets, lifestyle, networks, stuff, toys
22	awesome, forums, furniture, home, house, ia, interactive
23	browsers, extension, extensions, ie, mozilla, plugin, plugins
24	ai, essay, essays, noticias, publishing, theme, themes, wordpress
25	colour, gnu, mobile, pda, phone, wifi, wireless
26	brain, cooking, food, health, mind, psychology, recipe, recipes
27	algorithms, craft, crafts, make, paper, programs, projectmanagement, support
28	geo, gis, googlemaps, map, mapping, maps, mashup
29	creativecommons, data, geography, international, local, travel, world
30	ebook, ebooks, fiction, literature, pdf, visual, yahoo
31	analysis, engine, optimization, performance, searchengine, seo, statistics, stats, traffic
32	game, gaming, juegos, nyc, retro, urban, videogames
33	government, history, magazines, philosophy, religion, theory, usa
34	activism, alternative, economics, environment, green, law, politics, society, sustainability
35	entrepreneur, entrepreneurship, finance, financial, future, innovation, investing, money
36	chat, commentary, journalism, newspaper, newspapers, opinion, political
37	career, freelance, job, jobs, journal, kids, weekly, wikipedia
38	actionscript, blogger, cc, company, http, lessons, manual
39	best, content, gratis, guides, knowledge, libraries, lifehacker, time
40	digg, j2ee, japan, japanese, planning, utilidades, virtual
41	beta, cd, collaborative, email, firefoxtoolbar, mail, metadata, perl
42	apache, hosting, mysql, server, sql, storage, utils
43	forum, open, oss, portable, project, projects, review, source
44	cheatsheet, collection, desktop, help, install, office, privacy

Cluster ID.	Tags
45	apple, computing, hardware, it, mac, macosx, osx, pc
46	hack, hacking, linux, security, shell, sysadmin, unix, windows
47	architecture, color, faq, photoshop, tricks, widget, widgets
48	forms, gui, hci, interaction, navigation, presentation, site, websites
49	diseo, effects, example, examples, ui, www
50	accessibility, css, html, interface, standards, usability, w3c, webstandards, xhtml
51	generator, layout, prototype, style, template, templates, web_design, webmaster
52	agile, book, calendar, cms, firefox, language, programacion
53	api, framework, library, opensource, rails, ror, ruby, rubyonrails
54	browser, dom, js, script, scripting, scripts, xmlhttprequest
55	coding, dev, developer, dhtml, documentation, java, webdevelopment, xml
56	code, development, javascript, php, programming, tutorial, tutorials, webdev
57	archive, database, directory, download, downloads, freeware, search
58	advertising, communication, music, people, portal, trends, video, weblog
59	creative, digital, flash, graphic, graphics, visualization, website
60	flickr, gallery, image, images, photo, photography, photos, pictures
61	aggregator, blogging, rss, share, social, socialnetworking, socialsoftware
62	bookmark, bookmarking, bookmarks, delicious, folksonomy, tag, tagging, tags
63	collaboration, google, network, networking, sharing, startup, wiki
64	2.0, services, web_2, webapp, webapps, webservice, webservices, webtools
65	ajax, application, applications, apps, service, tool, utilities, utility
66	daily, links, magazine, media, news, reviews, safari_export
67	art, culture, entertainment, fun, funny, games, humor, interesting
68	community, computer, computers, geek, imported, inspiration, resource, useful
69	blog, blogs, design, reference, resources, web, web_2.0 , webdesign
70	cool, free, internet, online, software, tech, technology, tools
71	books, education, english, learning, read, reading, science, writing
72	creativity, info, information, marketing, misc, research
73	diy, hacks, life, lifehack, personal, todo
74	advice, business, gtd, ideas, lifehacks, management, organization, productivity, work
75	article, articles, guide, howto, list, lists, tips, toread