

# UC San Diego

## UC San Diego Electronic Theses and Dissertations

### Title

MaLCoN : Machine Learning analysis on Copyright Notices

### Permalink

<https://escholarship.org/uc/item/9vm6903r>

### Author

Kothari, Mohit Rajkumar

### Publication Date

2015

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

MaLCoN: Machine Learning analysis on Copyright Notices

A thesis submitted in partial satisfaction of the  
requirements for the degree of Master of Science

in

Computer Science

by

Mohit Rajkumar Kothari

Committee in charge:

Professor Lawrence K. Saul, Chair  
Professor Geoffrey M. Voelker  
Professor Stefan R. Savage

2015

Copyright

Mohit Rajkumar Kothari, 2015

All rights reserved.

The Thesis of Mohit Rajkumar Kothari is approved and is acceptable in quality and form for publication on microfilm and electronically:

---

---

---

Chair

University of California, San Diego

2015

## DEDICATION

*To my parents.*

## EPIGRAPH

The brain is a wonderful organ;  
It starts working the moment you get up in the morning  
and does not stop until you get into the office.

*Robert Frost*

## TABLE OF CONTENTS

Signature Page .....	iii
Dedication .....	iv
Epigraph .....	v
Table of Contents .....	vi
List of Figures .....	viii
List of Tables .....	x
Acknowledgements .....	xi
Abstract of the Thesis .....	xii
Introduction .....	1
Chapter 1 A Study in DMCA .....	3
1.1 Title II .....	5
1.1.1 Who is a <i>Service Provider</i> ? .....	5
1.1.2 How to become <i>Eligible for Limitation</i> ? .....	6
1.1.3 Limitations and Mechanisms .....	6
1.1.4 Counter-notices .....	8
Chapter 2 Background .....	11
2.1 Topic modeling .....	12
2.1.1 Bag-of-words representation .....	12
2.1.2 Multinomial distribution .....	14
2.1.3 Dirichlet distribution .....	15
2.1.4 Latent Dirichlet allocation .....	15
2.1.5 Collapsed Gibbs sampling .....	20
2.2 Clustering .....	22
2.2.1 K-means clustering .....	23
2.3 Classification .....	27
2.3.1 Support vector machine .....	28
2.3.2 Random forests .....	31
Chapter 3 Why so Cold? .....	35
3.1 Chilling Effects .....	35
3.2 Notices .....	36
3.2.1 Notice structure .....	37

3.3	Infringing URLs .....	38
3.4	SOPA/PIPA effect .....	39
3.5	Copyright holders .....	40
Chapter 4	Topical Paradise .....	42
4.1	Dataset .....	42
4.2	Pre-processing .....	43
4.2.1	Restricting to 200 OK .....	43
4.2.2	Curating “stopword” list .....	44
4.2.3	Cutting the tail .....	45
4.2.4	Document length distribution .....	46
4.3	Results .....	47
4.4	LDA on notices .....	48
Chapter 5	Clusters, Clusters, Everywhere .....	51
5.1	Dataset .....	54
5.1.1	HTML feature extraction .....	55
5.1.2	Domain splitting .....	56
5.1.3	Dimensionality reduction .....	56
5.2	Results .....	59
Chapter 6	Classifying Notices .....	64
6.1	Dataset .....	64
6.2	Feature extraction .....	66
6.2.1	Claim description .....	66
6.2.2	Tokenized URLs .....	66
6.3	Classifiers .....	67
6.4	Super Video category .....	68
6.5	Experiments .....	69
6.6	Results .....	69
Chapter 7	Conclusion .....	74
Appendix A	Additional topics .....	77
Appendix B	EMPCA MATLAB code for PCA .....	82
Bibliography	.....	86



## LIST OF FIGURES

Figure 2.1.	An illustration of how a document comprises of different topics. .	13
Figure 2.2.	An illustration of document-term matrix (Adapted from Do-kyum Kim, 2014[23]). . . . .	14
Figure 2.3.	Plate notation of LDA (Adapted from Blei et.al., 2009[6]) . . . . .	16
Figure 2.4.	Illustration of k-means clustering algorithm using randomly generated data. . . . .	25
Figure 2.5.	Illustration of a linear support vector machine. . . . .	29
Figure 2.6.	Illustration of a decision tree on car measurements dataset. . . . .	32
Figure 2.7.	Illustration of a random forest. . . . .	34
Figure 3.1.	A comparison of notices submitted per month by Google and Other sources in Chilling Effects. . . . .	36
Figure 3.2.	Number of URLs submitted per month. The trendline depicts the exponential rise in recent years. . . . .	39
Figure 4.1.	Distribution of HTTP response codes obtained after crawling 3 million URLs. . . . .	44
Figure 4.2.	Word count distribution in the dataset showing presence of <i>longtail</i> . . . . .	45
Figure 4.3.	A boxplot, without outliers, showing the document length distribution. . . . .	46
Figure 5.1.	Example of a <i>Not Found</i> page present in filetram.com. It is a search result of “mj fields jase”. . . . .	52
Figure 5.2.	Example of another <i>Not Found</i> page present in filetram.com. It is a search result of “tanya anne crosby perfect in my sight”. . . . .	53
Figure 5.3.	Example of a valid search result page for filetram.com. . . . .	54
Figure 5.4.	Screenshot of a parked page. . . . .	61
Figure 5.5.	Screenshot of an age verification page. . . . .	62

Figure 5.6.	Screenshot of a page showing unavailability of content due to copyright infringement. ....	63
Figure 6.1.	Confusion matrix of SVM on all Heldout notices, providing motivation for “Super Video” category. ....	68
Figure 6.2.	Accuracies for Test datasets using different features. ....	70
Figure 6.3.	Accuracies for Heldout datasets using different features. ....	70
Figure 6.4.	Comparison of confusion matrices from random forest (RF) classifier using 6 vs 7 categories ....	73

## LIST OF TABLES

Table 2.1.	Definition of symbols .....	17
Table 3.1.	Number of notices filed by different copyright holders along with the meta-level category the holder belongs to. ....	40
Table 4.1.	Top keywords of each topic learned by LDA with 10 topic configuration on crawled URLs dataset. <i>We try to figure out the meta-level label of each topic by looking at the keywords. First rows shows these labels and furthermore, we also highlight the important keywords present in each topic.</i> .....	49
Table 5.1.	List of domains used for k-means clustering. ....	57
Table 5.2.	Category wise distribution of 108 domains based on k-means clustering. ....	62
Table 6.1.	Set of labels used to classify the notices. ....	65
Table 6.2.	Manual labels of copyright holders .....	72
Table A.1.	A subset of topics from the topic model learned on crawled URL content and configured to find 50 topics. <i>Some of these topics represent different genres of “Music” and some of them are clearer than those in smaller models with only 10 topics.</i> .....	78
Table A.2.	A subset of topics from the topic model learned on description of notices and configured with 50 topics. <i>Interestingly, these topics are more coherent than their URL-related counterparts.</i> .....	80

## ACKNOWLEDGEMENTS

In the course of working towards this degree of Master of Science, I owe considerable debt of gratitude to the countless people. First and foremost, I would like to thank my advisors; Professor Lawrence Saul and Professor Geoffrey Voelker. Professor Lawrence Saul always provided me with right guidance whenever faced with technical roadblocks. Professor Geoffrey Voelker's devotion to academia has been a powerful source of inspiration for me. He was always there to calm me down and convince me that the world has not ended, be it with distributed systems project or this work. The trio of Lawrence Saul, Geoffrey Voelker and Stefan Savage is all you could ask for in mentors.

I wish to thank Gautam Akiwate, who also worked on this project. Working alongside him has always been exciting and he brought a new perspective to the problems.

I would also like to thank Matt Der and Tristan Halvorson. Matt Der provided me with the base code to scrape the HTML content and code to perform K-means clustering algorithm. Tristan helped me with the use of distributed crawler to gather the data for the project. Without both of them, I would still be working on my thesis.

All chapters, in part are currently being prepared for submission for publication of the material. Kothari, Mohit. The thesis author was the primary investigator and author of this material.

## ABSTRACT OF THE THESIS

MaLCoN: Machine Learning analysis on Copyright Notices

by

Mohit Rajkumar Kothari

Master of Science in Computer Science

University of California, San Diego, 2015

Professor Lawrence K. Saul, Chair

The adoption of digital computers and the digitization of record keeping marked the onset of the Digital Revolution bringing us into the Information Age. Among other things, the Internet is often regarded as a central force behind this revolution. As the name suggests, it means a web of interconnected computer networks. In recent years it has grown exponentially, allowing users to share and access information at an unprecedented scale. This freedom has its own set of challenges; the Internet unfortunately is often used for illegal sharing of copyrighted content and the traditional copyright laws were not well equipped to handle such scenarios. Hence, the **Digital Millennium Copyright**

**Act** was signed into law as an attempt to tackle these challenging issues. It provides an extra-judicial process, **Section 512**, by which copyright holders can issue takedowns notices of allegedly infringing material.

In this work we attempt to look at the takedown notices, available from online repository like **Chilling Effects**, and try to analyze them in a systematic fashion. We mainly focus on using machine learning techniques such as latent Dirichlet allocation, k-means, support vector machines and random forests to find interesting patterns in the dataset and try to reason about different challenges faced while working with this dataset.

# Introduction

The Internet may well be regarded as the invention of the last century, giving users a platform for creating, sharing, consuming, and accessing digital content at large scale. Such ease of accessibility also presents a myriad of challenges to curb illegal sharing or exchange of copyrighted materials such as movies, music, software, TV shows and books. Traditional copyright laws were not well equipped to handle such cases. Hence, in response to address these concerns and calm copyright holders, the **Digital Millennium Copyright Act** was passed. The law itself consists of five “*titles*”, but we will focus on **Title II**, also called **Section 512**, which lays out the mechanism by which a copyright holder can request takedown of an allegedly infringed material, from the Internet, bypassing the traditional judicial oversight. This extra-judicial process naturally, also raises concerns of being biased towards copyright holders or being abused by them.

This work is an attempt to systematically look at the *notices* which are issued under **Section 512**; in particular, we utilize data mining techniques to obtain insights into the current state of the law and how effectively it is being used. We mainly try to answer the following two questions:

1. Can we extract *latent topics* from the notices?
2. Can we *automatically classify* the notices?

The rest of thesis is structured as follows. Chapter 1 gives a brief overview of the **Digital Millenium Copyright Act** and the mechanism laid out in **Section 512** for taking

down an allegedly infringed content. In chapter 2, we provide some background on different machine learning techniques used in this work, namely latent Dirichlet allocation (LDA), k-means clustering, support vector machine and random forest classifiers. Chapter 3 describes the primary source of our dataset and the methodology used to gather the data. We also present some interesting properties of the dataset. Furthermore, we tackle the first question of finding *latent topics* by learning a topic model, described in chapter 4, on top of the crawled dataset. We also present different challenges we encounter during this experiment. This is followed by chapter 5, which describes our effort to remove some “noisy” data, in order to get more coherent topics, from the dataset by performing k-means clustering on per domain basis. Chapter 6 describes the classification experiment on notices to tackle the second question of *automated labeling*. Finally, we conclude with chapter 7.



# Chapter 1

## A Study in DMCA

In recent years, with the advancement in technology, the Internet has not only changed the way people create, share and consume content but has also made it much more accessible. The ease of copying and sharing in today's digital age has only increased the importance of copyright law. Some of the copyright issues include, sharing of copyrighted digital content such as movies, music, and books; liability concerns by copyright holders in an event of copyright infringement; establishment of internal policies regarding copyright ownership; and issues at the intersection of copyright law, advancing technology, and educational, research, and non-profit institutions<sup>1</sup>.

Furthermore, the issue of liability in cases of copyright violations became very complicated because of the complex nature of sharing and involvement of multiple entities in the eventual sharing of the content. With respect to the traditional copyright laws, the *Online Service Providers (OSPs)*, which includes *Internet Service Providers (ISPs)*, search engines (Google and Bing), libraries, educational institutions and others, were held liable on behalf of copyright infringement of their end users ( *secondary liability*). They were primarily held liable because they facilitated the transmission of digital content, namely the bits and bytes, that eventually translated to a third party copyrighted material. Hence, *OSPs* faced a lot of ambiguity in determining when they

---

<sup>1</sup>[http://www.asu.edu/counsel/brief/digital\\_copyright.html](http://www.asu.edu/counsel/brief/digital_copyright.html)

could be held liable for copyright infringement[38][26].

As a result, on one side of the ring were *OSPs*, trying to push for *safe harbor* provisions in the copyright laws from the *secondary liabilities* and on the other-side the copyright holders, who were trying to hold online service providers liable for the infringement because of their relationship with the users, usage of their software and infrastructure in aiding the infringement.

In response, on October 28, 1998, President Bill Clinton signed the **Digital Millennium Copyright Act (DMCA)**, limiting the liability of *OSPs* for copyright infringement by their users. The law became effective in October, 2000 and it has been incorporated into **Title 17** of the **United States Code**. The primary goal of the law is to “extend the reach of copyright, meet the demands of the Digital Age and to conform U.S. law to the requirements of the *World Intellectual Property Organization (WIPO)* and treaties signed in 1996.[4]”

The **DMCA** itself is divided into five major “*titles*” and addresses numerous other copyright related issues, such as imposing rules prohibiting the circumvention of technological protection measures (**Digital Rights Management**), mandating a study of the effects of anti-circumvention protection rules on the “first sale” doctrine and others[4]. For the purposes of this work, we will focus on the **Title II** of the **DMCA: Online Copyright Infringement Liability Limitation Act (OCILLA)**, otherwise known as “DMCA 512” or the “DMCA takedown provisions.” The **Title II** adds a new section to the **United States Copyright Act, Section 512**, which puts in limitations on liability for *OSPs* and lays down eligibility criterion for *safe harbor* in the event of copyright infringement.

In essence, *OCILLA* tries to achieve a balance between “the rights of authors and the larger public interest, particularly education, research and access to information”<sup>2</sup> by

---

<sup>2</sup>[http://www.wipo.int/treaties/en/text.jsp?file\\_id=295166](http://www.wipo.int/treaties/en/text.jsp?file_id=295166)

protecting *OSPs* from *secondary liability*, provided that *OSPs* adheres to the requirements laid down in order to protect the rights of authors. These requirements are further explained in section 1.1.3. Thus, *OCILLA* provides authors or copyright holders with the tool and process that ensures the rapid removal of allegedly infringing material while also providing *safe harbor* to the *OSPs*.

The remainder of this chapter tries to give a brief overview of **Section 512** and the process involved in taking down an allegedly copyright infringing material.

## **1.1 Title II**

The OCILLA is codified as Section 512 in **Title 17** of the **United States Code (Public Law No. 105-304, 112 Stat. 2860, 2877)**. Section 512 extends the limitations on liability for *OSPs*, in case of copyright infringement, by adding 4 new sections. These four new limitations are based on the nature of the services offered by the service providers[24],

1. **Transitory Digital Network Communications**
2. **System Caching**
3. **Information Residing on Systems or Networks At Direction of Users**
4. **Information Location Tools**

Before we look at these limitations and the mechanisms, let's first get an overview of what qualifies as a *service provider* and how they become *eligible for limitations*.

### **1.1.1 Who is a *Service Provider*?**

Since all of Section 512 is centered around the *Online Service Provider*, it is imperative to understand who or what qualifies as a *service provider* that is seeking

the benefits of liability limitations under **Title II**. The definition differs based on the kind of service offered by the provider, below we quote the definition provided in the legislation[24].

For purposes of the first limitation, relating to transitory communications, “service provider” is defined in **Section 512(k)(1)(A)** as “*an entity offering the transmission, routing, or providing of connections for digital online communications, between or among points specified by a user, of material of the user’s choosing, without modification to the content of the material as sent or received.*”

For purposes of the other three limitations, “service provider” is more broadly defined in **Section 512(k)(1)(B)** as “*a provider of online services or network access, or the operator of facilities therefore.*”

### **1.1.2 How to become *Eligible for Limitation*?**

After the party or entity seeking benefits of *safe harbor* qualifies as a *service provider*, the service provider must meet the following two general conditions[24, 38],

1. A service provider must adopt and reasonably implement a policy of terminating the accounts of subscribers and account holders of the service provider’s system or network who are “*repeat infringers*”<sup>3</sup> under appropriate circumstances.
2. A service provider must accommodate and not interfere with standard technical measures used by copyright holders to identify and protect copyrighted works.

Once these conditions are met, the *service provider* becomes eligible for the limitations of liability stated in Section 512.

### **1.1.3 Limitations and Mechanisms**

As stated earlier, Section 512 is usually best known for facilitating copyright holders to ask *OSPs* to take down infringed material without the involvement of judicial

---

<sup>3</sup>The definition of “repeat infringers” is often debated. When does an alleged infringer becomes a “repeat infringer”? After two complaints? More?

system. The *safe harbor* provisions acts as a strong incentive for *OSPs* to cooperate with the copyright holders in their fight against piracy. To qualify for these provisions, *OSPs* have to *expeditiously* takedown the allegedly infringed material in response to the notices submitted by the concerned copyright holders[38]. The process and protections differs based on the nature of service provider, as shown below,

### 1. **Transitory Digital Network Communications**

Also called *Section 512(a)*, provides “*safe harbor*” for service providers which provide transmission, routing or connections such as broadband, DSL, high-speed Internet. For services falling under this section, there is no requirement to “*take down*” material; the law simply gives them safe harbor from their users’ infringements as they are just acting as a basic conduit through which the content is flowing. Example of such a service is an *Internet Service Provider (ISP)*, like Time Warner Cable (TWC).

### 2. **System Caching**

Aliased as *Section 512(b)*, provides limitation of liability for the intermediate and temporary storage of material on a system or network controlled or operated by the service provider. *OSPs* qualifying under this section usually cache content for improving system performance and usability. Such *OSPs* are required to respond and remove or disable access to allegedly infringing material when certain conditions are met. For these conditions to be met, the material from the originating site must have been removed or atleast must have been ordered for removal. It is expected that the copyright holder filing the notice must also give a notification confirming the same to the service provider[24, 38]. Examples include *Content Delivery Networks (CDNs)* such as Akamai or Amazon Cloudfront.

### 3. **Information Residing on Systems or Networks At Directions of Users**

Coded as **Section 512(c)**, it limits the liability of service provider for hosting the content at the direction of the user. Such services, upon receiving notices of copyright infringement, are required to respond *expeditiously* by removing the infringing content. Examples of such providers include *content hosting services and forums* like Youtube and Blogger respectively.

#### 4. **Information Location Tools**

Finally, **Section 512(d)** covers *search engines*, where a service provider is linking or referring users to online location containing the infringed material. Popular examples of such service providers include Google and Bing. The purpose of this process is not to actually remove the content from the Internet but rather make it more difficult to locate the alleged infringing content. The copyright holders have to file a different notice under **Section 512(c)** with the service provider hosting the content to remove it from the Internet.

#### 1.1.4 **Counter-notices**

The procedure of *expeditious* removal of content upon receipt of takedown notice raised many concerns among academic institutes and other entities. These concerns were majorly related to the extra-judicial process proposed for *expeditious* removal of alleged infringing content from the Internet. According to them, it violated the constitutional provisions for due process of the claims. Hence, in an attempt to resolve these concerns, **Section 512(g)** was added, containing the additional procedural protections.

As mentioned earlier, service providers who qualify under **Section 512(c)** have to expeditiously remove the content upon receipt of a notice. In addition to the removal, in accordance with **Section 512(g)**, *OSPs* also have to notify the alleged infringer that the material has been removed, and act as an intermediary for any further communication between the complainant and the alleged infringer. Upon the receipt of the notice,

the alleged infringer can then file a counter-notice with the *OSP* which in turn has to forward it to the complainant. If within **10-14** days, after the receipt of counter-notice, the complainant has *not* notified the *OSP* regarding a lawsuit, the contested content or material can then be reinstated by the *OSP*<sup>4</sup>. It is important to note that a valid counter notice must include the following:

1. A physical or electronic signature
2. Identification of the material removed and its former location.
3. Statement under penalty of perjury that the user has a good faith belief the material was mistakenly removed.
4. The user's name, address, and phone number
5. Consent to the jurisdiction of Federal District Court

However, for *OSPs* qualifying under *Section 512(d)*, according to *Section 512(g)* they are not required to notify the alleged infringer of index removal as they likely have no service relationship with the alleged infringer and in any case they would rarely have the ability to notify. It is also important to note that the *OSPs* are exempt from the liability, as per *Section 512(g)(1)*, of a mistaken yet good faith removal of material based on a notice from a copyright holder[24].

Interestingly, *Section 512(h)* of the **DMCA** provide provisions by which a clerk of any United States district court can be requested, by the copyright holders, to issue a subpoena to a service provider to identify an alleged infringer. In response, the *OSP* is expected to *expeditiously* disclose the user's personal identifying information in accordance with a subpoena as long as it is accompanied by a valid notice[24][38].

---

<sup>4</sup><http://digitalcommons.law.scu.edu/cgi/viewcontent.cgi?article=1413&context=chtlj>

Looking back at the various sub-sections of **Section 512**, it appears to us that, they often don't address the deeper due processing concerns of the claims, demand stringent requirements for maintaining *safe harbor* qualification, they don't provide many incentives to question the takedown notices and give second thoughts to the claims. All of the above lead us to believe that there is a potential for abuse by copyright holders.



# Chapter 2

## Background

Text mining refers to the collection of techniques that extract and derive high-quality information from large unstructured text such as emails and technical documents. Today's corpora often consist of millions of documents, and there arises a need to interact with these documents in an automated fashion where a simple search is often not enough. The following are some of the most common tasks in text mining:

1. **Clustering**

Finding similarities and relationships between subsets of documents representing certain meta-level themes[16].

2. **Classification**

Classifying an unseen document to one of the known class of the document. An important application is spam-classification of emails[21].

3. **Topic Modeling**

Uncovering the underlying semantic structure of a document collection[6].

4. **Keyword extraction**

Finding a set of relevant terms that are representative of the given document[5].

and there are many more, like summarizing a document and predicting trends over time. This chapter gives a brief overview of some of the algorithms used for topic modeling, clustering and classification.

## 2.1 Topic modeling

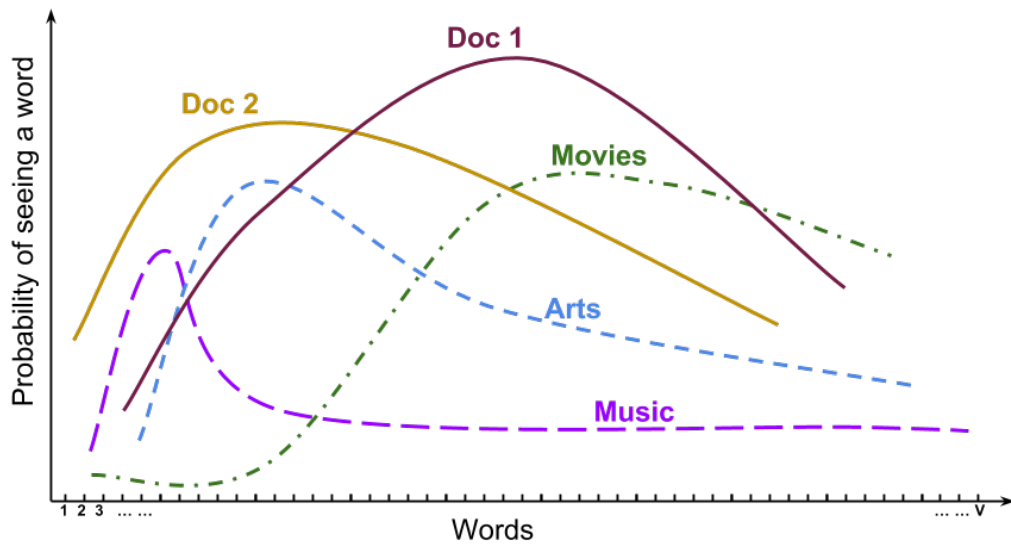
Topic modeling, as described earlier, tries to discover the main themes that pervade a large and otherwise unstructured collection of documents. They often use hierarchical Bayesian analysis of the original text to discover the underlying themes present in the corpus. The discovered themes are represented as “**topics**” which signify the hidden relationships between the documents and words in those documents. These topics are just the probability distribution of words in the corpus and a document is modeled as a mixture of topics consisting of different proportions of each topic.

Figure 2.1 represents a hypothetical distribution of topics over words and how a document can be represented as a distribution of multiple topics and in-turn a distribution over words. The dotted lines represent topic distribution and solid lines represent the documents.

The rest of the section gives an overview of one of the topic models, latent Dirichlet allocation (LDA)[7, 6], that has been used as basis for many other topic models. It is based on the work of latent semantic indexing (LSI)[13] and probabilistic LSI[19].

### 2.1.1 Bag-of-words representation

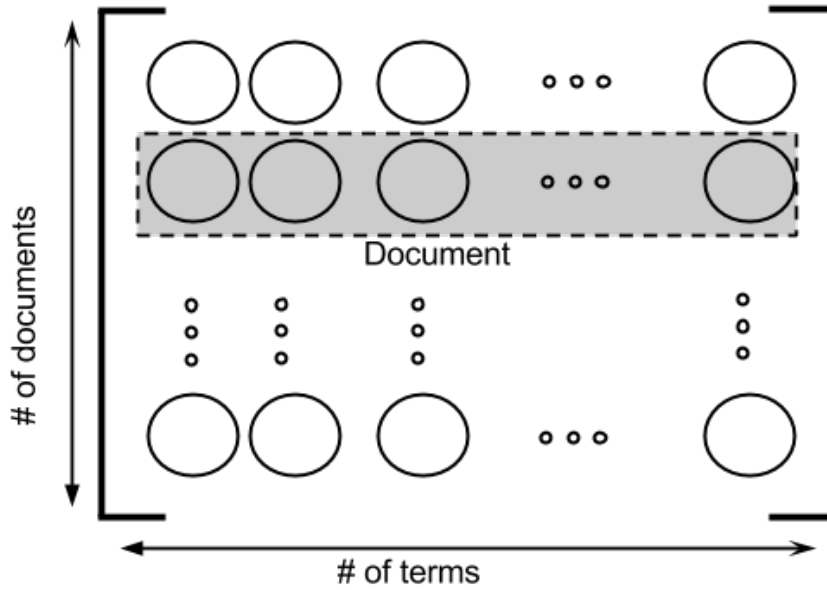
The first hurdle for any text mining task is to come up with a representation of the documents under consideration. Ideally, we would like to retain the semantic information represented by the order of words in the document. However, this often results in increased complexity of the algorithm. LDA relies on the the bag-of-words assumption[7], which means that it does not preserve the ordering of the words in the



**Figure 2.1.** An illustration of how a document comprises of different topics.

document and represents a document as a “bag-of-words.” Given a document, the bag-of-word representation is a feature vector of word counts in the document. The dimension of the vector represents the vocabulary of the underlying corpus, namely the set of unique words in the whole corpus. Using this representation, the corpus can be represented as a 2-dimensional matrix  $D$  where each row represents a document and each column represents a word. So any element of the matrix  $D_{i,j}$  represents the count of word  $j$  in document  $i$ . This matrix is often called document-term matrix (DTM) or term-document matrix (transpose of DTM); figure 2.2 gives an illustration of the same.

Since the model works on counts of word, frequently occurring words can introduce noise and reduce the quality of the algorithm. These words do not correspond to a unique topic or don’t have a distinguishing factor. Hence, most of the applications pre-process the documents by eliminating these so called “stop” words from the vocabulary. Some of the examples include pronouns (you, he, it), connectives (and, because), prepositions (to, the) and some applications curate their own “stop” words specific to the corpus. In addition to stopword removal some applications also “stem” words, to



**Figure 2.2.** An illustration of document-term matrix (Adapted from Do-kyum Kim, 2014[23]).

further reduce the noise, by stripping the word to its root.

### 2.1.2 Multinomial distribution

Now that we have a representation of documents as bags-of-words and given our assumption that a document consists of mixture of topics, we can model the document as a sequence of words drawn from (a mixture of) topics. For this, LDA uses a multinomial probability distribution. Mathematically this distribution is represented as,

$$p(d; \bar{\delta}) = \left( \frac{N_d!}{\prod_{j=1}^V d_j!} \right) \left( \prod_{j=1}^V \delta_j^{d_j} \right), \quad (2.1)$$

where given a document  $d$ ,  $\delta_j$  represents the probability of word  $j$  occurring in the document, such that  $\sum_{j=1}^V \delta_j = 1$ ,  $N_d$  represents the length of the document, and  $d_j$  is the count of word  $j$  in the document.

The first factor of equation 2.1 is called “multinomial coefficient” which represents the total permutations of words in the document. The second factor is the probability of any one of those permutations.

### 2.1.3 Dirichlet distribution

LDA makes a central use of the Dirichlet distribution[6], “the exponential family distribution over the simplex of positive vectors that sum to one.” It is a multivariate generalization of the beta distribution. The density function is given by,

$$p(\bar{\gamma}|\alpha_1, \alpha_2, \dots, \alpha_K) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K \gamma_i^{\alpha_i-1}, \quad (2.2)$$

where  $\bar{\alpha}$  is  $K$  dimensional vector with  $\alpha_i > 0 \forall i \in 1..K$  and  $\Gamma(\cdot)$  represents the Gamma function, which is a real-valued generalization of the factorial function. Equation 2.3 shows the integral for computing the gamma function:

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt \quad (2.3)$$

Dirichlet distribution is useful because it has finite dimensional sufficient statistics and is conjugate to the multinomial distribution (see previous section) which helps with inference and parameter estimation for LDA [7]. LDA uses a variant of the generalized Dirichlet distribution, the *symmetric Dirichlet*, in which all the parameter components have the same value. A greater sum of  $\bar{\alpha}$  leads to a sharper peak at the center whereas low values result in peaks at the extremes of the simplex.

### 2.1.4 Latent Dirichlet allocation

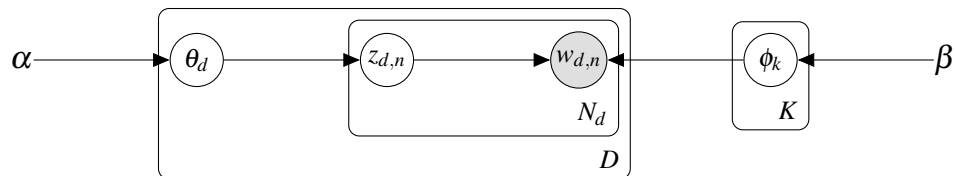
LDA is based on the intuition of hidden variable model where the observed data are the terms occurring in the documents and the hidden variables are the latent topics,

more specifically, the proportions of different topics present in each document. For a collection of documents, the posterior distribution of unseen variables given the observed data determines the topical decomposition of the collection. LDA models the way the documents are generated as an imaginary random process and assumes a particular interaction between the observed documents and hidden topical structure.

Table 2.1 describes meanings of the symbols used from now onwards. We now describe the generative process of LDA (adapted from [6])

1. For each topic  $k$ :
  - (a) Generate a word distribution for that topic ( $\bar{\phi}_k \sim Dir_V(\beta)$ ).
2. For each document  $d$ :
  - (a) Generate a topic distribution ( $\bar{\theta}_d \sim Dir_K(\alpha)$ ).
  - (b) For each word,  $w_{d,n}$ :
    - i. Select a topic  $z_{d,n} \sim Mult(\bar{\theta}_d), z_{d,n} \in (1, \dots, K)$ .
    - ii. For this topic, select a word ( $w_{d,n} \sim Mult(\bar{\phi}_{z_{d,n}}, w_{d,n} \in (1, \dots, V))$ ).

Since, this is a repetitive graphical model, it can also be represented using plate notation as shown in Figure 2.3. Here each node represents a random variable; shaded nodes are observed entities whereas unshaded nodes are latent variables. Edges are the dependence between nodes, and plates show repetition[11].



**Figure 2.3.** Plate notation of LDA (Adapted from Blei et.al., 2009[6])

**Table 2.1.** Definition of symbols

Symbols	Description
$V$	Vocabulary size
$D$	Corpus size
$K$	Number of topics
$d$	Auxiliary index for documents
$N_d$	Number of words in document $d$
$k$	Auxiliary index for topics
$v$	Auxiliary index for word
$m_{d,k}$	Number of times topic $k$ is assigned in document $d$
$q_{k,v}$	Number of times topic $k$ is assigned to word $v$ in corpus
$w_{d,n}$	Word at position $n$ in document $d$
$\bar{w}_{-(d,n)}$	All the words of document $d$ with $w_{d,n}$ removed
$z_{d,n}$	Assigned topic for word $w_{d,n}$
$\bar{z}_{-(d,n)}$	All the topic assignments of document $d$ except $z_{d,n}$
$\bar{\theta}_d$	Topic distribution for document $d$
$\bar{\phi}_k$	Word distribution for topic $k$
$\alpha$	Prior for symmetric Dirichlet over $\bar{\theta}_d$
$\beta$	Prior for symmetric Dirichlet over $\bar{\phi}_k$
$Dir_V(\beta)$	A $V$ -dimensional Dirichlet
$Dir_K(\alpha)$	A $K$ -dimensional Dirichlet

We can now deduce the joint probability distribution from the directed acyclic graph represented by figure 2.3 and apply d-separation rules on the resulting Bayesian network. Equation 2.4 represents the components of joint probability distribution of hidden and observed variables given the priors for Dirichlet distributions[33].

$$p(\bar{\theta}, z, w, \bar{\phi} | \alpha, \beta) = p(\bar{\theta} | \alpha) \times p(\bar{\phi} | \beta) \times p(z | \bar{\theta}) \times p(w | z, \bar{\phi}) \quad (2.4)$$

In equation 2.4,  $p(\bar{\theta} | \alpha)$  represents topic distribution of a document which is drawn from a  $K$ -dimensional Dirichlet distribution with  $\alpha$  as the prior, and we can also apply conditional independence because all the documents are assumed to be independent of each other. Equation 2.5 shows the computation:

$$p(\bar{\theta}_{1:D} | \alpha) = \prod_{d=1}^D \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \bar{\theta}_k^{\alpha_k - 1}. \quad (2.5)$$

$p(\bar{\phi} | \beta)$  is word distribution per topic which is drawn from a  $V$ -dimensional Dirichlet distribution give  $\beta$  as the prior. Equation 2.6 computes the probability:

$$p(\bar{\phi} | \beta) = \prod_{k=1}^K \frac{\Gamma(\beta_k)}{\prod_{v=1}^V \Gamma(\beta_{k,v})} \prod_{v=1}^V \bar{\phi}_{k,v}^{\beta_{k,v} - 1}. \quad (2.6)$$

Moving on,  $p(z | \bar{\theta})$  is topic to word assignments for the given collection of documents. Here each word in document  $d$  (i.e.  $w_{d,n}$ ) is assigned a topic, and its probability depends on the topic distribution  $\bar{\theta}_d$ . Equation 2.7 computes the probability:

$$p(z | \bar{\theta}) = \prod_{d=1}^D \prod_{k=1}^K \bar{\theta}_{d,k}^{m_{d,k}}. \quad (2.7)$$

Lastly,  $p(w | z, \bar{\phi})$  represents the probability of the term given the topic assignments



and topic distributions, in short the probability of given corpus:

$$p(w|z, \bar{\phi}) = \prod_{d=1}^D \prod_{v=1}^V \phi_{k,v}^{q_{k,v}}. \quad (2.8)$$

Combining eqs. (2.5) to (2.8) gives us the final joint distribution as shown in equation 2.9.

$$\begin{aligned} p(\bar{\theta}, z, w, \bar{\phi} | \alpha, \beta) &= \left( \prod_{d=1}^M \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \bar{\theta}_k^{\alpha_k - 1} \right) \\ &\times \left( \prod_{k=1}^K \frac{\Gamma(\beta_k)}{\prod_{v=1}^V \Gamma(\beta_{k,v})} \prod_{v=1}^V \bar{\phi}_{k,v}^{\beta_{k,v} - 1} \right) \\ &\times \left( \prod_{d=1}^D \prod_{k=1}^K \bar{\theta}_{d,k}^{n_{d,k}} \right) \times \left( \prod_{d=1}^D \prod_{v=1}^V \bar{\phi}_{k,v}^{n_{k,v}} \right) \\ &= \left( \prod_{d=1}^M \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \bar{\theta}_k^{\alpha_k + n_{d,k} - 1} \right) \\ &\times \left( \prod_{k=1}^K \frac{\Gamma(\beta_k)}{\prod_{v=1}^V \Gamma(\beta_{k,v})} \prod_{v=1}^V \bar{\phi}_{k,v}^{\beta_{k,v} + n_{k,v} - 1} \right) \end{aligned} \quad (2.9)$$

The key inferential problem that LDA needs to solve is the computation of posterior distribution of the hidden variables given a document as shown in equation 2.10:

$$p(\bar{\theta}, z | w, \alpha, \beta) = \frac{p(\bar{\theta}, z, w | \alpha, \beta)}{p(w | \alpha, \beta)}. \quad (2.10)$$

This distribution in general is intractable to compute because of the normalization constant in the denominator. To compute  $p(w | \alpha, \beta)$  we have to marginalize over the hidden variables and write it in terms of the model parameters, as shown in equation 2.11 and this equation is shown to be intractable due to the coupling between  $\bar{\theta}$  and  $\beta$ . For

detailed explanation, refer to latent Dirichlet allocation by David Blei, Andrew Ng and Jordan Michael[7, 15].

$$p(D|\alpha, \beta) = \int_{\bar{\phi}} \int_{\bar{\theta}} \sum_z \prod_{d=1}^M \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \bar{\theta}_k^{\alpha_k + n_{d,k} - 1} \times \prod_{k=1}^K \frac{\Gamma(\beta_k)}{\prod_{v=1}^V \Gamma(\beta_{k,v})} \prod_{v=1}^V \bar{\phi}_{k,v}^{\beta_{k,v} + n_{k,v} - 1} d\bar{\theta} d\bar{\phi} \quad (2.11)$$

Hence, the central computation problem for LDA is to approximate the posterior distribution given in equation 2.10. Some of the techniques developed for this problem are listed down below:

- Mean field variational inference[7].
- Collapsed Gibbs sampling[36].
- Collapsed variational inference[37].
- Expectation propagation[30].

Each method has its own pros and cons, and usually a selection is made based on different trade offs like complexity, performance time, implementation difficulty, and accuracy. In this work, we use MACHine Learning for Language Toolkit (MALLET)[29], which implements a distributed version of latent Dirichlet allocation using Collapsed Gibbs sampling[31]. We now give an overview of the same.

### 2.1.5 Collapsed Gibbs sampling

Collapsed Gibbs sampling uses the concept of ‘‘Gibbs sampling’’, a form of Markov chain Monte Carlo (MCMC) sampling, to extract a pre-defined set of topics from the underlying corpus. ‘‘MCMC refers to a set of approximate iterative techniques

designed to sample values from a high-dimensional distribution”[36]. Gibbs sampling tries to sample low-dimensional subsets of variables, where each subset is conditioned on the remaining set of variables, in order to approximate the high-dimensional distribution. It is an iterative procedure which terminates when the target high-dimensional distribution is fairly approximated.

In collapsed Gibbs sampling, direct estimates of  $\bar{\phi}_k$  and  $\bar{\theta}_d$  are not computed, rather they are approximated using the posterior distribution of  $\bar{z}$ .

For LDA, the algorithm takes into account each word token in the given corpus and estimates the probability of assigning the current word to each topic, conditioned on all the topic assignments except the current one. From this conditional distribution, the algorithm samples a new topic and assigns it to the current word. This conditional probability is given by  $p(z_{d,n} = j | \bar{z}_{-(d,n)}, \bar{w}, \alpha, \beta)$ . A series of manipulation using Bayes rule, marginalization and conditional independence, along with expansion gives the final probability:

$$p(z_{d,n} = j | \bar{z}_{-(d,n)}, \bar{w}, \alpha, \beta) \propto \frac{q'_{j,w_{d,n}} + \beta_{w_{d,n}}}{\sum_{v=1}^V q'_{j,v} + \beta_v} \frac{n'_{d,j} + \alpha_j}{\sum_{k=1}^K n'_{d,k} + \alpha_k} \quad (2.12)$$

where,  $q'_{j,w_{d,n}}$  represents the number of times word  $w_{d,n}$  is assigned topic  $j$  except the current instance, minus 1, and  $n'_{m,j}$  represents the number of times topic  $j$  is assigned in the document  $d$  except the current instance, minus 1. Equation 2.12 represents an un-normalized probability and the actual probability of assigning a word with topic  $j$  can be obtained by dividing the value in equation 2.12 for topic  $k$  by the sum over all the topics  $K$ [36]. For a more detailed explanation please refer to Steyvers and Griffiths[36, 17].

In a nutshell, the Gibbs sampling algorithm can be summarized as follows[33]:

1. Randomly initialize topic assignments for all the terms in the corpus
2. For each iteration of Gibbs sample:

- (a) For each term in the document and across all the documents:
- i. Decrement the matrices  $q'$  and  $n'$  by 1 corresponding to current topic assignment.
  - ii. Sample a topic according to equation 2.12.
  - iii. Increment the matrices  $q'$  and  $n'$  by 1 corresponding to new topic assignment.
  - iv. Update the topic assignment for the term.
3. Discard the initial Gibbs samples because they are poor estimates of the posterior (called the burnin period)[36].
  4. Repeat until a stable posterior distribution is reached.

Since, the Gibbs sampling algorithm gives the direct estimates of topics for every term. The word-topic distribution ( $\bar{\phi}_k$ ) and topic-document distribution ( $\bar{\theta}_d$ ) can be obtained from the count matrices  $q'$  and  $n'$ [36], given by:

$$\phi_{k,i} = \frac{q'_{k,i} + \beta_i}{\sum_{v=1}^V q'_{k,v} + \beta_v} \forall i \in (1, \dots, V) \quad (2.13)$$

$$\theta_{d,i} = \frac{n'_{d,i} + \alpha_i}{\sum_{k=1}^K n'_{d,k} + \alpha_k} \forall i \in (1, \dots, K) \quad (2.14)$$

As a side note on runtime complexity, the standard approach to Gibbs sampling described above takes  $\mathcal{O}(MK)$  for one epoch. Here  $M$  is the total number of tokens/terms in the corpus and  $K$  is the number of topics.

## 2.2 Clustering

As described earlier, clustering is a procedure for grouping data points or samples into groups, such that the members of a particular group are very similar to each other

and different from the members of other groups[20]. It is an example of an unsupervised algorithm which tries to automatically organize data and find hidden structure in the data. For example, clustering might involve creating multiple groups of users such that users in a single group share similar interests (like “automobiles”); which is different from the interests of other groups (“paintings”, “origami” etc.).

More formally, let the set of documents be represented as  $\{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_D\}$ , where each document  $\bar{x}_d$  is a  $V$  dimensional vector, and suppose there exists a distance function,  $dist(\bar{x}_{d_1}, \bar{x}_{d_2})$ , capturing the distance between the two documents. Given this, we would like to divide the corpus into  $K$  groups, different from  $K$  topics of LDA, such that every document  $\bar{x}_d$  has *exactly* one label  $k \in (1, 2, \dots, K)$  and documents belonging to same label have some semantic similarities between them. We now give a brief overview of k-means clustering algorithm which uses “*nearest mean*” approach to cluster documents.

### 2.2.1 K-means clustering

K-means clustering is an example of a clustering algorithm, which can be used to classify documents into pre-defined number of  $K$  groups such that each item belongs to the cluster with the *nearest mean*. Each cluster is identified by a “**centroid**”. The centroid is normally defined as “the center of mass of a geometric object of uniform density,” though here, we’ll consider it to be the **mean** of the documents present in the cluster. This results in clusters which are non-overlapping disjoint sets of items and hence, such algorithms are called partition-based clustering algorithms.

Finding an optimal solution to k-means for an arbitrary input is computationally NP-hard[1, 28]. However, there exist heuristic algorithms which converge quickly to local optima. One of the most widely used algorithm is Lloyd’s algorithm [25], often called **k-means algorithms** because of its popularity. It is an iterative approach which minimizes within-cluster “sum of squares distances” and guarantees a local optima but

not a global one.

In a nutshell, the algorithm proceeds as follows:

1. Initialization

(a) For every document  $\bar{x}_d$ :

i. Choose initial cluster identities  $z_d = k \in (1, 2, \dots, K)$ .

(b) For every cluster  $k$ :

i. Compute the centroid of the cluster  $\bar{m}_k$ .

2. Repeat

(a) For every document  $\bar{x}_d$ :

i. assign  $\bar{x}_d$  to its closest cluster:

$$z_d = \operatorname{argmin}_{i \in (1, \dots, K)} \operatorname{dist}(\bar{x}_d, \bar{m}_i),$$

$$\text{where } \operatorname{dist}(\bar{x}_d, \bar{m}_i) = \|\bar{x}_d - \bar{m}_i\|^2.$$

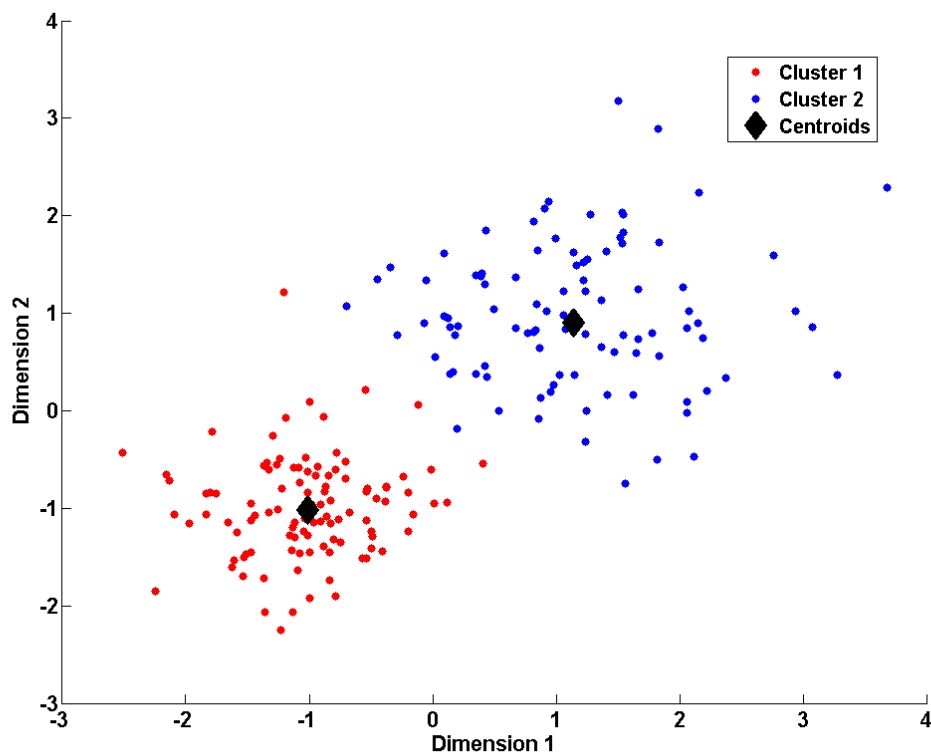
(b) For every cluster  $k$ :

i. Compute the new centroid by taking mean of all the documents to that cluster

$$\bar{m}_k = \frac{1}{N_k} \sum_{d: z_d=k} \bar{x}_d.$$

3. Repeat until convergence: no changes in  $\bar{z}_{1:D}$ .

As an illustration, we generate random data over two dimensions and run k-means (configured with  $K = 2$ ) on top of it. Figure 2.4 shows the final cluster assignments of the data and also plots the centroid of each cluster.



**Figure 2.4.** Illustration of k-means clustering algorithm using randomly generated data.

### Comparison with LDA

Both, LDA and K-means are *unsupervised learning* algorithms requiring a pre-defined  $K$  number of topics and clusters respectively. The main conceptual difference between the two is in terms of topic assignment and partitioning. K-means create  $K$  *disjoint* sets of documents whereas LDA works as a mixture membership model where each document consists of mixture of more than one topic and this does not result in disjoint set of documents.

Furthermore, it has been shown that the quality of the clusters largely depends on the way initial clusters are assigned[32]. There exist multiple ways of selecting initial clusters,

### 1. Randomly chosen

Almost self-explanatory, documents are initially assigned to the clusters randomly with uniform probability.

### 2. Forgy method

$K$  documents are randomly chosen with uniform probability and these  $K$  documents then act as the initial centroids for the clusters[2].

### 3. Macqueen approach

$K$  documents are randomly chosen and the rest of the documents are assigned, following the occurrence order, to the cluster with the nearest centroid. In this method, after each assignment a recalculation of the centroid is carried out[27].

For this work, we use MATLAB<sup>5</sup> implementation of k-means which uses *k-means++* for initialization. According to Arthur and Vassilvitskii[3], *k-means++* improves the running time of k-means algorithm along with the quality of the final clusters. *K-means++* initialization has following steps,

1. Select a document  $\bar{x}_d$  uniformly at random. This document is the first centroid, denoted by  $\bar{m}_1$ .
2. Compute distances from each document to  $\bar{m}_1$ ,

$$dist(\bar{x}_d, \bar{c}_1).$$

3. Select the next centroid,  $\bar{c}_2$  at random with probability proportional to the distance from  $\bar{c}_1$ ,

$$\frac{dist^2(\bar{x}_d, \bar{c}_1)}{\sum_{d'=1}^D dist^2(\bar{x}_{d'}, \bar{c}_1)}.$$

---

<sup>5</sup><http://www.mathworks.com/help/stats/kmeans.html>



4. To choose center  $j$ :
  - (a) Compute the distances from each observation to each centroid, and assign each observation to its closest centroid.
  - (b) For  $d \in (1, \dots, n)$  and  $p \in (1, \dots, j - 1)$ , select centroid  $j$  at random with probability:

$$\frac{\text{dist}^2(\bar{x}_d, \bar{c}_p)}{\sum_{h: x_h \in C_p} \text{dist}^2(\bar{x}_h, \bar{c}_p)}$$

where,  $C_p$  is the set of all observations closest to centroid  $\bar{c}_p$  and  $\bar{x}_m$  belongs to  $C_p$ . That is, select each subsequent center with a probability proportional to the distance from itself to the closest center that you have already chosen.

5. Repeat until  $K$  centroids are chosen.

## 2.3 Classification

Classification is one of the most common problems in machine learning. Generally speaking, given a set  $D$  of items;  $\{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_D\}$  and a set  $C$  of distinct classes  $\{c_1, c_2, \dots, c_C\}$ , the goal is to assign each instance in  $D$  to a unique class in  $C$ . Classification refers to the construction of a model that is able to correctly assign the classes for an observation not previously seen.

Some of the examples of classification include “click prediction” for web advertising companies, “face recognition” or tagging, and text categorization[21]. A general approach towards this problem is to “*learn from data*”, wherein the model learns (trains) from a set of observations along with their “*true*” class (training data) and then uses the trained parameters to “*predict*” classes of unseen data (test data). This approach of learning from training data is also called **supervised learning**.

### 2.3.1 Support vector machine

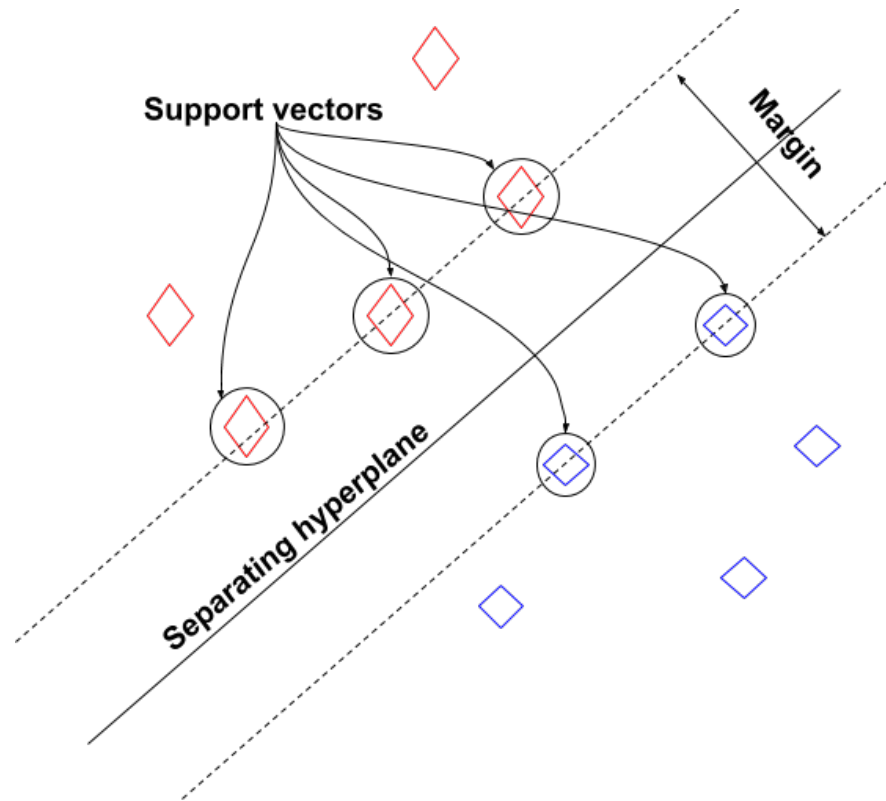
The support vector machine (SVM) is an instance of a supervised learning model which classifies observations by finding the best hyperplane that partitions the observations into different classes. They were first introduced in the early 90s[12] and gained popularity after their success for handwritten digit recognition[8]. Conventionally they are used as “*binary classifiers*” but a multi-class SVM can be built based on a “one-vs-rest scheme”, wherein a single classifier is trained for every class and while predicting a class for an unseen observation, the base classifier produces a real-valued score (confidence) for each class and based on this score the final class is predicted.

The remaining section will cover the basics of the binary SVM classifier. As defined earlier, an SVM works by finding the hyperplane that provides the largest margin between the two classes, wherein the margin is defined as the width of space parallel to the hyperplane such that there are no observation within that space. The points that lie on the boundary of the margin are called the *support vectors*. Figure 2.5 depicts a simple linear SVM with its separating hyperplane and support vectors.

There are different formulations of SVM depending on whether the data is linearly separable or not and whether there exists a linear hyperplane that separates the classes or a non-linear version is needed. For the purposes of this work, we use a linear SVM. The following section gives an overview of basic methodology behind the linear SVMs.

#### Linearly separable case

Let’s first define the terminology we use. For binary classification case, we will assume  $Y = \{+1, -1\}$  to be the set of classes and  $X = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_N\}$  to be our training set, where each  $\bar{x}_i$  is a  $V$  dimensional feature vector. Now, a general hyperplane can be described by equation 2.15, where  $\bar{w} \in \mathbb{R}^V$  represents the parameters and  $b \in \mathbb{R}$  is a



**Figure 2.5.** Illustration of a linear support vector machine.

constant<sup>6</sup>.

$$\bar{w}^T \cdot \bar{x} + b = 0 \quad (2.15)$$

Formally, the problem of finding the best hyperplane can be defined as follows,

$$\begin{aligned} & \text{Minimize } \|\bar{w}\| \\ & \text{Subject to } y_i(\bar{w}^T \bar{x}_i + b) \geq 1 \quad \forall i. \end{aligned}$$

Mathematically, we can change this to an equivalent problem of minimizing  $\frac{\|\bar{w}\|^2}{2}$ , as this results in an equivalent quadratic optimization problem. This problem is referred to as the **Primal**.

<sup>6</sup>The math has been adapted from <http://www.mathworks.com/help/stats/support-vector-machines-svm.html>

However, it is computationally easier to solve the **dual** of the primal. Dual can be obtained by using Lagrange multipliers  $\alpha_i$  for each observation. The resultant Lagrangian for primal is:

$$\mathcal{L}_P = \frac{\|\bar{w}\|^2}{2} - \sum_i \alpha_i (y_i (\bar{w}^T \bar{x}_i + b) - 1) \quad (2.16)$$

Now, setting the gradient of  $\mathcal{L}_P = 0$  and solving the equation, we obtain the following solutions:

$$\begin{aligned} \bar{w} &= \sum_i \alpha_i y_i \bar{x}_i \\ \sum_i \alpha_i y_i &= 0 \end{aligned}$$

Substituting these values in equation 2.16 results into the following optimization problem: dual of primal,

$$\begin{aligned} \text{Maximize } \mathcal{L}_D &= \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \bar{x}_i^T \bar{x}_j \\ \text{Subject to } \alpha_i &\geq 0, \sum_i \alpha_i y_i = 0 \quad \forall i. \end{aligned}$$

This is a standard quadratic programming problem for which the global maxima of  $\alpha_i$  can always be found[18]. The term  $(\bar{x}_i^T \bar{x}_j)$  is called the “**kernel**” of the linear SVM. For non-separable data, researchers often use “soft-margins” to solve the optimization problems and for the cases where a simple hyperplane is not enough there exists several non-linear transformations with the kernels[12].

After finding the optimal solution  $(\hat{w}, \hat{b})$ , the binary classification label of any

test observation can be predicted by looking at which side of the hyperplane does the observation lie:

$$pred(\bar{\mathbf{i}}) = sign(\hat{\mathbf{w}}^T \bar{\mathbf{x}}_i + \hat{\mathbf{b}}).$$

### 2.3.2 Random forests

Random forests are another method for supervised learning used for classification or regression. Initially developed by Breiman and Cutler[10], they perform classification by constructing multiple decision trees during training and using the majority vote of these decision trees to make new predictions. To further understand how random forests work, we will first look at some terminologies.

#### Decision tree

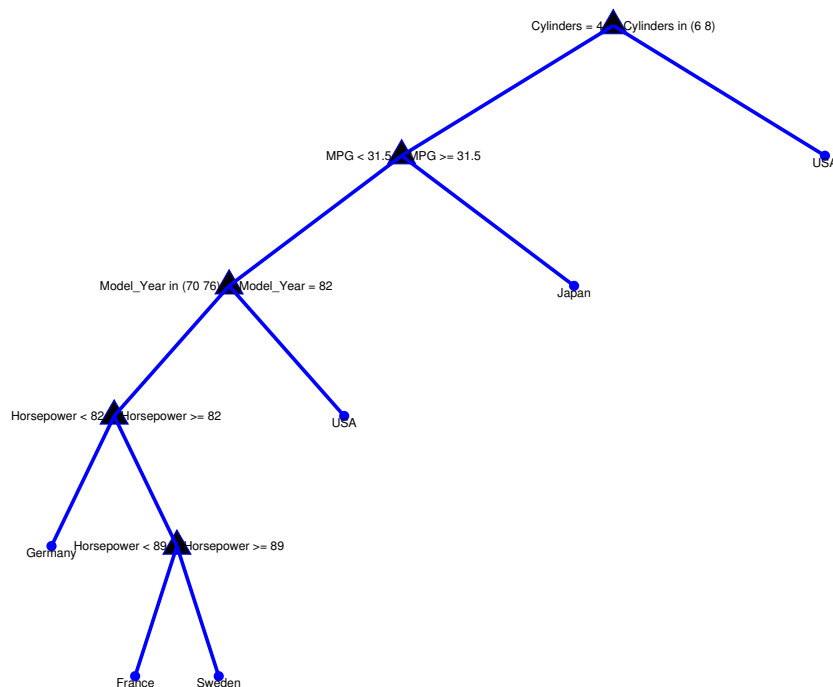
A decision tree is a predictive model which uses a set of decisions on an observation, based on the features, in order to provide final prediction. Decision trees take a finite number of classes as input. Once the decision tree is trained, the leaves of the tree represent the final class prediction and each internal root consists of a rule used to compare feature values and provide the direction for next level.

A simple decision tree is shown in figure 2.6, which is created using a MATLAB sample dataset on car measurements<sup>7</sup>. Each node describes the decision rule, and the leaf node represents the decision of the tree, which is the country of origin of the car.

#### Ensemble Learning

Ensemble learning methods usually involve learning multiple learners and using them in some weighted combination or in a voting mechanism to come up with final prediction. They usually combine a group of “weak learners” to form a “strong learner”.

<sup>7</sup>[http://www.mathworks.com/help/stats/\\_bq9uxn4.html](http://www.mathworks.com/help/stats/_bq9uxn4.html)



**Figure 2.6.** Illustration of a decision tree on car measurements dataset.

Here, “weak learners” are generally those classifiers which have very low accuracy, but higher than that of random guessing <sup>8</sup>.

## Bagging

Bagging is an example of a meta-algorithm that helps to reduce the variance of a model and avoid overfitting. It usually works by sampling the observations in the training set uniformly at random by replacement and training the model on this subset. This has been shown to improve the performance and stability of the resulting model[9].

In a nutshell, random forests are *ensemble of decision trees*, designed for classification or regression tasks, which use *bagging* to select a random subset of features for splitting and creating rules of decision trees, often referred to as “**feature bagging**.” Use

<sup>8</sup><https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/>

of multiple decision trees avoid overfitting and feature bagging improves the stability and performance of the model[10]. Generally, for feature bagging, out of  $V$  dimensions, a subset of  $\sqrt{V}$  features is used in each split[9] and the quality of the split is measured by Gini impurity[10] or Information gain.

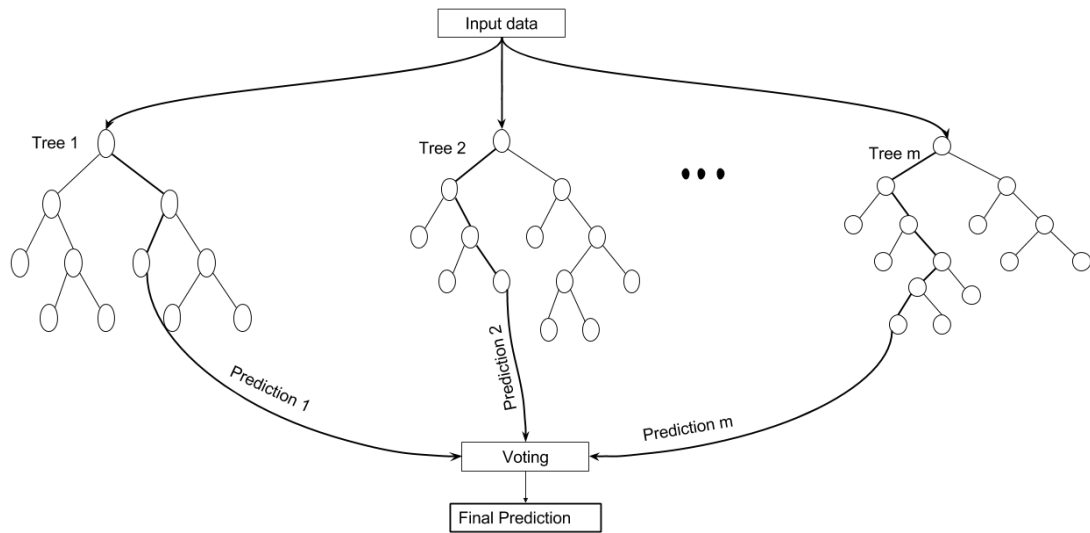
### **Cross validation**

As a side note, random forests don't require cross-validation because an unbiased estimate of the test set error is internally estimated by the algorithm. For every tree, a different bootstrap sample (random subset with replacement) is obtained from the training data and the heldout data is then used as test-set classification. This heldout data corresponds to approximately one-third of the input data. As the number of trees grow, this oob (out-of-bag) or heldout set helps in obtaining a running estimate of classification error<sup>9</sup>.

Finally, during testing, given a test observation, its class is determined by the majority vote prediction among all the trained decision trees. Figure 2.7 shows how a random forest model classifies an observation.

---

<sup>9</sup>[https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)



**Figure 2.7.** Illustration of a random forest.



# Chapter 3

## Why so Cold?

Equipped with the motivation to get some insights into the current state of the **Section 512** process, we started looking for a dataset to work on. Since the **DMCA** does not mandate making takedown notices part of public record, general public or research communities cannot analyse the effectiveness of the process in a well structured fashion. As a result, it becomes difficult to perform analysis and correlate the results.

### 3.1 Chilling Effects

**Chilling Effects** is an attempt to provide some insights into the takedown process. Among other things, it is primarily an online repository of the **DMCA** notices, initially created by Wendy Seltzer in collaboration with **Electronic Frontier Foundation (EFF)** and a consortium of law school clinics[38]. It is currently a project of Berkman Center for Internet & Society <sup>10</sup> with an aim of providing general public with a source of “cease-and-desist” letters, promoting research in this field and providing as much transparency as possible regarding the **Section 512** process.

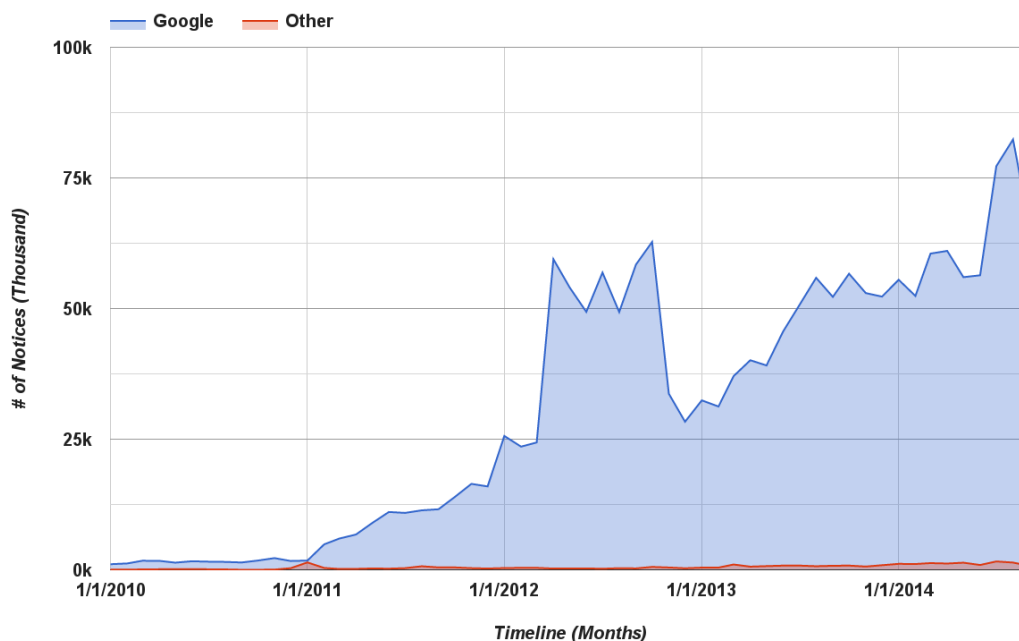
The remainder of the chapter provides an overview and a brief analysis of the datasets collected and created out of Chilling Effects.

---

<sup>10</sup>See <https://www.chillingeffects.org> for more information

## 3.2 Notices

In the Chilling Effects repository, one of the primary sources of **Section 512** notices is Google, which provides **Section 512(c) and (d)** notices submitted to them via Web forms. Google started submitting notices back in 2002 and eventually became the largest contributor of the notices as shown in figure 3.1. As an artifact, there is a probability that the results we get are biased and do not extrapolate to the notices from other sources.



**Figure 3.1.** A comparison of notices submitted per month by Google and Other sources in Chilling Effects.

Chilling Effects API<sup>11</sup> provides various formats in which the takedown notices can be viewed or downloaded. For the purposes of this work, we use the JSON (JavaScript Object Notation) format to download and parse the notices. In total, we crawl more than **2 million** notices from repository. This becomes our primary dataset from which all

<sup>11</sup>More information on the API [https://github.com/berkmancenter/chillingeffects/blob/master/doc/api\\_documentation.mkd](https://github.com/berkmancenter/chillingeffects/blob/master/doc/api_documentation.mkd)

the remaining datasets are derived. We now briefly describe the structure of the DMCA takedown notice.

### 3.2.1 Notice structure

Listing 1 provides the JSON structure of a typical notice obtained through the Chilling Effects API. These fields are on the lines of a valid takedown notice as coded in **Section 512(c)(3)(A)**[38].

Chilling Effects also tries to protect the privacy of the individual copyright owners by redacting any information that might disclose their names or identities before publishing any notice. Furthermore, as with any real world data, this dataset also presents its own set of challenges in terms of incomplete and malformed data; we prune it by skipping notices which contain malformed values in fields of our interest (for example, “works” field of the notice).

Among all the fields, the following are especially worth noting and are also important to us for further analysis:

1. **Sender Name**

The organization which filed the notice.

2. **Principal Name**

The organization claiming to be the copyright holder.

3. **Recipient Name**

The organization which received the notice (generally OSPs but not necessarily).

4. **Works**

The list of copyright claim(s) made by the copyright holder.

5. **Description**

A description of the work being infringed upon.

## 6. Infringing URLs

The list of infringing URLs per claim (works).

### 3.3 Infringing URLs

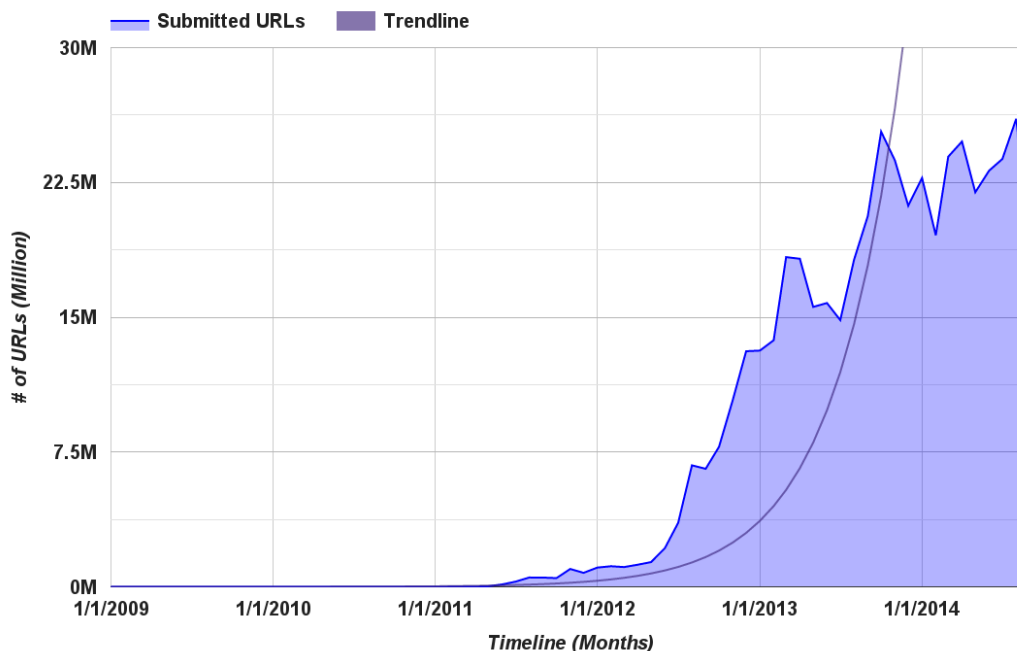
The second dataset that we use for this work is the raw HTML content of the infringing URLs that are part of the takedown notices. For reference, the total number of URLs present in all the crawled notices is 493,533,155 (approximately **500** million), and figure 3.2 shows the number of URLs submitted per month and the corresponding trendline depicts the exponential rise over past few years. As a starting point and before diving deeper into all the URLs, we sample a random subset of **10,000** notices and extract approximately **3 million** URLs from those notices.

We then crawl these 3 million URLs using a distributed crawler, part of another research project, which is written on top of the Hadoop framework<sup>12</sup>. In addition to getting the raw HTML content of the URL, the crawler also captures a screenshot of the completely loaded webpage (as seen on the Firefox browser), HTTP response status codes and several other fields. This content is then stored in a Hadoop Distributed File System (HDFS).

These two datasets, **Takedown notices** and **Crawled Infringing URLs**, now act as a source of other smaller more tailored datasets used in the analysis and applications of different machine learning techniques. These specialized datasets are further discussed in the experimentation sections.

---

<sup>12</sup>For information: <https://hadoop.apache.org>



**Figure 3.2.** Number of URLs submitted per month. The trendline depicts the exponential rise in recent years.

### 3.4 SOPA/PIPA effect

Figure 3.1 reveals an interesting correlation. If we examine the increase in notices from March to April 2012, there is an approximately **124%** increase in notices received by Google (from 24371 to 59450). Interestingly, this is around the same period when **Stop Online Piracy Act (SOPA)** and the **PROTECT IP Act (PIPA)**<sup>13</sup> were benched by the Congress (January, 2012). Copyright holders were lobbying for these laws to get passed as they contained provisions in their favor. This shows a strong correlation between this event and the drastic increase in **DMCA** notices being filed with Google, which suggests that copyright holders started using the existing laws more aggressively. However, we also see a dip appearing from October to November 2012, which we believe is due to an increase in number of URLs being submitted per notice, but we are not

<sup>13</sup>[http://en.wikipedia.org/wiki/Stop\\_Online\\_Piracy\\_Act](http://en.wikipedia.org/wiki/Stop_Online_Piracy_Act)

confident about this conclusion.

### 3.5 Copyright holders

From the dataset of more than 2 million notices, we take a closer look at the different copyright holders and the number of notices they filed; this additional information helps us to create a more uniform dataset for the classification task as explained in section 6.1. Table 3.1 shows some of the top copyright holders from different categories. Here, the copyright holders are manually labelled by using our own judgement. It is evident that there is an inherent skewness in the distribution of notices for different categories, and Music is the most dominant category.

**Table 3.1.** Number of notices filed by different copyright holders along with the meta-level category the holder belongs to.

<b>Copyright Owner</b>	<b>Number of notices</b>	<b>Category</b>
BPI (British Recorded Music Industry) Ltd	254,910	Music
Froytal Services Ltd	36,181	Adult
Microsoft Corp.	19,181	Software
IFPI	16,114	Music
Paramount	7,524	Movies
Fox	6,170	Movies
BangBros.com Inc.	5,543	Adult
Home Box Office	5,423	TV Shows
Random House	4,778	Books
Shueisha	3,926	Manga

```

{
  "dmca":{
    "id":"ID within Chilling Effects",
    "type":"DMCA",
    "title":"Notice title",
    "body":"Notice description (if any)",
    "date_sent":"Send Date",
    "date_received":"Receive Date",
    "topics":["List of topics"],
    "sender_name":"Filing organization",
    "principal_name":"Copyright Holder",
    "recipient_name":"Receiving entity",
    "works":[
      {"description":"Description of claim",
        "infringing_urls":[
          {"url":"Infringing URL"},
          {"url":"Infringing URL"},
        ],
        "copyrighted_urls":[
          {"url":"Original work URL"},
        ]
      }
    ],
    "tags":["List of tags (unreliable)"],
    "jurisdictions":["List of jurisdiction"],
    "action_taken":"Yes/No",
    "language":"Language of notice"
  }
}

```

**Listing 1.** Chilling Effects notice structure.

# Chapter 4

## Topical Paradise

As a first step, we would like to look at the crawled URLs from the notices and see what kind of *latent topics* are present in the dataset. Our initial hope is to get a representative set of topics such as ones related to Music, Books/Magazines, Movies, Comics and others. In order to do this, we learn a latent Dirichlet allocation model on top of the crawled content of the infringing URLs.

The remaining chapter gives an overview of the toolkit, dataset, and several pre-processing steps that we try in order to improve the results and final observations.

To learn the topic model, we use the LDA implementation from the Machine Learning for Language Toolkit (MALLET)[29]. It implements a distributed version of collapsed Gibbs sampling for learning. Collapsed Gibbs sampling has been explained in section 2.1.5.

### 4.1 Dataset

As described in section 2.1.1, we need a bag-of-word representation of each URL in order to learn the topic model. In order to achieve this, we use the **Crawled Infringing URLs** dataset described in section 3.3 and scrape the raw HTML content to extract **words**. Here, words represents the plain text present in the HTML content parsing



the HTML tag related information. To process HTML efficiently, we use HTMLParser<sup>14</sup>, a popular python library for parsing the “*body*” of HTML pages, which is quite robust on real world HTML content.

## 4.2 Pre-processing

Our initial experimentation on the extracted content yielded very noisy and nonsensical topics. For the sake of brevity, we don’t show these initial results. In an effort to reduce this noise we perform several pre-processing steps.

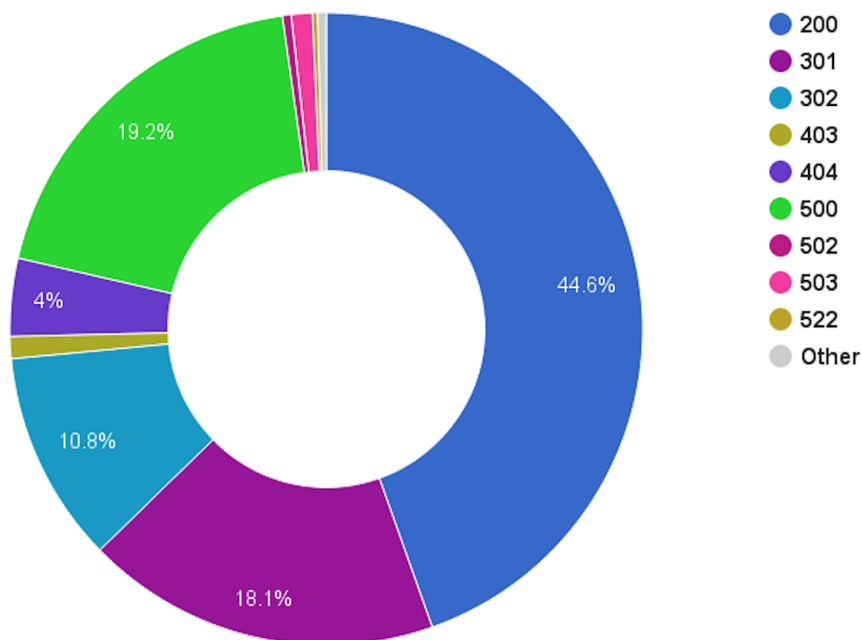
### 4.2.1 Restricting to 200 OK

Ideally, we would like to look at all the webpages whose content has not changed as a result of the DMCA notice. Since there is no practical way to know this, we make the following assumption: if the server of the URL does not return an error or the URL does not get redirected, we mark that URL as a valid one and restrict our dataset to only those URLs. In order to better understand this, we look at the distribution of the HTTP response codes that we obtain after crawling 3 million URLs. Figure 4.1 shows the overall distribution of different response codes. As it is evident from the pie chart, approximately 28.9% of the crawled URLs are a result of redirects (response code 301, 302) and 44.6% of the crawled URLs are valid.

We now create our dataset by removing all the invalid URLs (HTTP response code not equal to 200 OK), which also includes redirects. As a result, we are left with approximately 1 million URLs.

---

<sup>14</sup><https://docs.python.org/2/library/htmlparser.html>



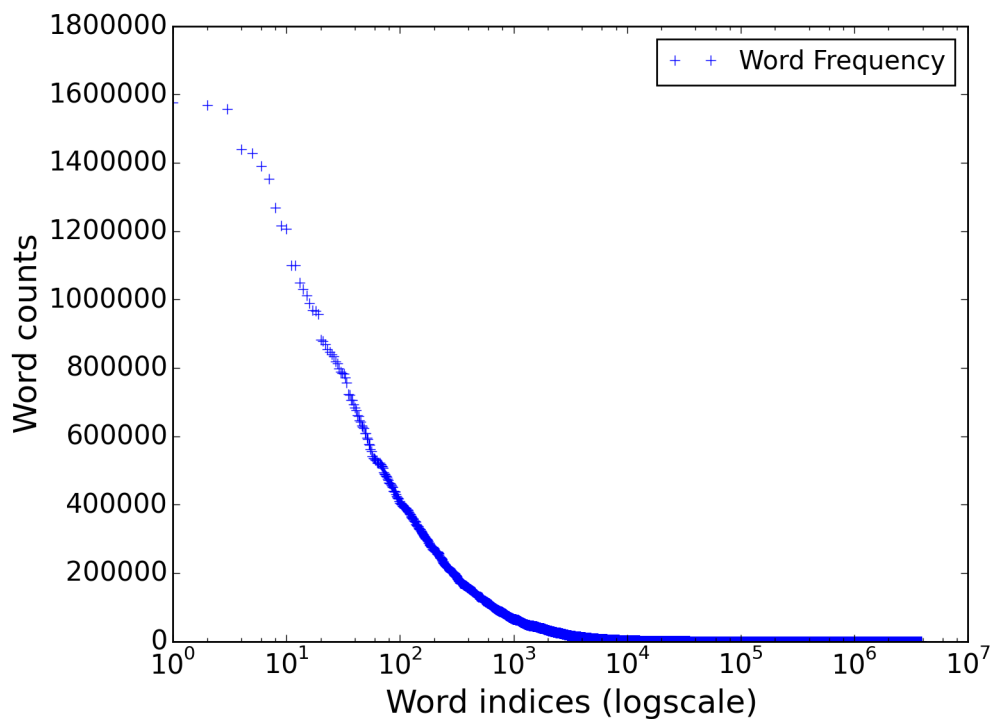
**Figure 4.1.** Distribution of HTTP response codes obtained after crawling 3 million URLs.

#### 4.2.2 Curating “stopword” list

Since we are working with a very specific dataset, we augment a secondary stopword list, beyond the standard list provided by MALLET. We look at the most frequent words in our dataset and add those words that don’t contribute anything meaningful. Some of the examples of such words are “mp3”, “downloads” and “login”. It is a challenging task to identify whether a word can be counted as a “stopword” or not, and for this we rely on our own judgement. In total, we have approximately **1200** words long “stopword” list.

### 4.2.3 Cutting the tail

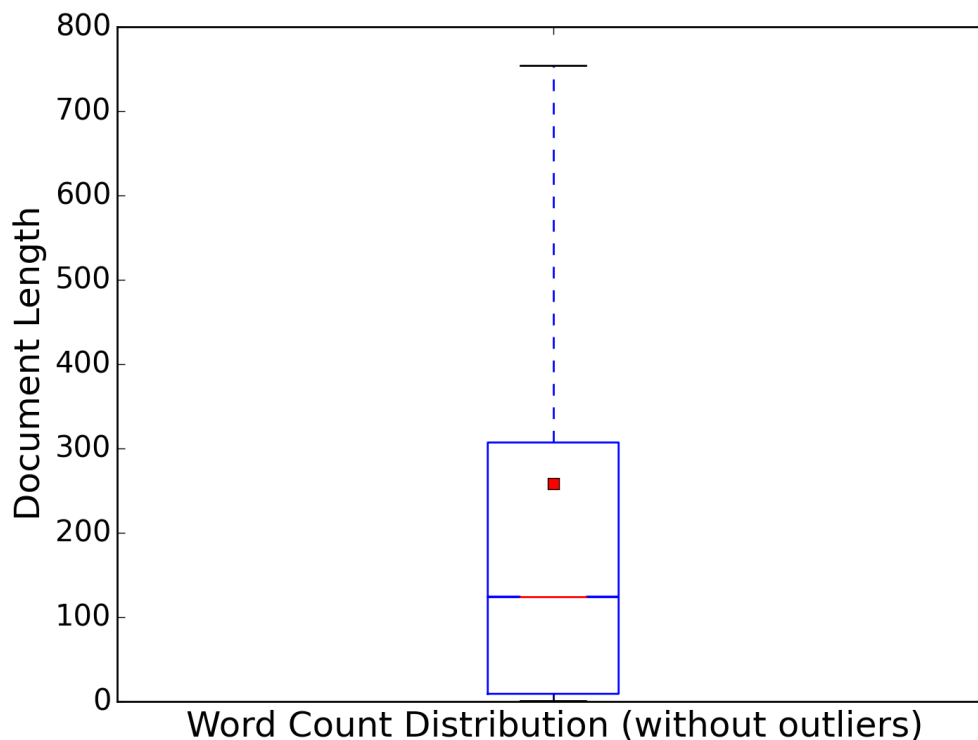
In order to reduce the computational time of the model, we further try to reduce the vocabulary of dataset. We start of by looking at the word count distribution. It is evident from the figure 4.2 that there exists a *longtail* of words that have very low frequencies. To get an idea of how *long* this tail is, we note that out of around 3.8 million words in the dataset, there are approximate 3.4 million words that have less than 10 occurrences. This constitutes around 88% of the total vocabulary size. Hence, we heuristically choose 10 as our threshold frequency to mark a given word for removal from the dataset.



**Figure 4.2.** Word count distribution in the dataset showing presence of *longtail*.

#### 4.2.4 Document length distribution

Finally, we also look at the document length distribution to see if there are a large number of documents which are either very small, in which case we can safely remove them, or are very large, in which case they might introduce more noise. Figure 4.3 shows a boxplot of the document length distribution. In this graph, we omit the outlier documents that have lengths on order of tens and hundreds of thousands. Based on this, we heuristically choose our minimum document length to be 25 words (approximately 29% of the documents) and maximum document length to be 15,000 words (approximately 0.03% of the documents) and discard all documents that don't lie within this window.



**Figure 4.3.** A boxplot, without outliers, showing the document length distribution.

Our final dataset for topic modeling consists of **967,131** documents with a vocabulary size of **457,723**. Since LDA requires a pre-defined number of topics, we experiment with **10, 20, 50** and **100** topics. We compare different models and get an idea of how the

topics change as the model is given more parameters to fit the dataset.

## 4.3 Results

Instead of showing topics from all the configurations, we present the results of LDA when trained with 10 topics. Table 4.1 shows the top keywords of each topic when LDA is run with a configuration of 10 topics. It is evident from the table that a lot of these topics do not always make sense even after performing several pre-processing steps. Many of these topics are either very generic or just simply random having no meta-level theme. Nevertheless, when running with different configurations, we do get topics that make more sense and have some high level theme which can be interpreted, but overall most of the topics seem to be very generic. Interestingly, on some configurations we do get different *genres* of *music* as different topics. We present some of these topics in appendix A.

In general, there still seems to be lot of noise present in the dataset. Some of the reasons that we believe might be causing the noise are:

### 1. Dominance of advertisement on the webpages

Many of these sites have 3<sup>rd</sup> party advertisements, and we think that these advertisements cause a lot of noise as the content itself doesn't make any sense and such kind of noise is difficult to identify and remove in the above mentioned pre-processing phases.

### 2. Lack of textual content on the webpage

Sites having digital media, such as video streaming sites or sites hosting manga comics, don't really have lot of textual content in them; we hoped to utilize the meta-data present as part of the digital content but overall it seems that this rich content is hard to model.

### 3. Presence of “Error” or “Not Found” pages

There are mainly two categories of URLs that we believe fall under this group:

- (a) Those URLs whose content has been taken down and are currently returning a “generic page” or “error page” specific to their domains. It is important to note that in such cases the HTTP response from the server is 200 OK, which makes them valid URLs, but the content does not correspond to the one targeted by the takedown notice.
- (b) URLs that correspond to **empty** search results on their respective domains and have no meaningful content in them. These URLs are directed towards the search engine within the domain. An example of such a URL is:  
<http://torrentz.eu/search?f=the.newsroom.2012.s02e09.720p.hdtv.x264>.

## 4.4 LDA on notices

As a side experiment, we also try running LDA on the descriptions of **50,000** notices. On one hand, these description don’t have different sources of noise as seen with URLs, but on the other hand they are usually very short or generic because they are not mandatory. Although the resulting topics are slightly more coherent (and topics like “Books” also emerge), the majority of the topics still don’t make much sense. For reference some of the interesting topics can be found in the appendix (table A.2).

To conclude, we believe that the current dataset needs to be refined further and probably a modified LDA algorithm might be required to extract meaningful topics out of it. In our next attempt to remove noise, we try to tackle the third issue of “Error” or “Not found” pages using a different machine learning approach. This is further explained in chapter 5.

**Table 4.1.** Top keywords of each topic learned by LDA with 10 topic configuration on crawled URLs dataset.

*We try to figure out the meta-level label of each topic by looking at the keywords. First rows shows these labels and furthermore, we also highlight the important keywords present in each topic.*

Generic	Generic	Generic	File sharing?	Torrents	Manga	Software?	Generic	Music	Generic
ago	link	enlace	check	<b>pirate</b>	part	part	<b>kimi</b>	<b>music</b>	whois
<b>movies</b>	embed	reproduccion	genre	<b>bay</b>	true	<b>windows</b>	hana	<b>remix</b>	icann
tv	track	descargar	net	music	man	dresses	kamisama	<b>mix</b>	<b>porn</b>
tracker	player	reproductor	<b>mb</b>	games	<b>manga</b>	uploaded	hajimemashita	<b>feat</b>	video
wav	code	cdigo	<b>search</b>	proxy	war	<b>adobe</b>	page	facebook	size
video	playlist	lista	facebook	shared	amp	information	<b>manga</b>	original	posts
search	registration	aadir	<b>shared</b>	beach	night	sp	episode	mp	videos
announce	edit	sitio	<b>filetram</b>	facebook	<b>karasumori</b>	<b>microsoft</b>	facebook	love	hd
<b>game</b>	diary	contrasea	link	finally	<b>masamori</b>	rapidgator	read	song	net
hikaru	pleer	pista	account	docked	years	<b>ultimate</b>	anime	video	dominio
<b>xxx</b>	maxpark	composicin	<b>rapidshare</b>	kopimi	change	<b>pro</b>	mib	net	rapidgator
udp	message	archivo	pop	<b>sharing</b>	ago	shipping	utsuho	dj	<b>sex</b>
mp	error	registrarse	<b>mediafire</b>	search	<b>omake</b>	video	itsuwaribito	songs	image
months	blog	embed	<b>depositfiles</b>	movies	zekkai	net	special	vip	part
size	twitter	insertar	hip	cloud	<b>yoshimoris</b>	links	drama	detail	mp
dvdrip	lorem	diary	source	video	human	site	ep	radio	post
view	ipsum	maxpark	feat	applications	child	office	images	house	<b>xxx</b>
xvid	livejournal	pleer	engine	tweet	box	full	home	live	view
latest	blogger	intprprete	hoprap	<b>bittorrent</b>	castle	pc	forum	back	website



## Chapter 5

# Clusters, Clusters, Everywhere

As previously discussed in section 4.3, one source of “noise” is the presence of “Error” or “Not Found” pages, where the page itself does not have any meaningful content in it or is polluted by other “generic” content. Here, by meaningful content we mean the original content that was targeted by the copyright holder for removal. Identifying whether the content on the page is actually the same content that was alleged to be infringing is in itself a challenging problem, which can be tackled in the future.

Figure 5.1 shows an example of a *Not Found* page. It is evident that there is no meaningful content on this page and that the page is polluted by “generic” content, whereby “generic” content is defined as any content that is not relevant to the current page and is present only to make sure the user stays on the page longer or gets redirected to other pages. This image forms the basis of our motivation to experiment further.

We believe that these *Not Found* pages have more or less the same structure and content. Figures 5.1 and 5.2 provide examples of *Not Found* pages present in `filetram.com`. These figures not only show the structural similarities between the *Not Found* pages but also the pollution by “generic” content. Another comparison between figures 5.1 and 5.3 reveals that the *Not Found* pages differ significantly, both structurally and textually, from other pages which have some meaningful content. We hope to leverage this 2 properties by using clustering algorithms, like k-means, to form clusters

of *Not Found* pages and weed them out from the dataset.

The screenshot shows the FileTram search interface. At the top, there are navigation links: All, Audio, Video, Images, Books, Add Source, Alerts, Direct Download, Premium, Log in, and Sign up. The search bar contains the text "mj fields jase". Below the search bar, it says "Results 0 - 0 of about 0 for 'furfk mj fields jase'". A yellow pop-up window is overlaid on the page, advertising a "Special offer!" of 500.00 MB for downloading daily, with a "Sign Up Now!" button. Below the pop-up, a message states: "No results containing all your search terms mj fields jase were found." To the left, there is a list of file-sharing sites including rapidshare.com, 4shared.com, depositfiles.com, mediafire.com, zshared.com, badongo.com, bitshare.com, box.com, crocko.com, extabit.com, fileden.com, filedropper.com, and filefactory.com. At the bottom, there are four columns: "Latest Searches" (panda, mr kitty, bob welch, pixel film studios, solitaire 8 in 1), "Latest Downloads" (4shared - free file sharing and storage, juego\_de\_tronos.mp3, end zone 3.rar, Marvel Zombies vs Army of Darkness #2 (of 5) (2006).cbz, juego\_de\_tronos.mp3, Canción de hielo y fuego 01 - Juego de tronos.pdf), "Hottest Searches" (c v the club, coldplay x y, index.php, depeche mode exciter, bd company, jack ryan), and "Hottest Downloads" (Dead\_Space\_2\_-\_Prima\_Official\_That-Boy.pdf, Jillian-Dodd-That-Boy-2-That-Wedding.pdf, Emma Starr & Xander Corvus - My first sex teacher - Userporn, Video Mesum RINADA PNS Bandung, Frank sinatra - Moonlight Serenade-(ESPECIAL MUSIC 2015)).

**Figure 5.1.** Example of a *Not Found* page present in filetram.com. It is a search result of “mj fields jase”.

All Audio Video Images Books | Add Source Alerts | Direct Download more ▾

Premium Log in Sign up

**File Tram**

tanya anne crosby perfect in my sight

Search Files

Results 0 - 0 of about 0 for 'audio tanya anne crosby perfect in my sight' in seconds

**Audio**

Also Try: [sight dvd](#) [anne](#) [sight dvd](#)

Direct Download  
Mirror #1 USA  
Mirror #2 Europe  
Mirror #3 Russia  
Mirror #4 Asia

Free Download

**Download without waiting from filesharing sites:**

4shared.com  
zippyshare.com  
netload.in  
rapidshare.com  
2shared.com

mediafire.com  
uploading.com  
turbobit.net  
filepost.com  
bitshare.com

**Special offer!**  
You get **500.00 MB** for Downloading daily.

**Sign Up Now!**

close

**No results containing all your search terms tanya anne crosby perfect in my sight were found.**

Your query - **tanya anne crosby perfect in my sight** - did not match any files in our base. Check the spelling of **tanya anne crosby perfect in my sight**. We have a **rip, perfect keylogger** in our base. Last month, our base is

Twitter 8+1 0

Bigger than 1GB

Reset options

**Link Us:**

```
<a target="" blank" href="http://filetram.com">File Search Engine</a>
```

**Search Widget:**

Search and Download.   
FileTram file search

[Get Code >](#)

**Latest Searches**

panda  
mr kitty  
bob welch  
pixel film studios  
solitaire 8 in 1

[more >](#)

**Latest Downloads**

4shared - free file sharing and storage  
juego\_de\_tronos.mp3  
end zone 3.rar  
Marvel Zombies vs Army of Darkness #2 (of 5) (2006).cbz  
juego\_de\_tronos.mp3  
Canción de hielo y fuego 01 - Juego de tronos.pdf

[more >](#)

**Hottest Searches**

c v the club  
coldplay x y  
depeche mode exciter  
bd company  
jack ryan  
inna sun is up

[more >](#)

**Hottest Downloads**

Dead\_Space\_2\_-\_Prima\_Official\_That-Boy.pdf  
Jillian-Dodd-That-Boy-1-That-Boy.pdf  
Jillian-Dodd-That-Boy-2-That-Wedding.pdf  
Emma Starr & Xander Corvus - My first sex teacher - Userporn  
Video Mesum RINADA PNS Bandung  
Frank Sinatra - Moonlight Serenade-(ESPECIAL MUSIC 2015)

[more >](#)

**Figure 5.2.** Example of another *Not Found* page present in filetram.com. It is a search result of “tanya anne crosby perfect in my sight”.

The screenshot displays the FileTram search results for the query 'wilfred'. The page layout includes a top navigation bar with links for 'All', 'Audio', 'Video', 'Images', 'Books', 'Add Source', 'Alerts', 'Direct Download', and 'more'. A search bar at the top left shows the query 'wilfred' and indicates 'Results 1 - 10 of about 1113 for 'wilfred' in 0.4s'. A sidebar on the left provides navigation options like 'All' and 'Premium', and lists various file-sharing sites such as 4shared.com, zippyshare.com, netload.in, rapidshare.com, 2shared.com, mediafire.com, uploading.com, turbobit.net, filepost.com, and bitshare.com. A central search widget is also present. The main content area displays search results with titles, source URLs, and download links. A prominent yellow pop-up box in the center-right offers a 'Special offer!' of 500.00 MB for downloading daily, with a 'Sign Up Now!' button. The search results include titles like 'wilfred.s1d2.tof-seyauno-Scene-W', 'Burn.Notice.S06E09.720p.HDTV.x264-IMMERSE.mkv', 'Wilfred.US.S02E01.Progress.PreAir.720p.WEBRip.H.264-BTN', 'Wilfred.US.S02E01.PreAir.720p.WEBRip.H.264-P2P.mkv', 'Wilfred.US.S02E01.720p.HDTV.X264-DIMENSION.mkv', and 'Wilfred.US.S02E01.Progress.720p.WEB-DL.DD5.1.H264-NTb.m...'. The page also features a sidebar with navigation options, a search widget, and a list of file-sharing sites.

Figure 5.3. Example of a valid search result page for filetram.com.

We now look at the dataset created for this experiment and then discuss some of interesting results obtained.

## 5.1 Dataset

One of the limitations of such kind of analysis is that it also depends on the domain currently being looked at, because each domain has a different mechanism to present a *Not Found* page. Hence, we can not perform clustering on the aggregated dataset of crawled URLs and have to take a *per domain* approach.

As described earlier, in addition to leveraging the textual content of the crawled

URLs (section 4.1), we also want to capture the structure of the page as each domain might be using a template to generate the *Not Found* page. Therefore, we create a new dataset from the raw HTML content that tries to capture the structure of the web page.

### 5.1.1 HTML feature extraction

Given a domain, we believe that the *Not Found* pages have a similar DOM (Data Object Model) structure<sup>15</sup>. We try to take advantage of this structural similarity to improve the cluster quality on a domain. To achieve this, we use the the same methodology as presented by Matt Der [14]. We extract HTML features using a bag-of-words approach ignoring the ordering in which the HTML elements occur and create instances of tokens out of the HTML tags. As an example, consider the following snippet of HTML code:

```

```

Using the feature extraction methodology on the above element, we get the following 4 new tokens for the webpage:

```
img:src=foo.jpg
img:alt=Barpic
img:height=100
img:width=200.
```

Using these two feature extraction techniques, we now have the following two datasets,

#### 1. Only text content

Consisting of just the plaintext obtained from HTML content of the URLs.

---

<sup>15</sup>DOM represents the webpage in a hierarchical format

## 2. Text and HTML features

Captures the DOM structure of the HTML content along with the plaintext content of the URLs.

### 5.1.2 Domain splitting

After creating the two datasets, we split them into sub-datasets on *per domain* basis and restrict them to only those domains that have at least 1,000 documents. These domains are extracted by parsing the URLs of the notices using python's `urlparse`<sup>16</sup> library. Table 5.1 lists all 108 domains that we select for our clustering experiment.

### 5.1.3 Dimensionality reduction

These datasets inherently have high dimensionality, therefore, we use Principal Component Analysis (PCA)[22] to reduce the dimensionality; particularly, we capture 99% of the variance present in the features by picking 200 principal components after normalizing each observation. This is done for all the domains, present in both the datasets, separately.

The traditional PCA algorithm requires a covariance matrix which is an  $F \times F$  matrix where  $F$  is total number of features. On a large scale dataset this results in a high computational overhead. In order to reduce this overhead, we use an Expectation Maximization algorithm for PCA (emPCA)[35]. The MATLAB implementation of emPCA<sup>17</sup> normalizes the input data by subtracting the mean and since, our dataset is very sparse, this step makes the algorithm memory inefficient. Using matrix manipulation we make the code more memory efficient by moving the normalization step inside the EM steps and doing some bookkeeping. The modified code is present in appendix B.

---

<sup>16</sup><https://docs.python.org/2/library/urlparse.html>

<sup>17</sup><http://www.cs.nyu.edu/~roweis/code/em pca2.tar.gz>

**Table 5.1.** List of domains used for k-means clustering.

## Domain Names

<p>pleer.com thepiratebay.net.co wapdam.net mp3vip.org search.4shared.com www.stafaband.info proxypirate.eu www.mangawindow.com www.mangatank.com mp3clan.com mp3sale.ru manga.animea.net bitshare.com kickass.to www.torrenthound.com catshare.net freakshare.com chomikuj.pl www.uploadable.ch raw.senmanga.com mangawall.com www.mangastream.to turbobit.net www.monova.org truemagnet.com torrentus.si vmsice.net</p>	<p>katproxy.com www.4shared.net getwapi.com www.torrentdownloads.me vk.com v2012.mangapark.com shairedir.com mp3forte.com www.mangaeden.com gosong.net extratorrent.cc zamob.com waptrick.com kickassunblock.net ryushare.com mp3wm.com mp3yjp.com www.torrentreactor.net www.citymanga.com www.readmanga.eu www.mangabb.me vodlocker.com h33t.to allmyvideos.net torrentz.eu mangadoom.co kickass torrents.link</p>	<p>rapidsharemix.com www.fullsongs.net rss.hu.lanunbay.org fleshut.com mangable.com www.mp3zone.sk muzofon.com www.mangahere.co terafle.co kickass.isohunt.to www.mangatown.com bitsnoop.com tpbunblocked.me www.ismanga.com thepiratebay.se search.4shared-china.com 5mp3.org www.torrentbit.net uploading.com mrtzcmp3.net wapdam.co torrentz-proxy.com btdigg.org www.dirtyremixes.com www.generalfil.es mp3clan.net sukebei.nyaa.se.proxy.dcmys.kr</p>	<p>www.katshore.org torrentproject.se www.downloads.nl citymanga.com mangaseven.com secureupload.eu vidspot.net mp3days.com youwatch.org www.88torrent.com mp3clan.it thepiratebay.sl kat.works www.sumotorrent.sx vibe3.com 1337x.to kickassstor.net download-torrents.org www.porn-w.org kat.gs vidbull.com torrentz.tf www.videoweed.es www.bayproxy.nl www.weblagu.com www.katshore.nl extratorrent.ee</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



To perform the clustering, we use the MATLAB implementation of k-means<sup>18</sup>. Since there may be other clusters present in the domain (such as valid search result pages vs actual content pages), running k-means with 2 cluster configuration may not always give a *Not Found* cluster even if there are such pages present within the domain. Hence, we run k-means with 3 different configurations:  $K = 2, 5$  and 10.

Once we have the clusters, based on the distance from the cluster centroid, we create a summarized page containing screenshots of the top 10, bottom 10, and random 10 URLs of each cluster. We use this output to perform further analysis.

## 5.2 Results

After clustering is performed, we manually look at the screenshot of the URLs present in the cluster to identify whether they belong to the *Not Found* category or otherwise. Based on this, we identify each domain as belonging/not belonging to the *Not Found* category. Interestingly, during this analysis we also come across 3 other types of domains related to parked pages, age verification and the DMCA compliance domains. In all, we categorize each domain into the following 5 categories:

### 1. Parked pages

These represent domains which have been either completely shut down, have same content on all the URLs, or is a parked domain: domains which have no services associated with them or might be reserved for future development. Figure 5.4 shows an example of a parked page.

### 2. Age verification domains

A handful of torrent related URLs are presented as “age verification” pages before they show the actual content. These pages gets clustered together and we mark such

---

<sup>18</sup><http://www.mathworks.com/help/stats/kmeans.html>

domains as “Age verification” domain. Figure 5.5 shows a sample age verification page.

### 3. **DMCA compliance**

We also come across several domains which have URLs that show unavailability of content due to a copyright infringement complaint and these URLs often get clustered together because they have the exact same structure with minor differences in the content. We mark such domains as “DMCA compliance” domain.

Presence of such clusters show that the copyright holders are not only targeting search engines to remove URLs from their search index but they are also targeting the services that host the content of those URLs. Figure 5.6 shows an example of a DMCA compliance page.

### 4. **Yes “Not Found”**

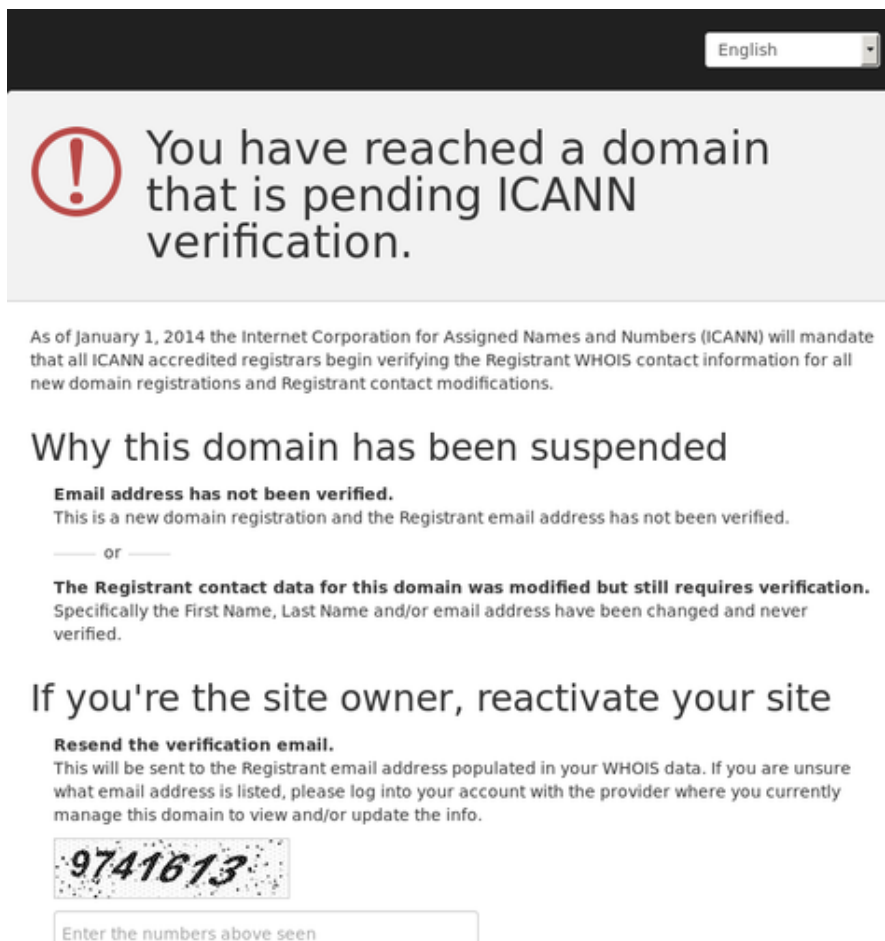
Domains in which there one or more clusters with *Not Found* pages.

### 5. **No “Not Found”**


Domains in which we cannot find any *Not Found* cluster. We believe these broadly represent two kinds of categories:

- (a) Domains that are not hosted in U.S. and therefore their content cannot be targeted under **Section 512**. Example of such a domain could be a Japanese **manga** related website.
- (b) Domains that always return some content for a search query regardless of its availability. Example of such a domain could be a music spamming site which return links for any search query.

Table 5.2 shows the distribution of domains across categories based on manual analysis. As expected, both the features perform equally well in identifying parked pages,



English

 You have reached a domain that is pending ICANN verification.

As of January 1, 2014 the Internet Corporation for Assigned Names and Numbers (ICANN) will mandate that all ICANN accredited registrars begin verifying the Registrant WHOIS contact information for all new domain registrations and Registrant contact modifications.

### Why this domain has been suspended

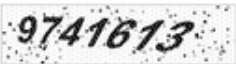
**Email address has not been verified.**  
This is a new domain registration and the Registrant email address has not been verified.

— or —

**The Registrant contact data for this domain was modified but still requires verification.**  
Specifically the First Name, Last Name and/or email address have been changed and never verified.

### If you're the site owner, reactivate your site

**Resend the verification email.**  
This will be sent to the Registrant email address populated in your WHOIS data. If you are unsure what email address is listed, please log into your account with the provider where you currently manage this domain to view and/or update the info.



**Figure 5.4.** Screenshot of a parked page.

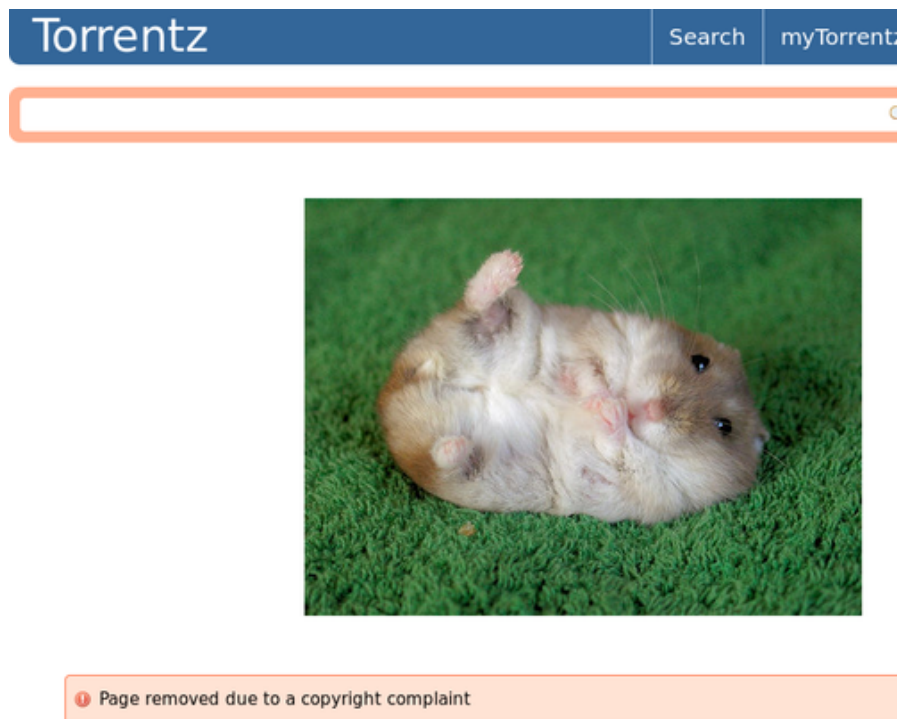
age verification, and DMCA compliance domains but HTML features are more helpful in identifying domains with *Not Found* clusters. In conclusion, it seems that clustering can help, but it is not scalable as there is no magic recipe to determine whether or not the resulting clusters contains a *Not Found* cluster.



**Figure 5.5.** Screenshot of an age verification page.

**Table 5.2.** Category wise distribution of 108 domains based on k-means clustering.

<b>Categories \ Features</b>	<b>Text only</b>	<b>Text and HTML features</b>
<b>Parked pages</b>	9	9
<b>Age verification</b>	5	5
<b>DMCA compliance</b>	7	7
<b>Yes "Not Found"</b>	36	49
<b>No "Not Found"</b>	51	38



**Figure 5.6.** Screenshot of a page showing unavailability of content due to copyright infringement.

# Chapter 6

## Classifying Notices

Revisiting the notices that are crawled from Chilling Effects repository, we now address the second question mentioned in the introduction, that is, we explore the feasibility of labelling the notices in an automated fashion by training a classifier. The primary motivation for this experiment is to obtain a *good enough* labeled dataset out of notices in a completely *automated* fashion. We believe that this dataset might help to learn a labeled variant of LDA[34], and that this labeled approach might reduce the noise and improve the quality of topics.

### 6.1 Dataset

In order to perform this experiment, we must create a new dataset out of the crawled notices from Chilling Effects. One of the prerequisites for training a classifier is the availability of a labeled dataset, which we don't have. Creating a labeled dataset is challenging because there is no ground truth available to us as part of the notices; moreover, we have no concrete idea about the total number of labels present in the dataset.

To tackle these challenges, we try to make some heuristic assumptions. First, as shown in table 6.1, we create a set of 7 labels which we think are more or less representative of the dataset.

**Table 6.1.** Set of labels used to classify the notices.

Labels
Music
Movies
TV Shows
Books
Manga (Comics)
Software
Adult

Second, we *manually* label the notices among those 7 classes. We use the “principal name” (copyright holder) field of notice for classification and in order to reduce the manual effort, we classify all the notices under the same principal name as belonging to the same class; for example, all the notices with *Microsoft* as the copyright holder are labeled as *Software*.

As a sanity check, we manually label a small subset of notices from selected copyright holders; this is done by looking at the content and then classifying it. The results are the same for most of the copyright holders. In the end, we select **40** different copyright holders and label them; table 6.2 contains the list of the copyright holders and their corresponding labels.

Given this list we now split the copyright holders into two sets,

### 1. **Train/Test Set**

Consisting of copyright holders on which the classifier is trained. We also create a validation set out of this to test the accuracy of the classifier on the same copyright holders.

### 2. **Heldout Set**

Consisting of the remaining copyright holders which are not seen by the classifier during the training. Since we would like to train the classifier on a small dataset and use it to classify the remaining notices, the accuracy on the heldout set helps

to measure the ability of the classifier to generalize new and unseen notices.

Referring back to section 3.5, there is an inherent skewness in the dataset based on the number of notices filed by copyright holders. In order to overcome this skewness, we create a uniform dataset by selecting same number of notices from each label. In all, we create the “Train/Test Set” with **5,000** notices from each label (total of **35,000** notices) and “Heldout Set” with **2,000** notices from each label (total of **14,000** notices).

## 6.2 Feature extraction

For this experiment, we continue using bag-of-word representation to extract features from the notices. Furthermore, we create two different datasets by extracting features from the description and URLs present in the claim(s) of the notices.

### 6.2.1 Claim description

For the first dataset, we tokenize the textual content of the descriptions in the claims (works) provided by the copyright holders. Note that, the **DMCA** does not mandate providing a detailed description of the claims; and hence, these descriptions tend to be very short and generic (e.g. “This is a DMCA copyright infringement notice”) or very short and specific (e.g. “The Lion King”).

Given a notice, we concatenate the description of all the claims, then tokenize and sanitize them; we also perform the pre-processing steps defined in section 4.2 to get the final feature vector. From here onwards, we refer to the dataset created using this methodology as **Description Only** dataset.

### 6.2.2 Tokenized URLs

For the second dataset, we leverage the URLs present in the notice. Many of these infringing URLs contain some information as part of their address itself. For example,



consider the following URL:

```
www.downtr.co/1526608-bbc-top-gear-magazine-uk-december-2012.html
```

Inspection of the URL itself gives some insight about the content it represents; clearly this URL is related to the “Top Gear TV Show.” We try to capture this information by tokenizing the URL and sanitizing it by removing “stopwords” and numbers. For the above example, we get the following tokens:

downtr

bbc

top

gear

magazine

uk

december

Given a notice, we now concatenate the tokens from the URLs along with the those obtained from the descriptions of the claims to form the final feature vector. From here onwards, we refer to the dataset created using this methodology as **URL Features** dataset.

## 6.3 Classifiers

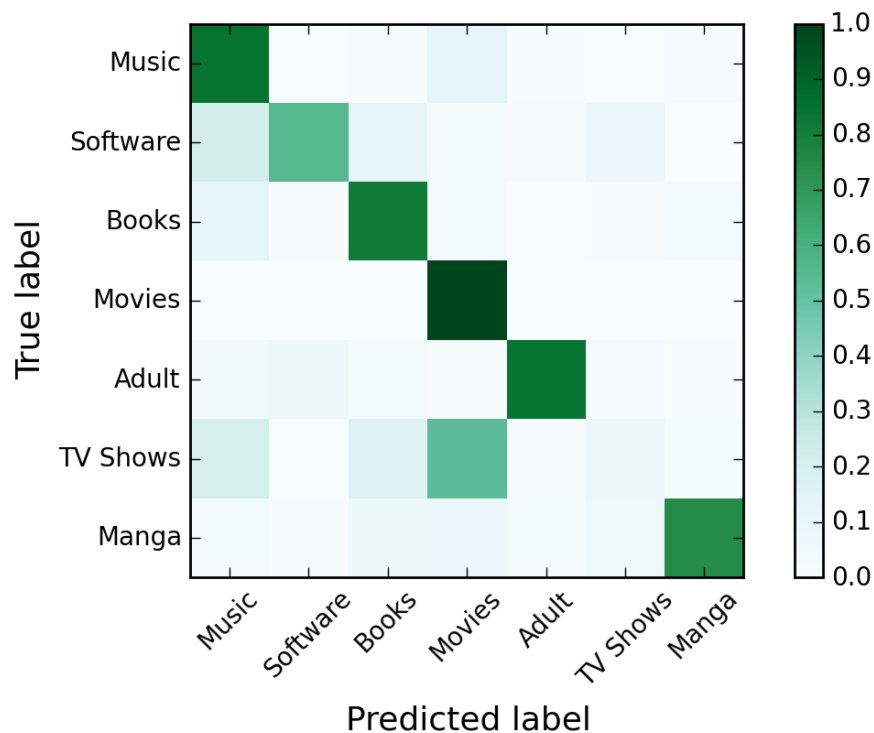
We use the two very popular classification techniques; Support Vector Machines (SVM) and Random Forests (RF), both of which have been briefly reviewed in section 2.3. Instead of implementing the algorithm from scratch, we use an opensource python framework, `scikit-learn`<sup>19</sup>, which has multi-class implementation of both the algorithms.

---

<sup>19</sup><http://scikit-learn.org/stable/>

## 6.4 Super Video category

Our initial experimentation revealed that the classifier is getting confused between TV Shows and Movies. Figure 6.1 shows a confusion matrix of the classifier on a heldout set of copyright holders depicting a high confusion between TV Shows and Movies. This is not totally surprising as both these meta-level classes represent media content. Therefore, as an additional experiment, we combine these two labels into a single category called “Super Video” and train/test our classifier on the above mentioned datasets. Overall, we now have 2 new datasets, based on the feature extraction, comprising of only 6 labels. We name them “Super Video Description only” and “Super Video URL Features.”



**Figure 6.1.** Confusion matrix of SVM on all Heldout notices, providing motivation for “Super Video” category.

## 6.5 Experiments

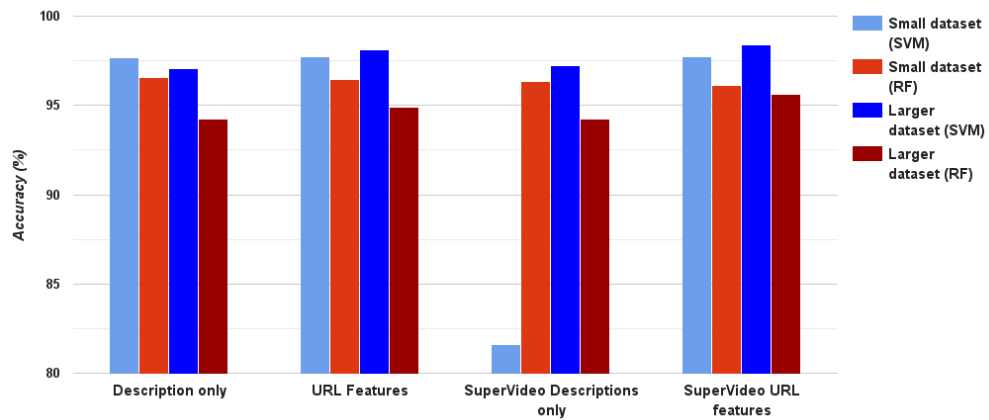
Combining everything, we test the accuracy of each classifier (SVM and RF) on 4 different feature related datasets (“Super Video Description only”, “Super Video URL Features”, “Description only”, and “URL Features”). Each of these feature related dataset have 2 different categories (“Train/Test” and “Heldout”). Furthermore, each of these two category has two different versions, a “Smaller Dataset” which has same number of notices for every label and a “Larger Dataset” which has *all* the notices for all the labeled copyright holders. For “Larger Dataset”, “Train/Test” consists of a total of **332,894** notices and “Heldout” consists of **106,639** notices.

## 6.6 Results

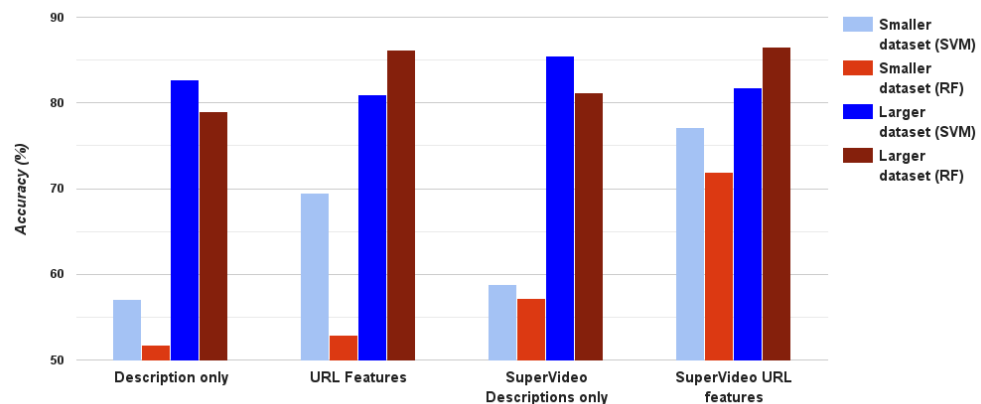
Figure 6.2 shows the accuracies on the test set, which contains the same copyright holders that is used to train the classifier and figure 6.3 shows the accuracies on the heldout set. Looking at the graphs we make couple of observations,

1. As expected, we obtain fairly high accuracy when testing the classifiers on “Test Set”. We get a maximum accuracy of 98.3% from SVM classifier on “Super Video URL Features” test set. This result provides a sanity check as we expect the classifier to correctly identify classes, with high accuracy, on the copyright holders which it has already seen.
2. Figure 6.3 shows, however, that the classifiers do not perform that well on unseen copyright holders. Introducing URL features has certainly helped improve the accuracy but it is still under 80% on smaller datasets. On the other hand, if we compare the accuracies of “Smaller Dataset” vs “Larger Dataset”, there seems to be a significant improvement in the accuracy, for example a rise 56% to 86% for

the random forest classifier using “URL Features.” We attribute this rise to the fact that there is an inherent skewness in the dataset as described in section 3.5 and if a classifier is able to correctly identify classes with more notices, such as Music and Adult, than the accuracy will naturally increase.



**Figure 6.2.** Accuracies for Test datasets using different features.



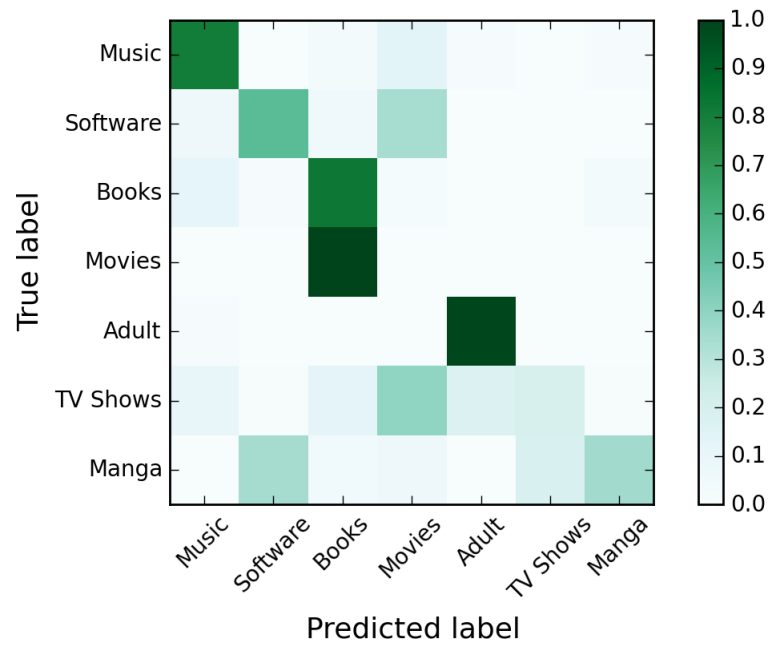
**Figure 6.3.** Accuracies for Heldout datasets using different features.

Furthermore, by looking at figures 6.4a and 6.4b we can compare the confusion matrices before and after combining **Movies** and **TV Shows** into **Super Video** category respectively. This comparison shows that fusion of the two categories does indeed help reduce the confusion the random forest classifier.

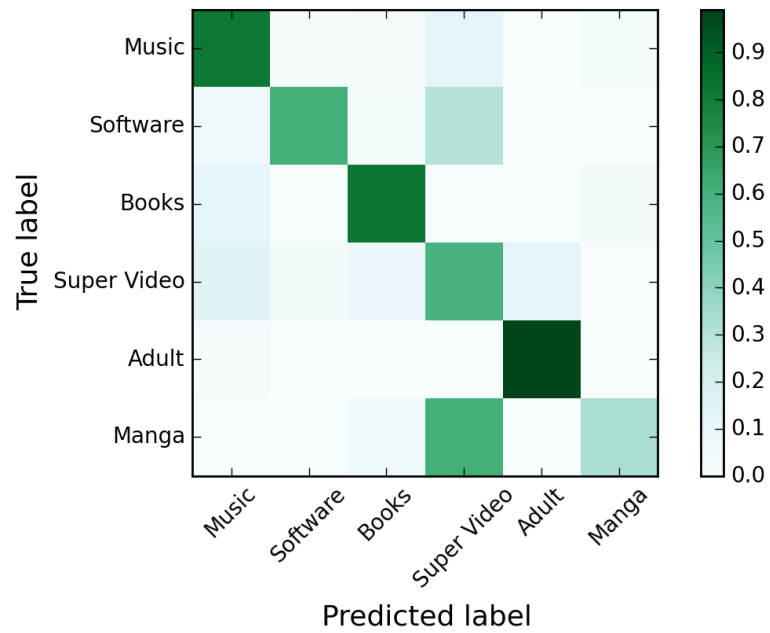
In summary, we observe that there exist some categories such as *Music* and *Adult* that are easy to classify and others such as *Movies*, *Software* and *TV Shows* that are not. We believe that one of the factors might be insufficient training data, for example *Software* category do not have lot of copyright holders; the dataset mainly consists of notices from Microsoft and Adobe, hence the classifier is not performing on those labels. Overall, we see that, SVMs outperform random forests in “Smaller Dataset(s)” and “Description only” features, whereas, random forests takes the lead with “Larger Dataset” and “URL Features.”

**Table 6.2.** Manual labels of copyright holders

<b>Copyright Holder Name</b>	<b>Label</b>
BPI (British Recorded Music Industry) Ltd	Music
Microsoft Corp.	Software
RIAA member companies	Music
BPI LTD MEMBER COMPANIES	Music
The Publishers Association	Books
Paramount	Movies
Twentieth Century Fox Film Corporation	Movies
BangBros.com Inc.	Adult
Home Box Office	TV Shows
Columbia Pictures Industries Inc	Movies
Random House	Books
Shueisha	Manga
Microsoft	Software
Paramount Pictures Corporation	Movies
Hachette Livre	Books
microsoft	Software
Pearson Education, Inc.	Books
Kodansha	Manga
MARC DORCEL	Adult
Pornostar Records	Adult
Adobe	Software
CBS	TV Shows
Froytal Services Ltd	Adult
CA Co., Ltd.	Adult
IFPI	Music
Reed Elsevier	Books
Beggars Group Digital Ltd	Music
TheEsa	Software
Hydentra L.P	Adult
Editis	Books
Harper Collins	Books
GGCash	Adult
METI	Manga
MX International Inc	Manga
Sony Pictures Classics Inc	Movies
Viacom Media Networks	TV Shows
VIZ Media LLC	Manga
Take Two Interactive	Software
Musical Freedom	Music
RCN TV	TV Shows



(a) URL Features



(b) Super Video URL Features

**Figure 6.4.** Comparison of confusion matrices from random forest (RF) classifier using 6 vs 7 categories

# Chapter 7

## Conclusion

In this work, we take a systematic look at the **Digital Millenium Copyright Act**, specifically **Section 512** of **DMCA**, which provides an extra-judicial mechanism for copyright holders to take down allegedly infringing material by issuing notices to **Online Service Providers**. We crawl more than 2 million notices from Chilling Effects repository; we then select a subset of 10,000 notices to crawl the content of approximately 3 million infringing URLs. We then apply different machine learning techniques in order to find interesting patterns within the dataset.

First, we try to extract *latent* topics present within the URLs by learning a topic model on the crawled content of the URLs. We perform several pre-processing steps, like custom stopword removal, cutting the *longtail* of low frequency words and removing very large and small documents, before learning the final model. The results show that the dataset is still dominated by “noisy” content. In an effort to remove this “noise”, we experiment with clustering techniques on *per domain* basis to identify *Not Found* (noisy) clusters and remove them from the dataset. Our results reveal that not all domains have *Not Found* clusters and the approach itself is not very scalable. However, this experiment served as a proof of concept that with effort we can remove some “noisy” content from the dataset.

Furthermore, we also try to automatically classify the notices based on features



extracted from the descriptions and the URLs of the claims. We find that, there exists an inherent difficulty in curating the dataset for such an experiment because we do not have ground truth labels or even know the total number of classes present in the dataset. Using some heuristic approximations, we create a different datasets with 6 and 7 categories, consisting of 40 different copyright holders. We obtain a maximum accuracy of around 86% on a heldout set consisting of copyright holders never seen by the classifier.

Overall, we conclude that applying machine learning techniques on this dataset proved to be more challenging than we expected. We believe this is because of the following inherent properties of the dataset:

1. Presence of heavy bias in terms of notices filed by copyright holders.
2. Lack of rich textual content in the notices as well as the URLs.
3. Dominance of “noise” in whatever content is available.

and many others. This in itself opens up lot of opportunities for future research.

Starting with classification, future research can involve:

1. Creating a more representative dataset by having more notices from under represented classes (Software, TV Shows and others).
2. Using an external signal for training, such as IMDB<sup>20</sup> to get the list of Movies and TV Shows.
3. Obtaining tags for URLs from services like AlchemyAPI<sup>21</sup>.

Clustering algorithms on this labeled dataset might be able to weed out *Not Found* pages within a given category. Once the dataset is less “noisy”, topic models can be

---

<sup>20</sup><http://www.imdb.com>

<sup>21</sup><http://www.alchemyapi.com/products/alchemy/language/keyword-extraction>

learned on a per category basis to reveal more fine grained patterns within each category. This in turn could answer some of the more interesting questions pertaining to the usage of **Section 512** by the copyright holders, for example:

1. What kind of music is mainly targeted by the copyright holders? Is it Rock, Electronica or Pop? Are they targeting more recent hits than classics?
2. Is there any particular pattern according to which Movie production houses target infringers? Like, what genre of movies are more heavily targeted? Are blockbuster movies targeted more aggressively?
3. Is DMCA being abused by copyright holders to take down content of competitors in order to gain an unfair advantage?

These and many more interesting questions are left for future work.

# Appendix A

## Additional topics

In this chapter we present additional topics that are learned by the topic models covered in chapter 4. Table A.1 lists down some interesting topics discovered when the topic model was configured with 50 topics. Table A.2 lists a subset of topics when we learn a topic model on the description of the notices instead of crawled content of URLs.

**Table A.1.** A subset of topics from the topic model learned on crawled URL content and configured to find 50 topics.

*Some of these topics represent different genres of “Music” and some of them are clearer than those in smaller models with only 10 topics.*

DJ Music	Pop-Rock Music	Adult	Metal/Rock Music	Movies	Software/Games	Software	File Sharing
remix mix original feat dj radio club edit house love music mp martin van alex extended mashup david vk	black man night heart fire baby net music wanna kiss britney spears facebook steve king coldplay brothers amy rain	porn sex xxx big anal video mp hd scene hot angel ass videos movies girls dvdrip fuck evil tits	day green dead report eur size link player date metallica nirvana forever queen aerosmith system webmoney checkout perfectmoney jovi	movies bluray ac xvid brrip dvdrip movie hdrrip age audio transformers extinction kill city video bdrip locke sin posted	part game games pc edition repack software read search rapidshare rapidgator full uploaded ebooks loading wait xbox ps music	windows part sp microsoft adobe pro ultimate office bit cs kb professional iso rus photoshop update applications mac letitbit	filetram search facebook account net shared engine google user log mb terms video mediafire sign zippyshare filepost searches netload

**Table A.2.** A subset of topics from the topic model learned on description of notices and configured with 50 topics.

*Interestingly, these topics are more coherent than their URL-related counterparts.*

<b>Books</b>	<b>Pop-Rock</b>	<b>Music</b>	<b>Rock Music</b>	<b>Generic</b>	<b>Games</b>	<b>TV Shows</b>	<b>Alternative Rock</b>	<b>Adult</b>	<b>Software</b>
james david robert michael patterson stephen king john kare kano connelly dean nicholas number series sparks night baldacci meyer	britney spears bruce springsteen bryan adams love brad paisley brunoz bruno mars bone men ii thugs jovi wow bow	snoop dogg pumpkins smashing patrol snow steve shakira system spice love girls smiths stevie stereophonics boy sum soundgarden life	pink paul floyd pitbull love offspring outkast nirvana phil jam pearl mccartney oasis collins prince paramore direction boys onerepublic	list representative url copyrighted website ii official includes sec pursuant usc works original dvd television broadcasts anime releases llc	xbox ps nds pc wii android psp ios mario mac ds super fifa grand dance rayman dead war dragon	worst runner american men family planet apes simpsons horror day days thief mr mother book dawn story dragon cleveland	david hill coldplay cypress bowie dave love cranberries cure craig daughtry matthews punk creedence clearwater daft revival counting common	de adult se yang dan website online caetano veloso entertainment graphic language activity sexual nature nudity explicit descriptions depictions	microsoft office sound age hammock original star trek windows movie life pure rus deep saint red golden transformers giant

## Appendix B

### EMPCA MATLAB code for PCA

```
function [evec ,eval] = empca_ef(data ,k,iter ,Cinit)
%[evec ,eval] = empca_ef(data ,k,iter ,Cinit)
%
% EMPCA Efficient
% finds the first k principal components of a dataset
% and their associated eigenvalues using the EM-PCA
  algorithm
% instead of creating a normalized matrix , it performs
% normalization at every iteration and hence is more
  memory
% efficient when input data is in sparse format
%
% Inputs:  data is a matrix holding the input data
%          each COLUMN of data is one data vector
%          k    is # of principal components to find
%
% optional:
%          iters is the number of iterations of EM to run
  (default 20)
%          Cinit is the initial (current) guess for C (
  default random)
%
% Outputs:  evec holds the eigenvectors (one per column)
%          eval holds the eigenvalues
%
```



```

[d,N] = size(data);

% setting default parameters
if(nargin < 4)
    Cinit = [];
end

if(nargin < 3)
    iter = 20;
end

% running the EM algorithm
C = empca_iter(data, Cinit, k, iter);

% extracting the eigenvectors and eigenvalues
[vec, eval] = empca_orth(data, C);

function [C] = empca_iter(data, Cinit, k, iter)
%[C] = empca_iter(data, Cinit, k, iter)
%
% EMPCA_ITER
% (re)fits the model
%
% data = Cx + gaussian noise
%
% with EM using x of dimension k
%
% Inputs: data is a matrix holding the input data
%         each COLUMN of data is one data vector
%         NB: DATA SHOULD BE ZERO MEAN!
%         k is dimension of latent variable space
%         (# of principal components)
%         Cinit is the initial (current) guess for C
%         iters is the number of iterations of EM to run
%
% Outputs: C is a (re)estimate of the matrix C
%          whose columns span the principal subspace
%
% check sizes and stuff

```

```

[p,N] = size(data);
assert(k<=p);
if isempty(Cinit)
    C = rand(p,k);
else
    assert(k==size(Cinit,2));
    assert(p==size(Cinit,1));
    C = Cinit;
end

% calculate mean of data
dmean = mean(data,2);

% business part of the code -- looks just like the
% math!
tic;
for i=1:iter
    % e step -- estimate unknown x by random
    % projection
    A = (C'*C)\C';
    x = bsxfun(@minus,A*data,A*dmean);
    % m step -- maximize likelihood wrt C given these
    % x values
    B = x'/(x*x');
    C = data*B - (dmean * (ones(1,N) * B));
toc
end

```

```

function [evec,eval] = empca_orth(data,C)
%[evec,eval] = empca_orth(data,Cfinal)
%
% EMPCA_ORTH
%
% Finds eigenvectors and eigenvalues given a matrix C
% whose columns span the principal subspace.
%
% Inputs:  data    is a matrix holding the input data
%          each COLUMN of data is one data vector
%          NB: DATA SHOULD BE ZERO MEAN!

```

```

%           Cfinal is the final C matrix from empca.m
%
% Outputs: evec,eval are the eigenvectors and eigenvalues
%         found
%         by projecting the data into C's column space
%         and finding and
%         ordered orthogonal basis using a vanilla pca
%         method
%
C = orth(C);
[xevec,eval] = truepca(C'*data);
evec = C*xvec;

function [evecs,evals] = truepca(dataset)
% [evecs,evals] = truepca(dataset)
%
% USUAL WAY TO DO PCA -- find sample covariance and
% diagonalize
%
% input: dataset
%       note, in dataset, each COLUMN is a datapoint
%       the data mean will be subtracted and discarded
%
% output: evecs holds the eigenvectors, one per column
%        evals holds the corresponding eigenvalues
%
[d,N] = size(dataset)

mm = mean(dataset',' );
dataset = dataset - mm*ones(1,N);

cc = cov(dataset',1);
[cvv,cdd] = eig(cc);
[zz,ii] = sort(diag(cdd));
ii = flipud(ii);
evecs = cvv(:,ii);
cdd = diag(cdd);
evals = cdd(ii);

```

# Bibliography

- [1] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.
- [2] Michael R Anderberg. Cluster analysis for applications. Technical report, DTIC Document, 1973.
- [3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
- [4] American Library Association. DMCA: The Digital Millennium Copyright Act. <http://www.ala.org/advocacy/copyright/dmca>.
- [5] Michael W Berry and Jacob Kogan. Text Mining. *Applications and Theory*. West Sussex, PO19 8SQ, UK: John Wiley & Sons, 2010.
- [6] David M Blei and John D Lafferty. Topic Models. *Text Mining: Classification, Clustering, and Applications*, 10:71, 2009.
- [7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning research*, 3:993–1022, 2003.
- [8] Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Lawrence D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *International Conference on Pattern Recognition*, pages 77–77. IEEE Computer Society Press, 1994.
- [9] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [10] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [11] Wray L. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

- [12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [13] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
- [14] Matthew F Der, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Knock it off: Profiling the Online Storefronts of Counterfeit Merchandise. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1759–1768. ACM, 2014.
- [15] James M Dickey. Multiple Hypergeometric Functions: Probabilistic Interpretations and Statistical Uses. *Journal of the American Statistical Association*, 78(383):628–637, 1983.
- [16] Weiguo Fan, Linda Wallace, Stephanie Rich, and Zhongju Zhang. Tapping the Power of Text Mining. *Communications ACM*, 49(9):76–82, September 2006.
- [17] Thomas L Griffiths and Mark Steyvers. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.
- [18] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The Elements of Statistical Learning*, volume 2. Springer, 2009.
- [19] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57. ACM, 1999.
- [20] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data Clustering: a Review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.
- [21] Thorsten Joachims. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Springer, 1998.
- [22] Ian Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2002.
- [23] Do-kyum Kim. Topic Modeling of Hierarchical Corpora, 2014. Copyright - Copyright ProQuest, UMI Dissertations Publishing 2014; Last updated - 2015-04-06; First page - n/a.
- [24] Legal Information Institute, Cornell Law University. 17 U.S. Code Section 512. <https://www.law.cornell.edu/uscode/text/17/512>.
- [25] Stuart Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

- [26] Arnold P. Lutzker. *Primer on the Digital Millenium*. Lutzker & Lutzker LLP, <http://goo.gl/ISqamh>, March 1999.
- [27] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.
- [28] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The Planar k-means Problem is NP-hard. In *WALCOM: Algorithms and Computation*, pages 274–285. Springer, 2009.
- [29] Andrew Kachites McCallum. Mallet: A Machine Learning for Language Toolkit. <http://mallet.cs.umass.edu>, 2002.
- [30] Thomas Minka and John Lafferty. Expectation-Propagation for the Generative Aspect Model. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.
- [31] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.
- [32] J.M Peña, J.A Lozano, and P Larrañaga. An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognition Letters*, 20(10):1027 – 1040, 1999.
- [33] Martin Ponweiser. Latent Dirichlet allocation in R. 2012.
- [34] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.
- [35] Sam Roweis. EM algorithms for PCA and SPCA. *Advances in Neural Information Processing Systems*, pages 626–632, 1998.
- [36] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 427(7):424–440, 2007.
- [37] Yee W Teh, David Newman, and Max Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1353–1360, 2006.

- [38] Jennifer M. Urban and Laura Quilter. Efficient Process or Chilling Effects - Take-down Notices under Section 512 of the Digital Millennium Copyright Act. *Santa Clara Computer & High Tech Law Journal*, 621(22), 2005.