Sentiment-driven Topic Analysis of Song Lyrics

A Thesis
Submitted For the Degree of
Master of Science (Engineering)
in the Faculty of Engineering

by

Govind Sharma

Department of Computer Science and Automation
Indian Institute of Science
BANGALORE – 560 012

AUGUST 2012

DEDICATED TO

my
Guru;
Maiyya Ji,
Baba and *Maa*,
Papa, Mummy, Uncle,
Gopal Uncle, Umesh Uncle,
Khushi Aunty and *Soni Aunty*,
Jitu, Shweta, Baabli and *Gunnu*,
Kaanhu and *Raghav* and *Chhutki*;
in short, my family members, who
blessed me with virtues of honesty
and truth, kindness and integrity,
moral values and selflessness,
and made a temple in
my heart.

।। मङ्गलाचरण ।।
गुरुर्ब्रह्मा गुरुर्विष्णुः गुरुर्देवो महेश्वरः ।
गुरुः साक्षात्परब्रह्मा तस्मै श्री गुरुवेनमः ।।

—— maṅgalācaraṇa ——

guruḥ brahmā guruḥ viṣṇuḥ guruḥ devo maheśvaraḥ

guruḥ sākṣātparabrahmā tasmai śrī guruvenamaḥ

Acknowledgements

It is impossible for me to express gratitude towards my well-wishers in mere words. But since it is customary, I would like to start with my *Guru*, Prof. M. Narasimha Murty, who spent an enormous amount of time nurturing my thoughts and was never too busy (no tooth too achy) to skip a meeting with we students. I cannot thank him here, not in words; but can only do so by doing something which he feels proud of.

Then comes my family, an infinite sea of love, affection, trust, devotion and what not. It will take volumes to express anything about them. It may seem that I am exaggerating the incapability of words, but believe me, words are shallow. The best I could do is to “speak by silence” in the following little space:

$\Big\{ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big\}.$

Now let me follow a chronological order. I would like to thank my Physics and Computer Science teacher Mr. Rajesh Keshav for teaching me how to learn. Next, I would express my gratitude to my friend and college-mate, Mr. Vasav Joshi for giving me an opportunity to work as an intern at CAIR, DRDO, which respectfully elevated my position in AI from ‘being-interested’ to ‘doing-something-about-it’. At CAIR, I was blessed to have got the guidance of Dr. Dipti Deodhare, who is an ocean of optimism and inspiration. I want to thank her, not because I was one of the authors in her research paper, but because she further filled me with excitement for AI and motivated me to pursue higher studies in this field.

I would like to go one step further to thank Prof. Narahari, Prof. Chiranjib, Prof. Shivani, Prof. S. Govindarajan and Prof. Raghavan at CSA, IISc who had interviewed me and had selected me for the research program at IISc. I can’t thank enough, Dr. Moon

Chetry of CAIR, DRDO for having suggested me to credit courses from the Dept. of Mathematics. For each and every course that I had attended during the first 3 semesters at IISc, I would like to deeply thank Prof. Bharali (Dept. of Mathematics), Prof. Vittal Rao (CEDT), Prof. Chiranjib, Prof. Indrajit, Prof. Veni Madhavan and Prof. Narasimha Murty. During the selection of an area for my research, Mr. Cohan Sujoy Carlos of AIAIOO Labs, Banglaore was really helpful, for which I am heartily grateful.

I cannot forget the time I had spent with Prof. Narasimha Murty, Prof. Veni Madhavan and Prof. Gopinath, endlessly discussing interesting topics. I still remember my first meeting with Prof. Veni Madhavan, which lasted for 4 continuous hours! I will always remember Prof. Indrajit for consoling me after I gave an extremely terrible performance in the Probability course. I would also like to thank him for his invaluable help in shaping up this thesis. I can't even imagine preparing for the General Test without the help and hard work of Prof. MNM. I am also grateful to Prof. Rajesh Sundaresan (ECE), Prof. Shevade and Prof. Narahari for passing me in the General Test.

During my stay at IISc, I made some great friends, who kept me oblivious of the fact that I wasn't at home. Some of them, whom I want to thank are Deepak *Bhaiya*, Rohit *Bhaiya*, Rajendu *Bhaiya*, Prashant, Avinash and Sharad. I would also like to thank my lab-mates, Karthik, Adi, Suresh, Madhav, Punya, Charan, Prabhakaran, Deepak and Jyoti for their charm which kept our lab fresh.

I can never forget Jitu, Shweta and Harish who are the best siblings one could ever imagine having. Shweta has even composed the three proses given at the end. My friends Kartik, Aayush, Abhimanyu, Varun, Vivek, Vikrant and Akanksha, whose ears always acted as sinks for my 'gibberish', are the best of friends one could ever have. Let me also express my gratitude towards the staff of CSA, including the security guards, who greeted each day of mine with a smiling face[1]. I still remember calling CSA office, being worried on finding the wrong spelling of my name in the list of "selected candidates for research." Lalitha Ma'm replied, "Don't worry. It's you only!"

[1]Might seem similar to the line in Mr. Rupesh Nasre's Ph.D. thesis. It's a universal truth.

Publications based on this Thesis

1. Govind Sharma and M. Narasimha Murty, "**Mining Sentiments from Songs using Latent Dirichlet Allocation**," In: *Proceedings of the 10th International Symposium on Intelligent Data Analysis, IDA 2011*, Porto, Portugal, pp. 328–339, LNCS 7014, Springer, October 2011

Abstract

Sentiment Analysis is an area of Computer Science that deals with the impact a document makes on a user. The very field is further sub-divided into Opinion Mining and Emotion Analysis, the latter of which is the basis for the present work. Work on songs is aimed at building affective interactive applications such as music recommendation engines. Using song lyrics, we are interested in both supervised and unsupervised analyses, each of which has its own pros and cons.

For an unsupervised analysis (clustering), we use a standard probabilistic topic model called Latent Dirichlet Allocation (LDA). It mines topics from songs, which are nothing but probability distributions over the vocabulary of words. Some of the topics seem sentiment-based, motivating us to continue with this approach. We evaluate our clusters using a gold dataset collected from an apt website and get positive results. This approach would be useful in the absence of a supervisor dataset.

In another part of our work, we argue the inescapable existence of supervision in terms of having to manually analyse the topics returned. Further, we have also used explicit supervision in terms of a training dataset for a classifier to learn sentiment specific classes. This analysis helps reduce dimensionality and improve classification accuracy. We get excellent dimensionality reduction using Support Vector Machines (SVM) for feature selection. For re-classification, we use the Naïve Bayes Classifier (NBC) and SVM, both of which perform well. We also use Non-negative Matrix Factorization (NMF) for classification, but observe that the results coincide with those of NBC, with no exceptions. This drives us towards establishing a theoretical equivalence between the two.

Contents

List of Tables

List of Figures

Keywords

Sentiment Analysis, Emotion Analysis, Music Information Retrieval, Music Recommendation Engine, Clustering, Classification, Latent Dirichlet Allocation, Dimensionality Reduction, Topic Models, Support Vector Machines, Naïve Bayes Classifier, Nonnegative Matrix Factorization

Chapter 1

Introduction and Motivation

1.1 Sentiment Analysis

The inherence and inevitability of sentiments in our life is squarely unquestionable. In addition to being omnipresent, sentiments are so subjective that conceptually, they even differ (though very subtly) from "emotions". Sentiment Analysis is an area of pattern recognition whose primary objective for a set of documents is to infer sentiments of their *users* and/or their *providers*. As examples, in case of movie-reviews (text) posted on the internet, *users* are readers and *providers* are writers; whereas for songs (audio), *users* are listeners and *providers* are none but song artists (singers, musicians, *etc.*). Using computer science for sentiment analysis is really the state-of-the-art, due to the humongous amount of data which is nearly impossible to be analysed by humans alone. Sentiment analysis gained popularity with the conception of Opinion Mining [2, 3], which is used to track the fame or infamy of a certain product by automatically analysing online user-opinions about it. For example, a movie producer would like to keep track of the popularity of his movie, for which he could resort to reading user-reviews on social websites. But analysing thousands of comments/reviews manually is not that trivial. Also, pin-pointing the pros and cons in terms of constituents (acting, direction, story, screenplay, *etc.*) of a movie is more difficult manually. Through opinion mining, it was possible to build models that tell a positive opinion from a negative one.

Recently, due to the dominance of its use, opinion mining has become almost a synonym for sentiment analysis, while it actually is a meronym. This is because analysing sentiments also refers to a more subjective domain that involves emotions. While opinions are classified as being *positive*, *negative* or *neutral*, emotions are richer, both quantitatively and qualitatively. A website [4] on songs reports as many as 150 emotions related to songs. However, one would require a highly subjective (nearly human!) computer to "understand" each one of them. Research in this field has not been on a very substantial scale. Mihalcea, *et al.* [5, 6] have been working in this field, dealing with emotions like happiness and sadness to draw interesting conclusions about human psychology. In addition to its applications in psychological studies, emotion analysis can largely benefit the entertainment/arts industry.

1.2 Music and Sentiment Analysis

Music is the most widely adopted form of entertainment people resort to. In the modern era, where we witness the existence of a large amount of music, it has become really challenging to choose from a bunch of unheard songs. Most commonly, we get introduced to new music by our acquaintances, who are generally aware of our "taste" for songs and suggest them to us appropriately. Otherwise, we stumble upon a random song, which we may like, and then start following the artist/album for other similar ones. But there seems to be a problem here. It totally depends on chance, whether a listener knows about a new song. For example, if I launch a new song in the market, it could very well get hidden amidst highly advertised songs; and a very few people get to know about it. Even with the advent of technology, there are no strong means to recommend a listener new songs qualitatively effectively. Moreover, from tens of thousands of songs in a song-library, most are never listened to; and we keep listening to favourite songs repeatedly. This leads to a large amount of songs being pushed towards "exile". There is thus a need for an automated subjective analyst which can recommend songs based on the "taste" of a listener. This calls for a sort of mood detector based on the songs one is listening to. In order to achieve this, we need to know the emotions associated with each song, which is the motivation for the present work.

1.3 Melody or Lyrics?

Songs can be looked upon as a combination of audio (melody) and text (lyrics). Melody comprises of the vocals, the instruments, the *raaga*, *etc.* whereas lyrics are just plain text. What exactly leads to our liking/disliking of a song and gives it the sentimental flavour it has? Actually, both melody and lyrics play their roles in making a song popular and defining its sentiment. Melody "attracts" people towards a song, playing the maximum role in the beginning. If the tune is catchy, people would want to listen to it. Lyrics are not paid much attention to initially. But as we continue listening to a song, we start liking/disliking the lyrics. Lyrics decide whether or not a song would last in the "favourites" playlist of a listener. As a dialogue from a popular Hollywood movie [7] goes by, "*A melody is like seeing someone for the first time. [...] But then, as you get to know the person, that's the lyrics, their story, who they are underneath.*"

Research goes on in both melody and lyrics for analysing sentiments. Signal analysts work successfully on melody and get good results; and as a matter of fact, most researchers have used melody for such analyses (see Sec. 1.4). We have chosen to work on lyrics alone, not to compare against, but to go hand in hand with melody followers! We feel that a combined approach using both melody and lyrics would integrate their pros and give us a better system.

1.4 Related Work

1.4.1 Sentiment Analysis' First Steps!

Music has been under the "microsonic"[1] ears of emotive psychologists for more than a century now [8, 9]. Almost all the work in this field was done manually, and with a

[1] (fictitious word) sonic analogue of microscopic

research (as opposed to application) perspective. It was in the late 1980's that these analyses started aiding the advertisement industry. But researchers looked at pure melody and did not pay much attention to lyrics. Even when lyrics were considered, only prosodic aspects were utilized, being oblivious to their meaning. Music was analysed for applications ranging from marketing to cultural studies. Balkwill, et al. [10] discuss the culture-specific perception of music, including Hindustani music and the concept of mood or *rasa*. With the advent of digital music, computational analysis became possible, involving computer programs to analyse music. Most applications used the melodic (audio) aspects of music and studied the variation of sentiments with it. The first (according to [11]) work on Music Information Retrieval was carried by Kageyama, et al. [12], who developed a system that took a hummed melody as input and gave melodically matching songs as output. By the end of 20th century, information retrieval in music had gained popularity and the first International Symposium on Music Information Retrieval (ISMIR) 2000 was conducted. Since then, ISMIR is conducted every year, with music information retrieval as their primary motive.

1.4.2 Present State-of-the-Art

Although similar works had been carried out by [13, 14] in the 1990's itself, opinion mining was initiated (to our knowledge) by Wiebe, et al. [15]. Then, suddenly the area gained huge attention by machine learning researchers [16, 3, 17, 18, 19] who took the field into limelight. Although the field of emotion analysis was not too slow [20, 21, 5, 6], it was overshadowed by opinion mining, due to the latter's demand in the market.

Emotional analysis of music went on in parallel with opinion mining. With the psychological studies on affective impact of music continuing from about a century ago [22, 23], research also started in automatic detection of genre, mood, *etc.* With the development of efficient pattern recognition algorithms, corpus-based and knowledge-based models started coming into being [24, 25, 26]. Also, a few works on building "emotion-aware" music players were carried out, which gave a push to the field. In the commercial world, online music vendors such as Last.FM [27], AllMusic [4], started providing emotion-based (though labelled manually) music recommendation to their users. Most of the algorithms built earlier were based on metadata (*e.g.*, title, artist, album, *etc.*). Systems such as MyStrands [28], iTunes Genius [29], iLike [30], Last.fm [27], *etc.* either use metadata, or keep track of ratings, play counts, like/dislike information, *etc.* to recommend music. Some recent systems such as Musicovery [31], iPlayr [32], MusicSense [33], LsM [34], LAMP [35], *etc.* analyse the subjective content, *i.e.*, melody and lyrics to recommend music based on the mood of a listener. Let us have a look at some of them to have an idea of the methodology used generally.

Lu, *et al.* [26] analyse the audio signals of music to decide its mood. They cover a lot of detail on the general methods used and the problems faced while extracting sentiments from auditory signals. From the signals, features such as intensity, timbre and rhythm are extracted, and then used to represent the music. Rui Cai, *et al.* [33] have developed a Contextual Music Recommendation System using Emotional Allocation Model that recommends music to web-users. The system analyses content of the pages being viewed by users (just as Google AdSense does), based on which, it decides which music to recommend. They use the lyrics and reviews of a song to derive its semantics. Theirs is a generative model where they picture each song as being generated from a mixture of emotions. iPlayr [32] is an Emotion-aware Music Platform that uses almost all the features of a musical piece such as lyrics, metadata and melody. They use ConceptNet [36] to analyse lyrics and some audio labelling toolkits for melody, to extract sentiments out of songs, and then recommend songs to users based on their queries. Chung-Yi Chi, *et al.* [37] focus their work on analysing lyrics' contribution in the emotional content of a song. They compile their own dataset by asking participants to rate pop songs based on their mood. To compare the impact of lyrics with that of audio signals, they conduct experiments in three different modes, *viz.*, "lyrics only", "music track only", and "both combined". Their conclusion favours the lyrics-feature and they claim that it can be used as a key design factor for mood-based music recommendation systems. However, there is a lot of scope for contributions involving lyrics.

1.4.3 Motivation and Scope for Contribution

In addition to these published works, there are a lot of modern online vendors [38, 39, 40, 41] supplying music based on user-mood. This shows that there is a lot of demand for automatic music recommenders, and emotion-based approach definitely turns out to be the best. We are motivated towards doing an unsupervised analysis, which can be useful when no (or very few) labelled songs are available. Also, never before has LDA been used in analysing emotions from songs in the way we have. The only work we have come across is the recent one by Yang, et al. [42] that uses LDA for regression in a continuous emotion setting. We on the other hand use discrete emotions that are more reliable to have been tagged by users. We also involve supervision and get an excellent emotion-based dimensionality reduction. Moreover, usage of Non-negative Matrix Factorization for classification has also been looked with a novel perspective. We are able to give a general picture of "topics," be it classification, or clustering or dimensionality reduction.

1.5 Our Contribution

Our contribution in the present work is in two parts. In the first part, we collect lyrics of about 60,000 songs and aim at clustering them. For this we use a probabilistic graphical model called Latent Dirichlet Allocation (LDA) [1], which estimates certain probability distributions called "topics" (see Sec. 2.5). We hope that some (not all) of the topics be sentiment-based, and find ourselves lucky (!) at the end. We also cluster songs using these topics and get a sentiment-oriented clustering. This unsupervised approach to sentiment clustering could be resorted to in case of absence of tagged data; however we require some supervision (manual or knowledge-based). To evaluate the clusters, we use songs labelled as one of six sentiments (moods) — *Happy*, *Sad*, *Angry*, *Funny*, *Tired*, *Love*.

The second part deals with a relatively small amount (about 1,300) of human-annotated songs, each labelled as being either *Happy* or *Sad*. After collecting these songs, we learn a Support Vector Machine (SVM) based classifier to separate them into their respective classes. The weight vector of the classifier could be used for ranking representative features for each class, to further reduce dimensionality by feature selection. We observe that the classification accuracy increases, with a decent decrease in dimensionality. We also learn a Naïve Bayes Classifier (NBC) for the data and see some variations with SVM results. We also tried using Non-negative Matrix Factorization (NMF) for classification by fixing one of its factors and interestingly enough, we find that it is exactly equivalent to using NBC. We establish the theoretical equivalence for both binary and term-frequency based term-weighting schemes. We conclude by saying that the supervised model can help in improving the performance of a recommender system by providing extra knowledge, especially when we have tagged data.

1.6 Organization of this Thesis

The underlying theme of this thesis is the notion of "topic," which is the reason the word occupies a place in the title. Note that the double quotes around the word *topic* mean a lot. They signify the polysemy of this concept, which is important to understand. We wish to glorify its unified view. It is strange how a word can be both polysemous and unified! What one needs to understand is that a topic is nothing but an assignment of certain numbers to words, each assignment possibly denoting a hidden concept in the corpus. This is the unified view. But it is polysemous in the sense of being omnipresent, be it a probabilistic or a deterministic setting. On one hand, we use a standard "topic" model called LDA for clustering, encountering topics in the way; while on the other hand, using an SVM or NBC for classification too follows the same route. Interestingly enough, we have an equivalence between NBC and NMF for classification, which further testifies the omnipresence of topics. This thesis disguises itself into "topics" to propagate the ideas of "sentiments".

Chapter 2 provides the theoretical background required, covering basics of data representation (2.1), preprocessing (2.2), clustering (2.3) and classification (2.4), along with their evaluation methods. Our work on unsupervised sentiment analysis has been covered in Chapter 3, which deals with collecting data and analysing it using Latent Dirichlet Allocation (LDA). Next comes supervised sentiment analysis, for which we devote Chapter 4, where we discuss the need for supervision and use a labelled dataset for classifying songs and reducing their dimensionality. In the next chapter, we theoretically deduce an equivalence between the Naïve Bayes Classifier and classification using Non-negative Matrix Factorization. We conclude in Chapter 6, along with future areas of interest in Sec. 6.2. At the end (Appendix A), we provide an illustration for the working of LDA using three short proses.

Chapter 2

Background Theory

2.1 Representing Data

In the domain of pattern recognition, we deal with documents that need to be (broadly) either clustered or classified. Many kinds of data could be treated as documents, depending upon the application. Typical types of documents are text, images, audio, video, *etc.* Data is represented using features, which are nothing but words when text documents are considered. We call a collection of documents (n in number) a "corpus," which could be denoted by $\mathcal{D} = \{D_1, D_2, \cdots, D_n\}$, where each document D_i is an ordered multiset[1]. We find words from a document using $Tokenization$, which removes punctuations and considers words as entities formed by letters and numbers, separated by white-spaces. If we define the vocabulary of l' words by a set $\Omega = \{\omega_1, \omega_2, \cdots, \omega_{l'}\}$, then each document can be represented as an ordered multiset $D_i = (d_{i1}, d_{i2}, \cdots d_{il_i})$ containing l_i tokens (not necessarily distinct), where $d_{ij} \in \Omega$ is the j^{th} token of document D_i.

This is the most "natural" form of a corpus, where no document has been tampered with. But, analysing documents in this form is really tedious and calls for some approximations to be made. The first (and the most famous) approximation we make is to drop the order in which words (tokens) appear in a document, called the Bag of Words approximation. In other words, each document is considered to be a bag containing words.

[1]A set in which members can appear multiple times

In order to do this, we take each of the ordered multisets D_i from $\mathcal{D}$ and convert them into sets $\tilde{D}_i = \{(\omega_1, f_{i1}), (\omega_2, f_{i2}), \cdots, (\omega_{l'}, f_{il'})\}$, where $\omega_1, \omega_2, \cdots, \omega_{l'}$ are words from vocabulary Ω and $f_{ij} \in \mathbb{N}$ denotes the frequency with which word ω_j occurs in document d_i. Thus, we have a new, approximated corpus $\tilde{\mathcal{D}} = \{\tilde{D}_1, \tilde{D}_2, \cdots, \tilde{D}_n\}$.

Analysing $\tilde{\mathcal{D}}$ may be straightforward for a human, but is too abstract for a computer. We need to further simplify the notion of a corpus so that a computer can deal with it. It seems convenient to represent the corpus $\tilde{\mathcal{D}}$ as a matrix $X' \in \mathbb{R}^{n \times l'}$ called the data matrix, which is defined as $X'_{ij} = f_{ij}$, $1 \leq i \leq n$, $1 \leq j \leq l'$, where the i^{th} row corresponds to document D_i and j^{th} column to word ω_j. Thus, we have converted the corpus $\mathcal{D}$ into a matrix X'. We say that the corpus contains n documents (also interpreted as n points), each lying in an l'-dimensional space.

2.2 Preprocessing

Once we have a concrete representation of a corpus in terms of data matrix X', we are ready to do computations. We have already dealt with *Tokenization* in the previous section. The next step is to remove highly-frequent words called *stop-words*, which, in general, do not take part in pattern recognition (an exception is their use in stylometry [43]). Then, it is customary and logical to do another approximation called *Morphological Analysis* over the whole vocabulary of words, so that different forms (*e.g*, `laugh`, `laughing`, `laughed`, `laughs`) of the same word (`laugh`) get clubbed together. Assuming each word to be a dimension, we have made words linearly independent, *i.e.*, totally unrelated. But different morphological forms of a word are related. In addition to building relationships between different forms of the same word by clubbing them together, morphological analysis also reduces the dimensionality of the corpus. Then comes *Dictionary Matching* where each word is checked for its existence in a lexicon and the violators are filtered out. The last step is to do a *Frequency Analysis* over the words, usually using the Zipf's curve [44], which relates frequencies of words to their ranks. In this step, we try to select the most relevant features for pattern recognition by filtering out extremely rare and extremely frequent words. For example, in computer science documents, the word “abracadabra” would be very rare (if at all it occurs) and the word “computer” would be very frequent; both of them should be filtered out. Preprocessing of the corpus represented by matrix X' gives us a new vocabulary $V = \{w_1, w_2, \cdots, w_l\}$ of l ($\leq l'$) words and a new data matrix $X \in \mathbb{R}^{n \times l}$ (Please note that each word $w_i \in V$ is either present as it is in Ω, or has a morphological variant in Ω).

2.3 Clustering

Clustering refers to forming groups of documents, such that documents in the same group are similar to each other and those in different groups are distant. Given the data matrix X (or the corpus $\mathcal{D}$), a clustering algorithm only demands k (denoting the number of clusters desired), and the notions of “similar” and “distant.” Once k is fixed, we actually need to find a relation $\kappa : \mathcal{D} \to \mathcal{K}$, where $\mathcal{K} = \{K_1, K_2, \cdots, K_k\}$ denotes the set of clusters, K_i denoting a cluster. If κ is many-to-one, we call it a *hard-clustering*, *i.e.*, each document can belong to only one cluster. On the other hand, if κ is many-to-many, we call it a *soft-clustering*, *i.e.*, multiple clusters allowed for a single document. For example, a document concerning *Sachin Tendulkar* could belong to two clusters, *Cricket* and *Politics*!

To decide whether two documents are similar or distant, we need to introduce the notion of a distance (or a similarity) measure. The most popular ones are *Euclidean Distance*, $d_E(\cdot\ ,\ \cdot)$ and *Cosine Similarity*, $s_C(\cdot\ ,\ \cdot)$ where,

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y})} \quad \text{and} \quad s_C(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\sqrt{(\mathbf{x}^\top \mathbf{x})(\mathbf{y}^\top \mathbf{y})}} \tag{2.1}$$

for any two documents (two rows in X) $\mathbf{x}, \mathbf{y} \in \mathbb{R}^l$, interpreted as vectors. There are other variants of distance measures such as the *Manhattan Distance*, *Hamming Distance*, *Mahalanobis Distance*, *etc.* Dealing with these measures also makes one ponder upon the term-weighting scheme used in X, *i.e.*, what should the entries in X actually be? Presently, they are term-frequencies, *i.e.*, X_{ij} denotes the number of times word w_j occurs in document D_i. X_{ij} could very well capture merely the "existence" of term w_j in document D_i, which makes X a binary matrix, X^{bin}. In other words, $X^{bin}_{ij} = \begin{cases} 1, & \text{if } X_{ij} \neq 0 \\ 0, & \text{if } X_{ij} = 0 \end{cases}$. Other term-weighting schemes are term-frequency (tf), document-frequency (*df*), tf-inverse-*df* ($tfidf$), *etc.* [45].

Popular clustering algorithms are Hierarchical Clustering, k-means Clustering, EM Clustering, *etc.* There are certain other techniques such as Non-negative Matrix Factorization (NMF), Topic Models, *etc.* which can be used for clustering [46]. Ding, *et al.* [47] have even established an equivalence between NMF and k-means clustering.

2.3.1 Cluster Evaluation

Once we have clusters, what remains to be done is their evaluation, for which we have to use some extra knowledge in terms of a labelled dataset. We need a set of m classes (or labels), $\mathcal{C} = \{C_1, C_2, \cdots, C_m\}$, along with the knowledge about the class of each document in X. Now, given a set of k clusters, $\mathcal{K} = \{K_1, K_2, \cdots, K_k\}$, we can use standard evaluation measures [45] like *Purity* (Eq. 2.2) and *Normalized Mutual Information (NMI*, Eq. 2.3). Since we seek soft-clusters in the present work, we have given the formulae for their "soft" versions.

$$purity(\mathcal{K}, \mathcal{C}) = \frac{1}{n} \sum_{j=1}^{k} \max_{i=1}^{m} |C_i \tilde{\cap} K_j|, \tag{2.2}$$

where $| \cdot \ \tilde{\cap} \ \cdot |$ is the weighted cardinality (based on membership) of the intersection between two sets. Basically, we try to associate with each class, the cluster that has the maximum documents from that class. The worst clustering has zero purity and the best clustering has purity equal to one. However, it is possible to get unit purity by assigning each document to a separate cluster, which is a drawback of the purity measure. To overcome this, we use *NMI* [45].

$$NMI(\mathcal{K}, \mathcal{C}) = \frac{I(\mathcal{K}; \mathcal{C})}{[H(\mathcal{K}) + H(\mathcal{C})]/2}, \tag{2.3}$$

where, I is the Mutual Information, given by:

$$I(\mathcal{K};\mathcal{C}) = \sum_{i=1}^{k}\sum_{j=1}^{m} \frac{|K_i \tilde{\cap} C_j|}{n} \log \frac{n \cdot |K_i \tilde{\cap} C_j|}{|K_i||C_j|}, \tag{2.4}$$

and H is the entropy, given by:

$$H(\mathcal{K}) = -\sum_{i=1}^{k} \frac{|K_i|}{n} \log \frac{|K_i|}{n} \qquad H(\mathcal{C}) = -\sum_{j=1}^{m} \frac{|C_j|}{n} \log \frac{|C_j|}{n}. \tag{2.5}$$

2.4 Classification

Classification is a supervised learning technique, wherein we seek best ways to tell one kind of documents from another. Being a supervised technique, it expects a labelled dataset, which is nothing but the data matrix X, along with a class-label vector $\mathbf{y} \in \mathbb{R}^n$, each entry y_i of whose, corresponds to the label of document D_i (or the i^{th} row of X). $\mathbf{y}$ takes entries from the set of classes, $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$. Sometimes, we append $\mathbf{y}$ to X as its last column $(X \vdots \mathbf{y})$ and call it the "label column." The entries in $\mathbf{y}$ could very well be strings such as 'Cricket' or 'Religion', but for simplicity, we denote them by numbers. Given X and $\mathbf{y}$, the aim is to find separator(s) to distinguish documents in one class from those in the others.

Because we are given the data matrix X, each document is a point in the l-dimensional space, and we need to find curves called decision boundaries that separate documents of different classes. The simplest such curve is an $(l-1)$-dimensional hyperplane, which is obtained when we aim at linear classification. An $(l-1)$-dimensional hyperplane is nothing but the locus of all points $\mathbf{x} \in \mathbb{R}^l$ satisfying $\mathbf{w}^\top \mathbf{x} = b$, for a given vector $\mathbf{w} \in \mathbb{R}^l$ and a scalar b. Hyperplanes are points when $l = 1$, lines when $l = 2$ and planes when $l = 3$. In the two-class scenario $(k = 2)$, we generally pronounce classes as being positive or negative. The set of all classes $\mathcal{C}$ becomes $\{C_+, C_-\}$ and the label vector $\mathbf{y} \in \{+1, -1\}^n$.

Popular classifiers include k-Nearest Neighbour Classifier (k-NNC), Support Vector

Machine (SVM), Naïve Bayes Classifier (NBC), *etc.*, of which SVM and NBC are examples of linear classifiers, which we discuss next.

2.4.1 Linear Support Vector Machines (SVM)

A Linear Support Vector Machine is a model that finds a hyperplane that "best" separates documents in one class from those in others. Here, "best" refers to a hyperplane that leaves the maximum margin between the classes. When multiple classes are considered, SVM breaks the problem into multiple two-class problems. So, we can look at the binary classification problem only. Given the data matrix $X \in \mathbb{R}^{n \times l}$ and the label vector $\mathbf{y} \in \{+1, -1\}^n$, we aim to find a hyperplane that separates documents in the two classes, $+1$ and -1, in such a way that the width of the margin between them is maximum. If such a classifier exists, the dataset is said to be "linearly separable" and documents of classes $+1$ and -1 lie in the positive and negative half spaces respectively. This means that if $\mathbf{w}^\top \mathbf{x} = b$ represents a decision boundary, then $\mathbf{w}^\top \mathbf{x} > b$ for a document $\mathbf{x}$ of class $+1$, and $\mathbf{w}^\top \mathbf{x} < b$ for a document $\mathbf{x}$ of class -1. Alternatively, $y \cdot (\mathbf{w}^\top \mathbf{x} - b) > 0$ for document $\mathbf{x}$ having label y.

Further, to characterize the maximum margin between classes, we introduce two support hyperplanes parallel to the decision boundary, one for each class. The documents through which the support hyperplanes pass are called support vectors. The region between the support hyperplanes is called the margin. Our aim is to maximize the margin such that no document lies in it. Incorporating these constraints, the SVM problem can be characterized as follows:

$$\min_{\mathbf{w},\ b} \|\mathbf{w}\|^2 \quad \text{subject to} \ \ y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1, (1 \leq i \leq n) \tag{2.6}$$

where $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \in \mathbb{R}^l$ represent documents in the data matrix, $X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix}$.

If the data is not linearly separable, there would be no hyperplanes possible, which could precisely separate the data. We need to ignore some documents for an approximate linear classification, *i.e.*, we should allow some violators. A scalar C is introduced, which denotes the cost of misclassification. And for each document $\mathbf{x}_i \in X$, slack variables ξ_i are introduced, to make the optimization problem change as follows:

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \left\{ \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \right\} \quad \text{s.t.} \;\; y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \xi_i \;\; \text{and} \;\; \xi_i \geq 0, \;\; (1 \leq i \leq n) \tag{2.7}$$

2.4.2 Naïve Bayes Classifier (NBC)

The Naïve Bayes classifier (NBC) is based on the assumption that words in a document occur independently of each other, once the class of that document is known. For each test document, we aim at finding the class which it *best* belongs to. This can be done by finding the conditional probability $P(C_i|D_j)$ for the document D_j, for all classes $C_i \in \mathcal{C}$. The best class for that document would be the one having the highest probability. We can formulate the problem using the Bayes' theorem as follows:

$$P(C_i|D_j) \propto P(C_i) \cdot P(D_j|C_i) = P(C_i) \cdot \prod_{w_k \in D_j} P(w_k|C_i) \tag{2.8}$$

This shows that it is enough to estimate the priors $P(C_i)$ and likelihoods $P(w_k|C_i)$ for all words w_k in vocabulary V and for all classes $C_i \in \mathcal{C}$. The priors $P(C_i)$ are estimated using the following:

$$\hat{P}(C_i) = \frac{|C_i|}{\sum_{C_j \in \mathcal{C}} |C_j|}, \quad \forall C_i \in \mathcal{C} \tag{2.9}$$

where $|C_i|$ denotes the number of documents in class C_i.

To estimate the likelihoods, the most popular versions are the Bernoulli and the Multinomial models.

Bernoulli Model

$$P^B_{zi} := P(w_z|C_i) = \frac{df_{C_i}(w_z)}{\sum_{w_j \in V} df_{C_i}(w_j)} \tag{2.10}$$

$\forall w_z \in V$ and $\forall C_i \in \mathcal{C}$, where $df_{C_i}(w_z)$ is the number of documents in C_i where w_z occurs, and $P^B \in \mathbb{R}^{l \times k}$, l being the vocabulary size and k, the number of classes.

Multinomial Model

$$P^M_{zi} := P(w_z|C_i) = \frac{tf_{C_i}(w_z)}{\sum_{w_j \in V} tf_{C_i}(w_j)} \tag{2.11}$$

$\forall w_z \in V$ and $\forall C_i \in \mathcal{C}$, where

$$tf_{C_i}(w_z) = \sum_{D \in C_i} tf(w_z, D) \tag{2.12}$$

where $tf(w_z, D)$ is the frequency of the term w_z in document D, and $P^M \in \mathbb{R}^{l \times k}$.

2.4.3 Evaluation of Classification

Linear classification for two classes gives us a hyperplane characterized by $\mathbf{w}$ and b. It claims that this hyperplane would correctly classify documents. If the data is linearly separable, the hyperplane obtained would certainly classify all the training documents correctly.

We are supposed to have two datasets X^{train} and X^{test}, with their corresponding label vectors, $\mathbf{y}^{train}$ and $\mathbf{y}^{test}$. In the training phase, we train the classifier using the training set (X^{train}, $\mathbf{y}^{train}$) and get a hyperplane $\mathbf{w}^\top \mathbf{x} = b$. Now comes the testing phase, in which we take each document $\mathbf{x}$ from the test set X^{test} and try to find its class label $y(\mathbf{x})$, which is nothing but $\texttt{sign}(\mathbf{w}^\top \mathbf{x} - b)$. If $y(\mathbf{x})$ is the same as the label specified in $\mathbf{y}^{test}$, we consider $\mathbf{x}$ to be classified correctly; otherwise its classification is termed incorrect. The percentage of test documents correctly classified is called "Classification Accuracy."

Generally, we randomly partition the given dataset X into datasets X^{train} and X^{test}.

If we let $X^{train} = X^{test} = X$, *i.e.*, test the classifier on the set of training documents itself, we get the ***Training-set Accuracy***. Another popular accuracy measure is the *10-fold Cross Validation* ***(10-fCV) Accuracy***. To find this, we first partition the dataset X into 10 (roughly) equal parts $X^1, X^2, \cdots, X^{10}$. Then we find 10 different accuracies, $a_1, a_2, \cdots, a_{10}$, taking $X_i^{test} = X^i$ and $X_i^{train} = \bigcup_{j \neq i} X^j$, for each i, $1 \leq i \leq 10$. The *10-fold Cross Validation Accuracy* is nothing but the average of accuracies $a_1, a_2, \cdots, a_{10}$.

2.5 Latent Dirichlet Allocation (LDA)

Generally, words in a document are considered independent of each other, given the document's class. Furthermore, this assumption can also be seen in some of the earliest works on information retrieval by Maron [48] and Borko [49]. But in real world documents, this is not the case, and, up to a large extent, words are related to each other, in terms of synonymy, hypernymy, hyponymy, *etc.* Also, their co-occurrence in a document definitely infers these relations. They are the key to some hidden semantics in documents.

To uncover these semantics, various techniques, both probabilistic and non-probabilistic have been used, few of which include Latent Semantic Indexing (LSI) [50], probabilistic LSI [51], Latent Dirichlet Allocation (LDA) [1], *etc.* Among them, LDA is the most recently developed and widely used technique that has been working well in capturing these semantics. It is a probabilistic graphical model that is used to find hidden semantics in documents. It is based on projecting words (basic units of representation) to topics (group of correlated words). Being a generative model it tries to find probabilities of features (words) to generate data points (documents). In other words, it finds a topical structure in a set of documents, so that each document may be viewed as a mixture of various topics.

Fig. 2.1 shows the plate representation of LDA. The boxes are plates representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document. There are three levels to the

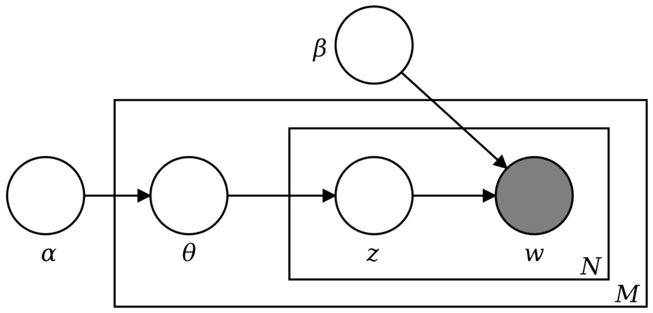

Figure 2.1: The Latent Dirichlet Allocation model

LDA representation. The parameters α and β are corpus-level parameters, variable θ is a document level variable, and z, w are word level variables.

According to the generative procedure, we choose the number of words, N as a Poisson with parameter ζ; θ is a Dirichlet with parameter vector α. The topics z_n are supposed to have come from a Multinomial distribution with θ as parameter. Actually, Dirichlet distribution is the conjugate of the Multinomial, which is the reason why it is chosen to be the representative for documents. Each word is then chosen from $p(w_i|z_n;\beta)$, where w_i are words. The number of topics, k are taken to be known and fixed.

We need to estimate β, an $l \times k$ matrix, l being the vocabulary size. Each entry in this matrix gives the probability of a word representing a topic ($\beta_{ij} = p(w_i = 1|z_j = 1)$). From the graphical structure of LDA, we have (2.13), which is the joint distribution of θ, $\mathbf{z}$ and $\mathbf{w}$, given the parameters α and β.

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \cdot \prod_{n=1}^{N} p(z_n|\theta) \cdot p(w_n|z_n, \beta) \tag{2.13}$$

To solve the inferential problem, we need to compute the posterior distribution of documents, whose expression, after marginalizing over the hidden variables, is an intractable one, when it comes to exact inference. Eq. (2.14) below shows this intractability due to coupling between θ and β.

$$p(\mathbf{w}|\alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{k} \theta_i^{\alpha_i - 1}\right) \left(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{v} (\theta_i \beta_{ij})^{w_n^j}\right) d\theta \tag{2.14}$$

For this reason, we move towards approximate inference procedures.

2.5.1 Variational Inference

Inferencing means to determine the distribution of hidden variables in a graphical model, given the evidences. There have been two types of inferencing procedures, exact and approximate. For exact inferencing, the junction tree algorithm has been proposed, which can be applied on directed as well as undirected graphical models. But in complex scenarios such as LDA, exact inferencing procedures do not work, as the triangulation of the graph becomes difficult, so as to apply the junction tree algorithm. This leads to using approximate methods for inferencing. Out of the proposed approximate inferencing techniques, one is Variational Inferencing [52]. The terminology has been derived from the field of calculus of variations, where we try to find bounds for a given function, or on a transformed form of the function (*e.g.*, its logarithm), whichever is concave/convex.

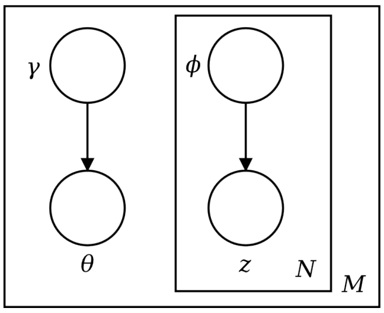

Figure 2.2: Approximation of the LDA model for variational inference [1]

A similar procedure is carried out for probability calculations. When applied to LDA, variational inferencing leads to simplifying our model by relaxing dependencies between θ and z, and z and w, just for the inferencing task. The simplified model is shown in Fig. 2.2, which can be compared with the original LDA model (Fig. 2.1).

2.6 Non-negative Matrix Factorization (NMF)

NMF was brought to the attention of the Pattern Recognition community by the work of Lee and Seung [53]. Given a positive integer r, it aims to approximate the data matrix

$X \in \mathbb{R}_*^{n \times l}$ [†] with the product of matrix $W \in \mathbb{R}_*^{n \times r}$ and the transpose of matrix $H \in \mathbb{R}_*^{l \times r}$ (*i.e.*, $X \approx WH^\top$). For instance, if X is a document-term matrix, then NMF tries to decompose it into a document-feature matrix W, and a feature-term matrix $H^\top$. Each row in $H^\top$ can be viewed as the direction along which the centroid of some cluster of documents (in term-space) would lie, while W captures the association of each document with these clusters. The objective of NMF can be formally stated as follows:

$$\min_{\substack{W \geq 0 \\ H \geq 0}} \|X - WH^\top\|_F \tag{2.15}$$

for a given non-negative matrix X, where $\| \cdot \|_F$ denotes the Frobenius norm.

The matrix H gives the exact location of r cluster-centroids, based on the data points we have in X. W captures the association of each document with these clusters. In other words, H gives l-dimensional basis vectors, r in number; and W gives the representation of each document in terms of the basis vectors of H.

[†] $\mathbb{R}_* = \mathbb{R}_+ \cup \{0\}$

Chapter 3

Sentiment Clustering

Given a collection of song lyrics, we aim to find sentiment-based clusters. For this, we first collect the to-be-clustered data and evaluation data from the web and then pre-process them. Once the data is ready, we use LDA, setting $k = 50$, which means we need 50 topics. Analysing them manually, we find some of them to be sentiment-oriented. We also cluster songs based on their association with topics and evaluate using an evaluation dataset of six emotions.

3.1 Collection and Preprocessing of Lyrics

For clustering, we required song lyrics in text format. There are websites where users post song lyrics. Over time, these websites have collected humongous amount of lyrics, but all embedded within their HTML pages. After a rough and quick survey on the number of songs and quality of data, we chose LyricsTrax[1]. We first crawled through it and then parsed the HTML pages for the lyrics and some metadata such as album-name and artist-name, along with title-name (as mere title-names could be conflicting). Saving each lyric as a text file to be treated as a separate document, we cover a total of 66,159 songs. Due to submission of lyrics by different users, the raw lyrics contain a lot of noise and inconsistency in spellings. This calls for a special pre-processing ("special" due to a

[1]`http://www.lyricstrax.com/`

specific morphological analysis) on the raw text in order to get meaningful results.

For each song, we carry out the following pre-processing steps, as mentioned in Sec. 2.2: *Tokenization, Stop-word Removal, Morphological Analysis, Dictionary Matching, and Frequency Analysis.* Everything else being in the standard way, some 'pre'-pre-processing (termed "special" earlier) needs to be done by using a variant of morphological analysis before tokenization, to resolve variants of words ending in '`ing`' (*e.g.*, `raining`, `rainin'`, `rainin`, `rain-ing`, *etc.*) by converting them to one standard form (`raining`). However, verbs would be further analysed by the standard morphological analyser for '`-ing`' forms, but forms like `everythin'` should not be filtered-off during *Dictionary Matching* at the end. The reason such forms occur is users literally following pronunciation (singers often do not pronounce the '`g`' of '`-ing`' for a poetic feel).

Once this preprocessing is done, we have a vocabulary set $V=\{w_1, w_2, \cdots, w_l\}$ containing all words (l in number) used in the dataset. What we also have, is a collection of songs, $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$, where each song $S_i = \{(w_1, f_{i1}), (w_2, f_{i2}), \cdots, (w_l, f_{il})\}$ is a set of cardinality l, f_{ij} denoting the frequency of word w_j in song S_i.

3.2 Topic Distillation using LDA

We fix the number of topics k and apply LDA using Variational Expectation Maximization (VEM) for parameter estimation over $\mathcal{S}$, to find vector α and matrix β (see Sec. 2.5). We get the topical priors in the form of the $k \times l$ matrix, β, where each row represents a distinct topic and each column, a distinct word. The values represent the probability of each word being a representative for the topic. In other words, β_{ij} is the probability of the j^{th} word representing the i^{th} topic, $P(w_j|z_i)$.

Once β is in place, we solve the posterior inference problem. This would give us the association of songs to topics. We calculate the approximate values in the form of the Dirichlet parameter γ, established in Sec. 2.5.1 (Fig. 2.2). It is an $n \times k$ matrix, wherein the rows represent songs, and columns, topics. Each value in γ is proportional (not equal) to the probability of association of a song to a topic. Thus, γ_{ij} is proportional to the probability of the i^{th} song S_i being associated to the j^{th} topic.

Now we can use β to represent topics in terms of words, and γ to assign topics to songs. Instead of associating all topics with all songs, we need to have a more precise association, that can form soft clusters of songs, each cluster being a topic. To achieve this, we impose a relaxation in terms of the probability mass we want to cover, for each song. Let π denote the probability mass we wish to cover for each song. First of all, normalize each row of γ to get exact probabilities and let the normalized matrix be $\tilde{\gamma}$. Then, sort each row of $\tilde{\gamma}$ in decreasing order of probability.

Now, for each song S_i, starting from the first column ($j = 1$) to the last column ($j = k$), add the probabilities $\tilde{\gamma}_{ij}$ and stop when the sum just exceeds π. Let that be the r_i^{th} column. All the topics covered before the r_i^{th} column should be associated to the song S_i, with probability proportional to the corresponding entry in $\tilde{\gamma}$. So, for each song S_i, we get a set of topics, $T_i = \{(t_1^i, m_1^i), (t_2^i, m_2^i), \cdots, (t_{r_i}^i, m_{r_i}^i)\}$ where $m_j^i = \tilde{\gamma}_{ij}$ gives the membership (not normalized) of song S_i to topic t_j^i. For each song S_i, we need to re-normalize the membership entries in T_i and let the re-normalized set be $\tilde{T}_i = \{(t_1^i, \mu_1^i), (t_2^i, \mu_2^i), \cdots, (t_{r_i}^i, \mu_{r_i}^i)\}$, where $\mu_j^i = \frac{m_j^i}{\sum_{n=1}^{r_i} m_n^i}$. Thus, for each song S_i, we have a set of topics $\tilde{T}_i$, which also contains the membership of the song to each one of the topics present in $\tilde{T}_i$.

3.2.1 An Illustration

Let us illustrate this by an example. We have three songs (refer Appendix A), which, after preprocessing, gives the vocabulary to be $V = \{\texttt{grow}, \texttt{wish}, \texttt{tree}, \texttt{free}, \texttt{meadow}\}$. Assuming $k = 2$ (2 topics), and fixing $n = 3$ (3 songs) and $l = 5$ (5 words), we can represent the song collection $\mathcal{S} = \{S_1, S_2, S_3\}$ as

$S_1 = \{(\texttt{grow}, 4), (\texttt{wish}, 6), (\texttt{tree}, 2), (\texttt{free}, 0), (\texttt{meadow}, 0)\}$

$S_2 = \{(\texttt{grow}, 5), (\texttt{wish}, 0), (\texttt{tree}, 4), (\texttt{free}, 0), (\texttt{meadow}, 1)\}$

$S_3 = \{(\texttt{grow}, 0), (\texttt{wish}, 0), (\texttt{tree}, 2), (\texttt{free}, 6), (\texttt{meadow}, 0)\}$, we get matrices

$$\beta = \begin{pmatrix} 0.28 & 0.00 & 0.33 & 0.33 & 0.06 \\ 0.33 & 0.50 & 0.17 & 0.00 & 0.00 \end{pmatrix}^{\top} \text{ and } \gamma = \begin{pmatrix} 0.35 & 12.29 \\ 10.27 & 0.37 \\ 8.31 & 0.33 \end{pmatrix} \text{ (not probabilities).}$$

Normalizing rows of γ, we get $\tilde{\gamma} = \begin{array}{c|cc} & z_1 & z_2 \\ S_1 & 0.03 & 0.97 \\ S_2 & 0.965 & 0.035 \\ S_3 & 0.96 & 0.04 \end{array} \overset{\text{sort rows}}{\Longrightarrow} \begin{pmatrix} z_2 : 0.97 & z_1 : 0.03 \\ z_1 : 0.965 & z_2 : 0.035 \\ z_1 : 0.96 & z_2 : 0.04 \end{pmatrix}$

Now, let $\pi = 0.6$ and for song S_1, let us start adding entries in the first row of $\tilde{\gamma}$ (sorted) till the sum reaches 0.6. Since for the first entry itself, we get the sum $0.97 > \pi$, we assign song S_1 to topic z_2. We assign songs S_2 and S_3 to topic z_1, going by the same argument. Thus we get $t_1^1 = z_2$, $t_1^2 = z_1$ and $t_1^3 = z_1$ as the only topics for S_1, S_2, S_3 with memberships $m_1^1 = 0.97$, $m_1^2 = 0.965$ and $m_1^3 = 0.96$ respectively. Thus,
$T_1 = \{(z_2, 0.97)\}$, $T_2 = \{(z_1, 0.965)\}$ and $T_3 = \{(z_1, 0.96)\}$.
Finally, $\tilde{T}_1 = \{(z_2, 1)\}$, $\tilde{T}_2 = \{(z_1, 1)\}$ and $\tilde{T}_3 = \{(z_1, 1)\}$ gives the topic association of songs (In this example, each song turns out to be associated with one topic only; but there can be multiple topic associations too).
We can now say that $\underline{\text{song } S_1 \text{ belongs to topic } z_2 \text{ and songs } S_2 \text{ and } S_3 \text{ belong to topic } z_1}$.

As far as analysing the topics is concerned, we can interpret the β matrix, which gives association of words to topics. In this example, sorting each row of

$$\beta^{\top} = \begin{array}{c|ccccc} & \texttt{grow} & \texttt{wish} & \texttt{tree} & \texttt{free} & \texttt{meadow} \\ z_1 & 0.28 & 0.00 & 0.33 & 0.33 & 0.06 \\ z_2 & 0.33 & 0.50 & 0.17 & 0.00 & 0.00 \end{array} \text{ in decreasing order of probabilities,}$$

we get

$$\beta^{\top}_{sorted} = \begin{pmatrix} (\texttt{tree}, 0.33) & (\texttt{free}, 0.33) & (\texttt{grow}, 0.28) & (\texttt{meadow}, 0.06) & (\texttt{wish}, 0.00) \\ (\texttt{wish}, 0.50) & (\texttt{grow}, 0.33) & (\texttt{tree}, 0.17) & (\texttt{free}, 0.00) & (\texttt{meadow}, 0.00) \end{pmatrix}, \text{ which}$$

can be interpreted as $z_1 = \boxed{\texttt{tree, free, grow}}$ and $z_2 = \boxed{\texttt{wish, grow, tree}}$ (by a particular protocol of using top 3 words), which concludes this topic (Topic 3.2.1)!

3.3 Results and Discussions

As mentioned earlier, we collected the dataset from LyricsTrax, a website for lyrics that provided us with the lyrics of ($n =$) 66,159 songs. After preprocessing, we get a total of ($l =$) 19,508 distinct words, which make the vocabulary V. Assuming different values of k ranging from 5 to 50 (in steps of 5), we applied LDA, which fetches two probability distributions in the form of a song-topic matrix γ and a topic-word matrix β. We have used β to realize topics in terms of words and γ to realize topics (soft-clusters) in terms of songs, by converting them into β_{sorted} and $\tilde{T}$ respectively. Let us analyse them one by one, and start by looking at β_{sorted}.

3.3.1 Topics as Distributions over Words

Table 3.1: Some of the 50 topics given by LDA

Topic #	Words	Sentiment[2]
1	sun, moon, blue, beautiful, shine, sea, angel, amor, summer, sin	Love
2	away, heart, night, eyes, day, hold, fall, dream, break, wait	Sad
3	time, way, feel, think, try, go, leave, mind, things, lose	Tired
5	good, look, well, run, going, talk, stop, walk, people, crazy	Happy
6	man, little, boy, work, woman, bite, pretty, hand, hang, trouble	-none-
7	yeah, alright, round, knock, spin, door, upside, feel, dizzy, right	-none-
9	god, child, lord, heaven, black, save, pray, white, thank, mother	Religious
11	sweet, ooh, music, happy, lady, morn, john, words, day, queen	Happy
12	sing, hear, song, roll, sound, listen, radio, blues, dig, bye	-none-
23	kid, happen, trip, laugh, billy, police, clown, sir, pig, famous	Funny
33	send, write, read, hip, book, hop, letter, message, nature, beats	-none-
50	kill, blood, fight, hate, death, hell, war, pain, fear, bleed	Angry

As illustrated in Sec. 3.2.1, β_{sorted} could be used to characterize topics. As hypothesized earlier, we get topics based on sentiments. It is practically difficult to look at all the words for a topic; so we consider only the top 10 words (for manual analysis) from each column of β_{sorted}. We also noticed (and so has been clear from [54]) that increasing k from 5 to 50 splits general topics into specific ones, losing their crisp nature. For $k = 50$

[2]Annotated manually

we have listed some of the topics in Tab. 3.1.

Please note that the sentiments provided in the last column were not a result of LDA. We have manually tagged the topics with the possible sentiment it corresponds to. Topic 1 contains the words sun, blue, beautiful, *etc.* and signifies a love/happy song. Likewise, Topic 50 reflects an angry mood, as it contains the words kill, blood, fight, death, *etc.* It is obvious to expect noisy topics, *e.g*, topics 6, 7, 12 and 33 do not correspond to any sentiment.

3.3.2 Validation of Clusters

This was the analysis based on β, which associates words with topics. From the posterior inference, we get γ, which assigns every topic to a document, with some probability. We normalize γ by dividing each row with the sum of elements of that row, and get the normalized matrix, $\tilde{\gamma}$. We then need to find the sorted order (decreasing order of probability) for each row of $\tilde{\gamma}$. Assume $\pi = 0.6$, *i.e.*, let us cover 60% of the probability mass for each document. Moving in that order, add the elements till the sum reaches π. As described in Sec. 3.2, we then find T_i for each song S_i and normalize it over the membership values to get $\tilde{T}_i$. To obtain $\mathcal{K}$ (clusters) from $\{\tilde{T}_i\}_{i=1}^{l}$, we need to analyse $\tilde{T}_i$ for all songs S_i, and associate songs to each topic appropriately.

For validation, we crawled through ExperienceProject[3], and downloaded a dataset of songs classified into 6 classes, *viz., Happy, Sad, Angry, Tired, Love* and *Funny.* From these songs, we pick the songs common to our original dataset S. This gives us $\tilde{S}$, which contain 625 songs, each associated with one of the 6 classes, from the set $\mathcal{C} = \{C_1, C_2, \cdots, C_6\}$ (*Happy,$\cdots$,Funny*). Each class C_j contains some songs from among the 625 songs in $\tilde{S}$. Intersecting $\mathcal{K}$ with $\tilde{S}$ gives $\tilde{\mathcal{K}}$, consisting of only those songs that are in the validation dataset. Now we are ready to validate the clusters, $\tilde{\mathcal{K}}$ against the annotated set $\mathcal{C}$ using each of the measures mentioned in Sec. 2.3.1.

We actually run LDA for different values of k, ranging from 5 to 50. For each of these values, we find the two evaluation measures, *Purity* and *NMI*. Then we plot

[3]`http://www.experienceproject.com/`

these measures against the number of topics, which has been summarised in Fig. 3.1 and Fig. 3.2.

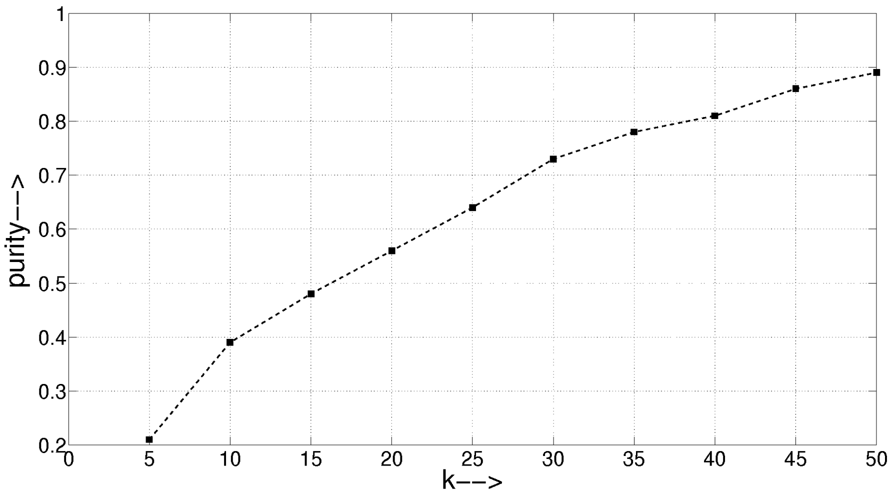

Figure 3.1: Purity

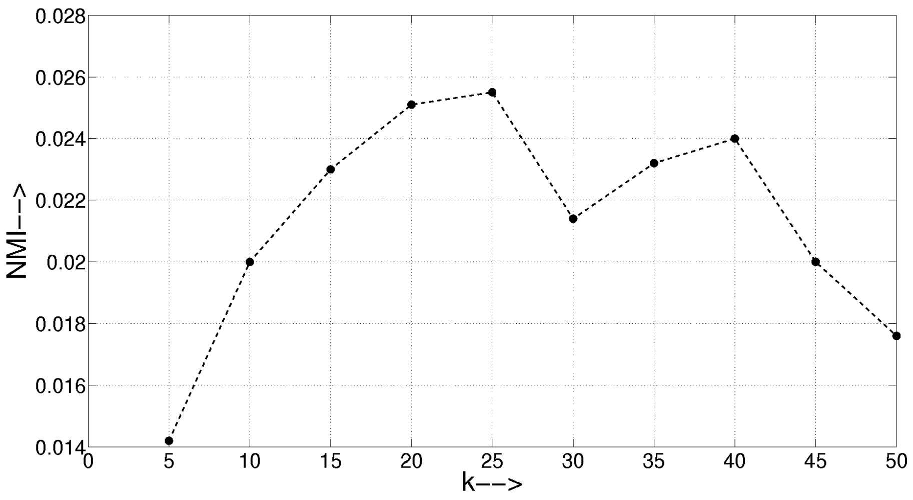

Figure 3.2: Normalized Mutual Information

It can be seen that the purity increases monotonically with the number of topics. As we already know from Sec. 2.3.1, if we had one separate cluster for each song, the purity would have been 1, i.e., highest. But that is obviously undesirable. Even if we try to find the mutual information between emotions and clusters, the problem persists. So, to penalize large values of k, we use NMI, which, being a normalized measure, can be used to compare clusterings better. In this, we have an entropy term in the denominator, which increases with increase in k, thus penalizing in the way we want it to. From the plot of NMI vs k (Fig. 3.2), it is clear that it first increases with the number of clusters k; achieves a maximum at $k = 25$, and then decreases, following a similar trend for $k \geq 30$. This gives us an idea that, according to the NMI measure, $k = 25$ is the best number of clusters that we should hunt for. Compared to $Purity$ (Fig. 3.1), which does not penalize high cluster cardinality, this measure is more reliable to decide k.

Chapter 4

Sentiment Classification

If we are provided with tagged data, we could be more confident in deciding the sentiment of a song. For a better analysis, we consider only two emotions, Happy and Sad.

4.1 Need and Inevitability of Supervision

We looked at an unsupervised learning method (LDA) to mine sentiments from songs based on their lyrics in Chapter 3. Notwithstanding the results, an unsupervised approach has its own advantages. In cases where any gold standard (annotated data) is not available (*e.g*, a not-so-famous language song-dataset), unsupervised methods have to be depended upon. Also, by using such an approach, we try to capture the actual sentimental structure of our dataset, as compared to a supervised approach, where the results may be biased towards the supervisor dataset.

Despite these advantages, there is a need for supervision. Ideally (deviating from the definition), all unsupervised learning methods use a supervisor, be it in terms of the term-weighting scheme, the distance-metric used, or the number of clusters specified. But the most unavoidable (taken for granted, but otherwise unapparent) use of supervision is in analysing results. For example, *k-means clustering*, which is an unsupervised technique, uses, say *Euclidean Distance* as the distance-metric and *tfidf* as the term-weighting scheme. Together with k, the number of clusters required, these form the 'obvious' supervisors for clustering. But we need another supervisor (human or mechanical) to legitimatise and label the clusters once they are obtained.

We thus motivate the need for supervision, either implicit or explicit, for unsupervised and supervised learning methods respectively. In the previous chapter, we had used the implicit human supervision to semantically judge topics. This is, no doubt, a very infeasible method, and needs to be replaced by an automatic supervisor (see Sec. 6.2). Coming to explicit supervision, which exists per se in the definition of supervised learning methods, an already labelled dataset (training set) is used to direct the learning method.

In this chapter, we focus on supervised learning methods using a human-annotated dataset as training set. In other (specific) words, we classify songs based on an already annotated dataset of songs. Our aim is to analyse the usage of supervision in classification and dimensionality reduction.

4.2 Collecting Happy and Sad Songs

We first collect details of happy and sad songs from the web and then look for their lyrics in the dataset of Sec. 3.1. One problem that we face is the matching of title-strings. We have, on one hand, a dataset containing names (title, album, artist) of songs, tagged as Happy/Sad. On the other hand, from Sec. 3.1, we have a dataset of lyrics of songs along with their names (title, album, artist). But because we download the two datasets from different websites, there is an inconsistency in the spellings of titles/artists/albums. So, we find the weighted edit distance between all pairs of titles, across each dataset and find matching songs. In this way, we get a dataset of song lyrics, with each song tagged as being either happy or sad.

Once the dataset is in hand (note that it has already been preprocessed in Sec. 3.1), we use it to classify songs based on sentiments. We aim at learning a classifier as well as study the words that make up the two (Happy and Sad) classes. As we have already discussed in Sec. 2.4, classifiers are broadly of two types, *viz.* descriptive and discriminative, of which NBC and SVM are the best examples.

4.3 Dimensionality Reduction using SVM

We first learn an SVM on the data and use the vector, $\mathbf{w} \in \mathbb{R}^l$ obtained, which we call as the "weight-vector" for dimensionality reduction. It is an l-dimensional vector (l being the vocabulary size) which gives a weight value to each word involved in classification. For simplicity, we denote Happy and Sad classes by C_+ and C_- (positive and negative classes) respectively. The weight-vector, $\mathbf{w}$ gives positive weights to words that "belong" to class C_+ and negative weights to those belonging to class C_-. We define two separate vectors, $\mathbf{w}^+$, $\mathbf{w}^- \in \mathbb{R}^l$ as follows ($1 \leq i \leq l$):

$$\mathbf{w}_i^+ = \begin{cases} \mathbf{w}_i, \text{ if } \mathbf{w}_i \geq 0 \\ 0, \text{ otherwise} \end{cases} \quad \text{and} \quad \mathbf{w}_i^- = \begin{cases} -\mathbf{w}_i, \text{ if } \mathbf{w}_i \leq 0 \\ 0, \text{ otherwise} \end{cases},$$

which means $\mathbf{w}^+$ contains weights of positive class words and $\mathbf{w}^-$, absolute values of the weights of negative class. These can be considered as distributions (not normalized though) over words. In other words, we have two topics, one for each class. The entries in these topics show the relative importance of words in their respective classes. We sort vectors $\mathbf{w}^+$ and $\mathbf{w}^-$ in decreasing order of their entries, calling them $\mathbf{w}^+_{sorted}$ and $\mathbf{w}^-_{sorted}$ respectively. Now, after we remove the zero-weighted words, we pick the top $p\%$ of words from each vector, and filtering out the rest, we get a reduced dimensional space. Lower the percentage p, better will be the dimensionality reduction. But we have to pay attention to the songs here. One is that as we filter out the rest $(100-p)\%$ words, we risk some songs getting filtered off too; because the songs that contain only the filtered-off words will not be retained. So we need to track the song-retention as we vary p. We hope that decrease in dimensions results in better classification, with only a negligible amount of songs filtered. We report an increase in accuracy with dimensionality reduction in Sec. 4.5, using both SVM and NBC on the reduced data.

4.4 Classification on Reduced Data

In the previous section, we reduced the dimensionality of our data from l to $p\%$ of l (roughly, as we would first remove the zero-weighted words), p varying from 100 to

3.125. Once the dimensionality is reduced we re-classify the data and observe the rise/fall in accuracy. For the re-classification, we used both SVM and NBC, to observe how discriminative and descriptive classifiers respond to the reduced data. We actually start varying p after removing the zero-weighted words (given by SVM) from the vocabulary. The results are discussed in Sec. 4.5. In the course of experiments, we also tried classifying using Non-negative Matrix Factorization (NMF) by fixing W. Interestingly, we had exactly the same results as we got using NBC. This drove us to delve deep into this issue and we ended up with finding a theoretical equivalence between the two (Chapter 5).

4.5 Results and Discussions

Tab. 4.1 shows the top ten words from $\mathbf{w}^+$ and $\mathbf{w}^-$. Not all of them actually denote exact sentiments, but we can see that a majority of them are associated to their respective classes.

Table 4.1: Top 10 words from Happy and Sad topics

Happy	`kinda infatuate future sunshine sick strike ride wake nearly fair`
Sad	`pain bye sad pink fright room storm fight reason someday`

Now, we pick top p% of these words and observe certain results. Tab. 4.2 shows the variation of classification accuracies (both 10-fold-cross-validation and training-set accuracy) of both SVM and NBC. The first column (**All**) corresponds to no dimensionality reduction, *i.e.*, all words considered. In the second column, we remove the zero-weighted words, corresponding to which, we say $p = 100$. Then, we vary p as 100, 100/2, 100/4, 100/8, 100/16, 100/32 in the subsequent columns.

The first row (**Song-Ret**) shows the percentage of songs retained when the dimensionality gets reduced (as discussed earlier). The next row corresponds to percentage dimensionality reduction (**Dim-Red**), which shows nothing but the percentage of words filtered out from the vocabulary after we reduce the dimensionality. The last four

Table 4.2: Variation of classification accuracies with p

$p \rightarrow$	**(All)**	**100**	**50**	**25**	**12.5**	**6.25**	**3.125**
Song-Ret(%)	100	100	100	**99.77**	99.32	98.26	93.49
Dim-Red(%)	0	24.12	62.05	**81.02**	90.50	95.25	97.62
SVM-10FCV(%)	68.43	68.74	79.64	**83.69**	83.61	77.58	73.77
SVM-training(%)	99.02	99.02	99.02	**99.09**	96.80	87.29	80.73
NBC-10FCV(%)	71.21	69.70	75.00	**81.68**	80.77	77.52	79.67
NBC-training(%)	89.02	91.07	91.90	**91.35**	87.73	82.28	78.46

rows report classification accuracies, both 10-fold-cross-validation and training-set (see Sec. 2.4.3) using SVM and NBC as classifiers.

As is clear from the table, no songs are filtered out when we remove the zero-weighted words, *i.e.*, we get a 100% song retention corresponding to $p = 100$. As we decrease p, *i.e.*, as more words get filtered out, we observe an excellent retention of songs. For example, even when 95.25% of the words are removed, 98.26% of the songs are retained (for $p = 6.25$). This means that SVM gives high weights to words which are present in most of the songs. This also means that because we have a good song retention, it will not be meaningless to compare accuracies across columns. Comparing the accuracies, we can see that for all the four rows at the last, it first increases and then decreases. We get maximum accuracies at $p = 25$, with 81.02% dimensionality reduction. The increase in (10-FCV) accuracy is substantial (*e.g*, from 68.43% to 83.69%) for SVM, with a decent decrease in dimensions. As the value of p further increases, we can see that accuracies decrease. But, it is not very meaningful to compare them, because even the songs get filtered out (which means we are removing the "wrong" words). NBC gives a low accuracy on the training dataset (last row), which means that it does not over-fit, and thus, manages to give a better 10-FCV accuracy when the dimensionality is high (in the first 2 columns (**All**) and **100**).

Chapter 5

Equivalence between NBC and NMF

5.1 Theoretical Formulation

Claim. *If $A \in \mathbb{R}^{m\times n}$ is a matrix and $\mathbf{a}_1,\ \mathbf{a}_2,\ \cdots,\ \mathbf{a}_n \in \mathbb{R}^m$ are its columns, then minimizing the Frobenius norm of A is equivalent to minimizing the Euclidean norm of each of $\mathbf{a}_1,\ \mathbf{a}_2,\ \cdots,\ \mathbf{a}_n$. That is,*

$$\{\min \|A\|_F\} \equiv \{\min \|\mathbf{a}_i\|,\ \forall i\}. \tag{5.1}$$

Proof. From the definition of Frobenius norm, we have $\|A\|_F^2 = \sum_{i=1}^{n} \|\mathbf{a}_i\|^2$, and since $\|\mathbf{a}_i\|^2 \geq 0$ for $1 \leq i \leq n$, minimizing $\|A\|_F$ is equivalent to minimizing $\|\mathbf{a}_i\|$ for $1 \leq i \leq n$. Hence the proof. □

As discussed in Sec. 2.6, given a non-negative matrix X, we find non-negative matrices W and H such that Eq. 2.15 is satisfied. Let $X = [\mathbf{x}_1,\ \mathbf{x}_2,\ \cdots,\ \mathbf{x}_l]$ and $H^\top = [\mathbf{h}_1,\ \mathbf{h}_2,\ \cdots,\ \mathbf{h}_l]$, where $\mathbf{x}_i \in \mathbb{R}_*^n$ and $\mathbf{h}_i \in \mathbb{R}_*^r$, for $1 \leq i \leq l$. From Eqns. 2.15 and 5.1, we get

$$\left\{ \min_{\substack{W\geq 0\\ H\geq 0}} \|X - WH^\top\|_F \right\} \equiv \left\{ \min_{\substack{W\geq 0\\ \mathbf{h}_i\geq \mathbf{0}}} \|\mathbf{x}_i - W\mathbf{h}_i\| \right\},\ \forall i. \tag{5.2}$$

5.1.1 Fixing W

If we fix matrix W, the problem in Eq. 2.15 becomes a convex optimization problem in H. As suggested by Eq. 5.2, we can equivalently solve l optimization problems each dealing with $\mathbf{h}_i$, a column vector of $H^\top$.

For each i, define a function $f_i : \mathbb{R}_*^r \to \mathbb{R}_*$ as follows:

$$f_i(\mathbf{z}) = \|\mathbf{x}_i - W\mathbf{z}\|^2, \ \forall \mathbf{z} \in \mathbb{R}_*^r. \tag{5.3}$$

Using Eqs. 5.2 and 5.3, the NMF problem (of Eq. 2.15) for a fixed W boils down to the following:

$$\min_{\mathbf{h}_i \geq \mathbf{0}} f_i(\mathbf{h}_i), \ \forall i \tag{5.4}$$

From Eq. 5.3, we have, for any $\mathbf{z} \in \mathbb{R}_*^r$,
$f_i(\mathbf{z}) \quad = \quad (\mathbf{x}_i - W\mathbf{z})^\top(\mathbf{x}_i - W\mathbf{z}) \quad = \quad \|\mathbf{x}_i\|^2 - 2 \cdot \mathbf{z}^\top W^\top \mathbf{x}_i + \mathbf{z}^\top W^\top W \mathbf{z}$

$$\implies \nabla f_i(\mathbf{h}_i) = -2W^\top \mathbf{x}_i + 2W^\top W \mathbf{h}_i, \ \forall i. \tag{5.5}$$

Let $\mathbf{h}_i^*$ be the optimal value of $\mathbf{h}_i$ in the problem defined by Eq. 5.4. It should thus satisfy the first order necessary conditions, *i.e.*, for each i, $\nabla f_i(\mathbf{h}_i^*) = \mathbf{0}$,

$\implies -2W^\top \mathbf{x}_i + 2W^\top W \mathbf{h}_i^* = \mathbf{0}$

$\implies W^\top W \mathbf{h}_i^* = W^\top \mathbf{x}_i$.

Now, if $W^\top W$ is invertible (see Sec. 5.1.2), we can have

$$\mathbf{h}_i^* = (W^\top W)^{-1} W^\top \mathbf{x}_i, \ \forall i. \tag{5.6}$$

Since (5.4) is a convex optimization problem, the first order necessary conditions are sufficient too. Thus, Eq. 5.6, gives $\mathbf{h}_1^*, \ \mathbf{h}_2^*, \ \cdots, \ \mathbf{h}_l^*$, which can be arranged as $H^* = [\mathbf{h}_1^*, \ \mathbf{h}_2^*, \ \cdots, \ \mathbf{h}_l^*]^\top$. We can rewrite H^* as

$$H^* = \begin{pmatrix} \mathbf{x}_1^\top W(W^\top W)^{-1} \\ \mathbf{x}_2^\top W(W^\top W)^{-1} \\ \vdots \\ \mathbf{x}_l^\top W(W^\top W)^{-1} \end{pmatrix} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_l^\top \end{bmatrix} W(W^\top W)^{-1} = X^\top W(W^\top W)^{-1}$$

Thus, we get the optimal $H^\top$ as:

$$H^{*\top} = (W^\top W)^{-1} W^\top X. \tag{5.7}$$

5.1.2 NMF $\equiv$ NBC[†]

If we assume that each class corresponds to a topic in NMF, we can fix the W matrix according to the training dataset. Let us first set $r = |\mathcal{C}|$, the number of classes. Now, let W be fixed as follows for $1 \leq i \leq n$ and $1 \leq j \leq r$:

$$W_{ij} = \begin{cases} 1, & d_i \in C_j \\ 0, & d_i \notin C_j \end{cases}. \tag{5.8}$$

Since each document belongs to only one class, each row of W should have exactly one entry as 1, and all others as 0. Furthermore, in a real dataset, there would be no empty classes. Therefore, no column of W would have all entries as 0. Given these constraints, we can assure that $L := W^\top W$ would be a diagonal matrix with positive diagonal entries. It can be further noticed that $L_{ii} = \sum_{j=1}^{n} (W_{ji})^2 = |C_i|$ for each class C_i. Also, L^{-1} is a diagonal matrix with $(L^{-1})_{ii} = 1/|C_i|$.

Fixing W as above, we have different interpretations of $H^{*\top}$ (Eq. 5.7) for different term-weighting measures used for the data matrix X. Let $H_b^{*\top} = L^{-1} W^\top X^b$ and $H_{tf}^{*\top} = L^{-1} W^\top X^{tf}$, where X^b and X^{tf} are binary-weighed and term-frequency-weighed data matrices respectively.

It is known that $X_{ij}^b = \begin{cases} 1, & w_j \in d_i \\ 0, & w_j \notin d_i \end{cases}$ and $X_{ij}^{tf} = tf(w_j, d_i)$. Therefore, $W^\top X^b$ and

[†]Classification using NMF is exactly identical to classification using NBC.

$W^\top X^{tf}$ will have entries as follows:

$$(W^\top X^b)_{pq} = \sum_{i=1}^{n} W_{ip} X^b_{iq} = df_{C_p}(w_q) \tag{5.9}$$

$$(W^\top X^{tf})_{pq} = \sum_{i=1}^{n} W_{ip} X^{tf}_{iq} = tf_{C_p}(w_q) \tag{5.10}$$

(refer Eq. 2.10 for df_c and Eq. 2.12 for tf_c). Further, from Eq. 5.7, we have

$$(H^*_b)_{ji} = (L^{-1} W^\top X^b)_{ij} = \frac{df_{C_i}(w_j)}{|C_i|} \tag{5.11}$$

and

$$(H^*_{tf})_{ji} = (L^{-1} W^\top X^{tf})_{ij} = \frac{tf_{C_i}(w_j)}{|C_i|}. \tag{5.12}$$

Normalizing the rows using L_1 norm, we get

$$H^*_{ji} = (H^*_b)_{ji} = \frac{df_{C_i}(w_j)}{\sum_{w_p \in V} df_{C_i}(w_p)}, \quad \text{when } X = X^b \tag{5.13}$$

$$H^*_{ji} = (H^*_{tf})_{ji} = \frac{tf_{C_i}(w_j)}{\sum_{w_p \in V} tf_{C_i}(w_p)}, \quad \text{when } X = X^{tf} \tag{5.14}$$

Now, if we look at Eqns. 2.10 and 2.11, we have $P^B = H^*_b$ and $P^M = H^*_{tf}$, corresponding to binary and term-frequency term-weighting schemes respectively.

5.2 Discussion

From the previous formulation, we have an equivalence between NMF and NBC for classification. The standard NMF is not a convex optimization problem, unless one of the factors are fixed. We fix W, interpreting each class as a cluster, which gives us the definition in Eq. 5.8. The gist of the problem is that we have a set of k abstract quantities, $\mathcal{T} = \{T_1, T_2, \cdots, T_k\}$, which W interprets them as being classes, $\mathcal{C} = \{C_1, C_2, \cdots C_k\}$, and H interprets as being representative features $\mathcal{R} = \{R_1, R_2, \cdots R_k\}$. The set $\mathcal{T}$ is the set of topics, each element of which may be interpreted either as containing documents

($\mathcal{C}$), or representative features ($\mathcal{R}$). If we fix one of the interpretations, the task of NMF is to find out the other. When we fix W according to $\mathcal{C}$ (*i.e.*, each column W_j of W fixed according to class C_j), H remains to be found (*i.e.*, $\mathcal{R}$).

When we calculate H from Eq. 5.7, each entry H_{ji} represents the number of documents of class C_i in which word w_j appears (if X is binary-weighted), and the number of times word w_j appears in class C_i (if X is term-frequency-weighted). The same is done while calculating likelihoods $P(w_z|C_i)$ using Bernoulli and multinomial models respectively. When we normalize H (as in Eqs. 5.13 and 5.14), we actually interpret them as probabilities. Thus we have $P^B = H_b^*$ and $P^M = H_{tf}^*$.

Let us illustrate the equivalence (only for term-frequency-weighting) with the following example. Assume that we have 6 documents $D_1, \cdots, D_6$ made of 2 words, w_1 and w_2. Let us also assume that documents D_1, D_2, D_3 belong to class C_1 and D_4, D_5, D_6 to class C_2. Let points (1,8), (1,6), (3,7), (6,0), (4,0) and (5,2) denote the term-frequencies of documents. That is, $X^\top = \begin{pmatrix} 1 & 1 & 3 & 6 & 4 & 5 \\ 8 & 6 & 7 & 0 & 0 & 2 \end{pmatrix}$. Now, term-frequencies,
$tf_{C_1}(w_1) = 1 + 1 + 3 = \mathbf{5}$, $tf_{C_1}(w_2) = 8 + 6 + 7 = \mathbf{21}$,
$tf_{C_2}(w_1) = 6 + 4 + 5 = \mathbf{15}$, $tf_{C_2}(w_2) = 0 + 0 + 2 = \mathbf{2}$.
Using Eq. 2.11, we have $P(w_1|C_1) = \mathbf{5/26}$, $P(w_2|C_1) = \mathbf{21/26}$, $P(w_1|C_2) = \mathbf{15/17}$ and $P(w_2|C_2) = \mathbf{2/17}$, which gives $P^M = \begin{pmatrix} \mathbf{5/26} & \mathbf{21/26} \\ \mathbf{15/17} & \mathbf{2/17} \end{pmatrix}$.
Now if we fix W as $W^\top = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$, denoting the containment of documents in classes, we get $W^\top W = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \implies (W^\top W)^{-1} = \begin{pmatrix} 1/3 & 0 \\ 0 & 1/3 \end{pmatrix}$.
Also, $W^\top X = \begin{pmatrix} 5 & 21 \\ 15 & 2 \end{pmatrix} \implies (W^\top W)^{-1} W^\top X = \begin{pmatrix} 5/3 & 7 \\ 5 & 2/3 \end{pmatrix}$, which actually denote the centroid vectors $(\frac{5}{3}, 7)$ and $(5, \frac{2}{3})$ for classes C_1 and C_2 respectively.
Normalizing each row of this, we get $H^* = \begin{pmatrix} \mathbf{5/26} & \mathbf{21/26} \\ \mathbf{15/17} & \mathbf{2/17} \end{pmatrix}$, which is nothing but P^M.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

If we take a look at our music players, and just think for a while about how they were "born," their non-triviality will be trivial. From the phonograph to the iPod, every kind of technology that delivers music underwent huge changes. Changes did not only take place in the music players, but in music formats too. A huge smooth leap from analog to digital music paved the way (and the need) for researchers to better analyse music, for any application, whatsoever. One such application is Automatic Music Recommendation, either subjective or objective.

We go for a subjective approach and try to find sentiments behind songs. We feel that sentiment-based (or mood-based) music recommendation is the most effective, compared to having other features such as metadata as the basis. Going along the state-of-the-art, we take songs and analyse them for sentiments, both in the supervised and the unsupervised setting. We replace a song (audio) by its lyrics (text).

When we are only given with a lot of song-lyrics, all we can do is to first cluster them using a heuristic; then manually analyse the clusters. So we take the dataset and use a topic model called Latent Dirichlet Allocation (LDA) over it. Some topics correspond to sentiments, which motivates us to go by the approach. We further use it to cluster sentimentally similar songs. What we can conclude from using LDA is that it tries to use word co-occurrence for soft-clustering. Using this approach, we would be able to deal with languages which have no tagged songs.

But in cases where it is possible to get supervision, it is always a better deal. We use a labelled dataset of songs and learn a classifier to differentiate between happy and sad songs. This gives us an insight into the constituents of various songs. We use them to reduce the dimensionality and get excellent accuracies in the reduced dimensional space. We have also seen that classification using Non-negative Matrix Factorization is equivalent to a Naïve Bayes Classifier. This strongly hints that classification is nothing but "matrix factorization," to some extent.

In all, we are able to find sentimentally oriented clusters and sentiment-based classifiers for songs by using only the lyrics. This could be embedded into a sentiment-based music recommender.

6.2 Future Work

In the future, we plan to work on phonetic languages such as Sanskrit, which have excellent prosodic features which have a lot of sentimental aspects embedded [55]. Same approaches could be followed for other artistic texts such as novels, poetry, *etc.* Moreover, we would like to involve natural language based approaches, as mere statistical analysis has worked more as an insight provider. For this, we would like to use knowledge-bases such as WordNet [56] and ConceptNet [36] which would more concretely define sentiments. We would like to build emotion-oriented knowledge bases such as SENTI-WORDNET [57], which could further help in better emotional analysis. As far as NMF is concerned, we would like to look at its equivalences (if any) with other classifiers including SVM.

Appendix A

Three Proses

A.1 The Proses

(1) Agglomerative Hierarchical Wishing

This is a wish I wish I got,

I grow, and grow, and grow.

No, here is a wish I wish I got,

to grow up as a tree.

Oh! No no, this is 'the' wish I wish I got,

I just could be a tree!

(2) A Child's Thought

Does that Tree grow up like me?

"Oh! Where did you see such a tree?"

Up in the meadows,

there's a Tree that grows,

and grows, and grows, and grows.

Why is it a Tree when it's just like me?

(3) A To-be-free Tree

There was a tree,

who wanted to be free,

and free every other tree.

All are made to be free,

but it isn't enough to only be free.

All have their own freedom,

all are free.

A.2 Preprocessing and Analysis

Assume S_1, S_2, S_3 denote proses **(1)**, **(2)** and **(3)** respectively. Please note that we do not consider the title to be a part of the song. After preprocessing (Tokenization, Stop-word Removal, Morphological Analysis, Dictionary Matching, and Frequency Analysis), we get the vocabulary $V = \{\texttt{grow}, \texttt{wish}, \texttt{tree}, \texttt{free}, \texttt{meadow}\}$ and the collection of songs, $\mathcal{S} = \{S_1, S_2, S3\}$ as:

$S_1 = \{(\texttt{grow}, 4), (\texttt{wish}, 6), (\texttt{tree}, 2), (\texttt{free}, 0), (\texttt{meadow}, 0)\}$

$S_2 = \{(\texttt{grow}, 5), (\texttt{wish}, 0), (\texttt{tree}, 4), (\texttt{free}, 0), (\texttt{meadow}, 1)\}$

$S_3 = \{(\texttt{grow}, 0), (\texttt{wish}, 0), (\texttt{tree}, 2), (\texttt{free}, 6), (\texttt{meadow}, 0)\}$

As discussed in Sec. 3.2.1, we find topics z_1 and z_2 to be $\boxed{\texttt{tree, free, grow}}$ and $\boxed{\texttt{wish, grow, tree}}$ respectively. Also, song S_1 gets associated with topic z_2 and S_2, S_3 with topic z_1. This reflects that the first prose, **Agglomerative Hierarchical Wishing** talks about "wishing" and "growing"; while the other two proses, **A Child's Thought** and **A To-be-free Tree** talk about "trees" and "freedom" roughly (which may not align with our intuition, but here the data is too small for the statistical analysis to be reliable; and this analysis was only an illustration).

Bibliography

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning and Research (JMLR)*, vol. 3, pp. 993–1022, January 2003. MIT Press, Cambridge, MA, USA.

[2] B. Liu, M. Hu, and J. Cheng, "Opinion Observer: Analyzing and Comparing Opinions on the Web," in *Proceedings of the 14th International Conference on World Wide Web*, WWW 2005, (Chiba, Japan), pp. 342–351, ACM Press, New York, NY, USA, May 2005.

[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs Up?: Sentiment Classification using Machine Learning Techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, vol. 10 of *EMNLP 2002*, (Philadelphia, PA, USA), pp. 79–86, ACL, Stroudsburg, PA, USA, July 2002.

[4] "AllMusic." `http://www.allmusic.com/`.

[5] R. Mihalcea and H. Liu, "A Corpus-based Approach to finding Happiness," in *Proceedings of the 2006 AAAI Spring Symposium on Computational Approaches to Weblogs*, AAAI-SSCW 2006, (Stanford, California), pp. 139–144, AAAI Press, Palo Alto, California, March 2006.

[6] C. Strapparava and R. Mihalcea, "Learning to Identify Emotions in Text," in *Proceedings of the 23rd Annual ACM Symposium on Applied Computing*, SAC 2008, (Ceará, Brazil), pp. 1556–1560, ACM Press, New York, NY, USA, March 2008.

[7] D. M. Lawrence, "Music and Lyrics," *Warner Bros*, 2007. `http://www.imdb.com/title/tt0758766/`.

[8] J. E. Downey, "A Musical Experiment," *The American Journal of Psychology (AJP)*, vol. 9, no. 1, pp. 63–69, October 1897. JSTOR, New York, NY, USA.

[9] K. Hevner, "The Affective Character of the Major and Minor Modes in Music," *The American Journal of Psychology (AJP)*, vol. 47, no. 1, pp. 103–118, January 1935. JSTOR, New York, NY, USA.

[10] L. L. Balkwill and W. F. Thompson, "A Cross-cultural investigation of the Perception of Emotion in Music: Psychophysical and Cultural cues," *Music Perception: An Interdisciplinary Journal*, vol. 17, no. 1, pp. 43–64, Fall 1999. JSTOR, New York, NY, USA.

[11] A. L. Uitdenbogerd, A. Chattaraj, and J. Zobel, "Music IR: Past, Present and Future," in *Proceedings of the 1st International Symposium on Music Information Retrieval*, ISMIR 2000, (Plymouth, USA), p. 2 (invited talk), ISMIR, October 2000. `http://ciir.cs.umass.edu/music2000/papers/invites/uitdenbogerd_invite.pdf`.

[12] T. Kageyama, K. Mochizuki, and Y. Takashima, "Melody Retrieval with Humming," in *Proceedings of the 1993 International Computer Music Conference*, ICMC 1993, (Tokyo, Japan), pp. 349–351, ICMA, San Francisco, CA, USA, September 1993.

[13] J. Karlgren and D. R. Cutting, "Recognizing Text Genres with Simple Metrics using Discriminant Analysis," in *Proceedings of the 15th International Conference on Computational Linguistics*, vol. 2 of *COLING 1994*, (Kyoto, Japan), pp. 1071–1075, ACL, Stroudsburg, PA, USA, August 1994.

[14] B. Kessler, G. Numberg, and H. Schütze, "Automatic Detection of Text Genre," in *Proceedings of the 8th Conference of the European Chapter of the Association for*

Computational Linguistics, EACL 1997, (Madrid, Spain), pp. 32–38, ACL, Stroudsburg, PA, USA, July 1997.

[15] J. Wiebe, T. Wilson, and M. Bell, "Identifying Collocations for Recognizing Opinions," in *Proceedings of the ACL/EACL 2001 Workshop on Collocation*, ACL 2001, (Toulouse, France), ACL, Stroudsburg, PA, USA, July 2001. `www.cs.pitt.edu/~wiebe/pubs/papers/acl01wkshop.pdf`.

[16] P. D. Turney, "Thumbs up or Thumbs down?: Semantic Orientation Applied to Unsupervised Classification of Reviews," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL 2002, (Philadelphia, PA, USA), pp. 417–424, ACL, Stroudsburg, PA, USA, July 2002.

[17] S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima, "Mining Product Reputations on the Web," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 2002, (Alberta, Canada), pp. 341–349, ACM Press, New York, NY, USA, July 2002.

[18] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews," in *Proceedings of the 12th International World Wide Web Conference*, WWW2003, (Budapest, Hungary), pp. 519–528, ACM Press, New York, NY, USA, May 2003.

[19] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 2004, (Washington, USA), pp. 168–177, ACM Press, New York, NY, USA, August 2004.

[20] G. Mishne, "Experiments with Mood Classification in Blog Posts," in *Proceedings of ACM SIGIR 2005 Workshop on Stylistic Analysis of Text for Information Access*, SIGIR 2005, (Salvador, Brazil), ACM Press, New York, NY, USA, August 2005. `www.science.uva.nl/~gilad/pubs/style2005-blogmoods.pdf`.

[21] S. Owsley, S. Sood, and K. Hammond, "Domain specific Affective Classification of Documents," in *AAAI Symposium on Computational Approaches to Analysing Weblogs*, AAAI-CAAW 2006, (Stanford, California, USA), pp. 181–183, AAAI Press, Palo Alto, California, March 2006.

[22] C. A. Anderson, N. L. Carnagey, and J. Eubanks, "Exposure to violent media: The effects of songs with violent lyrics on aggressive thoughts and feelings," *Journal of Personality and Social Psychology*, vol. 84, issue 5, pp. 960–971, May 2003. American Psychological Association.

[23] R. Omar, J. C. Hailstone, J. E. Warren, S. J. Crutch, and J. D. Warren, "The cognitive organization of music knowledge: a clinical analysis," *Brain*, vol. 133, issue 4, pp. 1200–1213, April 2010. Oxford University Press, Oxford, UK.

[24] T. Li and M. Ogihara, "Detecting Emotion in Music," in *Proceedings of the 4th International Conference on Music Information Retrieval*, ISMIR 2003, (Baltimore, Maryland, USA), pp. 239–240, ISMIR, October 2003.

[25] D. Liu, L. Lu, and H. J. Zhang, "Automatic mood detection from acoustic music data," in *Proceedings of the 4th International Conference on Music Information Retrieval*, ISMIR 2003, (Baltimore, Maryland, USA), pp. 81–87, ISMIR, October 2003.

[26] L. Lu, D. Liu, and H. Zhang, "Automatic mood detection and tracking of music audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, issue 1, pp. 5–18, 2006. IEEE.

[27] "Last.fm." `http://www.last.fm/`.

[28] "MyStrands." `http://strands.com/`.

[29] "iTunes Genius." `http://www.apple.com/itunes/features/#genius`.

[30] "iLike." `http://www.ilike.com/`.

[31] "Musicovery." `http://musicovery.com/`.

[32] D. C. Hsu, "iPlayr — An emotion-aware music platform," Master's thesis, National Taiwan University, Taipei, Taiwan, June 2007. `http://chaweihsu.com/home/assets/doc/thesis_iplayr.pdf`.

[33] R. Cai, C. Zhang, C. Wang, L. Zhang, and W.-Y. Ma, "MusicSense: Contextual music recommendation using emotional allocation modeling," in *Proceedings of the 15th International Conference on Multimedia*, ACM Multimedia 2007, (Augsburg, Germany), pp. 553–556, ACM Press, New York, NY, USA, September 2007.

[34] H. Liu, J. Hu, and M. Rauterberg, "LsM: A new location and emotion aware web-based interactive music system," in *2010 Digest of Technical Papers, International Conference in Consumer Electronics*, ICCE 2010, (Las Vegas, USA), pp. 253–254, IEEE, January 2010.

[35] W. R. Chu, R. Tsai, Y.-S. Wu, H.-H. Wu, H.-Y. Chen, and J.-J. Hsu, "LAMP, A Lyrics and Audio MandoPop Dataset for music mood estimation: Dataset compilation, system construction, and testing," in *The 2010 Conference on Technologies and Applications of Artificial Intelligence*, TAAI 2010, (Hsinchu, Taiwan), pp. 53–59, IEEE, November 2010.

[36] H. Liu and P. Singh, "ConceptNet — a practical commonsense reasoning toolkit," *BT technology journal*, vol. 22, no. 4, pp. 211–226, October 2004. Springer, Berlin/Heidelberg/New York.

[37] C.-Y. Chi, Y.-S. Wu, W. rong Chu, D. Wu, J.-j. Hsu, and R.-H. Tsai, "The power of words: Enhancing music mood estimation with textual input of lyrics," in *3rd International Conference on Affective Computing and Intelligent Interaction*, ACII 2009, (Amsterdam, The Netherlands), pp. 1–6, IEEE, September 2009.

[38] "AUPEO!." `https://www.aupeo.com/`.

[39] "MeeMix." `http://www.meemix.com/`.

[40] "Moodstream." `http://moodstream.gettyimages.com/`.

[41] "STEREOmood." `http://www.stereomood.com/`.

[42] D. Yang, X. Chen, and Y. Zhao, "A LDA-Based Approach to Lyric Emotion Regression," in *Knowledge Engineering and Management: Proceedings of the Sixth International Conference on Intelligent Systems and Knowledge Engineering, (Iske 2011)*, vol. 123 of *Advances in Intelligent and Soft Computing*, (Shanghai, China), pp. 331–340, Springer Berlin / Heidelberg, December 2012.

[43] A. Rajkumar, R. Saradha, V. Suresh, M. Narasimha Murty, and C. E. Veni Madhavan, "Stopwords and Stylometry: A Latent Dirichlet Allocation Approach," in *Proceedings of the NIPS 2009 Workshop on Applications for Topic Models: Text and Beyond (Poster Session)*, (Whistler, Canada), December 2009. `http://www.umiacs.umd.edu/~jbg/nips_tm_workshop/12.pdf`.

[44] G. Kingsley Zipf, *Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley Press, Reading, Massachusetts, USA, 1949. ASIN B001OHOXU6.

[45] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719.

[46] W. Xu, X. Liu, and Y. Gong, "Document Clustering based on Non-negative Matrix Factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, SIGIR 2003, (Toronto, Canada), pp. 267–273, ACM Press, New York, NY, USA, July–August 2003.

[47] C. Ding, X. He, and H. Simon, "On the Equivalence of Non-negative Matrix Factorization and Spectral Clustering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, SDM 2005, (Newport Beach, California, USA), pp. 606–610, SIAM, Philadelphia, Pennsylvania, USA, April 2005.

[48] M. E. Maron, “Automatic Indexing: An Experimental Inquiry,” *Journal of the ACM (JACM)*, vol. 8, issue 3, pp. 404–417, July 1961. ACM Press, New York, NY, USA.

[49] H. Borko and M. Bernick, “Automatic Document Classification,” *Journal of the ACM (JACM)*, vol. 10, issue 2, pp. 151–162, April 1963. ACM Press, New York, NY, USA.

[50] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by Latent Semantic Analysis,” *Journal of the American Society for Information Science*, vol. 41, issue 6, pp. 391–407, September 1990. Wiley-Blackwell, Hoboken, NJ, USA.

[51] T. Hofmann, “Probabilistic Latent Semantic Indexing,” in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, SIGIR 1999, (UCB, Berkeley, USA), pp. 50–57, ACM Press, New York, NY, USA, August 1999.

[52] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, “An introduction to variational methods for graphical models,” *Machine Learning*, vol. 37, no. 2, pp. 183–233, November 1999. Springer, Netherlands.

[53] D. D. Lee and H. S. Seung, “Learning the parts of objects by Non-negative Matrix Factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, October 1999. Nature Publishing Group, London, UK.

[54] R. Arun, V. Suresh, C. E. Veni Madhavan, and M. Narasimha Murty, “On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations,” in *Proceedings of The 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2010*, vol. 6118/2010 of *LNCS*, (Hyderabad, India), pp. 391–402, Springer, Berlin/Heidelberg/New York, June 2010.

[55] S. Mishra, *Chhandovallari: Handbook of Sanskrit prosody.* SABDA — Sri Aurobindo Ashram, Pondicherry, India, 1999. ISBN 8170601231.

[56] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "WordNet: An on-line lexical database," *International Journal of Lexicography*, vol. 3, issue 4, pp. 235–244, Winter 1990. Oxford University Press, Oxford, UK.

[57] A. Esuli and F. Sebastiani, "SENTIWORDNET: A publicly available lexical resource for opinion mining," in *Proceedings of the 5th International Conference on Language Resources and Evaluation*, LREC 2006, (Genoa, Italy), pp. 417–422, May 2006.