

# Mining Sentiments from Songs Using Latent Dirichlet Allocation

Govind Sharma and M. Narasimha Murty

Department of Computer Science and Automation,  
Indian Institute of Science, Bangalore, 560012, Karnataka, India  
{govindjsk,mmm}@csa.iisc.ernet.in

**Abstract.** Song-selection and mood are interdependent. If we capture a song’s sentiment, we can determine the mood of the listener, which can serve as a basis for recommendation systems. Songs are generally classified according to *genres*, which don’t entirely reflect sentiments. Thus, we require an unsupervised scheme to mine them. Sentiments are classified into either two (positive/negative) or multiple (happy/angry/sad/...) classes, depending on the application. We are interested in analyzing the feelings invoked by a song, involving multi-class sentiments. To mine the hidden sentimental structure behind a song, in terms of “topics”, we consider its lyrics and use Latent Dirichlet Allocation (LDA). Each song is a mixture of moods. Topics mined by LDA can represent moods. Thus we get a scheme of collecting similar-mood songs. For validation, we use a dataset of songs containing 6 moods annotated by users of a particular website.

**Keywords:** Latent Dirichlet Allocation, music analysis, sentiment mining, variational inference.

## 1 Introduction

It is required that with the swift increase in digital data, technology for data mining and analysis should also catch pace. Effective mining techniques need to be set up, in all fields, be it education, entertainment, or science. Data in the entertainment industry is mostly in the form of multimedia (songs, movies, videos, *etc.*). Applications such as recommender systems are being developed for such data, which suggest new songs (or movies) to the user, based on previously accessed ones. Listening to songs has a strong relation with the mood of the listener. A particular mood can drive us to select some song; and a song can invoke some sentiments in us, which can change our mood. Thus, song-selection and mood are interdependent features. Generally, songs are classified into genres, which do not reflect the sentiment behind them, and thus, cannot exactly estimate the mood of the listener. Thus there is a need to build an unsupervised system for mood estimation which can further help in recommending songs. Subjective analysis of multimedia data is cumbersome in general. When it comes to songs, we have two parts, *viz.* melody and lyrics. Mood of a listener depends on

both music as well as the lyrics. But for simplicity and other reasons mentioned in Sec. 1.2, we use the lyrics for text analysis.

To come up with a technique to find sentiments behind songs based on their lyrics, we propose to use Latent Dirichlet Allocation (LDA), a probabilistic graphical model, which mines hidden semantics from documents. It is a “topical” model, that represents documents as bags of words, and looks to find semantic dependencies between words. It posits that documents in a collection come from  $k$  “topics”, where a topic is a distribution over words. Intuitively, a topic brings together semantically related words. Finally, each document is a mixture of these  $k$  topics. Our goal is to use LDA over song collections, so as to get topics, which probably correspond to moods, and give a sentiment structure to a song, which is a mixture of moods.

### 1.1 Sentiment Mining

Sentiment Mining is a field wherein we are interested not in the *content* of a document, but in its *impact* on a reader (in case of text documents). We need to predict the sentiment behind it. State of the art [1], [2] suggests that considerable amount of work has been done in mining sentiments for purposes such as tracking the popularity of a product, analysing movie reviews, *etc.* Also, commercial enterprises use these techniques to have an idea of the public opinion about their product, so that they can improve in terms of user-satisfaction. This approach boils down sentiments into two classes, *viz.*, positive and negative. Another direction of work in sentiment mining is based on associating multiple sentiments with a document, which gives a more precise division into multiple classes, (*e.g.*, happy, sad, angry, disgusting, *etc.*) The work by Rada Mihalcea [3], [4] in Sentiment Analysis is based on multiple sentiments.

### 1.2 Music or Lyrics?

A song is composed of both melody and lyrics. Both of them signify its emotions and subjectivity. Work has been done on melody as well as on lyrics for mining sentiments from songs [5]. For instance, *iPlayr* [6] is an emotionally aware music player, which suggests songs based on the mood of the listener. Melody can, in general invoke emotions behind a song, but the actual content is reflected by its lyrics. As the dialogue from the movie, *Music and Lyrics* (2007) goes by, “*A melody is like seeing someone for the first time ... (Then) as you get to know the person, that’s the lyrics ... Who they are underneath ...*”. Lyrics are important as they signify feelings more directly than the melody of a song does. Songs can be treated as mere text documents, without losing much of their sentimental content. Also, a song is generally coherent, relative to any other sort of document, in that the emotions do not vary within a song, mostly. Thus, it is logical to demand crisp association of a song to a particular emotion. But the emotion-classes themselves are highly subjective, which makes the concept of emotions vary from person to person. In the present work, we assume that everyone has the same notion of an emotion.

### 1.3 Latent Dirichlet Allocation [7]

Generally, no relation between the words within a document is assumed, making them conditionally independent of each other, given the class (in classification). Furthermore, this assumption can also be seen in some of the earliest works on information retrieval by Maron [8] and Borko [9]. 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 the 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) [10], probabilistic LSI [11], Latent Dirichlet Allocation (LDA), *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. 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 LDA representation. The parameters  $\alpha$  and  $\beta$  are corpus-level parameters, variable  $\theta$  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  $\zeta$ ;  $\theta$  is a Dirichlet with parameter vector  $\alpha$ . The topics  $z_n$  are supposed to have come from a Multinomial distribution with  $\theta$  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

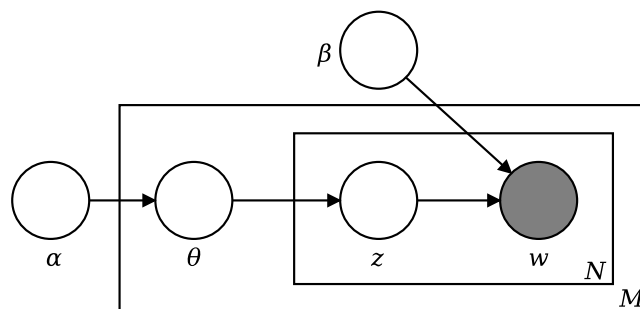


Fig. 1. The Latent Dirichlet Allocation model

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  $\beta$ , a  $v \times k$  matrix,  $v$  being the vocabulary size. Each entry in this matrix gives the probability of a word representing a topic ( $\beta_{ij} = p(w^j = 1|z^j = 1)$ ). From the graphical structure of LDA, we have (1), which is the joint distribution of  $\theta$ ,  $\mathbf{z}$  and  $\mathbf{w}$ , given the parameters  $\alpha$  and  $\beta$ .

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

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. Equation (2) below shows this intractability due to coupling between  $\theta$  and  $\beta$ .

$$p(\mathbf{w}|\alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left( \prod_{i=1}^k \theta_i^{\alpha_i - 1} \right) \left( \prod_{n=1}^N \sum_{i=1}^k \prod_{j=1}^v (\theta_i \beta_{ij})^{w_n^j} \right) d\theta \quad (2)$$

For this reason, we move towards approximate inference procedures.

#### 1.4 Variational Inference [12]

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. 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.

A similar procedure is carried out for probability calculations. When applied to LDA, variational inferencing leads to simplifying our model by relaxing dependencies between  $\theta$  and  $z$ , and  $z$  and  $w$ , just for the inferencing task. The simplified model is shown in Fig. 2, which can be compared with the original LDA model (Fig. 1).

## 2 The Overall System

Our goal is to first acquire lyrics, process them for further analysis and dimensionality reduction, and then find topics using LDA. We hypothesize that some of these topics correspond to moods. The overall system is shown in Fig. 3 and explained in the subsequent sections.

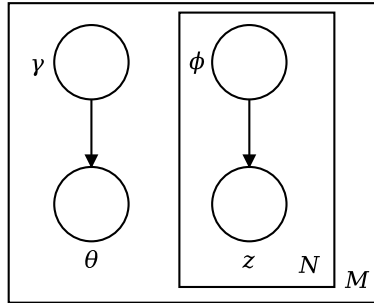


Fig. 2. Approximation of the LDA model for variational inference

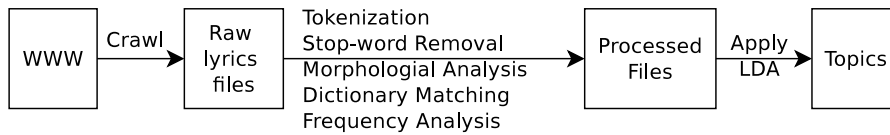


Fig. 3. Block diagram showing the overall system

### 2.1 Data Acquisition

There are a number of websites available on the internet which have provision for submission of lyrics. People become members of these websites and submit lyrics for songs. To collect this dataset, a particular website was first downloaded using *wget*, a Linux tool. Then, each file was crawled and lyrics were collected as separate text files. Let  $F = \{f_1, f_2, \dots, f_l\}$  be the set of  $l$  such files, which are in raw format (see Fig. 4a) and need to be processed for further analysis.

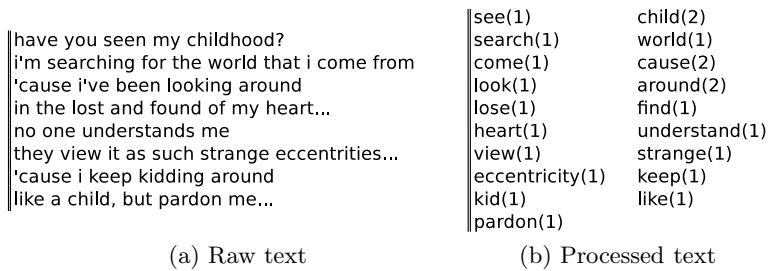


Fig. 4. A snippet of the lyrics file for the song, *Childhood* by *Michael Jackson*

### 2.2 Preprocessing

For each song,  $f_i$  in  $F$ , we carry out the following steps:

**Tokenization.** Here, we extract tokens from the songs, separated by white-spaces, punctuation marks, *etc.* We retain only whole words and remove all other noise (punctuation, parentheses, *etc.*). Also, the frequency of each token is calculated and stored along with  $f_i$ . For example, for the snippet in Fig. 4a, the tokens would be {have, you, seen, my, childhood, i, m, searching, ..., pardon, me}.

**Stop-word Removal.** Stop-words are the words that appear very frequently and are of very little importance in the discrimination of documents in general. They can be specific to a dataset. For example, for a dataset of computer-related documents, the word `computer` would be a stop-word. In this step, standard stop-words such as `a`, `the`, `for`, *etc.* and words with high frequency (which are not in the standard list) are removed. For the data in Fig. 4a, the following are stop words: {have, you, my, i, been, for, the, that, from, in, and, no, me, they, it, as, such, a, but}.

**Morphological Analysis.** The retained words are then analysed in order to combine different words having the same root to a single one. For example, the words `laugh`, `laughed`, `laughter` and `laughing` should be combined to form a single word, `laugh`. This is logical because all such words convey the same (loosely) meaning. Also, it reduces the dimensionality of the resulting dataset. For this, some standard morphological rules were applied (removing plurals, -ing rules, *etc.*) and then exceptions (such as `went` - `go`, `radii` - `radius`, *etc.*) are dealt with using the exception lists available from WordNet [13]. For Fig. 4a, after removal of stop-words, the following words get modified in this step: {seen - see, childhood - child, searching - search, looking - look, lost - lose, found - find, understands - understand, eccentricities - eccentricity, kidding - kid}

**Dictionary Matching and Frequency Analysis.** The final set of words are checked for in the WordNet index files and non-matching words are removed. This filters each song completely and keeps only relevant words. Then, rare words (with low frequency) are also removed, as they too do not contribute much in the discrimination of document classes. For the Fig. 4a data, the following words are *filtered out* according to dictionary matching: {m, ve}.

This leaves us with a corpus of songs which is noise-free (See Fig. 4b) and can be analysed for sentiments. Let this set be  $S = \{s_1, s_2, \dots, s_l\}$ . Each element in this set is a song, which contains words, along with their frequencies in that song. For example, each entry in the song  $s_i$  is an ordered pair,  $(w_j, n_{ji})$ , signifying a word  $w_j$  and  $n_{ji}$  being the number of times it occurs in  $s_i$ . Along with this, we also have a set of  $v$  unique words,  $W = \{w_1, w_2, \dots, w_v\}$  (the vocabulary), where  $w_i$  represents a word.

### 2.3 Topic Distillation

We now fix the number of topics as  $k$  and apply Variational EM for parameter estimation over  $S$ , to find  $\alpha$  and  $\beta$ . We get the topical priors in the form of the

$k \times v$  matrix,  $\beta$ . Each row in  $\beta$  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,  $\beta_{ij}$  is the probability of the  $j^{\text{th}}$  word,  $w_j$  representing the  $i^{\text{th}}$  topic.

Once  $\beta$  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  $\gamma$ , established in Sec.1.4. It is a  $l \times k$  matrix, wherein the rows represent songs, and columns, topics. Each value in  $\gamma$  is proportional (not equal) to the probability of association of a song to a topic. Thus,  $\gamma_{ij}$  is proportional to the probability of the  $i^{\text{th}}$  song,  $s_i$  being associated to the  $j^{\text{th}}$  topic.

Now we can use  $\beta$  to represent topics in terms of words, and  $\gamma$  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  $\pi$  be the probability mass we wish to cover for each song (This will be clearer in Sec. 4). First of all, normalize each row of  $\gamma$  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 ( $j = 1$ ), add the probabilities  $\tilde{\gamma}_{ij}$  and stop when the sum just exceeds  $\pi$ . Let that be the  $r_i^{\text{th}}$  column. All the topics covered before the  $r_i^{\text{th}}$  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,  $T_i = \{(t_1^i, m_1^i), (t_2^i, m_2^i), \dots, (t_{r_i}^i, m_{r_i}^i)\}$  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  $\tilde{T}_i = \{(t_1^i, \mu_1^i), (t_2^i, \mu_2^i), \dots, (t_{r_i}^i, \mu_{r_i}^i)\}$ , 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 Validation of Topics

Documents associated with the same topic must be similar, and thus, each topic is nothing but a cluster. We have refined the clusters by filtering out the topics which had less association with the songs. This results in  $k$  soft clusters of  $l$  songs. Let  $\Omega = \{\omega_1, \omega_2, \dots, \omega_k\}$  be the set of  $k$  clusters,  $\omega_i$  being a cluster containing  $n_i$  songs,  $s_1^i, s_2^i, \dots, s_{n_i}^i$ , with memberships  $u_1^i, u_2^i, \dots, u_{n_i}^i$  respectively. These memberships can be calculated using the sets  $\{\tilde{T}_i\}_{i=1}^l$ , which were formed in the end of Sec. 2.3.

For validation of the clusters in  $\Omega$ , let us have an annotated dataset which divides  $N$  different songs into  $J$  different classes. Let  $\tilde{S}(C, S)$  be the set of songs that are covered by this dataset. Let  $C = \{c_1, c_2, \dots, c_J\}$  be the representation of these classes, where  $c_i$  is a set of songs associated to a class  $i$ . Let  $\tilde{\Omega} = \{\tilde{\omega}_1, \tilde{\omega}_2, \dots, \tilde{\omega}_k\}$  be the corresponding clusters containing only songs from  $\tilde{S}$ . To validate  $\Omega$ , it is sufficient to validate  $\tilde{\Omega}$  against  $C$ .

We use two evaluation techniques [14, Chap. 16] for  $\tilde{\Omega}$ , *viz.* *Purity* and *Normalized Mutual Information* which are described in (3) - (6) below. In these equations,  $|\cdot \tilde{\cap} \cdot|$  is the weighted cardinality (based on membership) of the intersection between two sets.

$$purity(\tilde{\Omega}, C) = \frac{1}{N} \sum_{i=1}^k \max_{j=1}^J |\tilde{\omega}_i \tilde{\cap} c_j|. \quad (3)$$

$$NMI(\tilde{\Omega}, C) = \frac{I(\tilde{\Omega}; C)}{[H(\tilde{\Omega}) + H(C)]/2}, \quad (4)$$

where,  $I$  is the Mutual Information, given by:

$$I(\tilde{\Omega}; C) = \sum_{i=1}^k \sum_{j=1}^J \frac{|\tilde{\omega}_i \tilde{\cap} c_j|}{N} \log \frac{N \cdot |\tilde{\omega}_i \tilde{\cap} c_j|}{|\tilde{\omega}_i| |c_j|}, \quad (5)$$

and  $H$  is the entropy, given by:

$$H(\tilde{\Omega}) = - \sum_{i=1}^k \frac{|\tilde{\omega}_i|}{N} \log \frac{|\tilde{\omega}_i|}{N} \quad H(C) = - \sum_{j=1}^J \frac{|c_j|}{N} \log \frac{|c_j|}{N}. \quad (6)$$

Please note that these measures highly depend on the gold-standard we use for validation.

## 4 Experimental Results

As mentioned earlier, we collected the dataset from LyricsTrax [15], a website for lyrics that provided us with the lyrics of ( $l =$ ) 66159 songs. After preprocessing, we got a total of ( $v =$ ) 19508 distinct words, which make the vocabulary. Assuming different values of  $k$ , ranging from 5 to 50, we applied LDA over the dataset. Application of LDA fetches topics from it in the form of the matrices  $\beta$  and  $\gamma$  and as hypothesised, differentiates songs based on sentiments. It also tries to capture the theme behind a song. For example, for  $k = 50$ , we get 50 topics, some of which are shown in Table 1, represented by the top 10 words for each topic according to the  $\beta$  matrix. Topic 1 contains the words *sun, blue, beautiful, etc.* and signifies a happy/soothing song. Likewise, Topic 50 reflects a sad mood, as it contains the words *kill, blood, fight, death, etc.*

Changing the number of topics gives a similar structure and as  $k$  is increased from 5 to 50, the topics split and lose their crisp nature. In addition to these meaningful results, we also get some noisy topics, which have nothing to do with the sentiment (we cannot, in general, expect to get only sentiment-based topics).

This was the analysis based on  $\beta$ , which associates words with topics. From the posterior inference, we get  $\gamma$ , which assigns every topic to a document, with some probability. We normalize  $\gamma$  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



**Table 1.** Some of the 50 topics given by LDA. Each topic is represented by the top 10 words (based on their probability to represent topics).

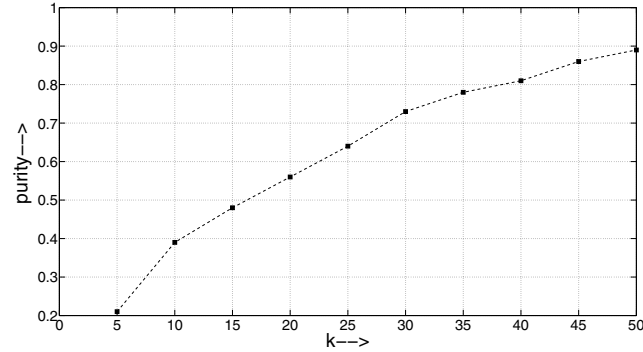
Topic #	Words
1	sun, moon, blue, beautiful, shine, sea, angel, amor, summer, sin
2	away, heart, night, eyes, day, hold, fall, dream, break, wait
3	time, way, feel, think, try, go, leave, mind, things, lose
5	good, look, well, run, going, talk, stop, walk, people, crazy
6	man, little, boy, work, woman, bite, pretty, hand, hang, trouble
8	ride, burn, road, wind, town, light, red, city, train, line
9	god, child, lord, heaven, black, save, pray, white, thank, mother
11	sweet, ooh, music, happy, lady, morn, john, words, day, queen
12	sing, hear, song, roll, sound, listen, radio, blues, dig, bye
50	kill, blood, fight, hate, death, hell, war, pain, fear, bleed

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  $\pi$ . As described in Sec. 2.3, we then find  $T_i$  for each song,  $s_i$  and normalize it over the membership values to get  $\tilde{T}_i$ . To obtain  $\Omega$  (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.

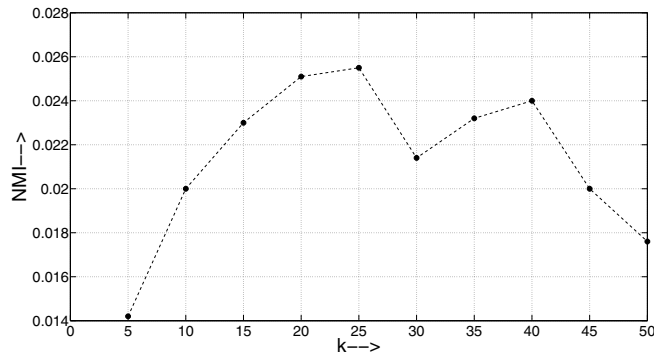
For validation, we crawled through ExperienceProject [16], and downloaded a dataset of songs classified into 6 classes, *viz.*, *Happy*, *Sad*, *Angry*, *Tired*, *Love* and *Funny*. From these songs, we pick the songs common to our original dataset  $S$ . This gives us  $\tilde{S}$ , which contain 625 songs, each associated with one of the 6 classes, by the set  $C = \{c_1, c_2, \dots, c_6\}$ . Each class  $c_j$  contains some songs from among the 625 songs in  $\tilde{S}$ . Intersecting  $\Omega$  with  $\tilde{S}$  gives  $\tilde{\Omega}$ , consisting of only those songs that are in the validation dataset. Now we are ready to validate the clustering,  $\tilde{\Omega}$  against the annotated set  $C$  using each of the measures mentioned in Sec. 3.

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*. These are summarised in Fig. 5a and Fig. 5b. It can be seen that purity of the clustering increases monotonically with the number of topics. If we had one separate cluster for each song, the purity would have been 1. Thus the variation in purity is as expected. On the other hand, NMI, being a normalized measure, can be used to compare clusterings better, with different values of  $k$ . It first increases then decreases, having a maximum at  $k = 25$ , but not much variation is observed.

The measures calculated above only partly evaluate the clusters. Having an actual look at the clusters provide better semantic information as to why certain songs were clustered together. It can be said that LDA captures the subjectivity up to some extent, but also lacks in disambiguating confusing songs which contain similar words (*e.g.*, the words *heart*, *break*, *night*, *eyes*, *etc.* could be part of happy as well as sad songs).



(a) Purity



(b) Normalized Mutual Information

**Fig. 5.** Validation Results

## 5 Conclusion and Future Work

Topic models work on the basis of word co-occurrence in documents. If two words co-occur in a document, they should be related. It assumes a hidden semantic structure in the corpus in the form of topics. Variational inference is an approximate inference method that works on the basis of finding a simpler structure for (here) LDA. As far as songs are concerned, they reflect almost coherent sentiments and thus, form a good corpus for sentiment mining. Sentiments are a semantic part of documents, which are captured statistically by LDA, with positive results. As we can intuitively hypothesize, a particular kind of song will have a particular kind of lyrical structure that decides its sentiment. We have captured that structure using LDA.

Then we evaluated our results based on a dataset annotated by users of the website [16], which gives 6 sentiments to 625 songs. Compared to the 66159 songs

in our original dataset, this is a very small number. We need annotation of a large number of songs to get better validation results.

LDA follows the bag of words approach, wherein, given a song, the sequence of words can be ignored. In the future, we can look at sequential models such as HMM to capture more semantics. Negation can also be handled using similar models. Also, (sentimentally) ambiguous words such as *heart*, *night*, *eyes*, *etc.*, whose being happy or sad depends on the situation of the song, can be disambiguated using the results themselves. In other words, if a word appears in a “happy” topic, is a “happy” word. SentiWordNet [17], which associates positive and negative sentiment scores to each word, can be used to find the binary-class classification of the songs. But that too requires extensive word sense disambiguation. Using the topics, we can improve SentiWordNet itself, by giving sentiment scores for words that have not been assigned one, based on their co-occurrence with those that have one. In fact, a new knowledge base can be created using the results obtained.

**Acknowledgements.** We would like to thank Dr. Indrajit Bhattacharya, Indian Institute of Science, Bangalore for his magnanimous support in shaping up this paper.

## References

1. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification using Machine Learning Techniques. In: Proceedings of the ACL 2002 Conference on Empirical Methods in Natural Language Processing, pp. 79–86. ACL, Stroudsburg (2002)
2. Liu, B.: Opinion Mining. In: Web Data Mining. Springer, Heidelberg (2007)
3. Mihalcea, R.: A Corpus-based Approach to Finding Happiness. In: AAAI 2006 Symposium on Computational Approaches to Analysing Weblogs, pp. 139–144. AAAI Press, Menlo Park (2006)
4. Strapparava, C., Mihalcea, R.: Learning to Identify Emotions in Text. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 1556–1560. ACM, New York (2008)
5. Chu, W.R., Tsai, R.T., Wu, Y.S., Wu, H.H., Chen, H.Y., Hsu, J.Y.J.: LAMP, A Lyrics and Audio MandoPop Dataset for Music Mood Estimation: Dataset Compilation, System Construction, and Testing. In: Int. Conf. on Technologies and Applications of Artificial Intelligence, pp. 53–59 (2010)
6. Hsu, D.C.: iPlayr - an Emotion-aware Music Platform. (Master’s thesis). National Taiwan University (2007)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, 993–1022 (2003)
8. Maron, M.E.: Automatic Indexing: An Experimental Inquiry. *J. ACM* 8(3), 404–417 (1961)
9. Borko, H., Bernick, M.: Automatic Document Classification. *J. ACM* 10(2), 151–162 (1963)
10. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41(6), 391–407 (1990)

11. Hofmann, T.: Probabilistic Latent Semantic Indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57 (1999)
12. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An Introduction to Variational Methods for Graphical Models. *Mach. Learn.* 37(2), 183–233 (1999)
13. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: WordNet: An Online Lexical Database. *Int. J. Lexicography* 3, 235–244 (1990)
14. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008)
15. LyricsTrax, <http://www.lyricstrax.com/>
16. ExperienceProject, [http://www.experienceproject.com/music\\_search.php](http://www.experienceproject.com/music_search.php)
17. Esuli, A., Sebastiani, F.: SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In: Proceedings of the 5th Conference on Language Resources and Evaluation, pp. 417–422 (2006)