Document Clustering

Anirban Chatterjee
ISI,Kolkata
anirbanchatterjee052@gmail.com

Abhinav Chakraborty
ISI,Kolkata
abhinavchakraborty3@gmail.com

Rohan Hore
ISI,Kolkata
mr.horerohan@gmail.com

Abstract

The report is a review of document clustering using direct methods and topic modelling. We introduce the idea of document clustering and present the basic prerequisites for using the topic modelling idea. We present all the pre-processing methods needed for document data-handling. Further we discuss about topic models LSA, PLSA and LDA. We apply the algorithms on data on song lyrics data, in order to compare with the natural division of songs w.r.t. genre.

Contents

1 Introduction

Clustering is an automatic unsupervised learning technique aimed at grouping a set of objects into subsets or clusters. The goal is to create clusters that are coherent internally, but substantially different from each other. Clustering can be adapted for textual data, with various goals like, similar document search, organization of large document collection, duplicate content detection, document recommendation system. The goal of a document clustering scheme is to minimize intra-cluster distances between documents, while maximizing inter-cluster distances (using an appropriate distance measure between documents). Information retrieval from textual data has its many challenges. One of the main problems is ambiguity of the language *i.e.* same word can be interpreted in two or more possible ways(*Polysemy*), whereas there can be multiple phrases associated with same meaning(*Synonymy*). Also there are words in documents outside the dictionary, for example abbreviations like "btw", "ppl" etc. Also there is the computational problem of dealing with huge data matrices. In the following section we will deal with these problems and finally work towards clustering documents.

2 Preprocessing Texual Data

Preprocessing is one of the key components in many text mining algorithms. For example a traditional text categorization framework comprises preprocessing, feature extraction, feature selection and classification steps. Although it is confirmed that feature extraction, feature selection and classification algorithm have significant impact on the classification process, the preprocessing stage may have noticeable influence on this success. Uysal et al. [7] have investigated the impact of preprocessing tasks particularly in the area of text classification. The preprocessing step usually consists of the tasks such as tokenization, filtering, lemmatization and stemming. In the following we briefly describe them

2.1 Tokenization

To work with text data,the first step is parsing the 'huge' document into smaller units(tokens) such as phrases or words. Most commonly use tokenization method is `Bag of Words(BOW)` model. We would discuss about this `BOW` model. Here the frequency or occurrence of words in a document is used to train the model. Using all the available words we form a `Vocabulary` of size N (say). Each document in the collection of documents are represented as a N-length vector w, whose i^{th} co-ordinate is number of times the i^{th} word occurs in the document i.e

$$\boldsymbol{w}_i = \#\{i^{th} \text{ word appears in the document}\}$$

An important highlight of the `BOW` model is that the vector(or list) representation completely ignores the order of occurrence of the words.

2.2 Corpus Cleaning

Cleaning is usually done on documents to remove some of the words and perhaps at the same time throw away certain characters such as punctuation marks. Corpus is defined as a large and structured collection of M documents denoted by $D = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M\}$. A common filtering/cleaning is stop-words removal. Stop words are the words frequently appear in the text without having much content information (e.g. prepositions, conjunctions, etc). With a purpose of cleaning corpus,we remove all the punctuation,numbers from the document too. Though not a cleaning method, but still we mention the step of modifying all words to same case, usually lowercase here.

2.3 Lemmatization

Lemmatization is the task that considers the morphological analysis of the words, i.e. grouping together the various inflected forms of a word so they can be analyzed as a single item. In other words lemmatization methods try to map verb forms to infinite tense and nouns to a single form. Lemmatization aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma*. This is sometimes tedious and error prone, so in practice *stemming* methods are preferred. `Examples`: am,are,is $\rightarrow$ be ; car,cars,car's$\rightarrow$ car

2.4 Stemming

For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivation-ally related words with similar meanings, such as democracy, democratic, and democratization. *Stemming* usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

3 Clustering Prerequisites

Before we proceed with the actual modelling and clustering we need some prerequisites. We need to specify how textual data is stored in the `BOW` model. Also some appropriate measure of distance between document and/or word vectors needs to be defined in order to apply prevalent clustering methods.

3.1 Term Frequency Matrix

As noted in the BOW model,each document is represented by a N-vector containing the word frequencies. Now,combining the N-vector of frequencies corresponding to all the M documents,we can form matrices. In **Document Term Matrix** rows correspond to documents in the collection and columns correspond to terms, the $(i, j)^{th}$ entry contains frequency of j^{th} term in i^{th} document. In **Term Document Matrix** rows correspond to terms and columns correspond to the documents. Rest is clear form context. For instance if one has the following two (short) documents:

- I like databases
- I hate databases

then the document term matrix would be

	I	like	hate	databases
D1	1	1	0	1
D2	1	0	1	1

which shows which documents contain which terms and how many times they appear.

3.2 Term Frequency-Inverse Document Frequency

A more intuitive choice is the **Tf-Idf Weighting**. Tf-idf or **term frequency-inverse document frequency** is a numerical statistic,intended to reflect how important a word is to a document in a corpus. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

3.2.1 Term Frequency

In the case of the term frequency $tf(t,d)$, the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the simplest tf scheme is $tf(t,d) = f_{t,d}$. For a detailed description of various scheme we refer to [6].

3.2.2 Inverse Document Frequency

The **Inverse Document Frequency** is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmic-ally scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

where

- N: total number of documents in the corpus i.e $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $tf(t,d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

3.2.3 Term Frequency-Inverse Document Frequency

Using above definitions of term and inverse document frequency the tf-idf if defined as:

$$tf - idf(t,d,D) = tf(t,d)idf(t,D)$$

a high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf–idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf–idf closer to 0.

3.3 Semantic Similarity

In order to apply clustering techniques we would need to develop a notion of similarity/dissimilarity among documents. Since,we have a vector representation of documents,we can try to define suitable distance measures between those vectors. We describe two commonly used measures:

3.3.1 Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. It is a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90 relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Between two vectors u and v the cosine similarity is defined as:

$$Cosine(u,v) := \frac{\sum_i u_i v_i}{\sqrt{\sum_i u_i^2}\sqrt{\sum_i v_i^2}}$$

3.3.2 Levenshtein Distance

Levenshtein Distance also referred as *edit distance*, is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. For example the Levenshtein Distance between ”kitten” and ”sitting” is 3. A minimal edit script that transforms the former into the latter is:

1 kitten→sitten (substitution of ”s” for ”k”)

2 sitten→sittin (substitution of ”i” for ”e”)

3 sittin→sitting (insertion of ”g” at the end).

3.4 Clustering Methods Used

In above sections we have defined a similarity/distance measure, now we are ready to use well known unsupervised learning methods for textual clustering purpose. We will be using the following two clustering algorithms:

- *K-means clustering method*
- *Hierarchical clustering method*

4 Topic Models

4.1 Latent Semantic Analysis

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text *(the distributional hypothesis)*. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and singular value decomposition (SVD) is used for dimension reduction while keeping the semantic structure. Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

4.1.1 Occurrence Matrix

LSA can use a term-document matrix which describes the occurrences of terms in documents; it is a sparse matrix whose rows correspond to terms and whose columns correspond to documents. A commonly used weight of matrix elements is the **tf-idf** weighting scheme.

4.1.2 Rank Lowering

After the construction of the occurrence matrix, LSA finds a low-rank approximation to the term-document matrix. There can be various intuitive reasons behind this:

- The original term-document matrix is presumed too large for the computing, hence the lower dimensional reduction gives an appropriate approximation.
- The original term-document matrix is presumed noisy: for example, anecdotal instances of terms are to be eliminated. From this point of view, the approximated matrix is interpreted as a de-noisified matrix.
- Lastly rank lowering is supposed to remove. It is expected to merge the dimensions associated with terms that have similar meanings.

4.1.3 Derivation

Suppose X= $\begin{bmatrix} x_{1,1} & \dots x_{1,j} & \dots x_{1,n} \\ \vdots & x_{i,j} & \vdots \\ x_{m,1} & x_{m,j} & x_{m,n} \end{bmatrix}$ denotes the term-document matrix (possibly weighted by tf-idf). Each row $t_i^\top = \begin{bmatrix} x_{i,1} \dots x_{i,j} \dots x_{i,n} \end{bmatrix}$ represents the term vector, giving its relation to each document. Each column $d_j^\top = \begin{bmatrix} x_{1,j} \dots x_{i,j} \dots x_{m,j} \end{bmatrix}$ represents the document vector, giving its relation to each term.
Now we can consider a SVD of X, say $X = U\Sigma V^\top$, where U and V are orthogonal matrices and Σ is a diagonal matrix.
Further we have columns of U are the eigenvectors of $XX^\top$ and columns of V are the eigenvectors of $X^\top X$. That is we have

$$\overset{X}{\overset{(\mathbf{d}_j)}{\downarrow}} \qquad\qquad\qquad\qquad U \qquad\qquad \Sigma \qquad\qquad \overset{V^T}{\overset{(\hat{\mathbf{d}}_j)}{\downarrow}}$$

$$(\mathbf{t}_i^T) \to \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix} = (\hat{\mathbf{t}}_i^T) \to \begin{bmatrix} \begin{bmatrix} \mathbf{u}_1 \end{bmatrix} \dots \begin{bmatrix} \mathbf{u}_l \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \cdot \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix}$$

in the above representation we can consider a lower dimensional representation. If we take the largest k many singular values then the k rank approximation of X will be

$$X_k = U_k \Sigma_k V_k^\top$$

where U_k contains the corresponding columns of U and $V_k^\top$ contains the corresponding rows of $V^\top$. Now observe that Σ_k and $V_k^\top$ contributes equally to all the rows of X_k, then the only differentiating factor comes from the rows of U_k, so we can consider the i^{th} row of U_k as the representation of term vector in the lower dimension. Similarly we can consider the columns of $V_k^\top$ to be the representation of document vector in the lower dimension. We should note that these lower dimensional vectors do no correspond to any comprehensible concepts, they are a lower-dimensional approximation of the higher-dimensional space.

4.1.4 Theoretical Justification

The main force behind LSA is the technique of SVD for dimension reduction. Following theorem justifies the use of SVD:

Theorem 1 *Suppose C is a matrix of rank r, and C_k is the $k(< r)$ dimensional reduction of C obtained via SVD, and $|| \quad ||_F$ is the Frobenius norm, then*

$$\min_{Z|rank(Z)=k} ||C - Z||_F = ||C - C_k||_F$$

4.1.5 Applications

We can use the above obtained lower dimensional representation in the follwing cases:

- Compare the documents in the low-dimensional space. We can compare documents j and l by computing the cosine distance between the lower dimensional representations scaled by the singular values. We can think of the singular values as the relative importance of each direction in the lower dimensional space.
- A similar comparison between terms can be followed.
- Given a query, it can be thought of as a mini document and can be compared to documents already in the training set. But before comparing we have to make the same transformation on the query document as the training documents. So if q is the query document then we have to consider
$$\hat{q} = \Sigma_k^{-1} U_k^\top q$$
and compare using this $\hat{q}$. A similar work can be done for term query.

4.1.6 Limitations

Some of the limitations of LSA include:

- The resulting dimensions may be difficult to interpret. LSA leads to results that can be justified on a mathematical level but may have no interpretable meaning in natural language.
- Limitations of bag of words model (BOW), where a text is represented as an unordered collection of words. To address some of the limitation of bag of words model (BOW), multi-gram dictionary can be used to find direct and indirect association as well as higher-order co-occurrences among terms.
- One of the main limitations of LSA is the absense of any generative model. This limitation will be addressed in the models described below.

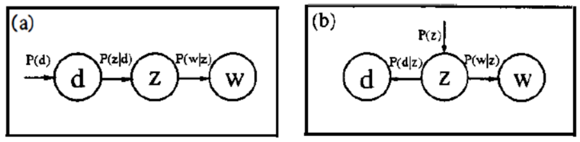


Figure 1: Graphical model representation for the equivalent parameterization

4.2 Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis (PLSA) is a statistical technique for the analysis of co-occurrence data, which has applications in information retrieval and filtering, natural language processing, machine learning from text, and in related areas. Compared to standard Latent Semantic Analysis which stems from linear algebra and performs a Singular Value Decomposition of co-occurrence tables, the proposed method is based on a mixture decomposition derived from a latent class model. This results. in a more principled approach which has a solid foundation in statistics.

4.2.1 Aspect Model

The starting point for Probabilistic Latent Semantic Analysis is a statistical model which has been called aspect model[5]. The aspect model is a latent variable model for co-occurrence data which associates an unobserved class variable $z \in \mathcal{Z} = \{z_1, \ldots, z_K\}$ with each observation. A joint probability model over $\mathcal{D} \times \mathcal{W}$ is defined by the mixture

$$P(d,w) = P(d)P(w|d), P(w|d) = \sum_{z \in \mathcal{Z}} P(w|z)P(z|d)$$

The aspect model introduces a conditional independence assumption, namely that d and w are independent conditioned on the state of the associated latent variable(the corresponding graphical model representation is depicted in Figure 1 (a)). Since the cardinality of z is smaller than the number of documents/words in the collection, z acts as a bottleneck variable in predicting words. It is worth noticing that the model can be equivalently parameterized by (see fig (b))

$$P(d,w) = \sum_{z \in \mathcal{Z}} P(z)P(d|z)P(w|z)$$

4.2.2 Likelihood

The likelihood of observed data can be written as

$$L = \prod_{(d,w)} P(w,d) = \prod_{d \in \mathcal{D}} \prod_{w \in \mathcal{W}} P(w,d)^{n(d,w)}$$

where $n(d, w)$ measures the frequency of word w in document d. The log-likelihood can be now written as

$$\ell = \log L = \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d,w) log\Bigg(\sum_{z \in \mathcal{Z}} P(w|z)P(z|d)P(d)\Bigg)$$

4.2.3 Model fitting with EM

Now that we have the expression of the Likelihood, we would like have the estimates of the parameters $P(w|z), P(d|z)$ and $P(z)$. The standard procedure for maximum likelihood estimation in latent variable models is the Expectation Maximization (EM) algorithm. EM alternates two coupled steps: an expectation (E) step where posterior probabilities are computed for the latent variables then an maximization (M) step, where parameters are updated. Standard calculations yield the E-step equation as

$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in \mathcal{Z}} P(z')P(d|z')P(w|z')}$$

as well as the following M-step formulae

$$P(w|z) \propto \sum_{d \in \mathcal{D}} n(d,w)P(z|d,w)$$

$$P(d|z) \propto \sum_{w \in \mathcal{W}} n(d,w)P(z|d,w)$$

$$P(z) \propto \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d,w)P(z|d,w)$$

4.2.4 Relation with LSA

Recall the model

$$P(d,w) = \sum_{z \in Z} P(z)P(d|z)P(w|z)$$

The joint probabilities can be interpreted as follows

$$P = U\Sigma V^\top \text{ With } U, V, \Sigma \text{ defined as}$$

$U_{d,z}$ contains the probabilities $P(d|z)$
$V_{w,z}$ contains the probabilities $P(w|z)$
Σ is a diagonal matrix of prior probabilities P(z)

4.2.5 Limitations

- The number of parameters grows linearly with the size of training documents.
- Although PLSA is a generative model of the documents in the collection it is estimated on, it is not a generative model of new documents.
- pLSA using EM often overfits the data therefore giving lacking generalisation power, one of the remedies is using a modification of EM-algorithm called the "tempered EM".

4.3 Latent Dirichlet Allocation

We start with a cooked-up example of LDA,showing what we want to achieve through LDA in document clustering case.Below follows the example.We expect that the example will motivate the reader to read further.

- I like to eat broccoli and bananas. (100% topic A)
- I ate a banana and spinach smoothie for breakfast.(100% topic A)
- Chinchillas and kittens are cute.(100% topic B)
- My sister adopted a kitten yesterday.(100% topic B)
- Look at this cute hamster munching on a piece of broccoli.(60% topic A,40% topic B)

Interpretations of topics can be provided as follows.Observing the sentences above,topics can be seen as a combination of words with each word having a probability to show up in a document.

- *Topic A:* 30% broccoli, 15% bananas, 10% breakfast, 10% munching,...(**Vegetable**)
- *Topic B*: 20% chinchillas, 20% kittens, 20% cute, 15% hamster,...(**Animals**)

There is an intrinsic assumption of documents in corpus,along with 'bag-of-words' model in LSA or PLSA.A classic representation theorem due to de Finetti(1990) establishes that any collection of exchangeable random variables has a representation as a mixture distribution—in general an infinite mixture. Thus, if we wish to consider exchangeable representations for documents and words, we need to consider mixture models that capture the exchangeability of both words and documents.This line of thinking leads us to LDA model.

Now,let's go deep into **Latent Dirichlet Allocation**.In natural language processing, latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics.Here each document is considered to have a set of topics that are assigned to it via LDA.Here each document is considered to have a set of topics that are assigned to it via LDA. This is identical to probabilistic latent semantic analysis (PLSA), except that in LDA the topic distribution is assumed to have a Dirichlet prior.

The basic notations we need to get in depth of LDA are as follows.We will use these following notations for further study in LDA.

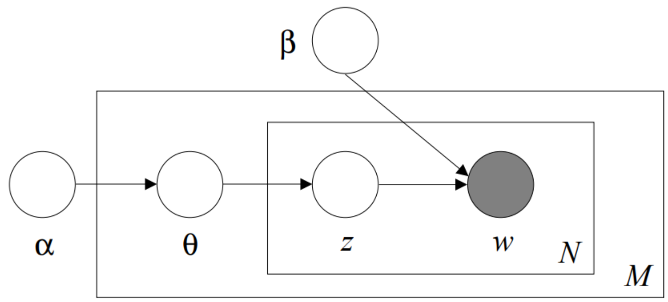


Figure 2: Graphical Model Representation of LDA. The outer plate represents documents, while the inner plate represents choice of topics and words within a document

- We represent words by unit-basis vectors that have a single component equal to one and all others equal to zero. So the i^{th} word in vocabulary V would be a vector w such that $w^i = 1$ and $w^j = 0 \quad \forall j \neq i$.
- A document is a sequence of N words denoted by $\boldsymbol{w} = \{w_1, w_2 \ldots, w_N\}$, where w_n is the n^{th} word in the sequence.
- A corpus is a collection of M documents denoted by $D = \{\boldsymbol{w_1, w_2} \ldots, \boldsymbol{w_M}\}$

4.3.1 Model

Now,we go through the LDA model and try to understand it.LDA assumes the following generative process for each document $\boldsymbol{w}$ in a corpus D.

- Choose $N \sim \text{Poisson}(\xi)$
- Choose $\theta \sim \text{Dir}(\alpha)$
- For each of the N words w_n:
  - Choose a topic $z_n \sim \text{Multinomial}(\theta)$
  - Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on topic z_n.
- The word probabilities are parametrized by a $k \times V$ matrix β, where $\beta_{ij} = p(w^j = 1|z^i = 1)$.

Here,the outer box is the document level and the inner box is the word level.The figure 1 is a nice graphical representation of the LDA model and easier to get hold of the model.

We would use the above generative model to find the probability of observing a corpus.Now,recall the dirchlet random variable is generated according to the probability

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \ldots \theta_k^{\alpha_k - 1}$$

where θ is a k-dimensional Dirichlet Random Variable and $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ are the dirichlet parameters. Given parameters α and β, the joint distribution of word and topic of N-word document and a topic mixture θ is given by

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

Then the marginal distribution of a N-word document becomes

$$p(\boldsymbol{w}|\alpha, \beta) = \int p(\theta|\alpha) \bigg(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \bigg) d\theta$$

Hence,the probability of a corpus can be written as

$$\begin{aligned} p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \\ \bigg(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \bigg) d\theta_d \end{aligned}$$

The parameters α and β are corpus level parameters, assumed to be sampled once in the process of generating a corpus. The variables θ_d are document-level variables, sampled once per document. Finally, the variables z_{dn} and w_{dn} are word-level variables and are sampled once for each word in each document.

Some of the basic models can be reviewed now from which we can build the LDA model sequentially as follows

- **Unigram Model:** The words in every document is drawn from a single multinomial distribution.

  $$p(\boldsymbol{w}) = \prod_{i=1}^{N} p(w_n)$$

- **Mixture of Unigrams:** Here each document is generated by first choosing a topic and generating N words from the conditional multinomial $p(w|z)$

  $$p(\boldsymbol{w}) = \sum_{z} p(z) \prod_{i=1}^{N} p(w_n|z)$$

- **Probabilistic Latent Semantic Ananlysis**

  $$p(d, w_n) = p(d) \sum_{z} p(w_n|z) p(z|d)$$

This shows the development of LDA model from the Unigram model by incorporating complex structures in the model.We have put pictures to show all the model and the LDA model in order to show the relative structural-complexity of the models.

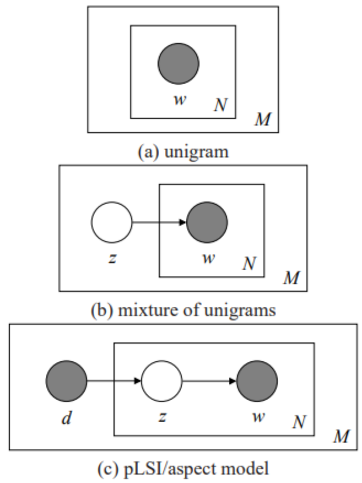


Figure 3: Graphical Representation of Other Models

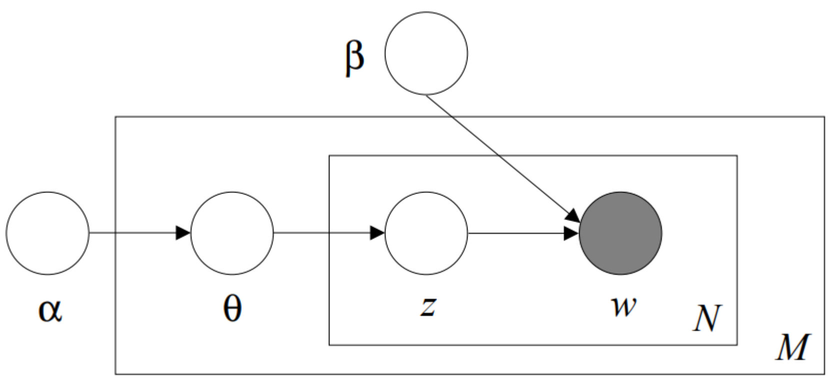


Figure 4: Graphical Representation of LDA Model

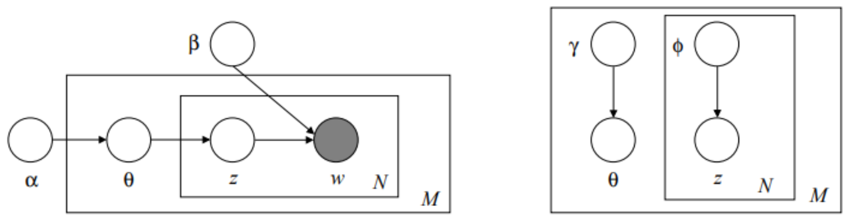


Figure 5: (Left) Graphical Model Representation of LDA. (Right) Graphical Model Representation of the variational distribution

4.3.2 Estimation

Here the main inferential problem is to compute the posterior distribution of hidden variable i.e. topic distribution θ given a document.

$$p(\theta, \boldsymbol{z}|\boldsymbol{w}, \alpha, \beta) = \frac{p(\theta, \boldsymbol{z}, \boldsymbol{w}|\alpha, \beta)}{p(\boldsymbol{w}|\alpha, \beta)}$$

This distribution is intractable to compute in general because of the coupling between θ and β.Indeed to normalize the distribution,we marginalize over the hidden variables in terms of model parameters.

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

as proposed in [2] we use **Variational Inference** method to estimate the LDA model as will be suggested below.Although the posterior distribution in intractable for inference,we can use a wide variety of approximate inference algorithms.The basic idea of variational inference is to make use of Jensen's Inequality to obtain an adjustable tight lower bound to the log-likelihood.A way to obtain tractable family of lower bounds is to consider simple modifications of the original graphical model in which some of the edges and nodes are removed.We have put a pictorial representation of the simplified graphical model assumed in the variational inference. From the resulting simplified graphical model,with free variational parameters,we obtain a family of distribution on the latent variables

$$q(\theta, z|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n)$$

where,the Dirichlet parameters γ and multinomial parameters $(\phi_1, \ldots, \phi_N)$ are free varaitional parameters.

One can show,finding a tight lower bound on the log-likelihood translates directly into the following optimization problem:

$$(\gamma^*, \phi^*) = arg \min_{\gamma, \phi} D(q(\theta, \boldsymbol{w}|\gamma, \phi)||p(\theta, \boldsymbol{z}|\boldsymbol{w}, \alpha, \beta)$$

where D is Kullback-Leibler Divergence between variational distribution and true posterior.

For the estimation purpose,we finally use a alternating variational EM procedure,as explained below

- **E-step**:For each document,find the optimizing values of the variational parameters $\{\gamma_d^*, \phi_d^* : d \in \mathcal{D}\}$This is done as discussed before
- **M Step**: Maximize the resulting lower bound on the log- likelihood w.r.t the model parameters α and β.

The two steps are repeated until the lower bound on the log-likelihood converges.For further in-depth understanding of LDA,we strongly recommend the reader to listen the lectures by david blei himself. [1]

4.3.3 Clustering

As long as we get hold of the estimate of posterior,we have the probabilities $p(\theta|\boldsymbol{w})$.This probability vector(document-topic vector) can be used as a representation for the document for normal clustering methods We can use **Jensen-Shannon Divergence** as a notion of dissimilarity between the document-topic vectors.

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M)$$

with $M = \frac{P+Q}{2}$

We can also possibly use,euclidean distance between the one-coordinate removed document-topic vector.The probability vectors are element of a simplex.So,removing one coordinate makes them an element of euclidean space of a less dimension.Also,another suggestion is if we choose the topic with maximum posterior probability $p(\theta|\boldsymbol{w})$ and report simply that as our cluster id.Here,some prior belief on the no of topics will possibly be helpful.

5 Evaluation Measures

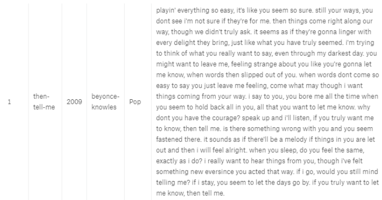
Measuring clustering accuracy corresponds to measuring how much "internally consistent" a cluster is.In presence of ground truth measures,we can compute measures of discrepancy/similarity between estimated clustering and ground-truth partition.

- Rand Index
- Normalized Mutual Information(NMI)

Rand Index:

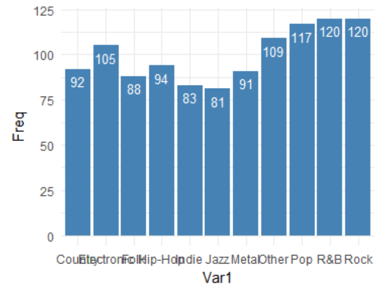
Given a set $S = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$,say we have two partitions of S,namely $X = \{X_1, X_2, \ldots, X_r\}$,partition into r subsets and
$Y = \{Y_1, Y_2, \ldots, Y_s\}$,partition into s subsets.If we define,a as the number of pairs of elements in S that are in the same subset in X and in the same subset in Y,b as the number of pairs of elements in S that are in the different subset in X and in the different subset in Y.The **Rand Index**,R is defined as

$$\frac{a+b}{\binom{n}{2}}$$



1	then-tell-me	2009	beyonce-knowles	Pop	playin' everything so easy, it's like you seem so sure. still your ways, you dont see i'm not sure if they're for me. then things come right along our way, though we didn't truly ask. it seems as if they're gonna linger with every delight they bring, just like what you have truly seemed. i'm trying to think of what you really want to say, even through my darkest day. you might want to leave me, feeling strange about you like you're gonna let me know, when words then slipped out of you. when words dont come so easy to say you just leave me feeling, come what may though i want things coming from your way. i say to you, you bore me all the time when you seem to hold back all in you, all that you want to let me know. why dont you have the courage? speak up and i'll listen, if you truly want me to know, then tell me. is there something wrong with you and you seem fastened there. it sounds as if there'll be a melody if things in you are let out and then i will feel alright. when you sleep, do you feel the same, exactly as i do? i really want to hear things from you, though i've felt something new eversince you acted that way. if i go, would you still mind telling me? if i stay, you seem to let the days go by. if you truly want to let me know, then tell me.

(a)



(b)

Figure 6: (a) One Typical Lyrics, (b) Representation of Genres

It lies within 0 and 1,while 0 indicating the clusters don't agree on any pairs of points and 1 indicates the clusters match exactly.

NMI:
Suppose,X and Y are random variables defined by X= the Class Labels and Y=the Cluster Labels.Then,**Normalized Mutual Information**(NMI) can be defined as

$$NMI(X,Y) = \frac{I(X;Y)}{min(H(x),H(Y))}$$

where,$H()$ represents the entropy and $I()$ represents the mutual information between two random variables.

6 Simulation

All of the methods will be now applied on a data,collected from "MetroLyrics Data".It is a kaggle dataset,posted with a purpose of genre identification from lyrics.There are around 3,80,000+ lyrics in the data set from a lot of different artists from a lot of different genres arranged by year. Every artist folder has a genre.txt file that tells what is the genre of the musician.We firstly have arranged a csv file with all the song id,lyrics and their respective genres and artists.We put one picture for one typical instancce of the csv file among many in the collection.

6.1 Preprocessing Steps

We did following pre-processing with the data to get the data easy to work with.

- We take the whole dataset and used stratified sampling to select our working dataset of size 1120.
- We used inbuilt corpus cleaning functions in **R** to remove punctuation,number,white-spaces and stop-words.
- We used the above stated tf-idf weighting function to form the term-document matrix.

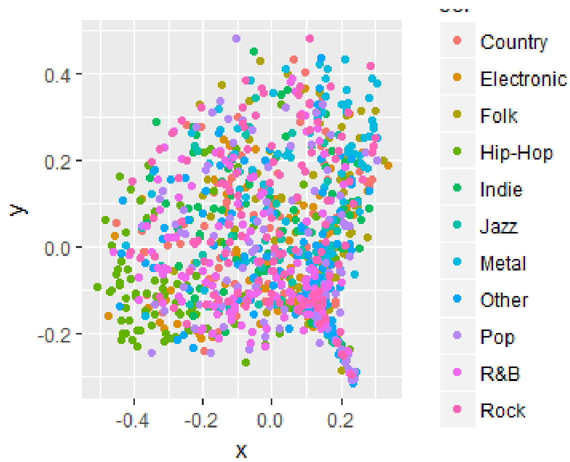
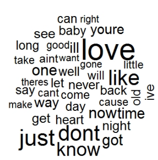
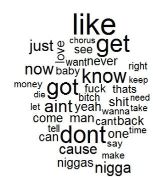


Figure 7: Actual Data

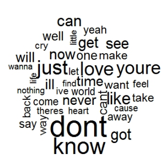
- We looked at a genre-wise and entire word-cloud to identify few important words. The word-clouds have been put in 8 here for a brief idea about what we are working with.



(a)



(b)



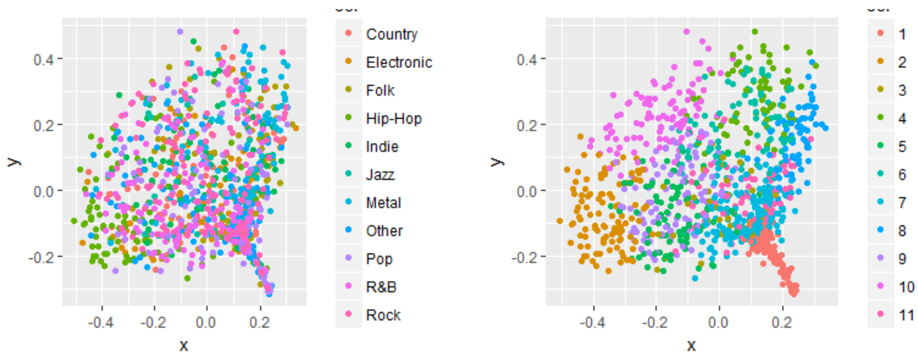
(c)

(d)

Figure 8: Most Important Words in (a) Country Genre, (b) Hip-Hop Genre, (c) Rock Genre, (d) Entire Lyrics Data

6.2 Implementing LSA

We used `LSA` package in **R** to produce the lower rank approximation to the term-document matrix weighted by tf-idf weighting. Further we used k-means clustering on the lower dimensional document vectors for the final clustering output. A comparison is shown in 9. Here in the picture we have plotted the 2 principal components of data,as found from MDS-scaling.

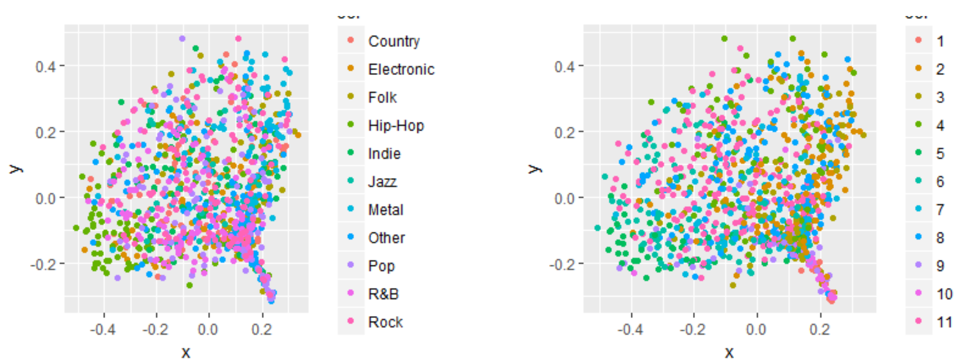


(a) Actual Clusters (b) k-means LSA Clusters

Figure 9: Comparison between Actual Clusters and Clusters identified by k-means and LSA

6.3 Implementing LDA

We used `LDA` function in `topicmodels` package in **R**. From there as an output we received the probability vector $P(\theta|\boldsymbol{w})$. Then used the above defined JS divergence as a similarity measure for clustering. We show a comparison in 10



(a) Actual Clusters (b) LDA Clusters

Figure 10: Comparison between Actual Clusters and Clusters identified by LDA

6.4 Choosing Hyper-parameters

Above we have used the k-means clustering algorithm. In order to use that we need to have prior knowledge of k,i.e the number of clusters. We describe the Elbow method for such an case. Further we would describe algorithms for choosing number of topics in case of LDA.

6.4.1 The Elbow Method

The elbow method looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion". For our data the plot of wss versus number of clusters is in 11.

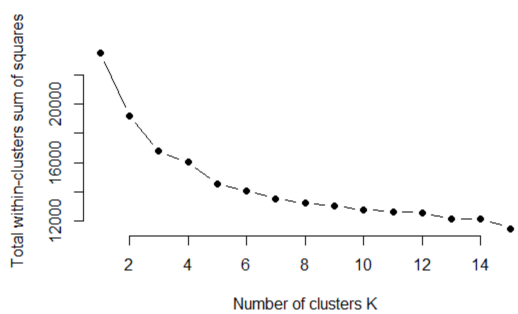


Figure 11: Elbow Method

6.4.2 Choosing Topic Number

Recall from the LDA model,the topic number is specified within the model.But to user,it is unknown.Various measures can be employed to find the optimal no of topics.We list a few of the masures

- Perplexity
- Griffith's measure [4]
- CaoJuan's Measure [3]

6.4.2.1 Griffith's Measure

To evaluate the consequences of changing the number of topics T,we use the Gibbs sampling algorithm to obtain samples from the posterior distribution of corpus words given no of topics at several choices of no of topics.The criteria is to be maximized in order to get our optimal choice.

For our dataset the plot is as in 12.We can clearly see that the measure keeps on increasing for our data-set,thus don't provide a optimal choice.

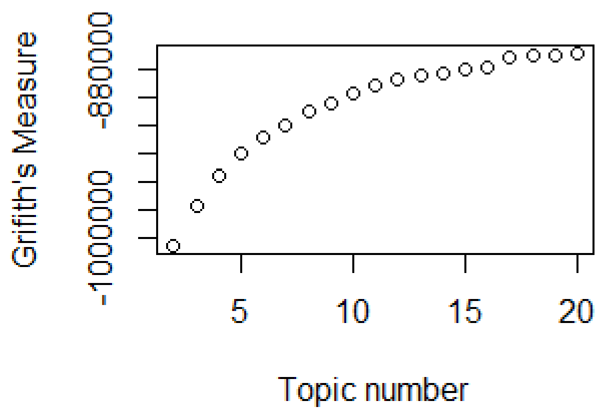


Figure 12: Griffith's Measure

6.5 Results Obtained

6.5.1 Topic Wise Most Probable Words

We here tabulate in 1 the most probable words ranked by the probability $P(w|z)$ for four topics, and observe that the words share a common topic between them example topic 4 has all spanish words mostly.

6.5.2 Comparison Between Methods

Using the above stated evaluation measures we show a comparison between the used methods. We would use the rand index and the NMI criterion and show obtained results in 2.

7 Remarks

It is quite clear that the clustering algorithms on the above data didn't work very good. Intuitively, it shouldn't have done because lyrics don't define a genre. Background scores, musical instruments and various other stuffs are constituent elements of genre.

Further, as a future work, we propose to use recent topic model methods such as *Non-Negative Matrix Factorization* on the dataset.

Top 10 words of each topic(ranked)			
Topic 1(likely pop)	Topic 2(Hip Hop)	Topic 3 (death metal)	Topic 4(Spanish songs)
"love"	"aint"	"die"	"que"
"youre"	"got"	"ich"	"con"
"ill"	"nigga"	"und"	"los"
"dont"	"like"	"der"	"amor"
"just"	"get"	"den"	"daddy"
"baby"	"shit"	"ist"	"como"
"make"	"niggas"	"nicht"	"quiero"
"cant"	"money"	"wie"	"las"
"way"	"big"	"instrumental"	"mas"
"can"	"fuck"	"auf"	"por"

Table 1: Topic Wise Most Probable Words

Method	**Rand Index**	**NMI**
LSA K-Means	0.8274845	0.1109488
LDA k-means(JS)	0.81754	0.09239208
LDA k-means(Euclidean)	0.81536	0.0728414
LDA MAP predictor	0.8294488	0.1001931

Table 2: Comparison between the methods

References

[1] David Blei. *Topic Models by Blei.* videolectures. 2009. URL: `http://videolectures.net/mlss09uk_blei_tm/`.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[3] Juan Cao et al. "A density-based method for adaptive LDA model selection". In: *Neurocomputing* 72.7-9 (2009), pp. 1775–1781.

[4] Thomas L Griffiths and Mark Steyvers. "Finding scientific topics". In: *Proceedings of the National academy of Sciences* 101.suppl 1 (2004), pp. 5228–5235.

[5] Thomas Hofmann. "Probabilistic latent semantic analysis". In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc. 1999, pp. 289–296.

[6] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. "Scoring, term weighting and the vector space model". In: *Introduction to information retrieval* 100 (2008), pp. 2–4.

[7] Alper Kursat Uysal and Serkan Gunal. "The impact of preprocessing on text classification". In: *Information Processing & Management* 50.1 (2014), pp. 104–112.

8 Code

```r

##packages to load
library(tm)
library(SnowballC)
library(ggplot2)
library(lsa)
library(cluster)
library(proxy)
library(wordcloud)
library(MASS)
library(svs)
library(topicmodels)
library(igraph)
library(ldatuning)

##lyrics data reading
lyrics_data_all = read.csv("lyrics.csv",stringsAsFactors = F)

#omitting those with no lyrics
to_omit=which(lyrics_data_all$lyrics=="")

#working full data set
lyrics_data_all=lyrics_data_all[setdiff(1:339277,to_omit),]

##omitting instances with genre not available
to_omit=which(lyrics_data_all$genre=="Not Available")
lyrics_data_all=lyrics_data_all[setdiff(1:dim(lyrics_data_all)[1],to_omit),]

##stratified sampling
n=dim(lyrics_data_all)[1] ##no of lyrics we will take
genre_names=names(table(lyrics_data_all$genre))
genre_size=as.numeric(table(lyrics_data_all$genre))
subset=numeric(0)
sampling_sizes=numeric(length(genre_names))
for (j in 1:length(genre_names)){
  sampling_set=lyrics_data_all[lyrics_data_all$genre==genre_names[j],]
  sampling_sizes[j]=sample(80:120,1) #change this according to the size of data
  subset=rbind(subset,sampling_set[sample(1:genre_size[j],sampling_sizes[j],rep=F),])
}
lyrics_data=subset

#str(text_data)
colnames(lyrics_data)=c("doc_id",names(lyrics_data)[2:5],"text")
df=as.data.frame(lyrics_data)

```

```
##for aaplying the code on kobita data comment above part and uncomment the
    below lines
#kobita_data
#df=as.data.frame(read.csv("C:/Users/lenovo/Downloads/Kobita_Data/kobita_and_
    porjaay.csv",header = T))
#colnames(df)=c("doc_id","text","genre")


#plotting representations of each genres in the data
to.plot=as.data.frame(table(df$genre))
ggplot(data=to.plot, aes(x=Var1, y=Freq)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=Freq), vjust=1.6, color="white", size=3.5)+
  theme_minimal()

#replacing \n with white spaces
df$text=gsub("[\r\n]", " ", df$text)

##making the corpus
df_source=DataframeSource(lyrics_data)
df_corpus = VCorpus(df_source)
print(df_corpus)

##pre-processing steps
clean_corpus <- function(corpus) {
  # Remove punctuation
  corpus <- tm_map(corpus, removePunctuation)
  #removing numbers
  corpus <- tm_map(corpus, removeNumbers)
  # Transform to lower case
  corpus <- tm_map(corpus, content_transformer(tolower))
  # Add more stopwords
  corpus <- tm_map(corpus, removeWords, c(stopwords("en")))
  # Strip whitespace
  corpus <- tm_map(corpus, stripWhitespace)
  #stemming (can ignore for now)
  #corpus <- tm_map(corpus, stemDocument, language = "english")
  return(corpus)
}
cleaned_corpus=clean_corpus(df_corpus)
cleaned_corpus

##genre wise wordcloud
for(id in unique(df$genre)){
  cat(id)
  dummy_df=df[df$genre==id,]
  dummy_df_corpus=VCorpus(DataframeSource(dummy_df))
  dummy_df_corpus=clean_corpus(dummy_df_corpus)
  wordcloud(dummy_df_corpus,max.words = 20)
}
```

```
#save(cleaned_corpus, file = "lyrics_corpus.rds")
#load("lyrics_corpus.rds")

#pictorial visualisation of important words
wordcloud(cleaned_corpus,max.words = 100)

#term document matrix
initial_td.mat<-TermDocumentMatrix(cleaned_corpus)

#removing empty documents after all the preprocessing
col_totals=apply(initial_td.mat,2,sum)
initial_td.mat=initial_td.mat[,col_totals>0]

#document_term_matrix
initial_dt.mat=DocumentTermMatrix(cleaned_corpus)

#removing empty documents
row_totals=apply(initial_dt.mat,1,sum)
to.omit=which(row_totals==0)
initial_dt.mat=initial_dt.mat[row_totals>0,]

#updating the data frame to work with
if(length(to.omit)!=0){
df=df[-(to.omit),]
}

#Term document matrix based on Tf-idf weighting
initial_td.mat.tfidf<-as.matrix(weightTfIdf(initial_td.mat))

##latent semantic analysis
lsa_space=lsa(initial_td.mat.tfidf,dims=100)
doc_vec=as.matrix(lsa_space$dk)

#cosine distance between feature vectors
dist.mat.tfidf <- dist((as.matrix(doc_vec)),method = "cosine")
dist.mat.tfidf  # check distance matrix

##original points with 2-D representation
#Multi-dimensional Scaling
fit <- cmdscale(dist.mat.tfidf, eig = TRUE, k = 2)
points <- data.frame(x = fit$points[, 1], y = fit$points[, 2])
ggplot(points, aes(x = x, y = y),color=df$genre) + geom_point(data = points,
    aes(x = x, y = y,color = df$genre))

#no of actual genres in the data we are using
k=length(unique(df$genre))

#clustering(hierarchical) using tf_idf value
groups <- hclust(dist.mat.tfidf,method="ward.D")
```

```r
plot(groups, cex=0.9, hang=-1)

#visualisation of clusters
rect.hclust(groups, k)

#clustering (k means) using tf_idf values
cluster<-kmeans(dist.mat.tfidf,k)
#cluster #inspection of cluster

table(cluster$cluster)

#visualisation of clusters using first 2 principal components
col=as.factor(cluster$cluster)
ggplot(points,aes(x = x, y = y,color=col)) + geom_point(data = points, aes(x
    = x, y = y,color = col))

#elbow methhod for optimal k choice
k.max <- 15
wss <- sapply(1:k.max,
              function(k){kmeans(dist.mat.tfidf, k)$tot.withinss})
wss
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")

#lda_model with no of topics=no of genres
model_lda=LDA(initial_dt.mat,k,method = "Gibbs",control = list(iter=1000,seed
    =33))

#matrix containing P(z|d) z being topic,d being documents
theta_topic=posterior(model_lda)$topics

#prediction of category of the documents with maximum posterior probability
pred_category=apply(theta_topic,1,function(x) as.numeric(which.max(x)))

#category wise mean theta
theta_topic_by=by(theta_topic,df$genre,colMeans)
theta_means=do.call("rbind",theta_topic_by)

##LDA K means with symmetrized KL divergence and one-column out k means
model_lda=LDA(initial_dt.mat,25,method = "Gibbs",control = list(iter=1000,
    seed=33))

#matrix containing P(z|d) z being topic,d being documents
theta_topic=posterior(model_lda)$topics

obs_no=nrow(df)
#Jensen-Shannon Divergence
FUN = function(i,j) {
```

```
  p=theta_topic[i,]
  q=theta_topic[j,]
  m=0.5*(p+q)
  JS <- 0.5 * (sum(p * log(p / m)) + sum(q * log(q / m)))
  return(JS)
}

#distance measures
dist_lda_1=outer(1:obs_no,1:obs_no,Vectorize(FUN))
dist_lda_2=dist(as.matrix(theta_topic[,-1]),method="euclidean")

#doing k means
lda_kmeans_1=kmeans(dist_lda_1,k)
lda_kmeans_2=kmeans(dist_lda_2,k)

#prediction
pred_category_1=lda_kmeans_1$cluster
pred_category_2=lda_kmeans_2$cluster

##evaluation
compare(as.numeric(as.factor(df$genre)),as.numeric(pred_category),method="rand")
compare(as.numeric(as.factor(df$genre)),as.numeric(pred_category),method="nmi")
compare(as.numeric(as.factor(df$genre)),as.numeric(pred_category_1),method="rand")
compare(as.numeric(as.factor(df$genre)),as.numeric(pred_category_1),method="nmi")
compare(as.numeric(as.factor(df$genre)),as.numeric(pred_category_2),method="rand")
compare(as.numeric(as.factor(df$genre)),as.numeric(pred_category_2),method="nmi")
compare(as.numeric(as.factor(df$genre)),as.numeric(cluster$cluster),method="rand")
compare(as.numeric(as.factor(df$genre)),as.numeric(cluster$cluster),method="nmi")

##topic wise most diagnostic words
most_diagnostic=function(n,vec){
  return(as.vector(order(vec,decreasing = T)[1:n]))
}
pw_z=posterior(model_lda)$terms
for(i in 1:k){
  index=most_diagnostic(10,pw_z[i,])
  print(initial_dt.mat$dimnames$Terms[index])
}

result <- FindTopicsNumber(
  initial_dt.mat,
  topics = seq(from = 2, to = 20, by = 1),
```

```
  metrics = c("Griffiths2004"),
  method = "Gibbs",
  control = list(seed = 33),
  mc.cores = 2L,
  verbose = TRUE
)

#visualisation of clusters using first 2 principal components
col=as.factor(pred_category_1)
ggplot(points,aes(x = x, y = y,color=col)) + geom_point(data = points, aes(x
    = x, y = y,color = col))
```

9 Credit Distribution

All three of us have worked hard and made equal contributions towards the completion of this project along with the corresponding report and presentations in time. We have distributed the workload equally amongst us. In order to understand and discuss the work at hand all of us had to study the literature stated above together. We divided the simulation part in between ourselves so as to reduce the workload on each. We divided the three main topic models among the three of us, each working with one of the topic models, while also brushing up on the other models for helping each other out.