ANALYSES OF LITERARY TEXTS BY USING STATISTICAL INFERENCE METHODS

by

Mehmet Can Yavuz

B. Eng. , Electrics and Electronics Engineering, FMV Işık University, 2009

B. S. , Physics, FMV Işık University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in FBE Program for which the Thesis is Submitted

Boğaziçi University

2016

ACKNOWLEDGEMENTS

I sincerely thank my advisor Prof. Dr. Muhittin Mungan who gives the main research idea, for his patience, his persistence on working. Besides, again thanks to him and his wife, Assist. Dr. F. Meltem Gürle for many insightful conversations during the development of the ideas in this thesis, and for helpful comments on the text. For financial support, I thank the Department of Physics and its professors at the Boğaziçi University.

Finally, I sincerely thank to my thesis jury, Assoc. Prof. Dr. Erkcan Özcan, Assoc. Prof. Dr. Taylan Cemgil and Prof. Dr. Muhittin Mungan for their attendance of the pre-presentation of the thesis.

ABSTRACT

ANALYSES OF LITERARY TEXTS BY USING STATISTICAL INFERENCE METHODS

Quantitative Analysis of Literature (QAL) [1] is a rapidly growing field and it deals with numerical analyses of literary texts. Our motivation in this thesis is to classify literary pieces of authors based on their choice of words. We examine, whether given a set of literary documents we can classify them by author, whether we can classify them depending on their genre or not and whether by treating speech by individual protagonists as texts we can infer something about the protagonists themselves. These are novel questions in the field of QAL [2].

Specifically, we would like to know whether we can say something about literary texts by only looking at the occurrences of words in these. Methodologically, we use two main approaches: Probabilistic Latent Semantic Analysis (pLSA) and Latent Dirichlet Allocation (LDA). Both of these use the Expectation-Maximization (EM) method in different settings but for the same purpose. We have analyses of plays by Shakespeare by choosing all plays among the three genres, History plays, Comedies and Tragedies. Taking a collection of plays from all three genres we apply LDA to classify the texts. We find that the classification by and large follows the genre. We next focus on individual plays. Here we treat the lines uttered by each of the characters of the play as a document and apply the LDA approach to classify the documents/protagonists. Lastly, we do these classifications for four plays (*Hamlet, Othello, Macbeth* and *Richard III*).

Consequently, although the results of the classification of the characters in the plays are not always what one would have expected based on the reading of the plays, these classifications are not entirely without meaning either.

ÖZET

EDEBİ METİNLERİN İSTATİSTİKSEL ÇIKARIM METODUYLA ANALİZİ

Niceliksel Edebiyat Analizi (NEA) [1], günümüzde hızla gelişen ve edebi metinlerin sayısal analizlerini yapan bir alandır. Bu tezdeki temel motivasyonumuz kelime seçimlerine bakarak yazarların edebi eserlerini tasnif etmektir. Bir grup edebi eseri yazarlarına göre tasnif edip edemeyeceğimizi ve türlerine göre eserleri ayırt edip edemeyeceğimizi ve her bir karakterin konuşmalarını bir metin olarak ele alırsak karakterler hakkında bilgi edinip edinemeyeceğizi inceliyoruz. Bu sorular NEA alanında özgün sorulardr [2].

Spesifik olarak, sadece kelimelerin dökümanlardaki kullanım sıklıklarına bakarak edebi metinler hakkında fikir yürütebilir miyiz, bununla ilgileniyoruz. Metodolojik olarak, iki farklı yaklaşımımız var: Olasılıksal Gizil Semantik Analizi (Probabilistic Latent Semantic Analysis, pLSA) ve Gizil Diriçlet Dağıtımı (Latent Dirichlet Allocation, LDA). Her iki program da temelde Beklenti-Maksimizasyonu (Expectation-Maximization, EM) metodunu kullanır ve aynı işleve sahiptir. Shakespeare'in üç türden, Tarihi, Komedi ve Trajedi oyunlarının analizi için LDA yaklaşımını oyunların tasnifine uyguladık. Tasnifin çoğunlukla türleri takip ettiğini bulduk. Daha sonra oyunları teker teker ele aldık. Her bir oyun karakteri tarafından söylenen cümleleri bir döküman olarak ele aldık ve LDA yaklaışımını döküman/karakterleri tasnif etmede kullandık. Son olarak, Bu tasnifi dört temel oyun (*Hamlet, Othello, Macbeth* and *Richard III*) üzerinde tüm karakterlere ve küçük bir grup temel karakterlere uyguladık.

Sonuç olarak, oyunlardaki karakterlerin tasnifi birinin metni okuması sonrasındaki beklentileriyle her zaman örtüşmese de, bu tasnifler anlamdan tümüyle yoksun da değiller.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

d_i	A documentary in $D = \{d_1, ..., d_N\}$
$\mathbb{E}$	Expectation
H(X)	The entropy of X
H	Hessian of a matrix
l	Log-likelihood of data
$\mathcal{L}$	Lagrangian
P(X_i)	The probability of X_i
w_j	A word in $W = \{w_1, ..., w_M\}$
X or Y	Observed random variables
z_k	A latent class variable in $Z = \{z_1, ..., z_K\}$
α or β	Hyper-parameters
γ	Dirichlet parameter
θ	The parameters to be estimated
ρ_i or τ_k	Lagrange multipliers
$\sum$	Total sum
ϕ	Multinomial parameter
$\frac{d}{dx}$	Derivative
$diag$	Diagonal of a matrix
Dir	Dirichlet distribution
$KL(P \mid\mid Q)$	The Kullback-Leibler divergence
log	Logarithm
$Multinomial$	Multinomial distribution
$\int$	Integral

1. INTRODUCTION

1.1. Probability

Probability is a measure of how likely an event will occur. Suppose that X and Y are random variables and can take any values in $\{x_1, ..., x_N\}$ and $\{y_1, ..., y_N\}$, correspondingly. *The probability of* X is simply represented as $P(X)$ while $P(X, Y)$ is called joint probability which can be verbalized as *the probability of* X *and* Y. Similarly, $P(Y \mid X)$ is a conditional probability and can be stated as *the probability of* Y *given* X occurred. There are two fundamental properties to mention here [3],

$$\textbf{sum rule} \qquad P(X) = \sum_{Y} P(X, Y) \tag{1.1}$$

$$\textbf{product rule} \qquad P(X, Y) = P(Y \mid X)P(X) \tag{1.2}$$

correspondingly.

1.2. Concave Functions and Jensen's Inequality

A real-valued function $f(x)$ on the interval $I = [a, b]$ is said to be concave if for every x_1, $x_2 \in [a, b]$ and $0 \leq \lambda \leq 1$,

$$f(\lambda x_1 + (1 - \lambda)x_2) \geq \lambda f(x_1) + (1 - \lambda)f(x_2) \tag{1.3}$$

Intuitively, the definition of concavity states that function never falls below the straight line between the points $(x_1, f(x_1))$ to $(x_2, f(x_2))$. Alternatively, if $f''(x) \leq 0$ on $[a, b]$, then $f(x)$ is concave on $[a, b]$. For example for $\log x$,

$$\frac{d^2}{dx^2} \log(x) = -\frac{1}{x^2} \leq 0, \quad \text{on} \quad x > 0 \tag{1.4}$$

Then, the natural logarithm is concave. If $f(x)$ is a concave function, and $X \in \{x_1, ..., x_N\}$ is a random variable with probabilities $P(x_i)$, where $\sum_i P(x_i) = 1$, then,

$$f(\mathbb{E}\{X\}) \geq \mathbb{E}\{f(X)\} \tag{1.5}$$

$$f\left(\sum_{i=1}^{N} x_i P(x_i)\right) \geq \sum_{i=1}^{N} f(x_i)P(x_i) \tag{1.6}$$

The proof can be given by the method of induction as follows. For the case $N = 1$, this is trivially true. For $N = 2$,

$$f(x_1P(x_1) + x_2P(x_2)) \geq f(x_1)P(x_1) + f(x_2)P(x_2) \tag{1.7}$$

which is guaranteed by the definition of concave functions Equation 1.3. Suppose the theorem holds for $N = k - 1$ and let $P'(x_i) = \frac{P(x_i)}{1-P(x_k)}$, where $i = 1, 2, ..., k - 1$,

$$\begin{aligned}
\sum_{i=1}^{k} f(x_i)P(x_i) &= (1 - P(x_k)) \sum_{i=1}^{k-1} f(x_i)P'(x_i) + f(x_k)P(x_k) \\
&\leq (1 - P(x_k)) f\left(\sum_{i=1}^{k-1} x_i P'(x_i)\right) + f(x_k)P(x_k) \\
&\leq f\left((1 - P(x_k)) \sum_{i=1}^{k-1} x_i P'(x_i) + x_k P(x_k)\right) \\
&\leq f\left(\sum_{j=1}^{k} x_j P(x_j)\right)
\end{aligned} \tag{1.8}$$

The theorem holds by induction. Since $\log x$ function is also a concave function, it can be written by Jensen's Inequality as follows [4],

$$\log\left(\sum_{i=1}^{N} x_i P(x_i)\right) \geq \sum_{i=1}^{N} \log(x_i)P(x_i) \tag{1.9}$$

1.3. Entropy and Kullback-Leibler Divergence

The entropy of a random variable $X \in \{x_1, ..., x_N\}$ is given by [5],

$$H[X] = -\sum_{i=1}^{N} P(x_i) \log_2 P(x_i) \tag{1.10}$$

Consider an unknown distribution $P(X)$ and suppose that it will be modelled using an approximated distribution $Q(X)$. The additional amount of information as a result of using $Q(X)$, instead of $P(X)$ is given by [5],

$$\begin{aligned} KL(P \,||\, Q) &= -\int P(X) \log Q(X) dX - (-\int P(X) \log P(X) dX) \\ &= -\int P(X) \log \left(\frac{Q(X)}{P(X)} \right) dX \end{aligned} \tag{1.11}$$

which is called relative entropy or Kullback-Leibler divergence. When $Q(X) = P(X)$, Equation 1.11 is zero.

1.4. Unsupervised Learning and Bayes' Theorem

In machine learning, the problem of unsupervised learning is that of trying to find hidden structures in unlabeled data. As an example, goals of unsupervised learning are of finding clusters, dimensionality reduction, building topographic maps, finding the hidden causes or sources of the data, modeling the data density. It will be adequate to mention Bayes' theorem since it helps in the following sections to understand the probabilistic generative models proposed. A generative model is a model for randomly generated observable data, typically given in terms of some hidden parameters.

Together with the symmetry property and the product rule Equation 1.2, the following relationship can be obtained [5],

$$P(Y \mid X) = \frac{P(X \mid Y) P(Y)}{P(X)} \tag{1.12}$$

which is called Bayes' theorem. Following Bayes' theorem, a prior probability can be converted to a posterior probability by using the data set which is represented by $P(X \mid Y)$, a conditional probability. $P(X \mid Y)$ is called *likelihood function* and indicates how probable the observed data set is for different settings of the parameter X. Bayes' theorem of Equation 1.12 can be read as follows,

$$posterior \propto likelihood \cdot prior \tag{1.13}$$

where the dominator of Equation 1.12, $P(X)$, is a normalization constant. In terms of the prior distribution and the likelihood function, $P(X)$ can be expressed as follows,

$$P(X) = \sum_{Y} P(X \mid Y)P(Y) \tag{1.14}$$

which form Equation 1.12 yields,

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{\sum_{Y} P(X \mid Y)P(Y)} \tag{1.15}$$

2. PROBABILISTIC LATENT SEMANTIC ANALYSIS

2.1. Introduction

Probabilistic Latent Semantic Analysis (pLSA) [6] is an unsupervised learning algorithm to mine keywords or main concepts in large data sets. The algorithm returns mixture probabilities based on a latent class model which can be used for classifications. A mixture model is a probabilistic model for representing the occurrences of sub-populations in a population.

In this chapter, we will present a pLSA approach to the 3-fold classification of the 6 novellas by J. Conrad, H.G. Wells and Ernest Hemingway.

2.2. The Aspect Model

The main idea of the pLSA approach by Hofmann [6] is based on a statistical model which is called the aspect model. The aspect model is a latent variable model that associates data sets with unknown latent parameters. Considering that a corpus is a collection of samples of "real world" texts. Suppose that we have a corpus, then we have a given collection of text documents $D = \{d_1, ..., d_N\}$ with a vocabulary $W = \{w_1, ..., w_M\}$. The particular occurrences of a word $w_j \in W$ in a document $d_i \in D$ is associated with a latent class variable $z_k \in Z = \{z_1, ..., z_K\}$ for the aspect model. A generative model for word/document co-occurrences can be defined as follows:

- selecting a document d_i with probability $P(d_i)$.
- picking a latent class z_k with probability $P(z_k \mid d_i)$.
- generating a word w_j with probability $P(w_j \mid z_k)$.

As a consequence, an observation pair (d_i, w_j) is obtained while the latent class variable z_k disappear, in terms of a joint probability model,

$$P(d_i, w_j) = P(d_i)P(w_j \mid d_i) \tag{2.1}$$

$$P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k)P(z_k \mid d_i) \tag{2.2}$$

The aspect model is based on two independence assumptions. The first one is called *bag of words* approach which assumes that document-word pairs (d_i, w_j) are generated independently. Second one is a *conditional independence* assumption which posits that conditioned on the latent class z_k, words w_j are generated independently of the specific document identity d_i. In a formal sense,

$$P(w_j, d_i \mid z_k) = P(w_j \mid z_k)P(d_i \mid z_k) \tag{2.3}$$

Notice that Equation 2.1 can be re-parametrized by using Bayes' rule for reversing the conditional probability $P(z \mid d)$ as $P(z \mid d)P(d) = P(d \mid z)P(z)$. Then Equation 2.1 becomes,

$$P(d_i, w_j) = \sum_{k=1}^{K} P(w_j \mid z_k)P(z_k \mid d_i)P(d_i) \tag{2.4}$$

$$= \sum_{k=1}^{K} P(z_k)P(w_j \mid z_k)P(d_i \mid z_k) \tag{2.5}$$

where Equation 2.4 is called the asymmetric parametrization and Equation 2.5 is called the symmetric parametrization. Please look at the Figure 2.1 for a graphical representation.

2.2.1. Likelihood Function

Letting P be the probability of having a collection of N documents with a vocabulary size of M and let $n(d_i, w_j)$ be the co-occurrence table which is a matrix specifying

frequencies of words in a specific document,

$$P = \prod_{i=1}^{N} \prod_{j=1}^{M} P(d_i, w_j)^{n(d_i, w_j)} \tag{2.6}$$

We can calculate the log-likelihood $\mathcal{L} = \log P$ as follows,

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log P(d_i, w_j) \\ &= \sum_{i=1}^{N} n(d_i) \left[\log P(d_i) + \sum_{j=1}^{M} \frac{n(d_i, w_j)}{n(d_i)} \log \sum_{k=1}^{K} P(w_j \mid z_k) P(z_k \mid d_i) \right] \end{aligned} \tag{2.7}$$

Here $n(d_i)$ is a column matrix that specifies how many words are in a document.

2.3. Model Fitting with the EM Algorithm

The Expectation Maximization (EM) algorithm is the standard procedure for inferring latent (unobserved) parameters with observed data for a given likelihood function [7]. EM is a technique that guarantees the local optimum (not necessarily global) and consists of two alternating steps: (i) an expectation (E) step which covers the computation of posterior probabilities for latent variables by using the current estimates of the parameters, (ii) a maximization (M) step which updates the parameters according to given posterior probabilities computed in the previous E-step [5].

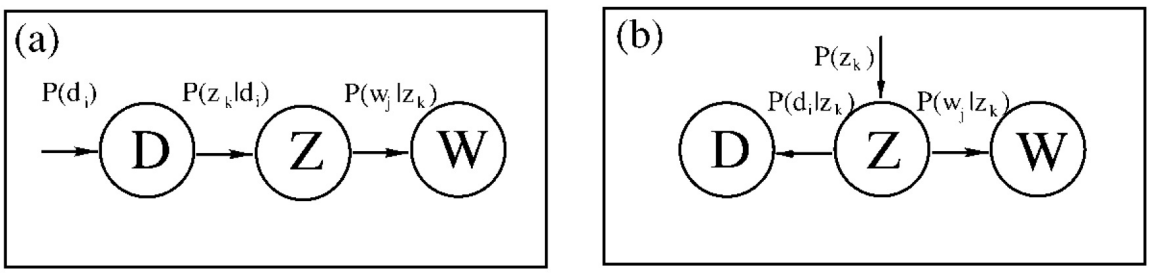

Figure 2.1: Graphical model representation of the aspect model in the asymmetric (a) and symmetric (b) parametrization [6] which are expressed in the form of Equation 2.4 and Equation 2.5, respectively. Arrows indicate conditioning.

2.3.1. Expectation-Maximization Formulation

In general, Expectation-Maximization is used to estimate likelihood of the data represented by $P(X \mid \theta)$. In our settings,

- $\theta \rightarrow$ The parameters to be estimated, *i.e.* the distributions $P(z)$, $P(z \mid d)$, $P(w \mid z)$.
- $X \rightarrow$ Observed Random Variables: frequencies of words in a specific document, captured by $n(d, w)$.
- $z \rightarrow$ Unobserved Random Variables, the latent classes $z \in Z = \{z_1, ..., z_K\}$.

In order to obtain a maximum likelihood estimator, maximizing $P(X \mid \theta)$ is the easiest way. However, adding a latent variable is generally inevitable for specific purposes, then the equation to be maximized becomes $P(X \mid \theta) = \sum_z P(X \mid z, \theta)P(z \mid \theta)$ which is difficult to maximize directly. An easier way to deal with the equations is to specify a lower bound with Jensen's Inequality.

$$\mathbb{E}[f(x)] \leq f(\mathbb{E}[f(x)]) \tag{2.8}$$

We represent a set of parameters so that $\mathcal{L}$ is maximized,

$$\begin{aligned} \log \sum_z P(X \mid z, \theta)P(z \mid \theta) &= \log \sum_z P(X \mid z, \theta)P(z \mid \theta)\frac{q(z)}{q(z)} \\ &= \log \mathbb{E}_q \left[\frac{P(X \mid z, \theta)P(z \mid \theta)}{q(z)} \right] \end{aligned} \tag{2.9}$$

Using Jensen's Inequality,

$$\log \mathbb{E}_q \left[\frac{P(X \mid z, \theta)P(z \mid \theta)}{q(z)} \right] \geq \mathbb{E}_q \left[\log \frac{P(X \mid z, \theta)P(z \mid \theta)}{q(z)} \right] \tag{2.10}$$

Basically, EM works as an iterative process to maximize the lower bound.

- Fix θ and maximize $\mathbb{E}_q \left[\log \frac{P(X|z,\theta)P(z|\theta)}{q(z)} \right]$ by varying $q(z)$.

- Take the new $q(z)$ to obtain a new θ.

$$\begin{aligned}
\mathbb{E}_q\left[\log \frac{P(X \mid z,\theta)P(z \mid \theta)}{q(z)}\right] &= \sum_z q(z) \log \frac{P(X \mid z,\theta)P(z \mid \theta)}{q(z)} \qquad (2.11)\\
&\quad \text{where Equation 2.11 follows from the Bayes' Rule}\\
&= \sum_z q(z) \log \frac{P(z \mid X,\theta)P(X \mid \theta)}{q(z)}\\
&= \sum_z q(z) \log P(X \mid \theta) + \sum_z q(z) \log \frac{P(z \mid X,\theta)}{q(z)}\\
&= \log P(X \mid \theta) - \sum_z q(z) \log \frac{q(z)}{P(z \mid X,\theta)}\\
&= \log P(X \mid \theta) - KL(q(z) \,||\, P(z \mid X,\theta) \qquad (2.12)\\
&\equiv \Lambda[q]
\end{aligned}$$

The first term in Equation 2.11 is independent of z and the lower bound calculation will remain the same. In other words, maximizing $\Lambda[q]$ with respect to q corresponds to minimizing the KL divergence. Under the $\sum_{X\in\mathcal{X}} q(x) = 1$ constraint, the *E-Step* becomes,

$$q^*(z) = P(z \mid X,\theta) \qquad (2.13)$$

- the distribution q^* that maximizes Λ is the *posterior* distribution of z given X and θ, that is, $P(z \mid X,\theta)$,
- the choice $q^*(z) = P(z \mid X,\theta)$ actually satisfies Jensen's inequality as an *equality*.

The last point can easily be shown as follows,

$$\begin{aligned}
\log P(X \mid \theta) &= \log \mathbb{E}_q\left[\frac{P(X \mid z,\theta)P(z \mid \theta)}{q^(z)}\right]\\
&\geq \mathbb{E}_q\left[\log \frac{P(X \mid z,\theta)P(z \mid \theta)}{q^(z)}\right]\\
&= \log \mathbb{E}_q\left[\frac{P(z \mid X,\theta)P(X \mid \theta)}{P(z \mid X,\theta)}\right]_{q=q*}\\
&= \mathbb{E}_q\left[\log \frac{P(z \mid X,\theta)P(X \mid \theta)}{P(z \mid X,\theta)}\right]_{q=q*}
\end{aligned}$$

$$
\begin{aligned}
&= \log \sum_z q(z) P(X \mid \theta) \\
&= \sum_z q(z) \log P(z \mid \theta) \\
&= \log P(X \mid \theta) \qquad (2.14)
\end{aligned}
$$

On the other hand the parameters θ are obtained by maximization of

$$\log P(X \mid \theta) = \left\langle \log \frac{P(z, X \mid \theta)}{q(z)} \right\rangle_{q=q^*} \qquad (2.15)$$

with respect to the variables θ for *M-Step*. From the definitions above we can now do the problem specific derivation of the Expectation and Maximization steps in detail.

2.3.2. Expectation Step

Here in E and M steps, we derive the asymmetric parametrization showed in Figure 2.1. For the E-step, probability of the distribution of z_k given d_i and w_j,

$$P(z_k \mid d_i, w_j) = \frac{P(z_k, d_i, w_j)}{P(d_i, w_j)}$$

By using Equation 2.4 for the denominator,

$$
\begin{aligned}
&= \frac{P(d_i, w_j \mid z_k) P(z_k)}{\sum_{l=1}^{K} P(w_j \mid z_l) P(z_l \mid d_i) P(d_i)} \\
&= \frac{P(w_j \mid z_k) P(d_i \mid z_k) P(z_k)}{\sum_{l=1}^{K} P(w_j \mid z_l) P(z_l \mid d_i) P(d_i)}
\end{aligned}
$$

By using Bayes' theorem for the numerator,

$$= \frac{P(w_j \mid z_k) P(d_i) P(z_k \mid d_i)}{\sum_{l=1}^{K} P(w_j \mid z_l) P(z_l \mid d_i) P(d_i)}$$

Finally, by dropping the $P(d_i)$s from the equation, $P(z_k \mid d_i, w_j)$ becomes,

$$P(z_k \mid d_i, w_j) = \frac{P(w_j \mid z_k)P(z_k \mid d_i)}{\sum_{l=1}^{K} P(w_j \mid z_l)P(z_l \mid d_i)} \tag{2.16}$$

2.3.3. Maximization Step

For the M-step, the expected complete log-likelihood $\mathbb{E}[\mathcal{L}^c]$ is to be maximized. Let us introduce hidden variables $P(z_k \mid d_i, w_j)$ to indicate which hidden topic z_k is selected to be generated w_j in d_i $\sum_k P(z_k \mid d_i, w_j) = 1$. The remaining non-trivial part then is

$$\mathcal{L} = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} P(z_k \mid d_i, w_j) \log[P(w_j \mid z_k)P(z_k \mid d_i)] \tag{2.17}$$

Considering the normalization constraints, we add the appropriate Lagrange multipliers τ_k and ρ_i to Equation 2.17,

$$\mathcal{L}^c = \mathcal{L} + \sum_{k=1}^{K} \tau_k \left(1 - \sum_{j=1}^{M} P(w_j \mid z_k)\right) + \sum_{i=1}^{N} \rho_i \left(1 - \sum_{k=1}^{K} P(z_k \mid d_i)\right) \tag{2.18}$$

Maximizing Equation 2.18 with respect to $P(w_j \mid z_k)$,

$$\begin{aligned} \frac{\partial \mathcal{L}^c}{\partial P(w_j \mid z_k)} &= \sum_{i=1}^{N} \frac{n(d_i, w_j)P(z_k \mid d_i, w_j)}{P(w_j \mid z_k)} - \tau_k = 0 \\ \sum_{i=1}^{N} n(d_i, w_j)P(z_k \mid d_i, w_j) - \tau_k P(w_j \mid z_k) &= 0 \end{aligned} \tag{2.19}$$

Summing Equation 2.19 over all possible w_j's in order to obtain τ_k,

$$\begin{aligned} \sum_{j=1}^{M} \left(\sum_{i=1}^{N} n(d_i, w_j)P(z_k \mid d_i, w_j) - \tau_k P(w_j \mid z_k)\right) &= 0 \\ \tau_k &= \sum_{j=1}^{M}\sum_{i=1}^{N} n(d_i, w_j)P(z_k \mid d_i, w_j) \end{aligned}$$

Substituting τ_k into Equation 2.19, we obtain

$$P(w_j \mid z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j) P(z_k \mid d_i, w_j)}{\sum_{m=1}^{M} \sum_{i=1}^{N} n(d_i, w_m) P(z_k \mid d_i, w_m)} \tag{2.20}$$

Maximizing Equation 2.18 with respect to the probability mass function $P(z_k \mid d_i)$,

$$\begin{aligned} \frac{\partial \mathcal{L}^c}{\partial P(z_k \mid d_i)} &= \sum_{j=1}^{M} \frac{n(d_i, w_j) P(z_k \mid d_i, w_j)}{P(z_k \mid d_i)} - \rho_i = 0 \\ \sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j) - \rho_i P(z_k \mid d_i) &= 0 \end{aligned} \tag{2.21}$$

In order to reach ρ, summing Equation 2.21 over all possible z_k's,

$$\begin{aligned} \sum_{k=1}^{K} \left(\sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j) - \rho_i P(z_k \mid d_i) \right) &= 0 \\ \rho_i &= n(d_i) \end{aligned}$$

Taking ρ back into Equation 2.21,

$$P(z_k \mid d_i) = \frac{\sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j)}{n(d_i)} \tag{2.22}$$

Finally, Equation 2.20 and Equation 2.22 are the re-estimation parameters of the M-step.

2.4. Implementation

In the following Algorithm shown in Figure 2.2, first we randomly initialize the conditional probability $p(z \mid d, w)$ and then iterate the Expectation-Maximization algorithm until convergence. The following pseudo-code performs first the Maximization step in order not to initialize $p(w \mid z)$ and $p(z \mid d)$, again. These can already be determined by the randomly initialized $p(z \mid d, w)$ and $n(d, w)$ matrices. In our actual Matlab implementation, for initialization of $p(z \mid d, w)$ the random matrix Matlab function "rand" is used, as shown in line 7 of Figure C.3.

We next examine the pseudo-code line by line. Line 3 of the code corresponds to Equation 2.20 and it is normalized in Line 4 according to the equation. The code in Line 5-6 is in accordance with Equation 2.22 and in Line 7-8 with Equation 2.16, respectively. In the actual, we iterate until the EM cycle converges, line 20 of Figure C.2. However in the below pseudo-code, there is an iteration number to reach for each cycle and it returns the variables $p(z \mid d, w)$, $p(w \mid z)$, $p(z \mid d)$. In addition to these variables, in the actual implementation the likelihood value history is also recorded.

The "main" part of the program loops over a fixed number of initial conditions to determine the classification which will have the likelihood maxima, as shown in line 30-45 of Figure C.2. If a word w_j occurs only in one document among documents, such that,

$$n(d_i, w_j) \geq 1 \; and \; n(d_{-i}, w_j) = 0 \tag{2.23}$$

(d_{-i} indicates documents that are not d_i.), the matrix elements satisfying Equation 2.23 are erased because it is related to only one document. We checked that including these did not change the results of the classification. Apparently, they are statistically irrelevant.

Input: word-document co-occurrence matrix $n(d, w)$ and topic number K

Output: conditional probabilities such as $p(z \mid d, w)$, $p(w \mid z)$, $p(z \mid d)$

1 *randomly initialize* $p(z \mid d, w)$;

2 **for** $t \leftarrow 1$ ***to*** *iteration number* **do**

3 $\quad p(w \mid z) \leftarrow \sum_d n(d, w) * p(z \mid d, w)$;

4 $\quad$ *normalize* $p(w \mid z)$;

5 $\quad p(z \mid d) \leftarrow \sum_w n(d, w) * p(z \mid d, w)$;

6 $\quad$ *normalize* $p(z \mid d)$;

7 $\quad p(z \mid d, w) \leftarrow p(w \mid z) * p(z \mid d)$;

8 $\quad$ *normalize* $p(z \mid d, w)$;

9 **return** $p(w \mid z)$, $p(z \mid d)$, $p(z \mid d, w)$;

Figure 2.2: The pLSA Algorithm

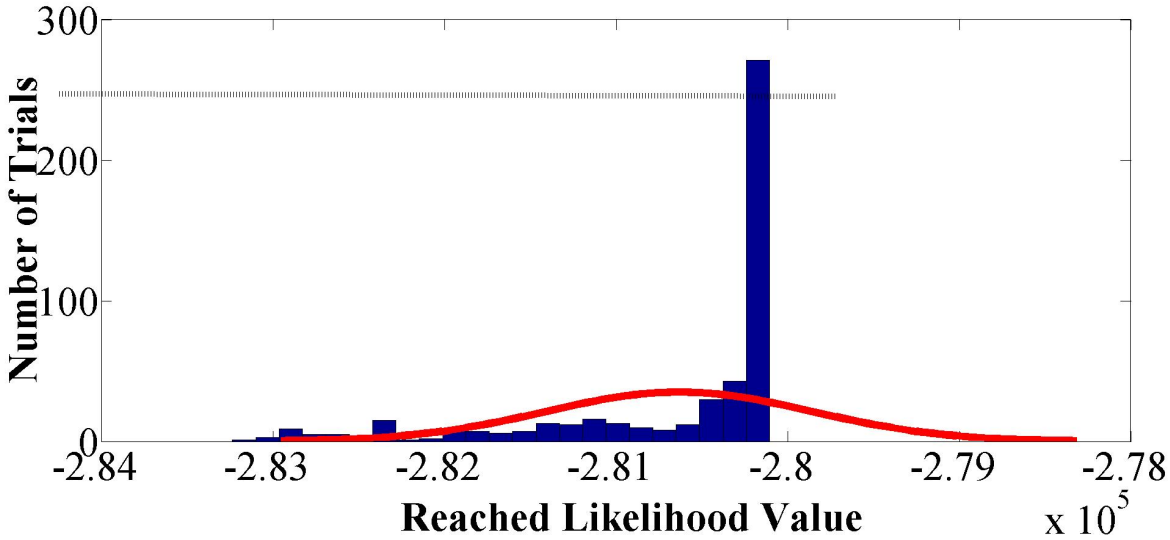

Figure 2.3: A histogram of reached likelihood values of the corpus six novellas by three authors with pLSA. The red curve is a Gaussian fit to the data. The dashed grey line corresponds to half of the trials (250).

2.5. Numerical Experiments

Our motivation here is to distinguish six pieces of three authors depending on their basic characteristics. English speaking authors are chosen in order not to fall into the translation mistakes. 500 trials were made using the MATLAB implementation described before. Elapsed time was 55712 seconds.[1]

A histogram of likelihood values obtained with a distribution fit is given in Figure 2.3. According to the graph, more than half of the trials falls into the highest region as shown with a dot line.

As it was discussed in the previous "Implementation" section, we choose the highest likelihood value's parameters $p(w \mid z)$, $p(z \mid d)$, $p(z \mid d, w)$. The evolution curve of the likelihood iterations of the EM loops is shown in Figure 2.4. Each line shows a run. There are 500 of them. Maximum reached likelihood curve is indicated with a red line.

Firstly, $p(z \mid d)$ of the highest achieved likelihood value is listed in Table 2.1 and it is obvious that rather crisp results are achieved. According to the table, pLSA program differentiates the authors and the corresponding entries are shown with a grey background. We named the z values according to their authors.

Table 2.1: Corresponding $p(z \mid d)$ to the highest achieved likelihood value of six novellas by three authors with pLSA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Authors (z)	Hemingway	Wells	Conrad
The Island of Doctor Moreau **1896**	0.00079	0.99834	0.00086
The Time Machine **1898**	0.00001	0.99950	0.00049
Heart of Darkness **1899**	0.00000	0.00000	1.00000
The Shadow Line **1915**	0.00000	0.00000	1.00000
The Snows of Kilimanjaro **1936**	0.99925	0.00001	0.00074
The Old Man and the Sea **1952**	0.99905	0.00009	0.00086

[1] in a 64Bit Matlab, 1.10GHz, 2GB-Ram Computer and this configuration is also used for all other experiments.

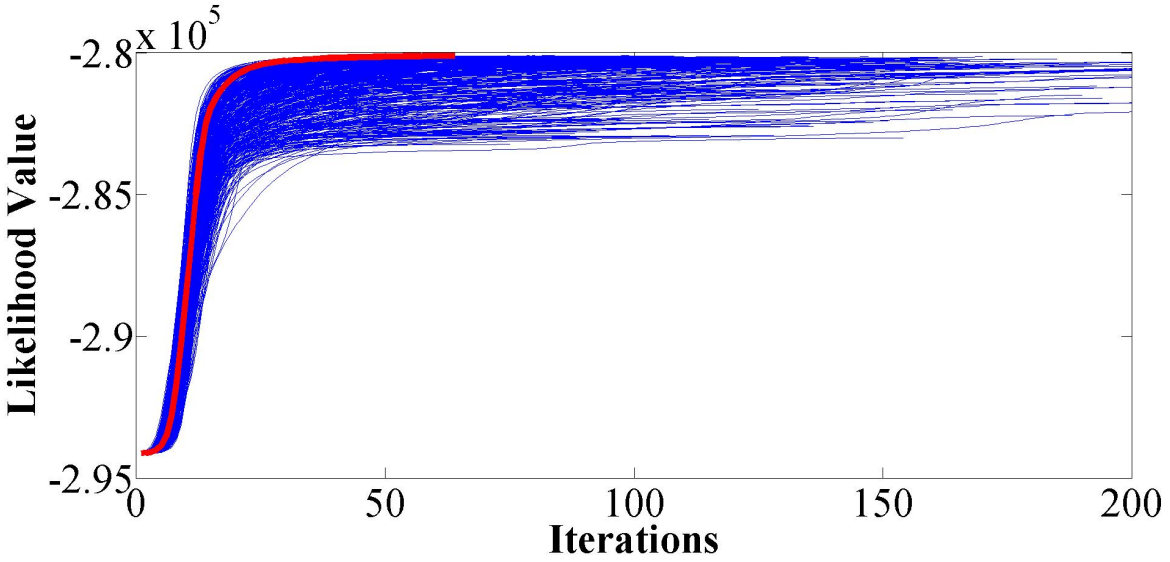

Figure 2.4: The likelihood evolution curves of the corpus six novellas by three authors with pLSA. Each line indicates the function $\mathcal{L}$ of Equation 2.17. There are 500 trials. Maximum reached likelihood curve is indicated with a red line.

Secondly, let us look at the top ten words in each topic z_k such that satisfying Equation 2.24. Table 2.2 shows the number of words satisfying Equation 2.24.

$$p(z_k \mid w_j) = 1 \; or \; equaivalently, \; p(z_{-k} \mid w_j) = 0 \tag{2.24}$$

which in practice we take to mean machine precision.

Table 2.2: Number of words satisfying Equation 2.24 as a result of pLSA calculations on six novellas by three authors.

Hemingway	Wells	Conrad
36	383	518

Since they are categorized into three, Table 2.3 gives us an idea about the main concepts in our corpus of six novellas. The words were chosen to machine precision from the $p(z \mid w)$ table satisfying Equation 2.24.

Table 2.3: Three aspects. Top ten words according to a three fold classification of six novellas by three authors with pLSA.

Hemingway	Wells	Conrad
'american'	'pursuit'	'helm'
'automatically'	'smaller'	'moments'
'belly'	'nerve'	'official'
'bought'	'pistol'	'indignant'
'brothers'	'pathway'	'indignation'
'bumping'	'bay'	'muttering'
'calloused'	'whips'	'bedplace'
'christ'	'habits'	'test'
'clubbed'	'bowed'	'illness'
'comfortably'	'scarcely'	'episode'

Let us look at the relationship between $n(d, w)$ and $p(z \mid w)$ of those picked top ten words in Table 2.6. In the table, the variables are : z_1 is 'Wells'; z_2 is 'Conrad'; z_3 is 'Hemingway', d_1 is 'Conrad-Heart of Darkness'; d_2 is 'Conrad-The Shadow Line'; d_3 is 'Hemingway-Old Man And The Sea'; d_4 is 'Hemingway-Snows of Klimanjero';, d_5 is 'Wells-The Island of Doctor Moreau'; d_6 is 'Wells-Time Machine'.

We can observe from Table 2.6 at the end of the chapter that if we group d_1 and d_2 as z_1, d_3 and d_4 as z_2, d_5 and d_6 as z_3, the new $n(z_k, w_j)$ is proportional to the $p(z \mid w)$. This is valid for all $p(z \mid w)$ and $n(d_i, w_j)$ tables. This is not a surprising fact since Equation 2.20 and Equation 2.22 relate $n(d, w)$ matrix with $p(z \mid w)$ and $p(z \mid d)$ matrices. Our program gives us results of the maximum information retrieval we can get, which means that it chooses the grouping parameters of d_is and w_js for us.

2.5.1. Results of Minimum Likelihood Case

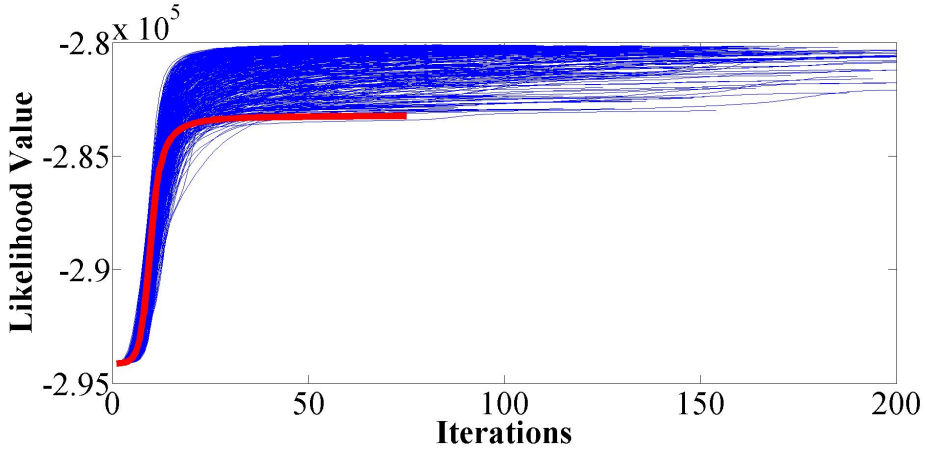

Figure 2.5: The worst case. The likelihood evolution curves of the corpus six novellas by three authors with pLSA. Each line indicates the function $\mathcal{L}$ of Equation 2.17. There are 500 trials. Minimum reached likelihood curve is indicated with a red line.

Let us look at the parameters of the lowest likelihood curve reached that can give an idea about how program works. Figure 2.5 indicates the lowest likelihood curve with a red line. Additionally, Table 2.4 shows the grouping parameter $p(z|d)$. However, there is no meaningful result.

Table 2.4: Corresponding $p(z \mid d)$ to the lowest achieved likelihood value of six novellas by three authors with pLSA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Authors (z)	z_1	z_2	z_3
The Island of Doctor Moreau **1896**	0.00000	0.73649	0.26350
The Time Machine **1898**	0.00000	0.00000	1.00000
Heart of Darkness **1899**	1.00000	0.00000	0.00000
The Shadow Line **1915**	0.00000	1.00000	0.00000
The Snows of Kilimanjaro **1936**	0.62730	0.37244	0.00026
The Old Man and the Sea **1952**	0.54786	0.45208	0.00006

2.5.2. Results of an Intermediate Likelihood Case

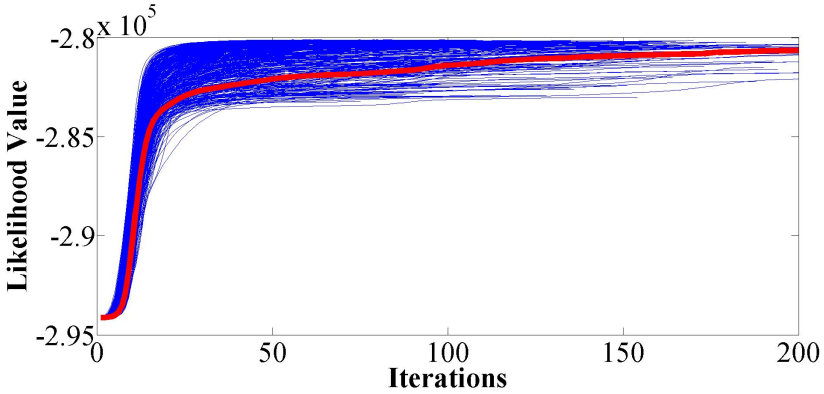

Figure 2.6: The likelihood evolution curves of the corpus six novellas by three authors with pLSA. Each line indicates the function $\mathcal{L}$ of Equation 2.17. There are 500 trials. An intermediate level likelihood curve is indicated with a red line.

Mean value of the likelihoods reached is calculated and corresponding mid-point likelihood curve is indicated in Figure 2.6 with a red line. In addition to that, Table 2.5 shows the grouping parameter $p(z|d)$. As the end likelihood value increases, more meaningful results we get. Grouping is correct, however there are deviations. Top words chosen by program are in accordance with the top words of highest likelihood achieved.

Table 2.5: Corresponding $p(z \mid d)$ to an intermediate level likelihood value of six novellas by three authors with pLSA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Authors (z)	z_1	z_2	z_3
The Island of Doctor Moreau **1896**	0.28398	0.71597	0.00005
The Time Machine **1898**	0.00000	1.00000	0.00000
Heart of Darkness **1899**	0.00000	0.00000	1.00000
The Shadow Line **1915**	0.00000	0.00000	1.00000
The Snows of Kilimanjaro **1936**	1.00000	0.00000	0.00000
The Old Man and the Sea **1952**	0.88808	0.11191	0.00001

2.6. Results and Discussion

We applied the algorithm on six texts of three different authors to check their authorship, and to get an idea about our pLSA implementation's speed and reliability. Using the corpus properties which are encoded in the $n(d, w)$ matrix, the pLSA algorithm is able to distinguish texts from the different authors.

"Heart of Darkness" and "The Shadow Line" are the novellas which have the most words satisfying Equation 2.24 according to the Table 2.2 and they unsurprisingly are classified with probability one. Why were not we surprised? Because it is natural that there are words only each author uses. From text to text some words remains and these remaining ones differentiate an author from the another author.

Another fact about our analyses is – as expected – that as the likelihood value increases more accurate classifications are obtained. At intermediate values of the achieved likelihood, we obtain classifications agreeing with that obtained from maximum likelihood case. However, the classification is less crisp. There are small deviations in "The Island of Doctor Moreau" and "The Old Man and the Sea". On the other hand, at the worst scenario which is classifications with a very small likelihood value lead to totally different grouping results.

As a final remark, applied to texts of different authors the pLSA approach seems to work well, however our implementation is slow, taking 15,5 hours. In the following chapter we turn to a faster implementation of a slightly modified probabilistic model, Latent Dirichlet Allocation (LDA).

Table 2.6: A comparison of $n(d, w)$ matrix and $p(z \mid w)$ matrix of a three fold classification of six novellas by three authors with pLSA. Picked words are from the Table 2.3

Words	$p(z_k \mid w_j)$			$n(w_j, d_i)$					
w_j	z_1	z_2	z_3	d_1	d_2	d_3	d_4	d_5	d_6
'american'	0.00000	0.00000	1.00000	0	0	1	1	0	0
'automatically'	0.00000	0.00000	1.00000	0	0	1	1	0	0
'belly'	0.00000	0.00000	1.00000	0	0	5	1	0	0
'bought'	0.00000	0.00000	1.00000	0	0	3	3	0	0
'brothers'	0.00000	0.00000	1.00000	0	0	5	1	0	0
'bumping'	0.00000	0.00000	1.00000	0	0	1	1	0	0
'calloused'	0.00000	0.00000	1.00000	0	0	1	1	0	0
'christ'	0.00000	0.00000	1.00000	0	0	2	3	0	0
'clubbed'	0.00000	0.00000	1.00000	0	0	4	1	0	0
'comfortably'	0.00000	0.00000	1.00000	0	0	2	1	0	0
'pursuit'	1.00000	0.00000	0.00000	0	0	0	0	9	1
'smaller'	1.00000	0.00000	0.00000	0	0	0	0	8	1
'nerve'	1.00000	0.00000	0.00000	0	0	0	0	6	1
'pistol'	1.00000	0.00000	0.00000	0	0	0	0	6	1
'pathway'	1.00000	0.00000	0.00000	0	0	0	0	6	1
'bay'	1.00000	0.00000	0.00000	0	0	0	0	6	1
'whips'	1.00000	0.00000	0.00000	0	0	0	0	5	1
'habits'	1.00000	0.00000	0.00000	0	0	0	0	5	1
'bowed'	1.00000	0.00000	0.00000	0	0	0	0	5	1
'scarcely'	1.00000	0.00000	0.00000	0	0	0	0	19	4
'helm'	0.00000	1.00000	0.00000	1	13	0	0	0	0
'moments'	0.00000	1.00000	0.00000	2	14	0	0	0	0
'official'	0.00000	1.00000	0.00000	2	11	0	0	0	0
'indignant'	0.00000	1.00000	0.00000	1	5	0	0	0	0
'indignation'	0.00000	1.00000	0.00000	1	5	0	0	0	0
'muttering'	0.00000	1.00000	0.00000	1	5	0	0	0	0
'bedplace'	0.00000	1.00000	0.00000	1	4	0	0	0	0
'test'	0.00000	1.00000	0.00000	1	4	0	0	0	0
'illness'	0.00000	1.00000	0.00000	1	4	0	0	0	0
'episode'	0.00000	1.00000	0.00000	1	4	0	0	0	0

3. LATENT DIRICHLET ALLOCATION

3.1. Introduction

Latent Dirichlet Allocation(LDA) is a generative probabilistic model which primarily aims to model the membership of text corpora while keeping the essential statistically relevant in relationships. In LDA, which is a three level hierarchical Bayesian model, each item of a collection is thought of as a finite mixture over an underlying set of topics and each topic as an infinite mixture over an underlying set of topic probabilities [8].

3.2. Notation and Terminology

Latent Dirichlet Allocation (LDA) as a probabilistic generative model uses notions such as "topics" for assumed hidden variable of the text collections, besides three observed entities such as "words", "documents" and "corpora". In a more formal sense,

- A *word* is the basic unit discrete data from the dictionary indexed by $\{1, \ldots, V\}$. Words are represented by V-dimensional unit-basis vectors which only a single component of the vector is one, all the rest is zero.
- A document is a sequence of words, namely N words, denoted by $\mathbf{w} = (w_1, w_2, ..., w_N)$, where w_n is the nth word in the sequence.
- A *corpus* is a collection of documents, namely M documents, denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$.
- A *topic* is k-dimensional unobserved grouping variable supposed to be derived from the observed variables, which indicates the membership of words and denoted by $Z = \{z_1, ..., z_k\}$.

3.3. Latent Dirichlet Allocation

The basic assumption of LDA is to interpret each topic as a distribution over the words, so to say each document is a mixture of a number of topics and that each word is attributable to one of the document topics. To clarify the generative process behind this, for each document $\mathbf{w}$ in a corpus D the following procedure is schematized [8]:

1. Choose $\theta \sim Dir(\alpha)$
2. For each of the N words w_n:
   (a) Choose a topic $z_n \sim Multinomial(\theta)$.
   (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

The word probabilities are parameterized by a $k \times V$ matrix $\beta_{ij} = p(w^j \mid z^i)$, which is the collection of parameters to be estimated. θ is a k-dimensional Dirichlet random variable which lies in the $(k-1)$-simplex and has the following probability density,

$$p(\theta \mid \alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1} \tag{3.1}$$

where the parameter α is a k-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function.

As has been mentioned, LDA assumes that words are generated from topics, and the topics are exchangeable within a document which is known as the ex-changeability assumption for words in a document. de Finetti's representation theorem [9] establishes that if the joint distribution of a set of random variables is invariant under permutations, these random variables could be considered as independent and identically distributed conditioned on a latent parameter, which is drawn from a certain distribution. In LDA, the random variables in question are the topics corresponding to the words and the latent parameter is θ for the discrete distribution, which is drawn from the Dirichlet distribution; Dirichlet(α). For the given parameters α and β, the

joint distribution is given by:

$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta) \tag{3.2}$$

where $p(z_n \mid \theta)$ is simply θ_i for the unique i such that $z_n^i = 1$. Integrating over θ and summing over z, the marginal distribution of a document:

$$p(\mathbf{w} \mid \alpha, \beta) = \int p(\theta \mid \alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n \mid \theta) p(w_n \mid z_n, \beta) \right) d\theta \tag{3.3}$$

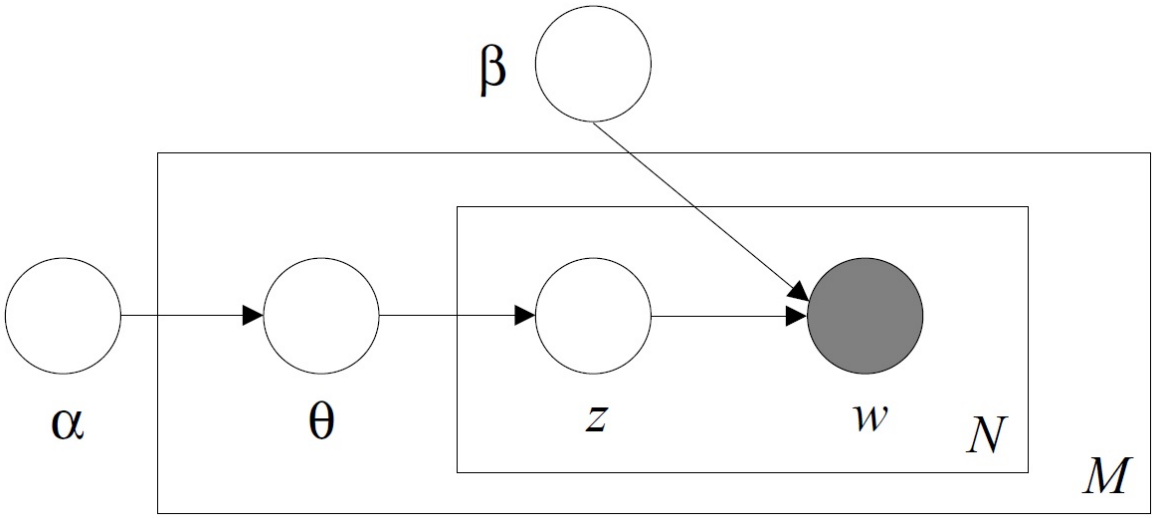

Figure 3.1: Graphical model representation of LDA. The boxes are plates representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

By multiplying marginal probabilities of single documents, we obtain the probability of a corpus:

$$p(\mathbf{D} \mid \alpha, \beta) = \prod_{d=1}^{N} \int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{d_n}} p(z_{d_n} \mid \theta_d) p(w_{d_n} \mid z_{d_n}, \beta) \right) d\theta_d \tag{3.4}$$

3.4. Inference

The basic inferential problem to use LDA is to solve posterior distribution of the hidden variables for a given document:

$$p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)}{p(\mathbf{w} \mid \alpha, \beta)} \tag{3.5}$$

However, the normalization part of Equation 3.5 is intractable due to the coupling between θ and β under the summation over latent topic parameters which becomes clear when we rewrite Equation 3.3 in terms of model parameters:

$$p(\mathbf{w} \mid \alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{k} \theta^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j} \right) d\theta \tag{3.6}$$

Since the exact inference is not possible, a variational approximation algorithm can be used to solve above equation.

3.4.1. Variational Inference

Aim of the variational inference algorithm is to reach a lower bound on the log likelihood function with the aid of Jensen's inequality. Namely, via an optimization procedure a tightest possible lower bound is chosen from a family of lower bounds which are functions of variational parameters.

The coupling between β and θ which does not allow to exact inference is a consequence of the edges between $\theta, \mathbf{z}, \mathbf{w}$. By eliminating these obstacles and reorganizing the model with free variational parameters, the following family of distributions on the latent variables are imposed:

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = q(\theta \mid \gamma) \prod_{n=1}^{N} q(z_n \mid \phi_n) \tag{3.7}$$

where the Dirichlet parameter γ and the multinomial parameters $(\phi_1, ..., \phi_N)$ are the free variational parameters. Having a simplified family of probability distributions allows for an optimization procedure searching for the values of the variational parameters γ and ϕ.

Bounding the log likelihood of a document using Jensen's inequality,

$$\begin{aligned}
\log p(\mathbf{w} \mid \alpha, \beta) &= \sum_{d=1}^{D} \log \int \sum_{\mathbf{z}} p(\theta, \mathbf{z}, \mathbf{w_d} \mid \alpha, \beta) d\theta \\
&= \sum_{d=1}^{D} \log \int \sum_{\mathbf{z}} \frac{p(\theta, \mathbf{z}, \mathbf{w_d} \mid \alpha, \beta) q(\theta, \mathbf{z})}{q(\theta, \mathbf{z})} d\theta \\
&\geq \sum_{d=1}^{D} \int q(\theta) q(z) \\
&\quad \left[p(\theta \mid \alpha) + \sum_{n} \log p(z_n \mid \theta) + \sum_{n} \log p(w_n \mid z_n, \beta) \right] d\theta \\
&\quad - \int \sum_{z} q(\theta) q(z) \log q(\theta) q(z) d\theta \\
&= \sum_{d=1}^{D} E_q[\log p(\theta, \mathbf{z}, \mathbf{w_d} \mid \alpha, \beta)] - E_q[\log q(\theta, \mathbf{z} \mid \gamma, \phi)] \qquad (3.8)
\end{aligned}$$

Jensen's inequality yields a lower bound on the log likelihood for any variational distribution $q(\theta, \mathbf{z} \mid \gamma, \phi)$.

In Equation 3.8, the difference equation corresponds to KL divergence between the variational posterior probability and the correct posterior probability. Rewriting $E_q[\log q(\theta, \mathbf{z} \mid \gamma, \beta)$ as $L(\gamma, \phi \mid \alpha, \beta)$,

$$\log p(\mathbf{w} \mid \alpha, \beta) = L(\gamma, \phi \mid \alpha, \beta) + D(q(\theta, \mathbf{z} \mid \gamma, \phi) \mid\mid p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)) \qquad (3.9)$$

That is to say, maximizing the lower bound $L(\gamma, \phi \mid \alpha, \beta)$ with respect to γ and ϕ corresponds to minimizing the KL divergence.

$$(\gamma^*, \phi^*) = arg \min_{\gamma, \phi} D(q(\theta, \mathbf{z} \mid \gamma, \phi) \mid\mid p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)) \qquad (3.10)$$

So the optimization problem becomes maximization of the lower bound.

$$\begin{aligned} L(\gamma,\phi;\alpha,\beta) &= E_q[\log p(\theta \mid \alpha)] + E_q[\log p(\mathbf{z} \mid \theta)] + E_q[\log p(\mathbf{w} \mid \mathbf{z},\beta)] \\ &\quad -E_q[\log q(\theta)] - E_q[\log q(\mathbf{z})] \end{aligned} \tag{3.11}$$

First term of the above expression can be expanded as,

$$\begin{aligned} E_q\Bigg[\log p(\theta \mid \alpha)] &= E_q[\log \exp\left(\left(\frac{\Gamma(\sum_{i=1}^{k}\alpha)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\right)\theta_1^{\alpha_1-1}...\theta_k^{\alpha_k-1}\right)\Bigg] \\ &= \sum_{i=1}^{k}(\alpha_i-1)E_q[\log(\theta_i)] + \log(\sum_{i=1}^{k}\Gamma\alpha_i) - \sum_{i=1}^{k}\log\Gamma(\alpha_i) \end{aligned}$$

where the general fact that the derivative of the log normalization factor with respect to natural parameter (here it is γ) is equal to expectation of sufficient statistics ($T(x)$). $E_q[\log\theta_i] = \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}\gamma_j)$

$$= \sum_{i=1}^{k}(\alpha_i-1)\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}\gamma_j)\right) + \log\left(\sum_{i=1}^{k}\Gamma(\alpha_i)\right) - \sum_{i=1}^{k}\log\Gamma(\alpha_i) \tag{3.12}$$

The second term can be expanded as,

$$\begin{aligned} E_q[\log p(\mathbf{z} \mid \theta)] &= \int\sum_{\mathbf{z}} q(\theta \mid \gamma)q(\mathbf{z} \mid \phi)\log p(\mathbf{z} \mid \theta)d\theta \\ &= \sum_{\mathbf{z}} q(\mathbf{z} \mid \phi)\int q(\theta \mid \gamma)\log p(\mathbf{z} \mid \theta)d\theta, \\ &= \sum_{n=1}^{N}\sum_{i=1}^{k}\phi_{ni}E_q[log\theta_i \mid \gamma] \\ &= \sum_{n=1}^{N}\sum_{i=1}^{k}\phi_{ni}\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}\gamma_j)\right) \end{aligned} \tag{3.13}$$

The third term can be expanded as,

$$\begin{aligned} E_q[\log p(\mathbf{w} \mid \mathbf{z}, \beta)] &= \int \sum_{\mathbf{z}} q(\theta \mid \gamma) q(\mathbf{z} \mid \phi) \log p(\mathbf{w} \mid \mathbf{z}, \beta) d\theta \\ &= \int q(\theta \mid \gamma) \sum_{\mathbf{z}} q(\mathbf{z} \mid \phi) \log(\beta^{\mathbf{w}}) d\theta \\ &= \sum_{n=1}^{N} \sum_{i=1}^{k} \sum_{j=1}^{V} \phi_{ni} w_n^j \log \beta_{ij} \end{aligned} \tag{3.14}$$

the fourth term as,

$$\begin{aligned} E_q[\log q(\theta)] = \sum_{i=1}^{k} (\gamma_i - 1) \left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \right) \\ + \log \left(\sum_{i=1}^{k} \Gamma(\gamma_i) \right) - \sum_{i=1}^{k} \log \Gamma(\gamma_i) \end{aligned} \tag{3.15}$$

and finally the last term as,

$$E_q[\log q(\mathbf{z})] = \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni} \log \phi_{ni} \tag{3.16}$$

Summing all terms,

$$\begin{aligned} L(\gamma, \phi; \alpha, \beta) &= \sum_{i=1}^{k} (\alpha_i - 1) \left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \right) + \log \left(\sum_{i=1}^{k} \Gamma(\alpha_i) \right) \\ &\quad - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni} \left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \right) \\ &\quad + \sum_{n=1}^{N} \sum_{i=1}^{k} \sum_{j=1}^{V} \phi_{ni} w_n^j \log \beta_{ij} \\ &\quad - \sum_{i=1}^{k} (\gamma_i - 1) \left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \right) + \log \left(\sum_{i=1}^{k} \Gamma(\gamma_i) \right) \\ &\quad - \sum_{i=1}^{k} \log \Gamma(\gamma_i) - \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni} \log \phi_{ni} \end{aligned} \tag{3.17}$$

In particular, given a corpus of documents $D = \mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_M}$, aim is to find parameters α and β that maximize the log likelihood of the data:

$$l(\alpha, \beta) = \sum_{d=1}^{M} \log p(\mathbf{w_d} \mid \alpha, \beta) \tag{3.18}$$

We obtain an approximate empirical Bayes estimates for the LDA model via an alternating variational EM procedure that maximizes a lower bound with respect to variational parameters γ and ϕ, and then, for fixed values of the variational parameters, maximizes the lower bound with respect to the model parameters α and β. Respectively,

1. (E-step) For each document, find the optimizing values of the variational parameters $\{\gamma_d^*, \phi_d^* \, d \in D\}$
2. (M-step) Maximize the resulting lower bound on the log likelihood with respect to the model parameters α and β.

3.4.2. Variational Multinomial (E-step)

The aim in this section is to maximize with respect to ϕ_{ni} with the constraint of $\sum_{i=1}^{k} \phi_{ni} = 1$ and where $w_n^j \log \beta_{ij} = \log \beta_{i\nu}$.
For simplicity dropping out the non-ϕ_{ni} parameters,

$$L_{[\phi_{ni}]} = \phi_{ni} \left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \right) + \phi_{ni} \log \beta_{i\nu} - \phi_{ni} \log \phi_{ni} + \lambda_n \left(\sum_{i=1}^{k} \phi_{ni} - 1 \right) \tag{3.19}$$

Taking derivative with respect to ϕ_{ni} and equate to zero,

$$\frac{\partial L}{\partial \phi_{ni}} = \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) + \log \beta_{i\nu} - \log \phi_{ni} - 1 + \lambda = 0 \tag{3.20}$$

Maximazing value of the variational parameter ϕ_{ni},

$$\phi_{ni} \propto \beta_{i\nu} \exp \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) \tag{3.21}$$

3.4.3. Variational Dirichlet(E-step)

The aim in this section is to maximize with respect to γ_i. For simplicity dropping out the non-γ_i parameters,

$$\begin{aligned} L_{[\gamma]} &= \sum_{i=1}^{k}(\alpha_i - 1)\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}\gamma_j)\right) \\ &\quad + \sum_{n=1}^{N}\phi_{ni}\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}\gamma_j)\right) - \sum_{i=1}^{k}(\gamma_i - 1)\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}\gamma_j)\right) \\ &\quad + \log\left(\sum_{i=1}^{k}\Gamma(\gamma_i)\right) - \sum_{i=1}^{k}\log\Gamma(\gamma_i) \end{aligned} \tag{3.22}$$

Taking the derivative with respect to γ_i and equating to zero,

$$\frac{\partial L}{\partial \gamma_i} = \Psi'(\gamma_i)(\alpha_i + \sum_{n=1}^{N}\phi_{ni} - \gamma_i) - \Psi'(\sum_{j=1}^{k}\gamma_j)\sum_{j=1}^{k}(\alpha_j + \sum_{n=1}^{N}\phi_{nj} - \gamma_j) = 0 \tag{3.23}$$

So the maximizing value of the variational parameter γ_i

$$\gamma_i = \alpha_i + \sum_{n=1}^{N}\phi_{ni} \tag{3.24}$$

3.5. Parameter Estimation

The aim in this section is to obtain empirical Bayes' estimates of the model parameters α and β by using the variational lower bound with the fixed variational parameters ϕ and γ found by variational inference.

From the ex-changeability assumption for the documents, the overall log likelihood of a corpus $D = \{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_M}\}$ is the sum of log likelihoods for individual documents; moreover, the overall variational lower bound is the sum of the individual variational bounds. In this section, L is used for the total variational bound.

3.5.1. Conditional Multinomial (M-step)

For simplicity dropping out the non-β parameters and adding constraint $\sum_{j=1}^{V} \beta_{ij} = 1$,

$$L_{[\beta]} = \sum_{d=1}^{M} \sum_{n=1}^{N_d} \sum_{i=1}^{k} \sum_{j=1}^{V} \phi_{dni} w_{dn}^{j} \log \beta_{ij} + \sum_{i=1}^{k} \lambda_i \left(\sum_{j=1}^{V} \beta_{ij} - 1 \right) \tag{3.25}$$

Taking the derivative with respect to β_{ij} and equating it to zero:

$$\beta_{ij} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni} w_{dn}^{j} \tag{3.26}$$

3.5.2. Dirichlet (M-step)

For simplicity dropping out the non-α parameters,

$$\begin{aligned} L_{[\alpha]} = \sum_{d=1}^{M} \Big\{ & \log \Gamma(\sum_{j=1}^{k} \alpha_j) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) \\ & + \sum_{i=1}^{k} (\alpha_i - 1)(\Psi(\gamma_{di}) - \Psi(\sum_{j=1}^{k} \gamma_{dj})) \Big\} \end{aligned} \tag{3.27}$$

Taking the derivative with respect to α_i,

$$g(\alpha) = \frac{\partial L}{\partial \alpha_i} = M(\Psi(\sum_{j=1}^{k} \alpha_j) - \Psi(\alpha_i)) + \sum_{d=1}^{M} (\Psi(\gamma_{di}) - \Psi(\sum_{j=1}^{k} \gamma_{dj})) \tag{3.28}$$

and the Hessian is,

$$H(\alpha) = \frac{\partial^2 L}{\partial \alpha_i \partial \alpha_j} = \delta(i,j) M \Psi'(\alpha_i) - M \Psi'(\sum_{j=1}^{k} \alpha_j) \tag{3.29}$$

The Newton-Raphson optimization technique finds a stationary point of a function by iterating:

$$\alpha_{new} = \alpha_{old} - H(\alpha_{old})^{-1} g(\alpha_{old}) \tag{3.30}$$

The Hessian matrix is in the form:

$$H = diag(\chi) + \mathbf{1}\lambda\mathbf{1}^T \tag{3.31}$$

then $\lambda = \Psi'(\sum_{j=1}^{k} \alpha_j)$ and the inverse of the Hessian,

$$H^{-1} = diag(\chi)^{-1} - \frac{diag(\chi)^{-1}\mathbf{1}\mathbf{1}^T diag(\chi)^{-1}}{\lambda^{-1} + \sum_{j=1}^{k} \chi_j^{-1}} \tag{3.32}$$

multiplying by the gradient, for the ith component,

$$(H^{-1}g)_i = \frac{g_i - c}{\chi_i}, \;\; where \;\; c = \frac{\sum_{j=1}^{k} g_j/\chi_j}{\lambda^{-1} + \sum_{j=1}^{k} \chi_j^{-1}} \tag{3.33}$$

3.6. Implementation

Our LDA code uses a variational EM inference procedure. In the following Algorithm shown in Figure 3.2, firstly we initialize ϕ_{ni}^0, γ_i, α, β_{ij} from the Line 1-4. In the actual implementation, we set γ_i to a zero matrix and α, β_{ij} to some random matrices. For the generation of a random matrix Matlab function "rand" is used and they are normalized by `normalizer.m`. For ϕ_{ni}^0, we set it to the zero matrix. γ_i initialization above does not contain alpha in order not to waste of calculation because it is already initialized in m-step. Similarly, ϕ_{ni}^0 initialized in E-step. We initialize them in the iterations.

Now we examine the pseudo-code line by line. The code in Line 11 corresponds to Equation 3.21 and it is normalized in Line 12 according to the equation. Line 13 is in accordence with Equation 3.24. This is the E-step and it iterates over documents and words and topics. On the other hand, Line 21 with Equation 3.26, respectively. This is the M-step and it iterates over documents and words and topics, as well. Both E and M step calculations are iterated until the parameters ϕ_{ni}, γ_i and β_{ij} converge. Finally, if the likelihood converges, then the run is finished.

In our actual implementation, ϕ and γ is updated, respectively. On the other hand, β is updated in accordance with Equation 3.26 and finally, α is updated according to Equation 3.30.

The program iterates until the EM cycles converges. We repeat the same cycle again and again, then we choose the least recorded perplexity. The perplexity is equivalent to the inverse of the geometric mean per-word likelihood.

Input: Number of topics K, Corpus with M documents and N_d words in document d

Output: Model parameters: β, θ, z

initialize ϕ^0_{ni} :=1/k for all i in k and n in N_d;
initialize γ_i :=α_i+N/k for all i in k;
initialize α :=50/k;
initialize β_{ij} :=0 for all i in k and j in V;
//E-Step(determine ϕ and γ and compute expected likelihood);
loglikelihood := 0 ;
for $d \leftarrow 1$ ***to*** M **do**
    repeat ;
    for $n \leftarrow 1$ ***to*** N_d **do**
        for $i \leftarrow 1$ ***to*** K **do**
            $\phi^{t+1}_{ni} := \beta_{iw_n} exp(\psi(\gamma^t_{di}))$
        normalize ϕ^{t+1}_{dni} to sum to 1
    $\gamma_{t+1} := \alpha + \sum_{n=1}^{N} \phi^{t+1}_{dni}$;
    until convergence of ϕ_d and γ_d ;
    loglikelihood := loglikelihood + $L(\gamma, \phi; \alpha, \beta)$
//M-Step(maximize the log likelihood of variational distribution);
loglikelihood := 0 ;
for $d \leftarrow 1$ ***to*** M **do**
    for $i \leftarrow 1$ ***to*** K **do**
        for $n \leftarrow 1$ ***to*** N_d **do**
            $\beta_{ij} := \phi_{dni} w_{dnj}$
        normalize β_i to sum to 1
if loglikelihood converged **then** ;
return parameters ;
else ;
go back to E-step ;
endif

Figure 3.2: The LDA Algorithm

3.7. Numerical Experiments of Variational Bayesian LDA

The same corpus of six novellas by three authors is used in order to compare the performance of the pLSA and LDA implementation. 500 trials were made using the MATLAB implementation described before[2] . The elapsed time was 19456 seconds which is more than two times faster than pLSA code.

The perplexity is a decreasing function of the likelihood. Perplexity measures listed in a histogram with a distribution fit is given in Figure 3.3. The figure shows that more than one fourth of the trials falls into the lowest region. A lower perplexity score indicates better generalization performance of the topic model, so we try to minimize perplexity.

$$\log(\text{perplexity}) = \frac{1}{W}\sum_{m}\sum_{n}\log p(w_{m,n}|\text{training model}) \tag{3.34}$$

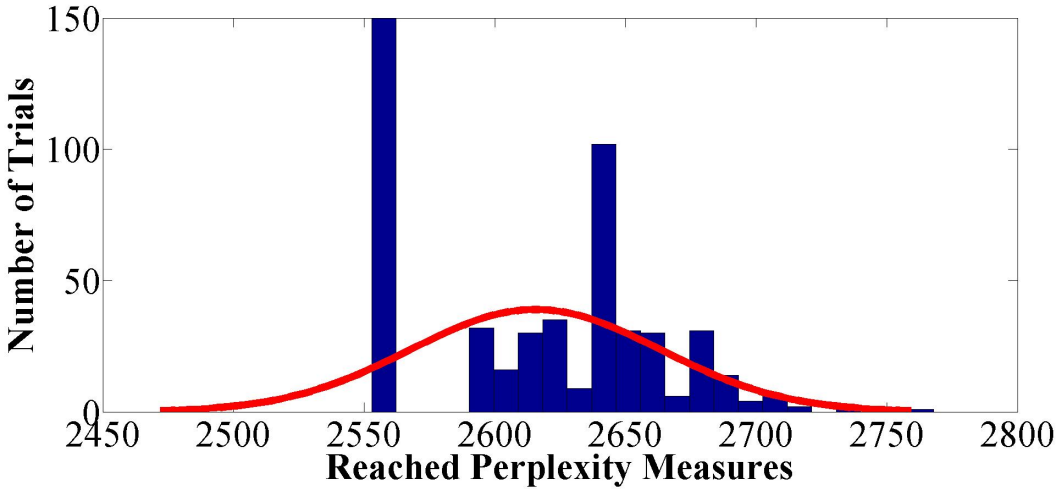

Figure 3.3: A histogram of reached perplexity measures of the corpus six novellas by three authors with LDA. The red curve is a Gaussian fit to the data.

[2]in a 64Bit Matlab, 1.10GHz, 2GB-Ram Computer and this configuration is also used as in other experiments.

In the previous "Implementation" section we pointe out that the lowest perplexity measure's variables such as γ and β are chosen. The evolution of perplexity curves are shown in Figure 3.4. 500 trials are shown as blue thin lines. The minimum reached perplexity curve is indicated with a red line.

Firstly, Table 3.2 shows $p(z \mid d)$ of the lowest achieved likelihood value and we have crisp results. LDA program separates the authors according to their authors.

Table 3.1: Number of words satisfying Equation 2.24 as a result of LDA calculations on six novellas by three authors.

Hemingway	Wells	Conrad
1063	3827	4499

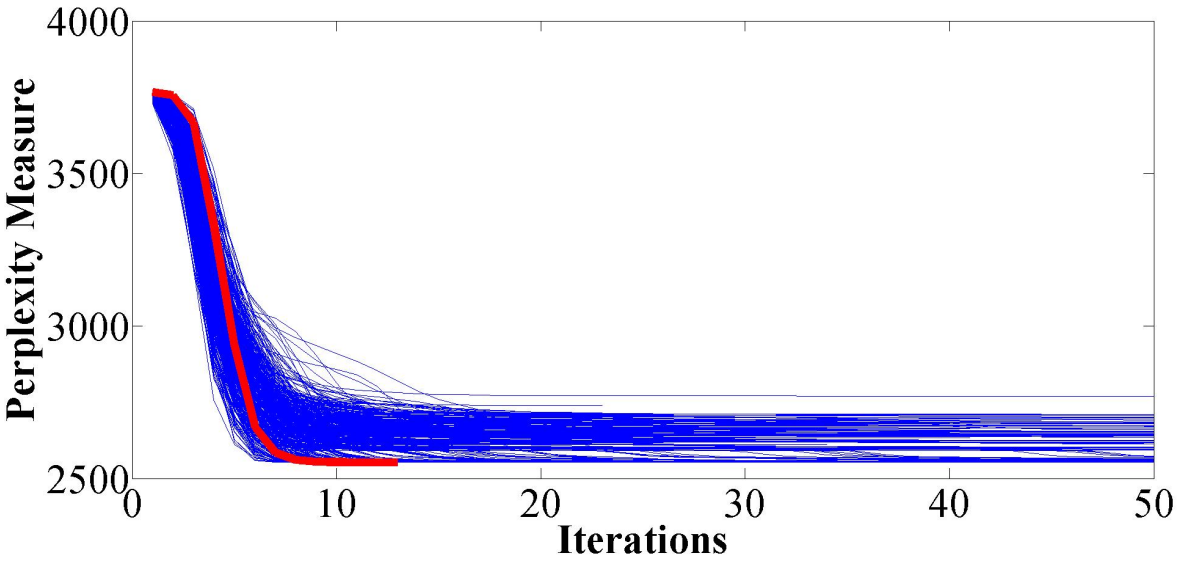

Figure 3.4: The perplexity evolution curves of the corpus six novellas by three authors with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 3.1 shows the number of words satisfying Equation 2.24. The number of words satisfying Equation 2.24 is much higher than 2.2, since we do not eliminate the words satisfying Equation 2.23 this time. However, they are statistically irrelevant. The categorization parameter is dominated by the words belong to and occur in the every element of the same topic.

Table 3.2: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of six novellas by three authors with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Authors (z)	Hemingway	Conrad	Wells
The Island of Doctor Moreau **1896**	0.00006	0.00005	0.99989
The Time Machine **1898**	0.00000	0.00013	0.99987
Heart of Darkness **1899**	0.00001	0.99998	0.00001
The Shadow Line **1915**	0.00002	0.99997	0.00001
The Snows of Kilimanjaro **1936**	0.99989	0.00010	0.00001
The Old Man and the Sea **1952**	0.99992	0.00007	0.00001

Table 3.3 shows the main concepts of the 3-fold categorization. The words are from the $p(z \mid w)$ table satisfying Equation 2.24. The words are chosen with machine precision from the $p(z \mid w)$ table satisfying Equation 2.24.

The overall calculations are identical with pLSA. Let us again look at the relationship between $n(d, w)$ and $p(z \mid w)$ of those picked top ten words in Table 3.6 at the end of the chapter. In the table, the variables are : z_1 is 'Wells'; z_2 is 'Conrad'; z_3 is 'Hemingway', d_1 is 'Conrad-Heart of Darkness'; d_2 is 'Conrad-The Shadow Line'; d_3 is 'Hemingway-Old Man And The Sea'; d_4 is 'Hemingway-Snows of Klimanjero';, d_5 is 'Wells-The Island of Doctor Moreau'; d_6 is 'Wells-Time Machine'.

Relevant statements are hold here as well. We can observe from Table 3.6 at the end of the chapter that if we group d_1 and d_2 as z_1, d_3 and d_4 as z_2, d_5 and d_6 as z_3, the new $n(z_k, w_j)$ is proportional to the $p(z \mid w)$. This is valid for all $p(z \mid w)$

and $n(d_i, w_j)$ tables. This is not a suprising fact since Equation 2.20 and Equation 2.22 relates $n(d, w)$ matrix with $p(z \mid w)$ and $p(z \mid d)$ matrices. Our program gives us results of the maximum information retrieval we can get, which means that it chooses the grouping parameters of d_is and w_js for us.

Table 3.3: Top ten words according to a three fold classification of six novellas by three authors with LDA.

Hemingway	Wells	Conrad
'tommies'	'diamonds'	'dump'
'carcasses'	'doubled'	'flatly'
'locataire'	'lockingup'	'flyer'
'henriquatre'	'slumberous'	'denied'
'regence'	'slunk'	'trident'
'unhooked'	'revulsion'	'officera'
'ditches'	'evillooking'	'sulkily'
'skied'	'injunction'	'mistakes'
'potatoes'	'foreheads'	'nests'
'potato'	'dingey'	'cork'

3.7.1. Results of Maximum Perplexity Case

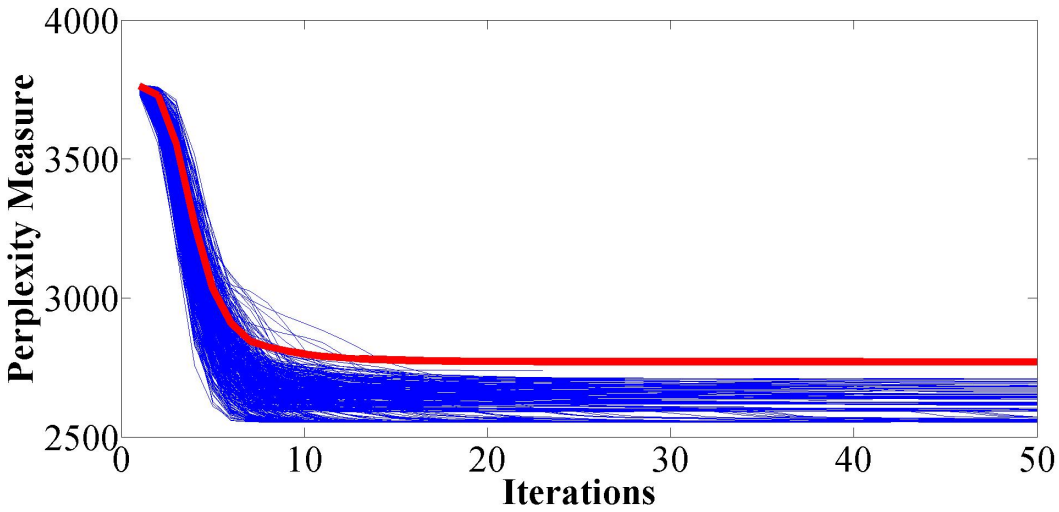

Figure 3.5: The perplexity evolution curves of the corpus six novellas by three authors with LDA. There are 500 trials. Maximum reached perplexity curve is indicated with a red line.

Let us again look at the parameters of the highest perplexity curve reached to get an idea about how program works. Figure 3.5 indicates the highest perplexity curve with a red line. Additionally, Table 3.4 shows the grouping parameter $p(z|d)$. Some items are grouped correctly, though.

Table 3.4: Corresponding $p(z \mid d)$ to the highest achieved perplexity value of six novellas by three authors with LDA.

Documents (d) \ Authors (z)	z_1	z_2	z_3
The Island of Doctor Moreau **1896**	0.00016	0.71967	0.28017
The Time Machine **1898**	0.00003	0.00003	0.99994
Heart of Darkness **1899**	0.00000	1.0000	0.00000
The Shadow Line **1915**	0.00000	1.0000	0.00000
The Snows of Kilimanjaro **1936**	0.05602	0.94391	0.00007
The Old Man and the Sea **1952**	1.0000	0.00000	0.00000

3.7.2. Results of an Intermediate Perplexity Case

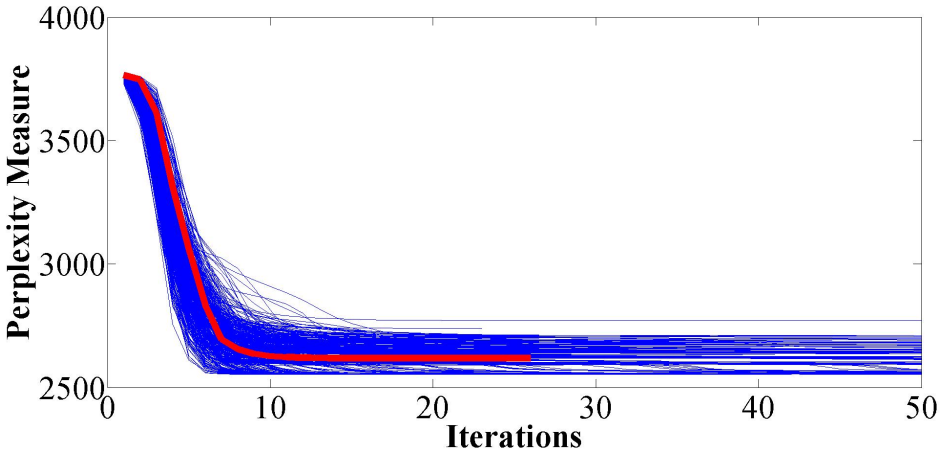

Figure 3.6: The perplexity evolution curves of the corpus six novellas by three authors with LDA. There are 500 trials. Mid-point reached perplexity curve is indicated with a red line.

Mean value of the perplexities reached is calculated again and corresponding mid-point perplexity curve is indicated in Figure 2.6 with a red line. In addition to that, Table 2.5 shows the grouping parameter $p(z|d)$. As the end perplexity value decreases,

more meaningful results we get. Grouping is correct, however there are deviations and "The Island of Doctor Moreau" grouped in a different topic.

Table 3.5: Corresponding $p(z \mid d)$ to the mid-point achieved perplexity value of six novellas by three authors with LDA.

Documents (d) \ Authors (z)	z_1	z_2	$z_3 3$
The Island of Doctor Moreau **1896**	0.00002	0.99968	0.0030
The Time Machine **1898**	0.00001	0.00015	0.99985
Heart of Darkness **1899**	0.00001	0.00002	0.99997
The Shadow Line **1915**	0.00001	0.99999	0.00000
The Snows of Kilimanjaro **1936**	0.99998	0.0001	0.00001
The Old Man and the Sea **1952**	0.99998	0.0001	0.00001

3.8. Results and Discussion

Using Latent Dirichlet Allocation for the six texts, we recover the classification by authorship as in the case of pLSA. For the minimum perplexity, we get rather crisp results. However, probabilities are not exact, there are non-zero ϵ values. For pLSA there is a direct relationship between the likelihood value and results. That holds also for LDA. As the perplexity measure decreases, we get more significant results.

Let us look at a different question. Although we did not eliminate the words satisfying Equation 2.23, so there are more words in the corpus, the total calculation is faster than pLSA, taking only 5,5 hours. Why is our LDA implementation faster than pSLA? The answer is simple and obvious. Let us compare Algorithm shown in Figure 2.2 and Algorithm shown in Figure 3.2. Since K is the number of topics, the corpus contains M documents and N_d is the words in document d, for pLSA there are $(K * M * N_d + M + N_d) + (K * M * N_d + N_d + K) + (K + K)$ calculations per iteration amounting to $2 * K * M * N_d + M + 2 * N_d + 3 * K$ calculations per iteration in total. On the other hand, for LDA there are $(M * K * N_d + N_d) + (M * K * N_d + M * K)$ calculations per iteration which is $(2 * K * M * N_d + N_d + M * K)$ calculations per iteration in total. Both are nearly equal. All else being equal, these two algorithms should have similar performance times then. However in the Matlab calculations, we use matrix multiplications for pLSA, there are also zero components. However, in LDA multiplications is programmed as element-wise and zero components are not included, explaining why our LDA implementation is faster than our pLSA implementation. So this is more an issue of how we implemented the two algorithms.

To conclude, applied to the same corpus of texts, both pLSA and LDA produce identical classifications, but our LDA implementation is faster. Hence, we will use LDA for the analysis of the plays of Shakespeare in the next chapter.

Table 3.6: A comparison of $n(d, w)$ matrix and $p(z \mid w)$ matrix of a three fold classification of six novellas by three authors with LDA. Picked words are from the Table 3.3.

Words	$p(z_k \mid w_j)$			$n(w_j, d_i)$					
w_j	z_1	z_2	z_3	d_1	d_2	d_3	d_4	d_5	d_6
'diamonds'	1.00000	0.00000	0.00000	0	0	0	0	1	0
'doubled'	1.00000	0.00000	0.00000	0	0	0	0	2	0
'lockingup'	1.00000	0.00000	0.00000	0	0	0	0	1	0
'slumberous'	1.00000	0.00000	0.00000	0	0	0	0	1	0
'slunk'	1.00000	0.00000	0.00000	0	0	0	0	2	0
'revulsion'	1.00000	0.00000	0.00000	0	0	0	0	1	0
'evillooking'	1.00000	0.00000	0.00000	0	0	0	0	1	0
'injunction'	1.00000	0.00000	0.00000	0	0	0	0	1	0
'foreheads'	1.00000	0.00000	0.00000	0	0	0	0	3	0
'dingey'	1.00000	0.00000	0.00000	0	0	0	0	8	0
'tommies'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'carcasses'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'locataire'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'henriquatre'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'regence'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'unhooked'	0.00000	1.00000	0.00000	0	0	1	0	0	0
'ditches'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'skied'	0.00000	1.00000	0.00000	0	0	0	2	0	0
'potatoes'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'potato'	0.00000	1.00000	0.00000	0	0	0	1	0	0
'dump'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'flatly'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'flyer'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'denied'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'trident'	0.00000	0.00000	1.00000	0	2	0	0	0	0
'officera'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'sulkily'	0.00000	0.00000	1.00000	0	3	0	0	0	0
'mistakes'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'nests'	0.00000	0.00000	1.00000	0	1	0	0	0	0
'cork'	0.00000	0.00000	1.00000	0	2	0	0	0	0

4. STATISTICAL ANALYSES OF SHAKESPEAREAN PLAYS

4.1. Introduction

In this chapter, we investigate whether we can classify the plays of Shakespeare based on the choice of words in the texts. Methodologically, we apply the LDA approach to the texts. Traditionally, Shakespeare's plays are divided into three: History plays, Comedies and Tragedies [10]. We examine whether we can recover this traditional division by applying LDA. Secondly, we consider the lines uttered by each of the characters within play as a document and try to classifiy the protagonists through their lines. In particular, we are interested to what extent such a classification can tell us something about the characters and relationships between them. Lastly, we perform the same analysis off all the major characters across four plays: *Richard III*, *Othello*, *Hamlet* and *Macbeth*.

4.2. A 3-Fold LDA Classification of All Plays

Here our aim is to see to what extent a 3-fold LDA classfication of the plays, each treated as a document, correlates with the tradional classification of these as 'Tragedy', 'History' and 'Comedy'. The 3-fold LDA classification of all 37 plays (*see* Table 4.1) with 500 runs takes about 40 hours in our LDA implementation. A frequency histogram of the obtained perplexity measures and perplexity development curves are provided in Figure 4.1 and Figure 4.2, respectively. Although there are deviations, the histogram bars follow closely the Gaussian fit curve.

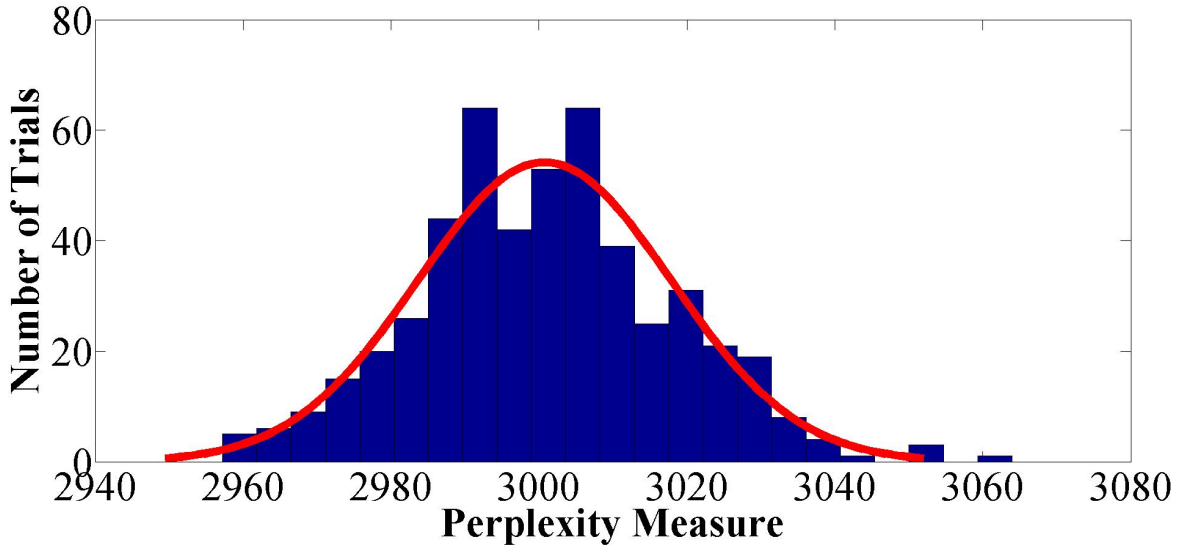

Figure 4.1: A histogram of reached perplexity measures of the corpus Shakespeare's all plays in three fold classification with LDA. The red curve is a Gaussian fit to the data.

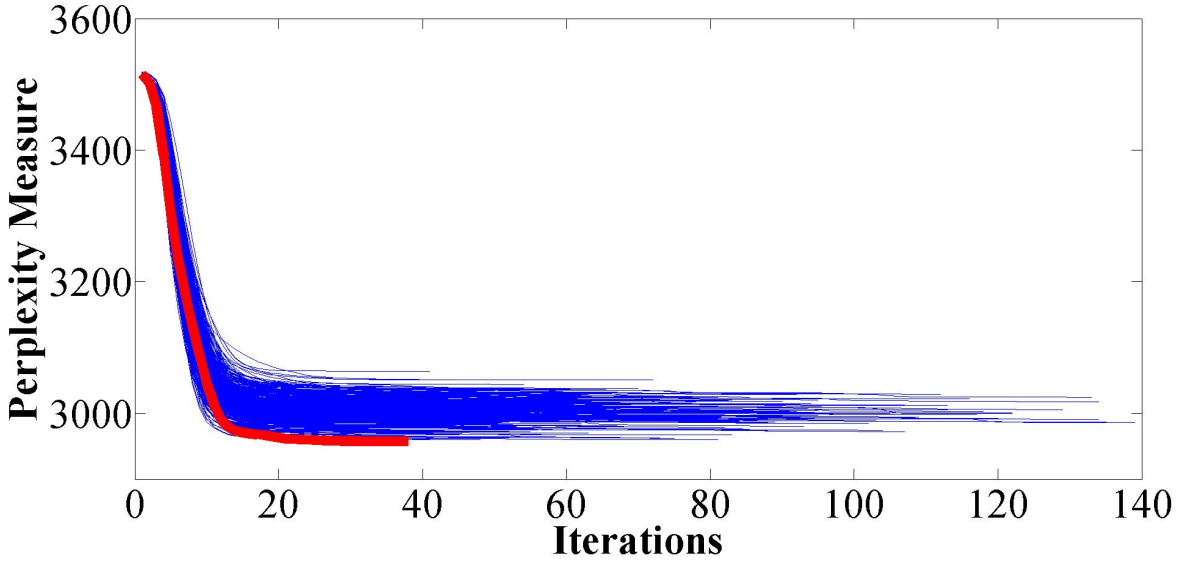

Figure 4.2: The perplexity evolution curves of the corpus Shakespeare's all plays in three fold classification with LDA. We performed 500 classifications. Minimum reached perplexity curve is indicated with a red line.

Table 4.1: Three-fold classification of all of Shakespeare's 37 plays using LDA. Shown in the table is the conditional probability $p(z \mid d)$ obtained from the LDA classification with lowest achieved perplexity value. For each play, the topic variable z_i with highest probabilities is highlighted with a grey background.

Documents (d) \ Category (z)	z_1	z_2	z_3
'Comedy-All's Well That Ends Well'	0.66456	0.00767	0.32777
'Comedy-As You Like It'	0.99825	0.00031	0.00144
'Comedy-Comedy of Errors'	0.99960	0.00015	0.00025
'Comedy-Cymbeline'	0.00163	0.00013	0.99824
'Comedy-Love's Labour's Lost'	0.99813	0.00185	0.00002
'Comedy-Measure for Measure'	0.95878	0.00110	0.04012
'Comedy-Merchant of Venice'	0.99739	0.00061	0.00201
'Comedy-Merry Wives of Windsor'	0.99993	0.00005	0.00002
'Comedy-Midsummer Night's Dream'	0.99950	0.00007	0.00043
'Comedy-Much Ado About Nothing'	0.99960	0.00030	0.00011
'Comedy-Pericles'	0.00298	0.00166	0.99536
'Comedy-Taming of the Shrew'	0.99985	0.00005	0.00010
'Comedy-The Tempest'	0.00072	0.00003	0.99924
'Comedy-Troiles and Cressida'	0.99221	0.00189	0.00590
'Comedy-Twelfth Night'	0.99402	0.00001	0.00596
'Comedy-Two Gentlemen of Verona'	0.99741	0.00088	0.00171
'Comedy-Winter's Tale'	0.00369	0.00001	0.99630
'History-Henry IV, part 1'	0.01490	0.98359	0.00152
'History-Henry IV, part 2'	0.12018	0.86996	0.00986
'History-Henry V'	0.00096	0.99859	0.00045
'History-Henry VI, part 1'	0.00001	0.99987	0.00011
'History-Henry VI, part 2'	0.00003	0.99972	0.00025
'History-Henry VI, part 3'	0.00002	0.99988	0.00010
'History-Henry VIII'	0.00907	0.63102	0.35991
'History-King John'	0.00301	0.99447	0.00253
'History-Richard II'	0.00001	0.99961	0.00038
'History-Richard III'	0.00050	0.99903	0.00048
'Tragedy-Antony and Cleopatra'	0.00010	0.00001	0.99989
'Tragedy-Coriolanus'	0.00003	0.00004	0.99993

Documents (d) \ Category (z)	z_1	z_2	z_3
'Tragedy-Hamlet'	0.18146	0.56617	0.25237
'Tragedy-Julius Caeser'	0.00016	0.00078	0.99906
'Tragedy-King Lear'	0.00400	0.00173	0.99427
'Tragedy-Macbeth'	0.00003	0.01047	0.98950
'Tragedy-Othello'	0.94838	0.00021	0.05141
'Tragedy-Romeo and Juliet'	0.98821	0.01028	0.00152
'Tragedy-Timon of Athens'	0.00107	0.00001	0.99892
'Tragedy-Titus Andronicus'	0.00184	0.98830	0.00986

We can observe from Table 4.1 that most of the plays follow the traditional classification. History, for example, is a rigid block. However, *Hamlet* is grouped with the History plays. That might arise from the fact that *Hamlet* is a loose adaptation of a historical event. Moreover, it shows characteristics of a History play: the play involves kings and queens along with their power relations. In *H*amlet in particular, the major conflict arises over the ownership of the throne. What else we can say about the Comedies? For example. J. Spens in his book about Elizabethan Drama [11] argues that although the plays are divided into three in the First Folio,[3] there is a sub-genre called Romance in Comedies. Romances are four according to J. Spens: *Cymbeline*, *Winter's Tale*, *The Tempest* and *Pericles*. They are classified with Tragedies in the Table 4.1. As a matter of fact, *Cymbeline* was classified as a Tragedy in the First Folio. These might be the reasons for the deviations from the genre Comedy. For the deviations from Tragedy to Comedy, there is research that regards *Othello* and *Romeo and Juliet*[4] as Comedies ending with tragic events [12], [13].

We next focus on the 2-fold classification of all plays.

[3]The First Folio is commonly referred by modern scholars to the book published in 1623: *Mr. William Shakespeare's Comedies, Histories, and Tragedies*. It is a collection of Shakespeare's plays.

[4]"Romeo and Juliet is different from Shakespeare's other tragedies in that it becomes, rather than is, tragic. Other tragedies have reversals, but in Romeo and Juliet the reversal is so radical as to constitute a change of genre: the action and the characters begin in familiar comic patterns, and are then transformed – or discarded – to compose the pattern of tragedy." Snyder, Susan. "Romeo and Juliet: Comedy into Tragedy." Essays in Criticism 20 (1970): 391-402.

4.3. A 2-Fold Analysis of All Plays

A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.3 and Figure 4.4, respectively.

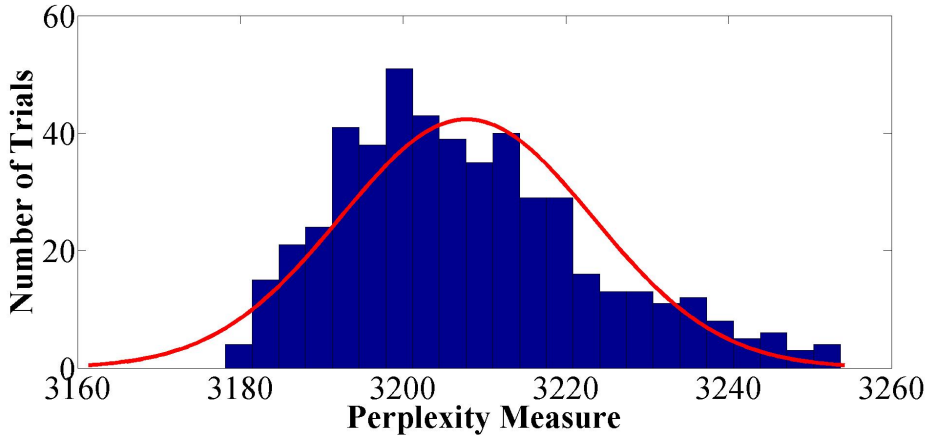

Figure 4.3: A histogram of reached perplexity measures of the corpus Shakespeare's all plays in two fold classification with LDA. The red curve is a Gaussian fit to the data.

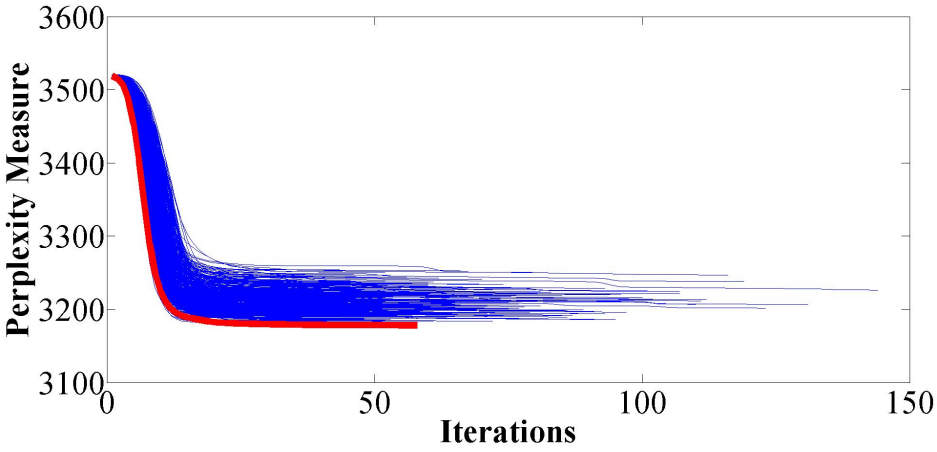

Figure 4.4: The perplexity evolution curves of the corpus Shakespeare's all plays in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.2: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of Shakespeare's all plays with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Category (z)	z_1	z_2
'Comedy-All's Well That Ends Well'	0.01102	0.98898
'Comedy-As You Like It'	0.00111	0.99889
'Comedy-Comedy of Errors'	0.00127	0.99873
'Comedy-Cymbeline'	0.03473	0.96527
'Comedy-Love's Labour's Lost'	0.00052	0.99948
'Comedy-Measure for Measure'	0.00227	0.99773
'Comedy-Merchant of Venice'	0.00148	0.99852
'Comedy-Merry Wives of Windsor'	0.00023	0.99977
'Comedy-Midsummer Night's Dream'	0.00022	0.99978
'Comedy-Much Ado About Nothing'	0.00153	0.99847
'Comedy-Pericles'	0.02225	0.97775
'Comedy-Taming of the Shrew'	0.00055	0.99945
'Comedy-The Tempest'	0.00423	0.99577
'Comedy-Troiles and Cressida'	0.00271	0.99729
'Comedy-Twelfth Night'	0.00015	0.99985
'Comedy-Two Gentlemen of Verona'	0.00141	0.99859
'Comedy-Winter's Tale'	0.00347	0.99653
'History-Henry IV, part 1'	0.98658	0.01342
'History-Henry IV, part 2'	0.92933	0.07067
'History-Henry V'	0.99921	0.00079
'History-Henry VI, part 1'	0.99980	0.00020
'History-Henry VI, part 2'	0.99976	0.00024
'History-Henry VI, part 3'	0.99980	0.00020
'History-Henry VIII'	0.87758	0.12242
'History-King John'	0.99522	0.00478
'History-Richard II'	0.99975	0.00025
'History-Richard III'	0.99904	0.00096
'Tragedy-Antony and Cleopatra'	0.12538	0.87462
'Tragedy-Coriolanus'	0.93925	0.06075

Documents (d) \ Category (z)	z_1	z_2
'Tragedy-Hamlet'	0.01712	0.98288
'Tragedy-Julius Caeser'	0.99314	0.00686
'Tragedy-King Lear'	0.03357	0.96643
'Tragedy-Macbeth'	0.92003	0.07997
'Tragedy-Othello'	0.00048	0.99952
'Tragedy-Romeo and Juliet'	0.01452	0.98548
'Tragedy-Timon of Athens'	0.01412	0.98588
'Tragedy-Titus Andronicus'	0.99723	0.00277

According to the Table 4.2, Comedies and Histories are two rigid groups and Tragedies are scattered among them. Although *Anthony and Cleopatra*, *Hamlet*, *King Lear*, *Othello*, *Romeo and Juliet*, *Timon of Athens* and *Titus Andronicus* are mostly (except *Romeo and Juliet*) based on historical events, they are grouped with Comedies. We know that some of the Comedies such as *Much Ado About Nothing* starts as a Comedy and then turn into tragedies. These plays have been referred to as Traji-Comedy [14]. This might be the reason why they are classified in the same group as the tragedies. On the other hand, it is not surprising that some of the Tragedies are grouped with Histories such as *Coriolanus*, *Julius Ceaser* and *Macbeth*, since these two genres are close to each other. If we compare the two tables, Table 4.1 and Table 4.2, *Othello* and *Romeo and Juliet* are again classified as Comedy, while *Hamlet* this time is classified as Comedy.

It appears that history plays are a rigid category appearing together in the 2- and 3-fold classifications, while the Comedies and Tragedies seem to mix. This is an agreement with literary scholarship which points out that the border between Comedies and Tragedies, as well as Histories and Tragedies is not clean cut. This might also explain partly the classifications we have found. We also attempted a 4-way classification, however this did not result in meaningful results.

4.4. A 2-Fold Analysis of The Play Hamlet

4.4.1. All Characters

The aim of this section is to find out whether treating the lines uttered by each character of a play as a document, the LDA classification of these can tell us something about the relation amongst the protagonists themselves. In this section we consider the play *Hamlet*. In order to do that we perform a 2-fold categorization of all characters in the play. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.5 and Figure 4.6, respectively.

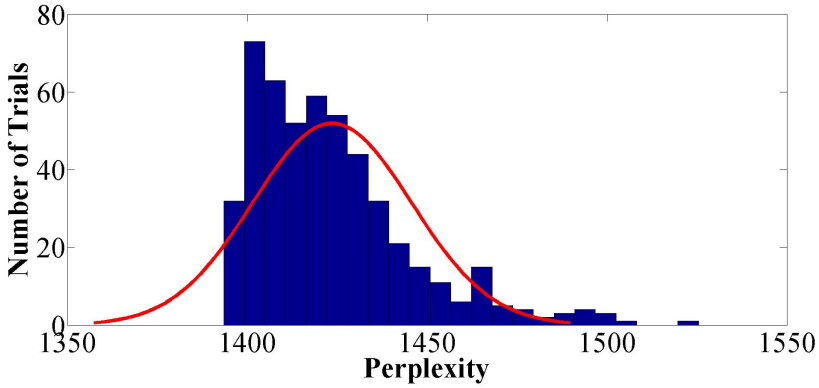

Figure 4.5: A histogram of reached perplexity measures of the corpus Hamlet in two fold classification with LDA. The red curve is a Gaussian fit to the data.

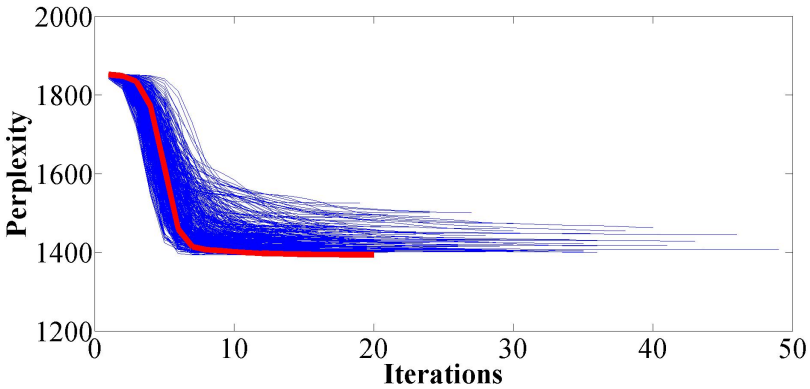

Figure 4.6: The perplexity evolution curves of the corpus Hamlet in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.3: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of all characters in the play Hamlet with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'BERNARDO'	0.00022	0.99978
'Captain'	0.99915	0.00085
'Danes'	0.00432	0.99568
'First Ambassador'	0.46547	0.53453
'First Clown'	0.99989	0.00011
'First Player'	0.00010	0.99990
'First Priest'	0.00046	0.99954
'First Sailor'	0.99771	0.00229
'FRANCISCO'	0.00067	0.99933
'Gentleman'	0.99957	0.00043
'Ghost'	0.00005	0.99995
'GUILDENSTERN'	0.99970	0.00030
'HAMLET'	0.99999	0.00001
'HORATIO'	0.00002	0.99998
'KING CLAUDIUS'	0.00142	0.99858
'LAERTES'	0.00003	0.99997
'LORD POLONIUS'	0.00002	0.99998
'Lord'	0.99839	0.00161
'LUCIANUS'	0.00053	0.99947
'MARCELLUS'	0.00010	0.99990
'Messenger'	0.00158	0.99842
'OPHELIA'	0.00004	0.99996
'OSRIC'	0.99977	0.00023
'Player King'	0.00010	0.99990
'Player Queen'	0.00017	0.99983
'PRINCE FORTINBRAS'	0.00023	0.99977
'QUEEN GERTRUDE'	0.00004	0.99996
'REYNALDO'	0.00124	0.99876
'ROSENCRANTZ'	0.00007	0.99993
'Second Clown'	0.99918	0.00082
'Servant'	0.00432	0.99568
'VOLTIMAND'	0.99951	0.00049

From the $p(z \mid d)$ table of the play *Hamlet*, we can observe that main characters such as King Claudius, Lord Polonius, Ophelia, Laertes and Queen Gertrude are classified together, while Hamlet is classified as belonging to another group. In terms of the play, this result makes sense, if antagonistic relations drive the classification. King Claudius is the uncle who occupies the throne and Queen Gertrude is the mother of Hamlet and wife of the King. Ophelia loves Hamlet – tragically. While Laertes kills Hamlet at the end of the play. Lord Polonius, father of Laertes and Ophelia, is killed by Hamlet. Interestingly, the play contains a "play within a play": Hamlet is trying to show that King Claudius killed his father by a mise en scène. The Player King and Player Queen are in the same group with their resemblances. However, for example, Prince Fortinbras is against the throne, the King. In addition to that two friends of Hamlet are grouped in the opposite groups: while Guildenstern is with Hamlet, Rosencrantz is against him. These are weaknesses of our approach. It is hard to tell anything about secondary characters, they don't have much conflict with the major characters. For instance, Captain of Prince Fortinbras is classified in the same group with Hamlet. This also is in disagreement with assuming that antagonistic relations drive the classification. However, for the main characters the classification obtained by LDA is compatible with one based on antagonistic relationships in the play.

We next focus on subsets of the dramatis personae, by considering protagonist/documents that contain a minimum number of words. As word thresholds we use 248, 500 and 1200 words and refer to the resulting set of protagonist/documents as large, medium and small.

4.4.2. Large Group of Characters

In order to continue our investigation on the play, we make 2-fold categorization of a large set of dramatis personae. 248 words are chosen as a threshold. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.7 and Figure 4.8, respectively.

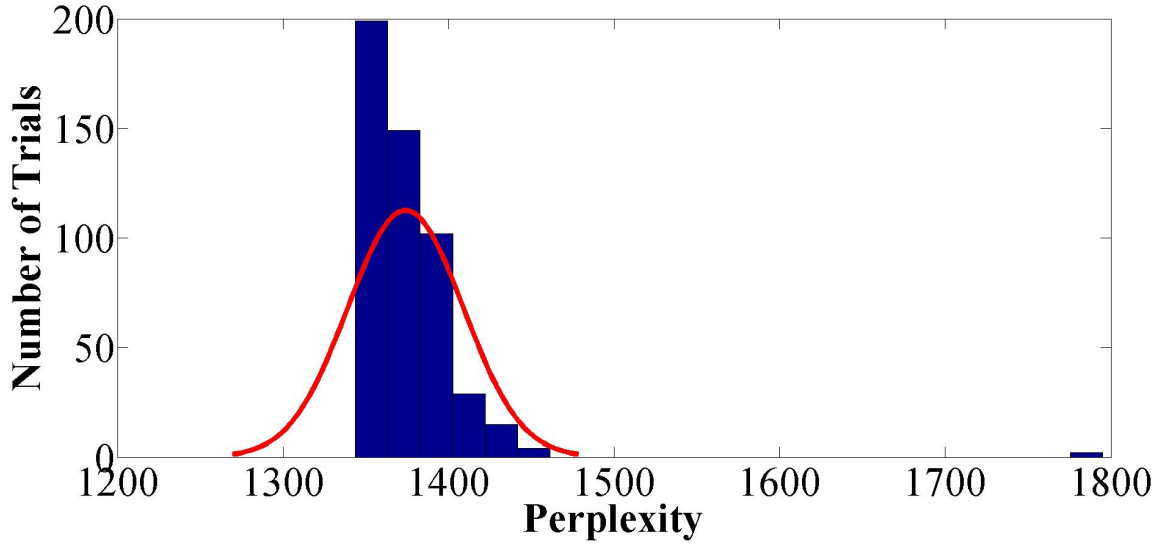

Figure 4.7: A histogram of reached perplexity measures of the corpus large set of characters of Hamlet in two fold classification with LDA. The red curve is a Gaussian fit to the data.

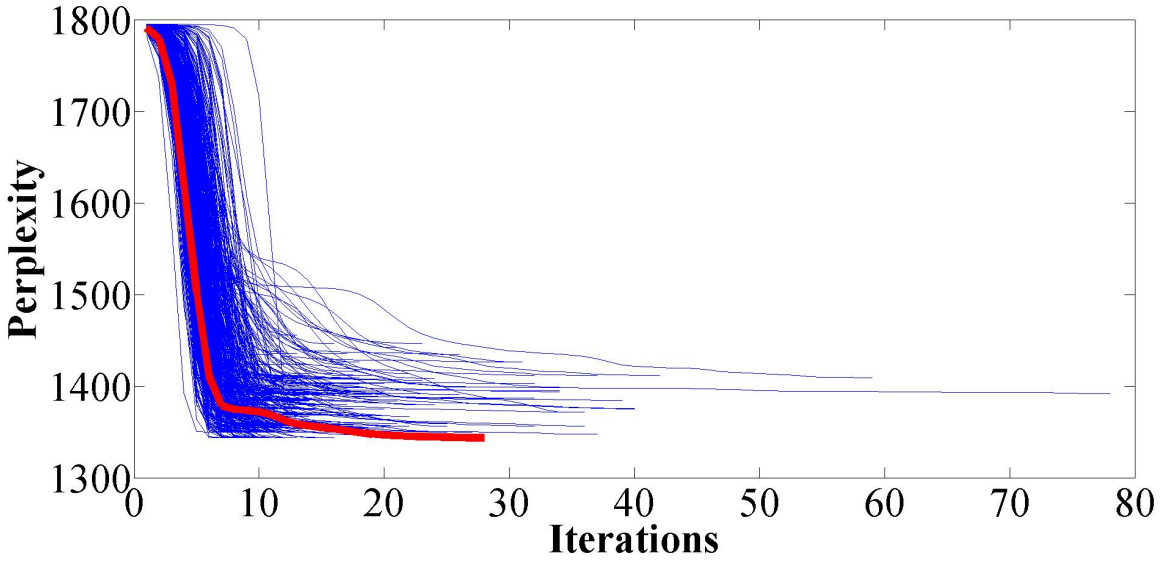

Figure 4.8: The perplexity evolution curves of the corpus large set of characters of Hamlet in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.4: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of large set of characters in the play Hamlet with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'BERNARDO'	0.00019	0.99981
'First Clown'	0.99991	0.00009
'First Player'	0.99986	0.00014
'Ghost'	0.00004	0.99996
'GUILDENSTERN'	0.99977	0.00023
'HAMLET'	0.99999	0.00001
'HORATIO'	0.00002	0.99998
'KING CLAUDIUS'	0.00001	0.99999
'LAERTES'	0.00002	0.99998
'LORD POLONIUS'	0.00001	0.99999
'MARCELLUS'	0.99986	0.00014
'OPHELIA'	0.00003	0.99997
'OSRIC'	0.99982	0.00018
'Player King'	0.00008	0.99992
'Player Queen'	0.00014	0.99986
'QUEEN GERTRUDE'	0.00003	0.99997
'ROSENCRANTZ'	0.00006	0.99994

The classification of this subset of protagonists does not show much differences with the calculations that considered all characters.

4.4.3. Medium Group of Characters

We next turn to a 2-fold categorization of a medium subset of dramatis personae. 500 words are chosen as a limit. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.9 and Figure 4.10, respectively.

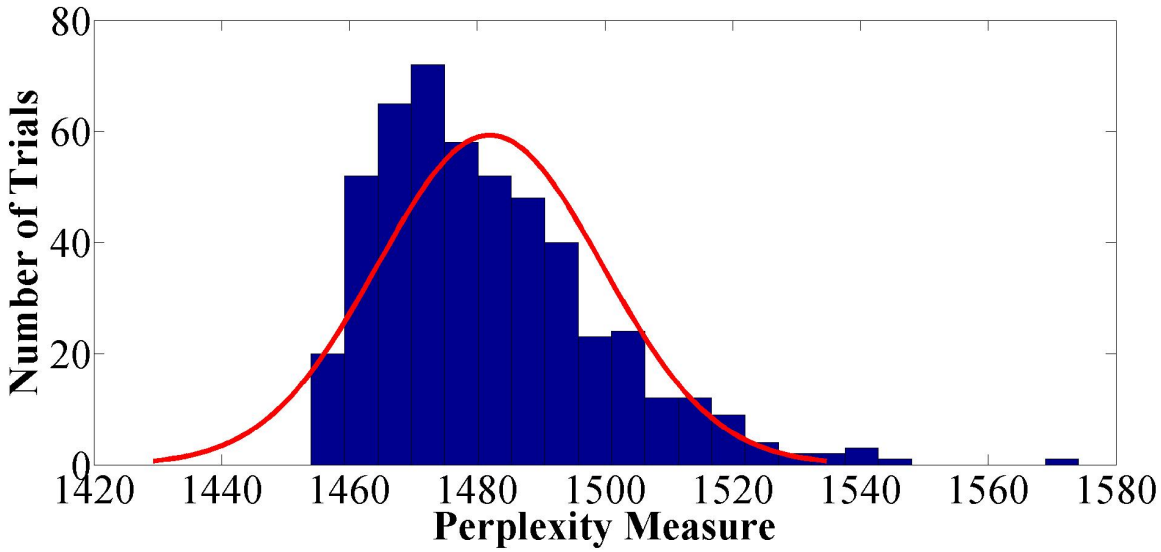

Figure 4.9: A histogram of reached perplexity measures of the corpus medium set of characters of the play Hamlet in two fold classification with LDA. The red curve is a Gaussian fit to the data.

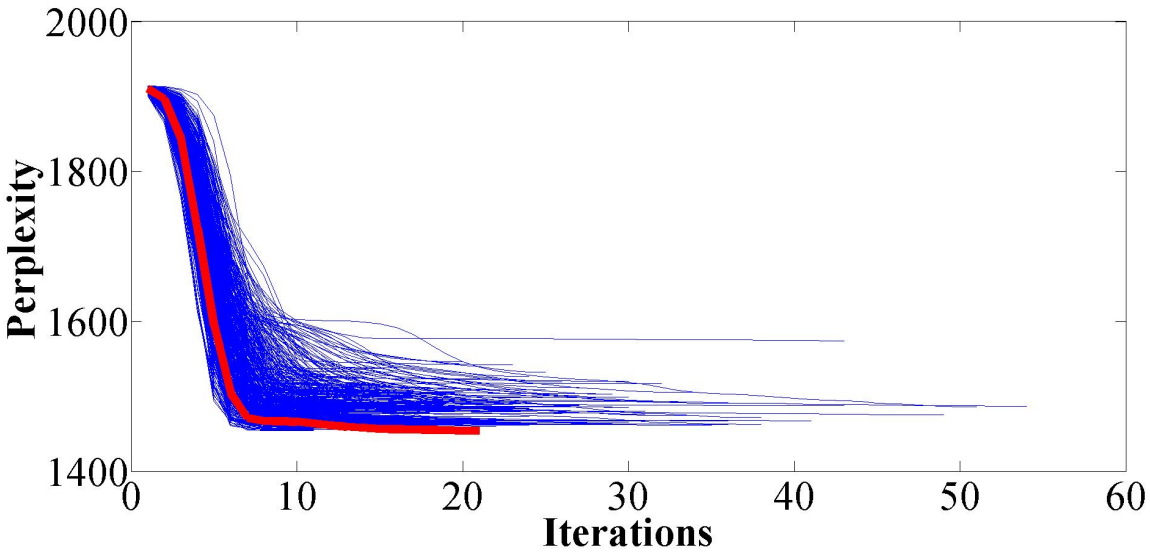

Figure 4.10: The perplexity evolution curves of the corpus medium set of characters of Hamlet in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.5: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of medium set of characters in the play Hamlet with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'First Clown'	0.99985	0.00015
'Ghost'	0.00004	0.99996
'HAMLET'	0.99999	0.00001
'HORATIO'	0.00002	0.99998
'KING CLAUDIUS'	0.00001	0.99999
'LAERTES'	0.00002	0.99998
'LORD POLONIUS'	0.00001	0.99999
'OPHELIA'	0.00003	0.99997
'QUEEN GERTRUDE'	0.00003	0.99997
'ROSENCRANTZ'	0.00006	0.99994

For the medium set of characters, all are classified as belonging into one group except Hamlet and the Fist Clown. Arguably, this classification follows the conflicts of the play.

We next focus on a small subset of the dramatis personae.

4.4.4. Small Group of Characters

In order to continue our investigation on the play, we make 2-fold categorization of a small subset of dramatis personae. 1200 words are chosen as a limit, since there is a gap between QUEEN GERTRUDE and First Clown as shown in Table A.3. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.11 and Figure 4.12, respectively.

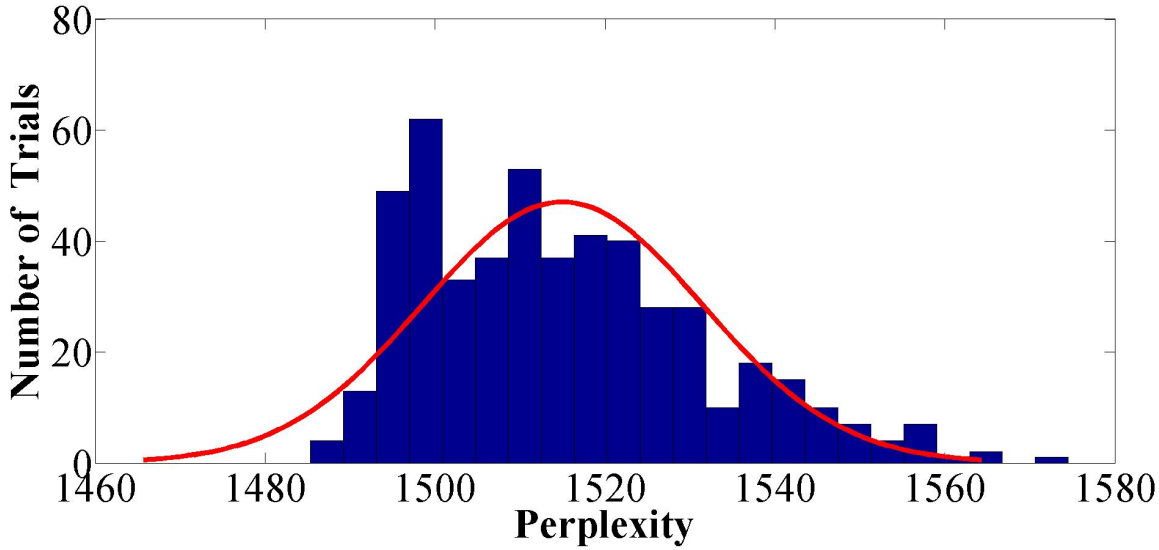

Figure 4.11: A histogram of reached perplexity measures of the corpus for the small set of characters of the play Hamlet in two fold classification with LDA. The red curve is a Gaussian fit to the data.

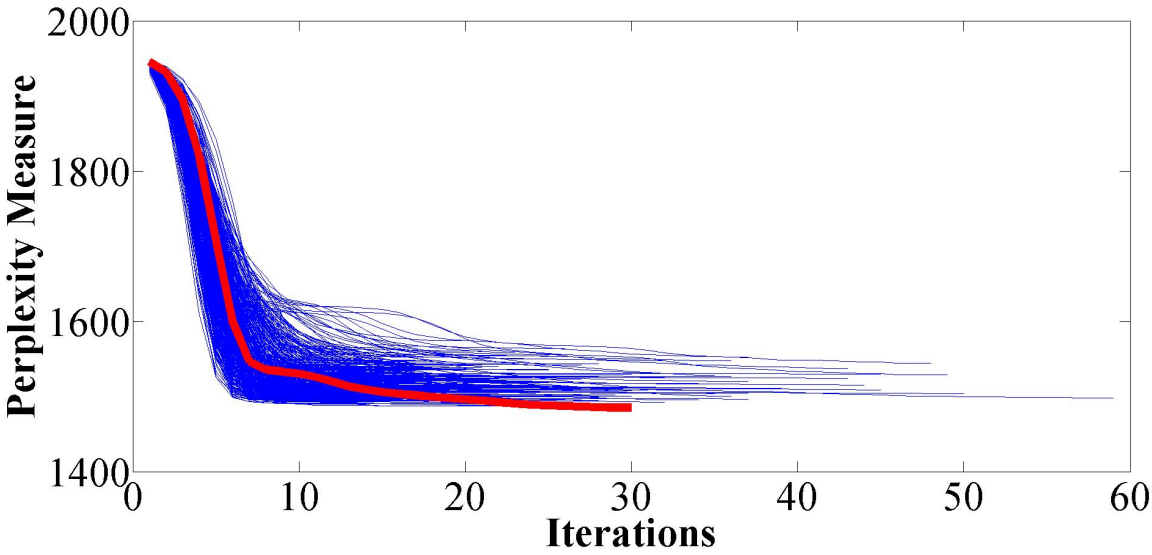

Figure 4.12: The perplexity evolution curves of the corpus for the small set of characters of Hamlet in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.6: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of small set of characters in the play Hamlet with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'HAMLET'	0.00001	0.99999
'HORATIO'	0.99998	0.00002
'KING CLAUDIUS'	0.99999	0.00001
'LAERTES'	0.99998	0.00002
'LORD POLONIUS'	0.99999	0.00001
'OPHELIA'	0.99997	0.00003
'QUEEN GERTRUDE'	0.99997	0.00003

For the small set of characters, all characters except Hamlet are classified in one group. Again, the play puts Hamlet against the other members of the throne and thus our LDA classification can be argued to follow the play's conflicts.

4.5. A 2-Fold Analysis of The Play Othello

4.5.1. All Characters

In this section we repeat the previous analysis this time consindering the play Othello. In order to do that we make 2-fold categorization of all characters in the play. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.13 and Figure 4.14, respectively.

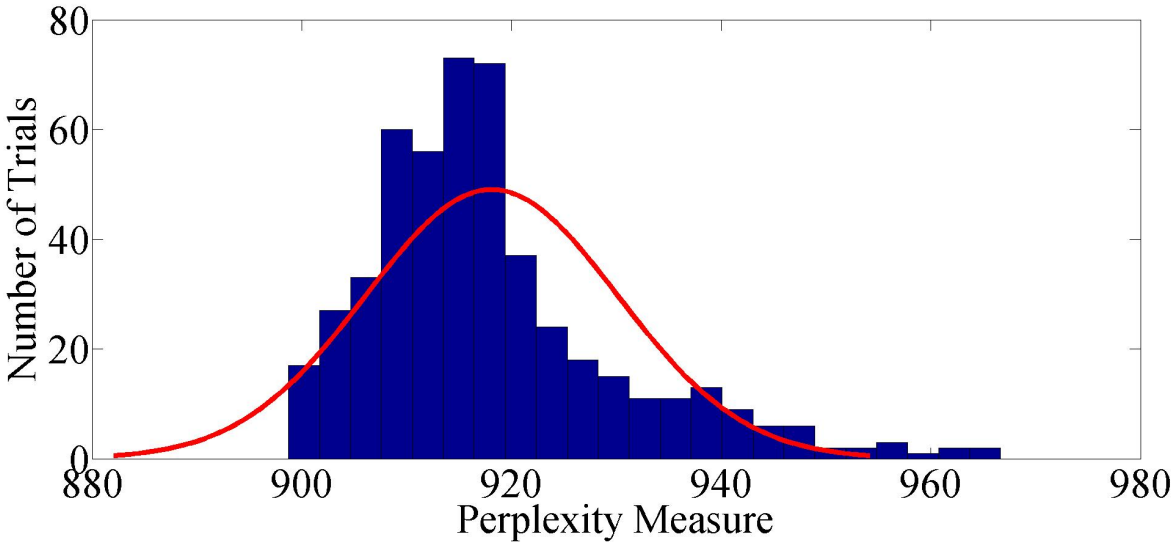

Figure 4.13: A histogram of reached perplexity measures of the corpus Othello in two fold classification with LDA. The red curve is a Gaussian fit to the data.

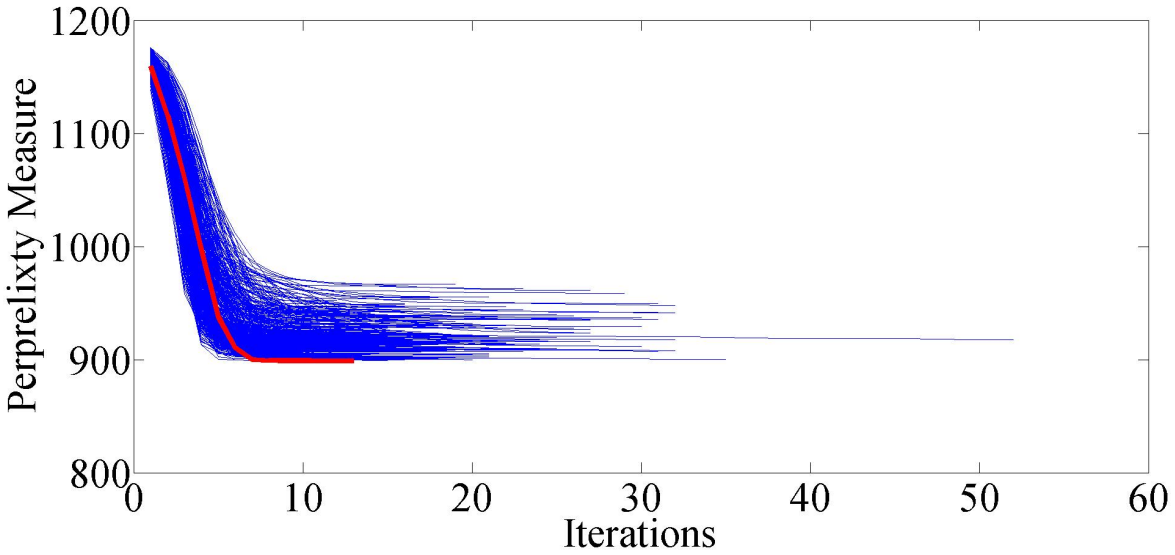

Figure 4.14: The perplexity evolution curves of the corpus Othello in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.7: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of all characters in the play Othello with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'BIANCA'	0.00019	0.99981
'BRABANTIO'	0.00005	0.99995
'CASSIO'	0.00002	0.99998
'Clown'	0.00027	0.99973
'DESDEMONA'	0.99995	0.00005
'DUKE OF VENICE'	0.00009	0.99991
'EMILIA'	0.99993	0.00007
'First Gentleman'	0.00325	0.99675
'First Musician'	0.00163	0.99837
'First Officer'	0.99613	0.00387
'First Senator'	0.00026	0.99974
'Fourth Gentleman'	0.00253	0.99747
'Gentleman'	0.01103	0.98897
'GRATIANO'	0.00020	0.99980
'Herald'	0.99865	0.00135
'IAGO'	0.00001	0.99999
'LODOVICO'	0.00009	0.99991
'Messenger'	0.99800	0.00200
'MONTANO'	0.00011	0.99989
'OTHELLO'	0.99998	0.00002
'RODERIGO'	0.00006	0.99994
'Sailor'	0.00190	0.99810
'Second Gentleman'	0.00060	0.99940
'Second Gentlemen'	0.00452	0.99548
'Second Senator'	0.00152	0.99848
'Senator'	0.00452	0.99548
'Third Gentleman'	0.00043	0.99957

From the Table 4.7, we can observe that the main characters Othello, Desdemona, Emilia are groupe together, while Iago is classified in opposition. It is in accordence with the character conflicts of the play *Othello*. For the group z_1, it is hard to analyze side characters such as First Officer, Herald and Messenger. Rest of the group, we mean the three characters, Othello, Desdemona and Emilia, are victim of Iago, the evil character. Although Captain Cassio is another victim of Iago's plan, he is classified in the second group, z_2, with Iago. This contradicts our assumption that LDA classification follows the antagonisms of the play. Iago shows himself as a good person to the other characters but we know from his monologues that he has an evil plan for Othello. This makes reasonable why Iago is in the second group, z_2. There is a conflict between Brabantio and Othello and Desdemona from the beginning. Roderido, who is in love with Desdemona, has a conflict with Othello, but they don't have a one-on-one speech. These characters are opponents and they are correctly classified as belonging to different groups. It is hard to analyze conflicts of Duke of Venice in the play. But he is classified in the group opposite to Othello.

4.5.2. Small Group of Characters

In order to continue our investigation on the play, we make 2-fold categorization of a small subset of dramatis pesonae. 900 words are chosen as a limit, since there is a gap between RODERIGO and LODOVICO as shown in Table A.4. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.15 and Figure 4.16, respectively.

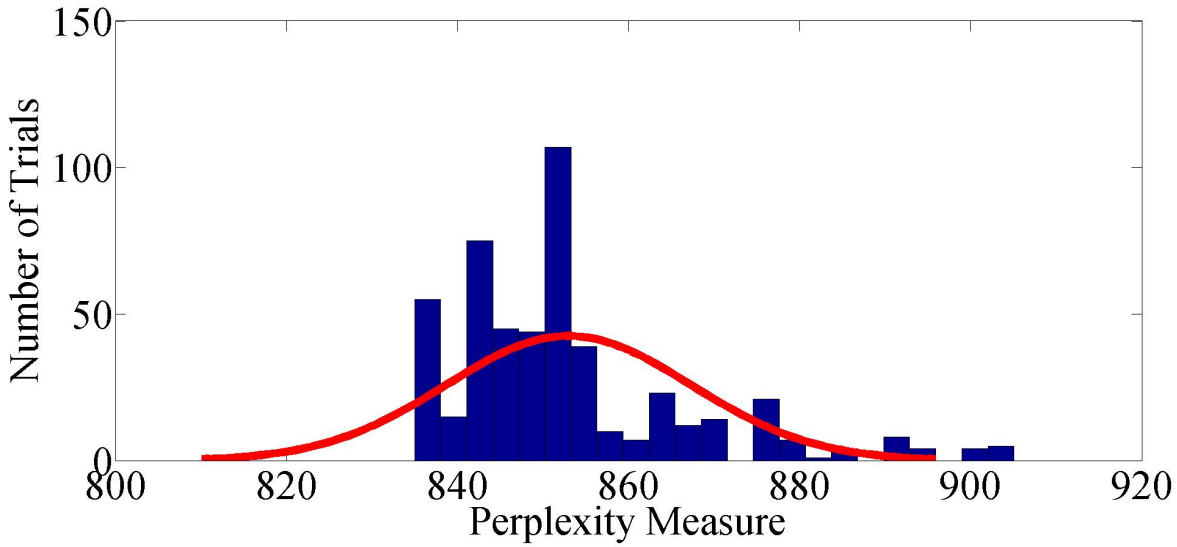

Figure 4.15: A histogram of reached perplexity measures of the corpus of small set of characters of Othello in two fold classification with LDA. The red curve is a Gaussian fit to the data.

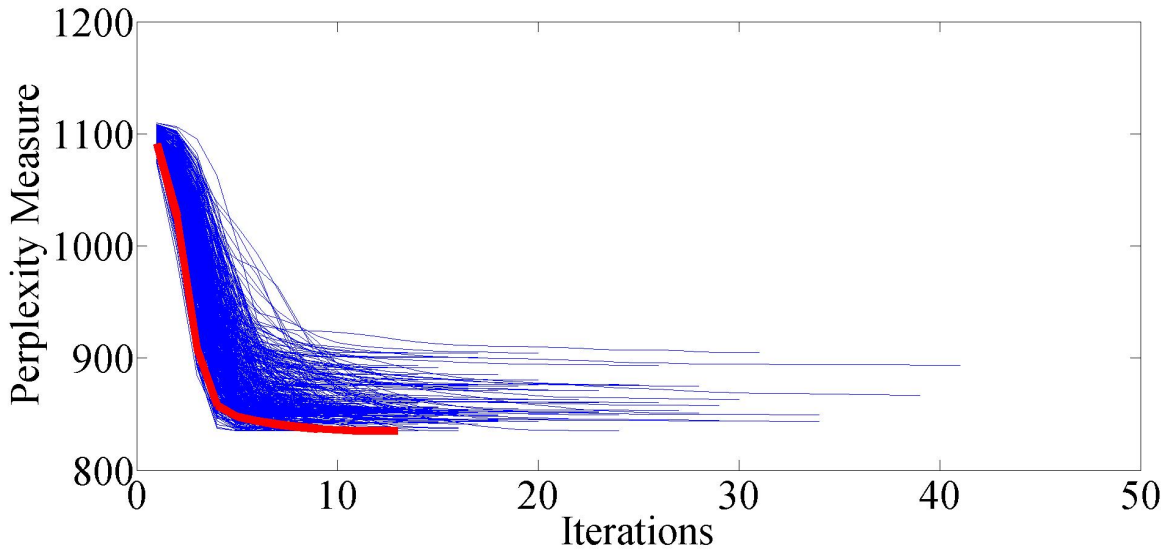

Figure 4.16: The perplexity evolution curves of the corpus small set of characters of Othello in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.8: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of small set of characters in the play Othello with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'BRABANTIO'	0.00005	0.99995
'CASSIO'	0.00003	0.99997
'DESDEMONA'	0.99997	0.00003
'EMILIA'	0.99996	0.00004
'IAGO'	0.00001	0.99999
'OTHELLO'	0.99999	0.00001
'RODERIGO'	0.00007	0.99993

Our Tables 4.7 and 4.8 are similar for the main characters.

4.6. A 2-Fold Analysis of The Play Macbeth

4.6.1. All Characters

Aim of this section is to find out which characters of the play Macbeth are close to each other. In order to do that we make 2-fold categorization of all characters in the play. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.17 and Figure 4.18, respectively.

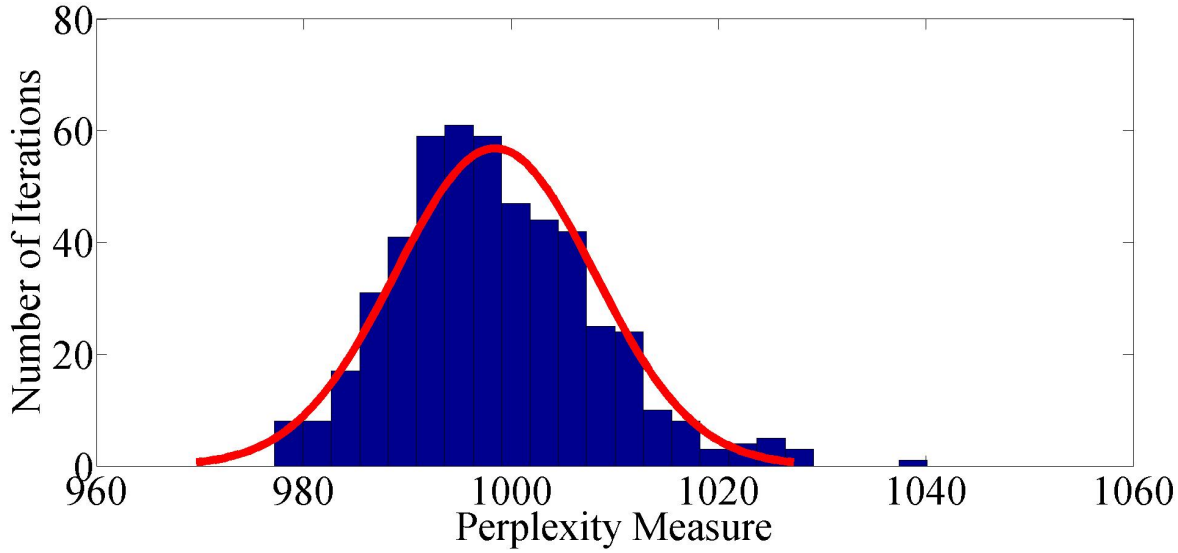

Figure 4.17: A histogram of reached perplexity measures of the corpus of Macbeth in two fold classification with LDA. The red curve is a Gaussian fit to the data.

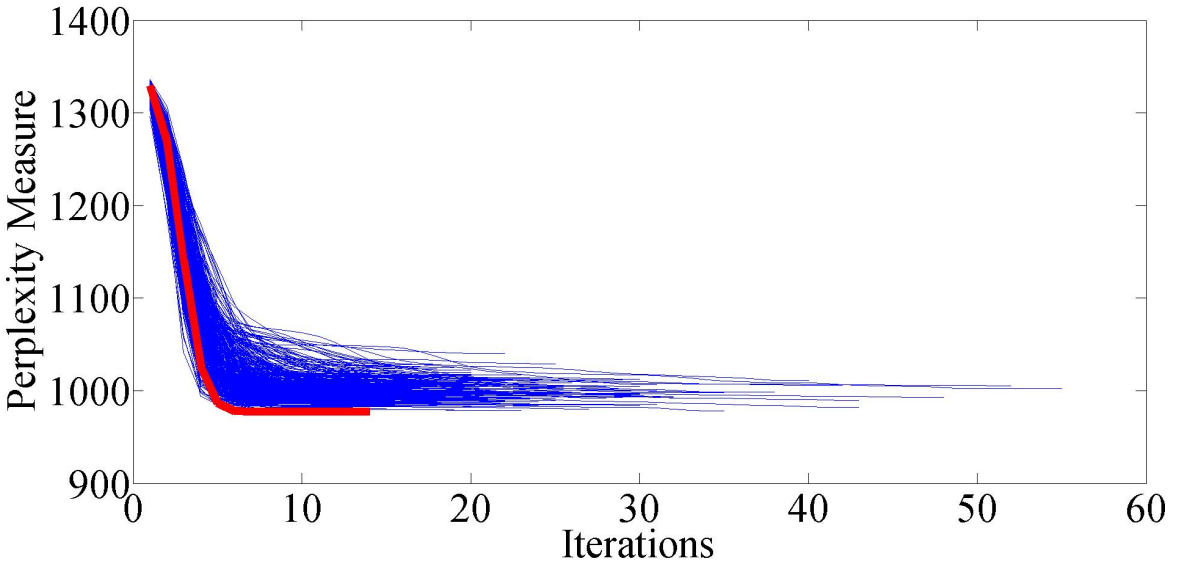

Figure 4.18: The perplexity evolution curves of the corpus Macbeth in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.9: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of all characters in the play Macbeth with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'ANGUS'	0.00037	0.99963
'ATTENDANT'	0.00842	0.99158
'BANQUO'	0.00007	0.99993
'Both Murderers'	0.00634	0.99366
'CAITHNESS'	0.99902	0.00098
'Doctor'	0.99976	0.00024
'DONALBAIN'	0.99874	0.00126
'DUNCAN'	0.99984	0.00016
'First Apparition'	0.99649	0.00351
'First Murderer'	0.00029	0.99971
'First Witch'	0.99983	0.00017
FLEANCE'	0.99611	0.00389
'Gentlewoman'	0.00032	0.99968
'HECATE'	0.99971	0.00029
'LADY MACBETH'	0.99996	0.00004
'LADY MACDUFF'	0.00017	0.99983
'LENNOX'	0.00010	0.99990
'Lord'	0.00038	0.99962
'Lords'	0.00320	0.99680
'MACBETH'	0.99999	0.00001
'MACDUFF'	0.00004	0.99996
'MALCOLM'	0.00004	0.99996
'MENTEITH'	0.00070	0.99930
'Messenger'	0.00036	0.99964
'Old Man'	0.99923	0.00077
'Porter'	0.99975	0.00025
'ROSS'	0.00006	0.99994
'Second Apparition'	0.99793	0.00207
'Second Murderer'	0.00073	0.99927
'Second Witch'	0.99958	0.00042

Documents (d) \ Categories (z)	z_1	z_2
'Sergeant'	0.00020	0.99980
'Servant'	0.00256	0.99744
'SEYTON'	0.99749	0.00251
'SIWARD'	0.00029	0.99971
'Soldiers'	0.01250	0.98750
'Son'	0.00038	0.99962
'Third Apparition'	0.39752	0.60248
'Third Murderer'	0.00103	0.99897
'Third Witch'	0.99960	0.00040
'YOUNG SIWARD'	0.00089	0.99911

Our LDA approach to the play *Macbeth* successfully distinguished the antagonistic relationships. Witches, Hecate, Lady Macbeth, Macbeth and Seyton are classified together. King Duncan is in this group too. He is mischievously killed one night by the Macbeths and the King had no conflicts with the characters in the group. Being a victim does not necessarily throw him to the other group. On the other side, there are Banquo, Lady Macduff, Lennox, Lords, Macduff, Malcolm, Menteith, Ross, Son, Siward and Young Siward. Both of these characters makes war on Macbeth or they are killed by Macbeth. Within this setting arguably the first, second and third murderer are misclassified as belonging to the second group.

4.6.2. Small Group of Characters

In order to continue our investigation on the play, we make 2-fold categorization of a small subset of dramatis personae. 800 words are chosen as a limit, since there is a gap between BANQUO and LENNOX as shown in Table A.5. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.19 and Figure 4.20, respectively.

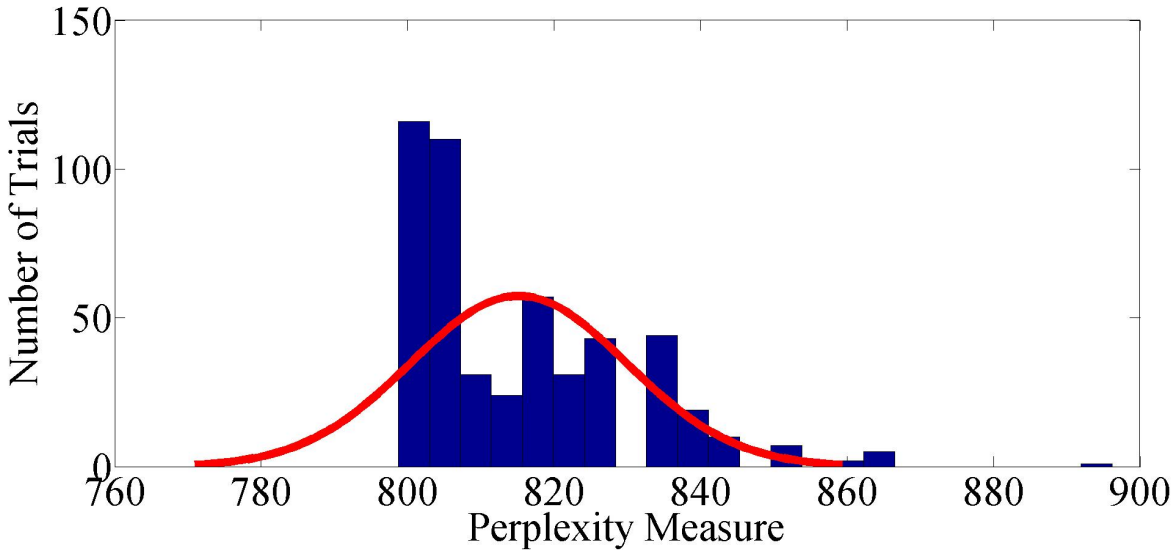

Figure 4.19: A histogram of reached perplexity measures of the corpus of small set of characters of Macbeth in two fold classification with LDA. The red curve is a Gaussian fit to the data.

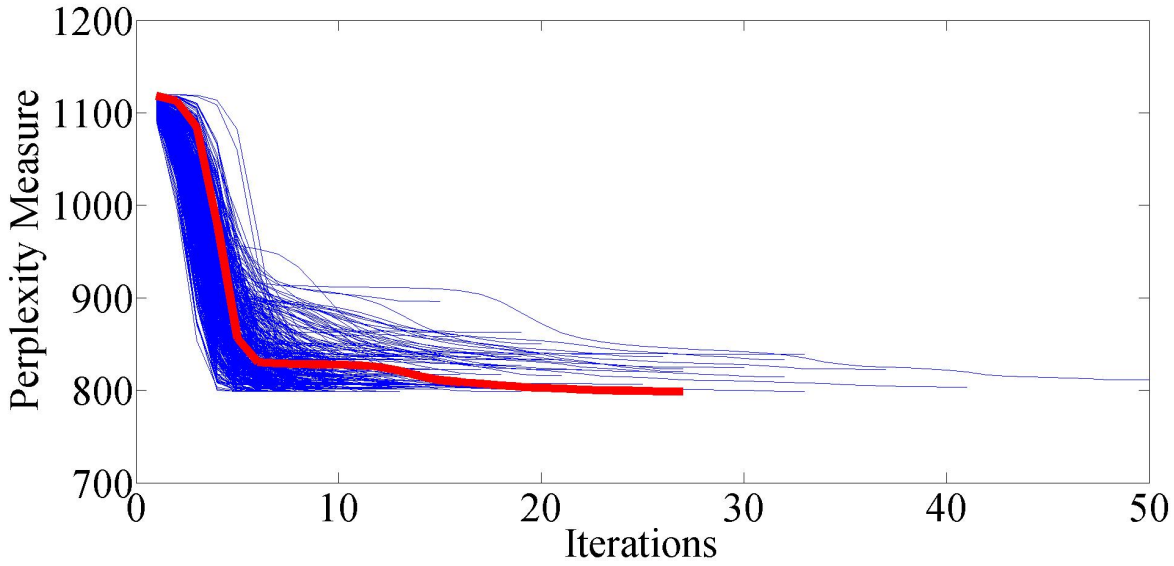

Figure 4.20: The perplexity evolution curves of the corpus Macbeth in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.10: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of small set of characters in the play Macbeth with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'BANQUO'	0.00004	0.99996
'LADY MACBETH'	0.99996	0.00004
'MACBETH'	0.99999	0.00001
'MACDUFF'	0.00003	0.99997
'MALCOLM'	0.00002	0.99998
'ROSS'	0.00003	0.99997

With the small set of characters the classification again follows the antagonistic relations in the play. Macbeth and Lady Macbeth are in conflict with the rest of the characters in the small set.

4.7. A 2-Fold Analysis of The Play III. Richard

4.7.1. All Characters

Aim of this section is to find out which characters of the play III.Richard are close to each other. In order to do that we make 2-fold categorization of all characters in the play. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.17 and Figure 4.18, respectively.

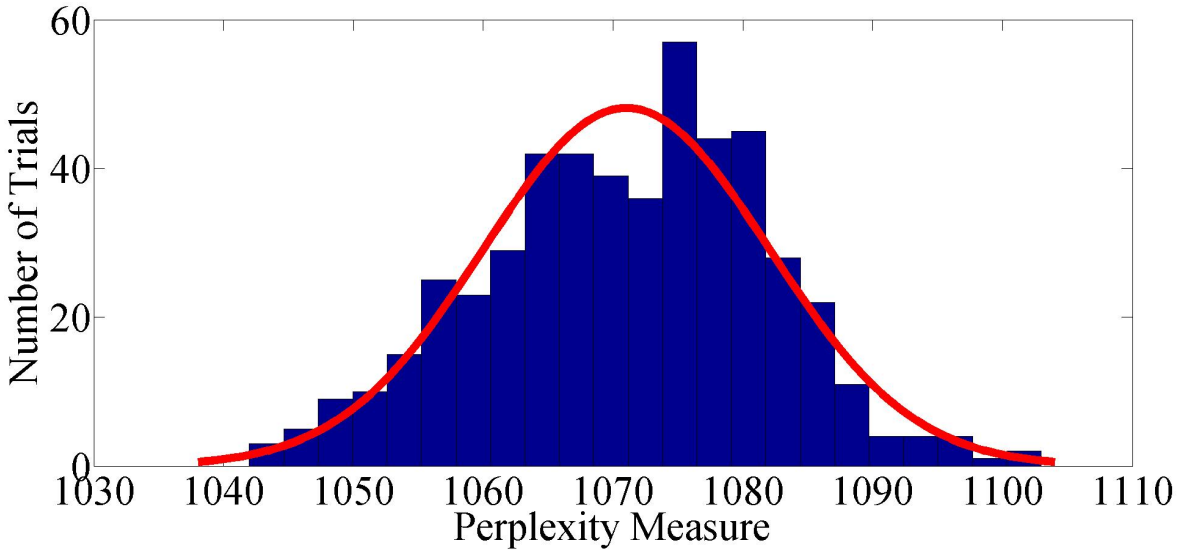

Figure 4.21: A histogram of reached perplexity measures of the corpus of III.Richard in two fold classification with LDA. The red curve is a Gaussian fit to the data.

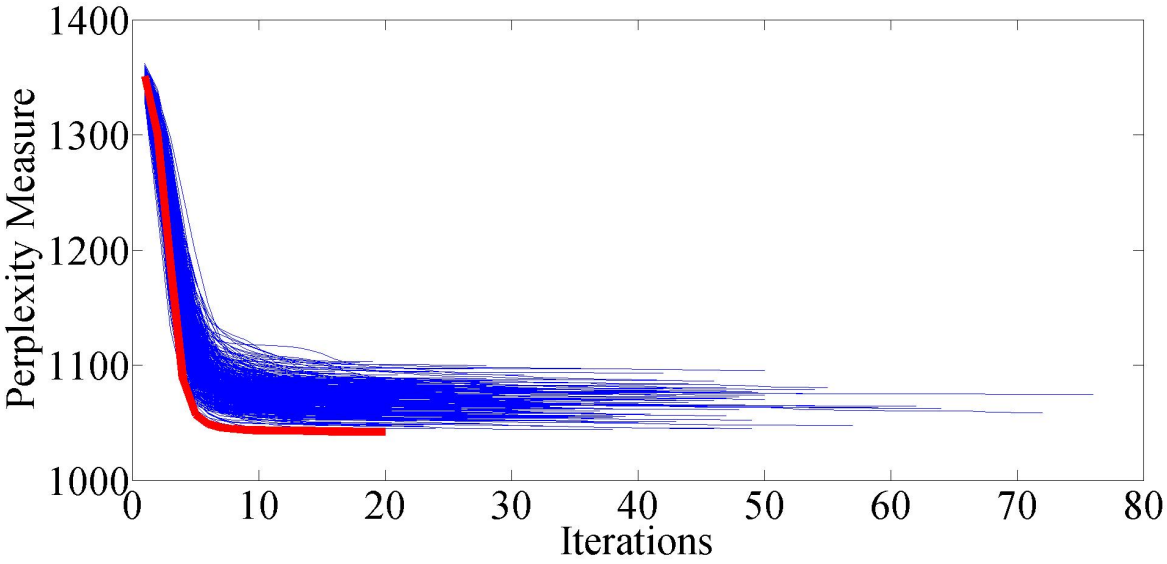

Figure 4.22: The perplexity evolution curves of the corpus III.Richard in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.11: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of all characters in the play III.Richard with LDA. Highest probabilities in a row are shown with a grey background. Continues in the next page.

Documents (d) \ Categories (z)	z_1	z_2
'ANOTHER'	0.00663	0.99337
'ARCHBISHOP OF YORK'	0.66637	0.33363
'BISHOP OF ELY'	0.00090	0.99910
'BLUNT'	0.00096	0.99904
'Both'	0.00878	0.99122
'Boy'	0.99938	0.00062
'BRAKENBURY'	0.99969	0.00031
'BUCKINGHAM'	0.00002	0.99998
'CARDINAL'	0.00087	0.99913
'CATESBY'	0.00013	0.99987
'Children'	0.99689	0.00311
'CHRISTOPHER'	0.00075	0.99925
'Citizens'	0.01302	0.98698
'CLARENCE'	0.99993	0.00007
'DERBY'	0.00012	0.99988
'DORSET'	0.00034	0.99966
'DUCHESS OF YORK'	0.99993	0.00007
'First Citizen'	0.00071	0.99929
'First Murderer'	0.99979	0.00021
'Fourth Messenger'	0.00069	0.99931
'Gentleman'	0.00663	0.99337
'GENTLEMEN'	0.98575	0.01425
'Ghost of CLARENCE'	0.99896	0.00104
'Ghost of GREY'	0.99381	0.00619
'Ghost of HASTINGS'	0.99832	0.00168
'Ghost of LADY ANNE'	0.99898	0.00102
'Ghost of RIVERS'	0.51281	0.48719
'Ghost of VAUGHAN'	0.00244	0.99756
'Girl'	0.99771	0.00229
'GLOUCESTER'	0.99998	0.00002
'GREY'	0.89910	0.10090

Documents (d) \ Categories (z)	z_1	z_2
'HASTINGS'	0.00005	0.99995
'HERBERT'	0.00663	0.99337
'KING EDWARD IV'	0.17112	0.82888
'KING RICHARD III'	0.00001	0.99999
'LADY ANNE'	0.99994	0.00006
'Lord Mayor'	0.00040	0.99960
'LORD STANLEY'	0.00048	0.99952
'LORDS'	0.00334	0.99666
'LOVEL'	0.00192	0.99808
'Messenger'	0.00029	0.99971
'NORFOLK'	0.00077	0.99923
'of BUCKINGHAM'	0.72736	0.27264
'of King Henry VI'	0.99872	0.00128
'of Prince Edward'	0.99850	0.00150
'of young Princes'	0.99907	0.00093
'OXFORD'	0.00334	0.99666
'Page'	0.00158	0.99842
'Priest'	0.00878	0.99122
'PRINCE EDWARD'	0.28880	0.71120
'Pursuivant'	0.00192	0.99808
'QUEEN ELIZABETH'	0.99996	0.00004
'QUEEN MARGARET'	0.99995	0.00005
'RATCLIFF'	0.00026	0.99974
'RICHMOND'	0.00005	0.99995
'RIVERS'	0.00014	0.99986
'Scrivener'	0.00054	0.99946
'Second Citizen'	0.00051	0.99949
'Second Messenger'	0.00244	0.99756
'Second Murderer'	0.99980	0.00020
'Sheriff'	0.98925	0.01075
'STANLEY'	0.00025	0.99975
'SURREY'	0.00532	0.99468
'Third Citizen'	0.00024	0.99976
'Third Messenger'	0.00112	0.99888
'TYRREL'	0.00021	0.99979
'VAUGHAN'	0.00663	0.99337
'YORK'	0.99971	0.00029

It is hard to say that the classification of the dramatis personae in *III.Richard* reproduces successfully their antagonistic relationships. Gloucester is the name of Richard before he becomes a king. They are in the opposite groups. Obviously, Richard changes the way he uses the language. Duke of Clarence is killed by two murderers with the command of Gloucester. However, Clarence, first and second murderers and Gloucester are classified together. Queen Margaret condemns Gloucester and the rest of the characters and she is obviously against him in the play. She is also classified with Gloucester. The relationship between Gloucester (King Richard III) and Lady Anne is complex so it is hard assess whether they are antagonists or not. Buckingham supports Gloucester and yet they are classified in opposite groups. Lord Hastings is killed with the command of Gloucester and they are in the opposite groups which is correct. King Edward dies in the middle of the play and had no conflict with the characters. However, Prince Edward is killed and mistakenly grouped with King Richard. In addition to that Buckingham is in riot against King Richard but classified in the same group with him. Richmond kills King Richard yet is grouped with him.

4.7.2. Small Group of Characters

In order to continue our investigation on the play, we make 2-fold categorization of a small set of dramatis personae. 1000 words are chosen as a limit, since there is a gap between RICHMOND and KING EDWARD IV as shown in Table A.6. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.23 and Figure 4.24, respectively.

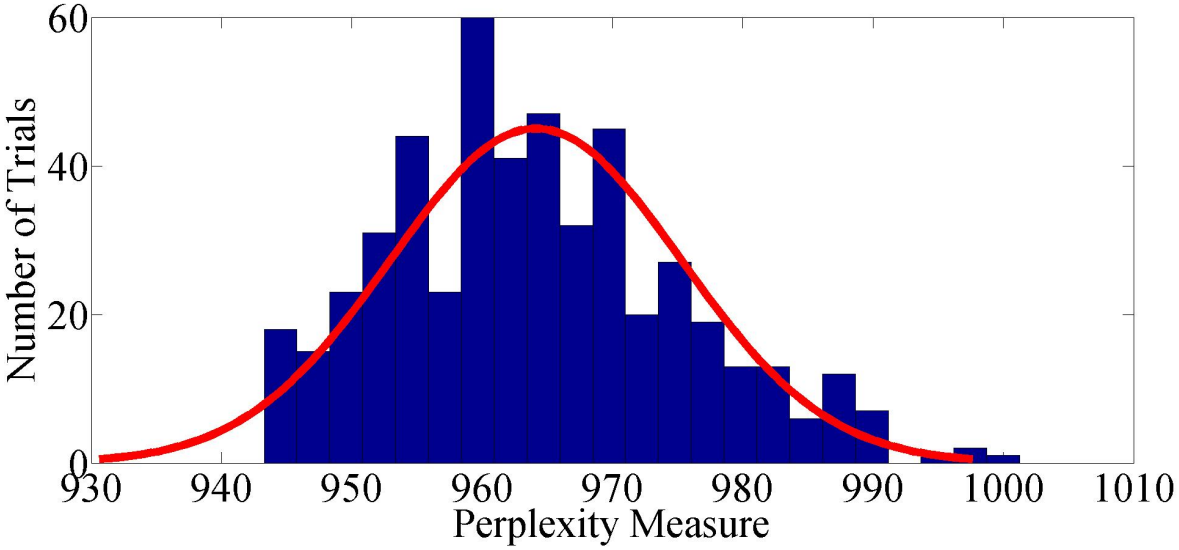

Figure 4.23: A histogram of reached perplexity measures of the corpus of small set of characters of III.Richard in two fold classification with LDA. The red curve is a Gaussian fit to the data.

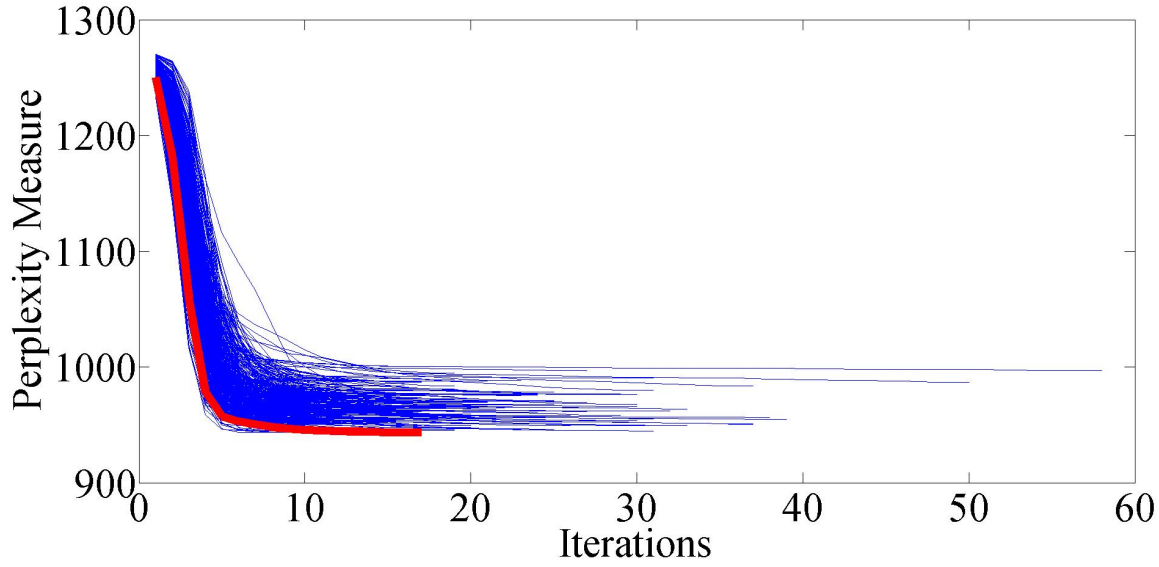

Figure 4.24: The perplexity evolution curves of the corpus small set of characters of III.Richard in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.12: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of small set of characters in the play III.Richard with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
'BUCKINGHAM'	0.99998	0.00002
'CLARENCE'	0.99996	0.00004
'DUCHESS OF YORK'	0.00002	0.99998
'GLOUCESTER'	0.99999	0.00001
'HASTINGS'	0.99996	0.00004
'KING RICHARD III'	0.00001	0.99999
'LADY ANNE'	0.00002	0.99998
'QUEEN ELIZABETH'	0.00001	0.99999
'QUEEN MARGARET'	0.00002	0.99998
'RICHMOND'	0.00003	0.99997

4.8. A 2-Fold Analysis of all Main Characters in the Four Plays

In this section we perform a 2-fold classification of the main characters across the four plays we considered in the previous sections, Hamlet, Othello, Macbeth and III.Richard. In order to do that we make 2-fold categorization of all characters in the plays. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.25 and Figure 4.26, respectively.

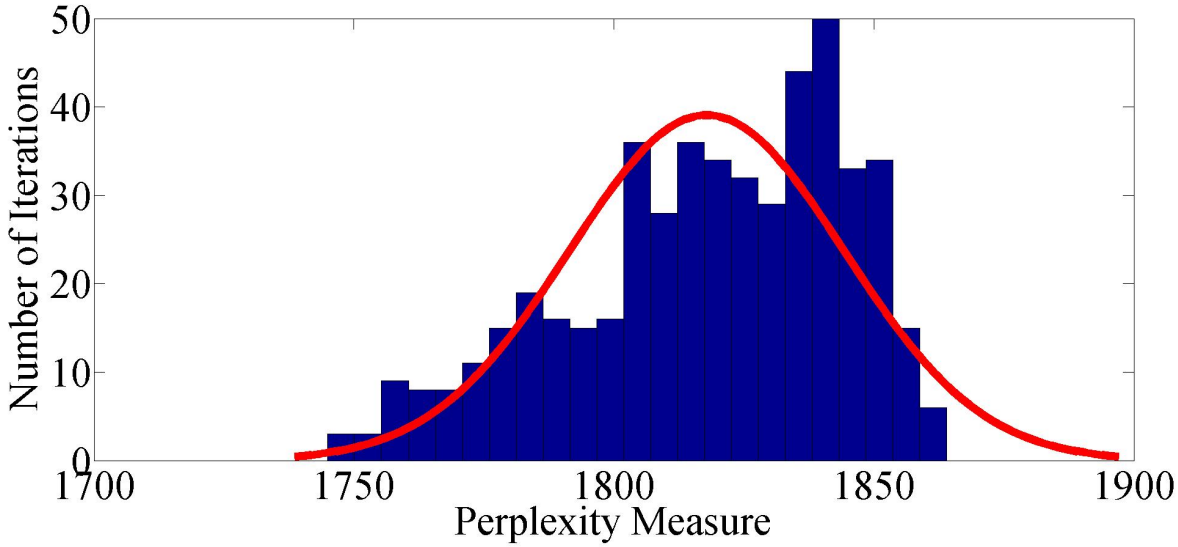

Figure 4.25: A histogram of reached perplexity measures of the corpus of mixture of major characters of the four plays in two fold classification with LDA. The red curve is a Gaussian fit to the data.

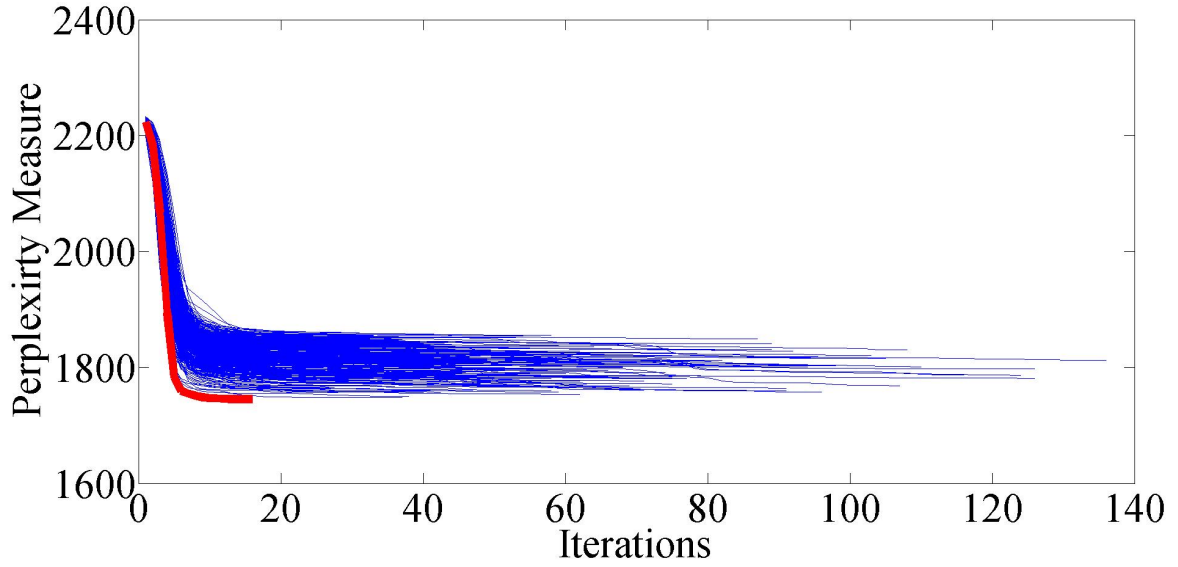

Figure 4.26: The evolution curves of the corpus mixture of major characters of the four plays in two fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.13: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of two fold classification of mixture of major characters in the four plays with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2
Hamlet-HAMLET	0.99999	0.00001
Hamlet-HORATIO	0.99997	0.00003
Hamlet-KING CLAUDIUS	0.99999	0.00001
Hamlet-LAERTES	0.99997	0.00003
Hamlet-LORD POLONIUS	0.99998	0.00002
Hamlet-OPHELIA	0.99996	0.00004
Hamlet-QUEEN GERTRUDE	0.99996	0.00004
Macbeth-BANQUO	0.00005	0.99995
Macbeth-LADY MACBETH	0.00002	0.99998
Macbeth-MACBETH	0.00001	0.99999
Macbeth-MACDUFF	0.00003	0.99997
Macbeth-MALCOLM	0.00002	0.99998
Macbeth-ROSS	0.00004	0.99996
Othello-BRABANTIO	0.97363	0.02637
Othello-CASSIO	0.99998	0.00002
Othello-DESDEMONA	0.99998	0.00002
Othello-EMILIA	0.99998	0.00002
Othello-IAGO	0.99999	0.00001
Othello-OTHELLO	0.99999	0.00001
Othello-RODERIGO	0.99994	0.00006
Richard III-BUCKINGHAM	0.00001	0.99999
Richard III-CLARENCE	0.00003	0.99997
Richard III-DUCHESS OF YORK	0.00003	0.99997
Richard III-GLOUCESTER	0.00001	0.99999
Richard III-HASTINGS	0.00003	0.99997
Richard III-KING RICHARD III	0.00001	0.99999
Richard III-LADY ANNE	0.00003	0.99997
Richard III-QUEEN ELIZABETH	0.00002	0.99998
Richard III-QUEEN MARGARET	0.00002	0.99998
Richard III-RICHMOND	0.00003	0.99997

4.9. A 3-Fold Analysis of Mixture of The Basic Characters in Four Plays

We next investigate the 3-fold classification. Aim of this section is to find out which basic characters of the play Hamlet, Othello, Macbeth and III.Richard are close to each other. In order to do that we make 3-fold categorization of all characters in the plays. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.27 and Figure 4.28, respectively.

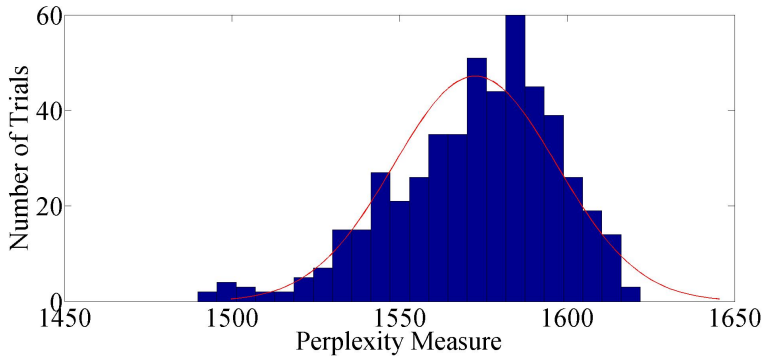

Figure 4.27: A histogram of reached perplexity measures of the corpus of mixture of major characters of the four plays in three fold classification with LDA. The red curve is a Gaussian fit to the data.

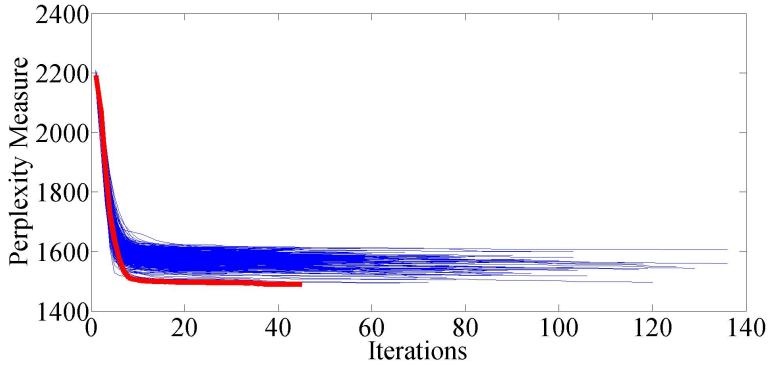

Figure 4.28: The evolution curves of the corpus mixture of major characters of the four plays in three fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

Table 4.14: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of three fold classification of mixture of major characters in the four plays with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2	z_3
Hamlet-HAMLET	0.00000	1, 00000	0.00000
Hamlet-HORATIO	0.00001	0.99998	0.00001
Hamlet-KING CLAUDIUS	0.00000	0.99999	0.00001
Hamlet-LAERTES	0.00001	0.99997	0.00001
Hamlet-LORD POLONIUS	0.99998	0.00001	0.00001
Hamlet-OPHELIA	0.00002	0.99997	0.00002
Hamlet-QUEEN GERTRUDE	0.00001	0.42023	0.57975
Macbeth-BANQUO	0.42929	0.57069	0.00002
Macbeth-LADY MACBETH	0.00001	0.99998	0.00001
Macbeth-MACBETH	0.00000	0.99999	0.00000
Macbeth-MACDUFF	0.00001	0.99997	0.00001
Macbeth-MALCOLM	0.00001	0.99998	0.00001
Macbeth-ROSS	0.00002	0.99997	0.00002
Othello-BRABANTIO	0.99996	0.00002	0.00002
Othello-CASSIO	0.86867	0.13132	0.00001
Othello-DESDEMONA	0.99998	0.00001	0.00001
Othello-EMILIA	0.99998	0.00001	0.00001
Othello-IAGO	0.99999	0.00000	0.00000
Othello-OTHELLO	0.99999	0.00000	0.00000
Othello-RODERIGO	0.99995	0.00003	0.00002
Richard III-BUCKINGHAM	0.00001	0.00001	0.99999
Richard III-CLARENCE	0.00001	0.00002	0.99997
Richard III-DUCHESS OF YORK	0.00001	0.00002	0.99997
Richard III-GLOUCESTER	0.00000	0.00000	0.99999
Richard III-HASTINGS	0.00001	0.00002	0.99997
Richard III-KING RICHARD III	0.00000	0.00001	0.99999
Richard III-LADY ANNE	0.00001	0.00002	0.99997
Richard III-QUEEN ELIZABETH	0.00001	0.00001	0.99998
Richard III-QUEEN MARGARET	0.00001	0.00001	0.99998
Richard III-RICHMOND	0.00001	0.00002	0.99997

4.10. A 4-Fold Analysis of Mixture of The Basic Characters in Four Plays

We next investigate a 4-fold classification. A histogram of the reached perplexity measures and perplexity development curve are graphed in Figure 4.29 and Figure 4.30, respectively.

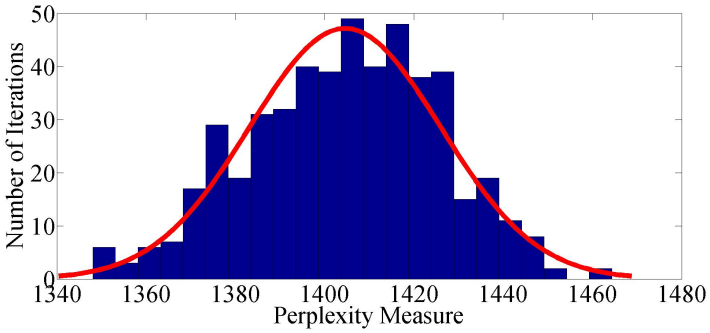

Figure 4.29: A histogram of reached perplexity measures of the corpus of mixture of major characters of the four plays in four fold classification with LDA. The red curve is a Gaussian fit to the data.

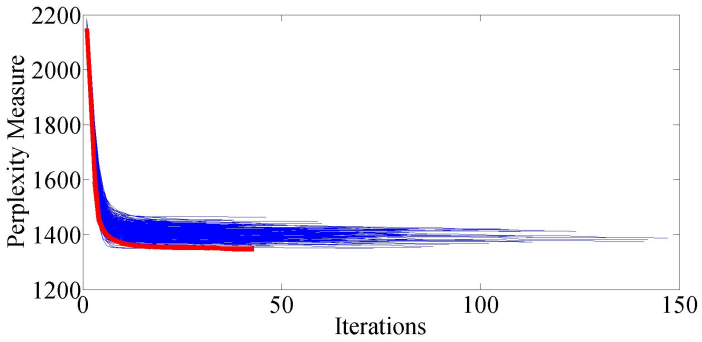

Figure 4.30: The evolution curves of the corpus mixture of major characters of the four plays in four fold classification with LDA. There are 500 trials. Minimum reached perplexity curve is indicated with a red line.

In the Table 4.15, Richard III seems perfectly grouped together with its characters. However Macduff, Desdemone and Roderigo are grouped with Hamlet's characters and Malcolm of Macbeth is with Othello's characters. Brabantio, Queen Gertrudeand King Claudius are grouped with Macbeth.

Table 4.15: Corresponding $p(z \mid d)$ to the lowest achieved perplexity value of four fold classification of mixture of major characters in the four plays with LDA. Highest probabilities in a row are shown with a grey background.

Documents (d) \ Categories (z)	z_1	z_2	z_3	z_4
Hamlet-HAMLET	0.00000	0.00000	0.99999	0.00000
Hamlet-HORATIO	0.00002	0.00001	0.99996	0.00002
Hamlet-KING CLAUDIUS	0.99998	0.00001	0.00001	0.00001
Hamlet-LAERTES	0.00002	0.00001	0.99995	0.00002
Hamlet-LORD POLONIUS	0.00001	0.00001	0.99997	0.00001
Hamlet-OPHELIA	0.00003	0.00002	0.99993	0.00003
Hamlet-QUEEN GERTRUDE	0.99993	0.00002	0.00002	0.00003
Macbeth-BANQUO	0.99991	0.00002	0.00003	0.00003
Macbeth-LADY MACBETH	0.99997	0.00001	0.00001	0.00001
Macbeth-MACBETH	0.99999	0.00000	0.00000	0.00000
Macbeth-MACDUFF	0.07637	0.00001	0.92359	0.00002
Macbeth-MALCOLM	0.09049	0.90948	0.00002	0.00002
Macbeth-ROSS	0.99993	0.00002	0.00002	0.00003
Othello-BRABANTIO	0.99993	0.00002	0.00002	0.00003
Othello-CASSIO	0.00001	0.99996	0.00001	0.00001
Othello-DESDEMONA	0.00001	0.15659	0.84339	0.00001
Othello-EMILIA	0.00001	0.99996	0.00001	0.00001
Othello-IAGO	0.00000	0.99999	0.00000	0.00000
Othello-OTHELLO	0.00000	0.99999	0.00000	0.00000
Othello-RODERIGO	0.00003	0.08273	0.91721	0.00003
Richard III-BUCKINGHAM	0.00001	0.00001	0.00001	0.99997
Richard III-CLARENCE	0.00002	0.00001	0.00002	0.99995
Richard III-DUCHESS OF YORK	0.00002	0.00001	0.00002	0.99995
Richard III-GLOUCESTER	0.00000	0.00000	0.00000	0.99999
Richard III-HASTINGS	0.00002	0.00002	0.38383	0.61614
Richard III-KING RICHARD III	0.00001	0.00000	0.00001	0.99998
Richard III-LADY ANNE	0.00002	0.00001	0.00002	0.99995
Richard III-QUEEN ELIZABETH	0.00001	0.00001	0.00001	0.99997
Richard III-QUEEN MARGARET	0.08540	0.00001	0.00001	0.91458
Richard III-RICHMOND	0.00002	0.00002	0.00002	0.99994

4.11. Results and Discussion

In this chapter, we performed LDA classifications on the corpus of plays by Shakespeare. We then considered the lines spoken by individual characters of the play as documents and performed LDA classifications of these character/documents within the plays *Hamlet*, *Othello*, *Macbeth* and *Richard III*, as well as across these four plays.

For the 3-fold analysis of all plays, we find that the Historical plays are a rigid category. Additionally, the play *Hamlet* is also grouped with the Historical plays. This could be, since Hamlet after all is a liberal adaptation of a historical event. On the other hand, in the literature we find that *Othello* and *Romeo and Juliet* are grouped with the Comedies, while some of the Comedies are grouped with Tragedies. These two plays are discussed as Comedies ending with tragic events,[12], [13]. The category of Comedy includes Romances [11]. The Romances are four: *Cymbeline*, *Winter's Tale*, *The Tempest* and *Pericles*. These are the four last pieces of Shakespeare. This perfectly agrees with our results of three-fold categorization. These four plays are grouped with Tragedies. Romances are closer to Tragedies than Comedies.

We then turned to the 2-fold classification of all plays. According to our results, Tragedy seems to be an inter-category, while History and Comedy are major categories. *Anthony and Cleopatra*, *Hamlet*, *King Lear*, *Othello*, *Romeo and Juliet* and *Timon of Athens* are grouped with Comedies. Except Romeo and Juliet, all the plays mentioned are based on historical events. Thus the LDA method, or more precisely the corpus of these plays is not sensitive to whether the plays have a historical context or not.

For the 2-fold "character" analysis for each of the four plays, we can say that mostly the aim of recognizing the main conflicts is achieved except in *III.Richard*. We found that antagonistic relations are more difficult to detect by LDA when the roles are minor and thus their texts contain less words. This is in accordance with our expectation that main characters use a more elaborate language and also that the author spends more time and care developing these characters in the play, so that they might indeed have a distinct corpus.

Finally, we considered the classfication of all major characters across the four plays. In the 2-fold category, *Hamlet* and *Othello* are in one group while *Macbeth* and *III.Richard* are in the other group.

5. CONCLUSIONS

In this thesis, we implemented two different document classification techniques, Probabilistic Latent Semantic Analysis (pLSA) and Latent Dirichlet Allocation (LDA), to analyze literary texts. Both techniques are based on the frequency distribution of words occuring in the documents. Documents are classified based on the similarity or dissimilarity of their respective word count distributions. We considered two applications of these techniques to literary texts. Assuming that different authors use a different vocabulary to write their works, we considered a corpus of six novellas by three authors and asked whether we could classify these texts according to authorship, using the pLSA and LDA techniques. We found that a 3-fold classification of these novellas provides indeed a grouping by authorship. Our implementation of LDA was about 3 times faster then our pLSA implementation, so we continued our analysis using the LDA technique.

We next applied LDA to the plays of W. Shakespeare. Generally the plays of Shakespeare are classified as Historical, Comedy and Tragedy. We first asked to what extent LDA reproduces this classification. Applying 2- and 3-fold classification of all Shakespeare plays we found that all History plays fall into one of the classes. Interestingly, the LDA technique does not seem to be able to distinguish Comedy from Tragedy. We found that in the case of a two-fold classification, one of the classes contains all history plays while the other contains the comedies. The tragedies however were grouped as belonging to one or the other class. In the case of a three-fold classification however, the history plays again formed a single class, while another class contained most of the comedies and again, the tragedies where mostly in their own class, with a few classified either along with the history plays or the comedies. We conclude that the choice of vocabulary does indeed single out the history plays from all other plays of Shakespeare. However this vocabulary does not seem to be sufficiently strong to distinguish the comedies from the tragedies. As the list of words most closely associated with a given topic show, these words are rather generic and by themselves do not suggest whether they might have come form a comedy, tragedy or a history play.

We next asked whether the choice of words is correlated with features of characters in a play. To this end, we broke down a play into its characters. Here each character corresponds to a document, containing all the lines spoken by that character in the play. LDA is used to classify the characters. The plays we considered were *Hamlet*, *Othello*, *Macbeth* and *Richard III*. We considered two-, three- and four-fold classifications of these players with the full *dramatis personae* and smaller subsets containing the major characters. We find that particularly when using the smaller subset of characters, a two-fold classifications is mostly along the line of antagonistic relations between the characters, except for *III.Richard*.

Lastly, we performed a classification of all the major characters across the four plays. In the 2-fold category, the characters of *Hamlet* and *Othello* are classified in one group, while those of *Macbeth* and *III.Richard* are in the other group. For the 3-fold categorization, we see the traces of that observation, but there are some deviations.

To conclude, applying LDA and pLSA to literary texts, we have shown that these approaches can successfully distinguish genres and have the potential to uncover relationships among characters in the plays. Although the results of the classification of the characters in the plays are not always in accordance with the expectation based on the reading of the plays, these classifications are meaningful. Therefore, literary texts can be profitably investigated through statistical inference methods and can assist in gaining alternative perceptions of these.

REFERENCES

1. Hoover, D. L., "Quantitative Analysis and Literary Studies", *A Companion to Digital Literary Studies, ed. Susan Schreibman and Ray Siemens. Oxford: Blackwell*, 2008. `http://www.digitalhumanities.org/companionDLS/`, accessed at December 2015.

2. Hoover, D. L., J. Culpeper, K. O'Halloran. *Digital Literary Studies: Corpus Approaches to Poetry, Prose, and Drama*, Routledge, New York, NY, 2014.

3. Feller, William. *An Introduction to Probability Theory and Its Applications.* New York, NY, Wiley, 3rd ed., 1968.

4. Konstantinos G. D., *Jensen's Inequality*, `http://www.cse.yorku.ca/~kosta/CompVis_Notes/jensen.pdf`, accessed at December 2015.

5. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer-Verlag, New York, NY, 2006.

6. Hofmann, T. (1999). "Probabilistic Latent Semantic Indexing", *Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval, (SIGIR-99)*, 1999.

7. Dempster, A. P., N. M. Laird, D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *J. Royal Statist. Soc. B 39*, pp. 1-38, 1977.

8. Blei, et al. "Latent Dirichlet Allocation", *J. Mach. Learn. Res.*, 2003, pp. 993-1022, 2003.

9. de Finetti, B., *Theory of Probability*, John Wiley Sons Ltd., Chichester, 1990.

10. Shakespeare, W., *Mr. William Shakespeare's Comedies, Histories and Tragedies, published according to the true original copies*, William Jaggard, Edward Blount,

L. Smithweeke, and W. Aspley, London, UK, 1623.

11. Spens, J., *Elizabethan Drama*, Metheun Co., London, 1922.

12. Snyder, S., "Romeo and Juliet: Comedy into Tragedy", *Essays in Criticism*, XX: 391-402, 1970.

13. Rogers, S., "Othello: Comedy in Reverse" *Shakespeare Quarterly*, Vol.24, No.2, pp. 210-220, Spring 1973.

14. Birch, D., *The Oxford Companion to English Literature*, Oxford University Press, Oxford, UK, 2009.

APPENDIX A: WORD-COUNT TABLES

Table A.1: Word-Count table of six novellas

CONRAD Heart of Darkness:	Words: 38109
CONRAD The Shadow Line:	Words: 39764
HEMINGWAY The Old Man and the Sea:	Words: 26583
HEMINGWAY The Snows of Kilimanjaro:	Words: 9164
WELLS The Island of Doctor Moreau:	Words: 43399
WELLS The Time Machine:	Words: 32296

Table A.2: Word-Count table of All Plays

Comedy-All's Well That Ends Well:	Words: 22548
Comedy-As You Like It:	Words: 21303
Comedy-Comedy of Errors:	Words: 14518
Comedy-Cymbeline:	Words: 26695
Comedy-Love's Labour's Lost:	Words: 21058
Comedy-Measure for Measure:	Words: 21237
Comedy-Merchant of Venice:	Words: 20920
Comedy-Merry Wives of Windsor:	Words: 21049
Comedy-Midsummer Night's Dream:	Words: 16058
Comedy-Much Ado About Nothing:	Words: 20765
Comedy-Pericles:	Words: 17460
Comedy-Taming of the Shrew:	Words: 20414
Comedy-The Tempest:	Words: 16004
Comedy-Troiles and Cressida:	Words: 25240
Comedy-Twelfth Night:	Words: 19423
Comedy-Two Gentlemen of Verona:	Words: 16873
Comedy-Winter's Tale:	Words: 24504

Table A.2: Word-Count table of All Plays (cont.)

History-Henry IV, part 1:	Words: 23923
History-Henry IV, part 2:	Words: 25837
History-Henry VIII:	Words: 23199
History-Henry VI, part 1:	Words: 20574
History-Henry VI, part 2:	Words: 24451
History-Henry VI, part 3:	Words: 23299
History-Henry V:	Words: 25832
History-King John:	Words: 20378
History-Richard III:	Words: 28438
History-Richard II:	Words: 21795
Tragedy-Antony and Cleopatra:	Words: 23699
Tragedy-Coriolanus:	Words: 26510
Tragedy-Hamlet:	Words: 29514
Tragedy-Julius Caeser:	Words: 19105
Tragedy-King Lear:	Words: 25267
Tragedy-Macbeth:	Words: 16423
Tragedy-Othello:	Words: 25774
Tragedy-Romeo and Juliet:	Words: 23777
Tragedy-Timon of Athens:	Words: 17447
Tragedy-Titus Andronicus:	Words: 20030

Table A.3: Word-Count table of Hamlet

HAMLET:	Words: 11901	Second Clown:	Words: 121
KING CLAUDIUS:	Words: 4257	First Priest:	Words: 94
LORD POLONIUS:	Words: 2832	Captain:	Words: 91
HORATIO:	Words: 2150	REYNALDO:	Words: 78
LAERTES:	Words: 1501	Lord:	Words: 70
OPHELIA:	Words: 1236	FRANCISCO:	Words: 62
QUEEN GERTRUDE:	Words: 1188	First Sailor:	Words: 43
First Clown:	Words: 800	First Ambassador:	Words: 42
ROSENCRANTZ:	Words: 738	LUCIANUS:	Words: 41
Ghost:	Words: 691	Messenger:	Words: 38
MARCELLUS:	Words: 450	Prologue:	Words: 17
First Player:	Words: 381	Danes:	Words: 16
OSRIC:	Words: 381	Servant:	Words: 10
GUILDENSTERN:	Words: 372		
Player King:	Words: 345		
Player Queen:	Words: 248		
BERNARDO:	Words: 243		
PRINCE FORTINBRAS:	Words: 199		
Gentleman:	Words: 179		
VOLTIMAND:	Words: 161		

Table A.4: Word-Count table of Othello

IAGO:	Words: 8616	First Officer:	Words: 34
OTHELLO:	Words: 6490	First Musician:	Words: 33
DESDEMONA:	Words: 2912	Sailor:	Words: 27
CASSIO:	Words: 2053	First Gentleman:	Words: 21
EMILIA:	Words: 1910	Fourth Gentleman:	Words: 21
BRABANTIO:	Words: 1071	Second Gentlemen:	Words: 13
RODERIGO:	Words: 918	Senator:	Words: 8
LODOVICO:	Words: 562	Gentleman:	Words: 6
DUKE OF VENICE:	Words: 552		
MONTANO:	Words: 438		
BIANCA:	Words: 280		
Clown:	Words: 239		
GRATIANO:	Words: 216		
First Senator:	Words: 196		
Third Gentleman:	Words: 121		
Herald:	Words: 102		
Second Gentleman:	Words: 85		
Messenger:	Words: 64		
Second Senator:	Words: 39		

Table A.5: Word-Count table of Macbeth

MACBETH:	Words: 5435	Third Witch:	Words: 155
LADY MACBETH:	Words: 2009	ANGUS:	Words: 141
MALCOLM:	Words: 1541	Second Murderer:	Words: 98
MACDUFF:	Words: 1215	Old Man:	Words: 89
ROSS:	Words: 944	MENTEITH:	Words: 76
BANQUO:	Words: 807	CAITHNESS:	Words: 76
LENNOX:	Words: 524	DONALBAIN:	Words: 60
DUNCAN:	Words: 490	Third Murderer:	Words: 54
First Witch:	Words: 395	YOUNG SIWARD:	Words: 51
Doctor:	Words: 347	SEYTON:	Words: 37
LADY MACDUFF:	Words: 329	Third Apparition:	Words: 34
Porter:	Words: 309	Second Apparition:	Words: 27
HECATE:	Words: 267	Servant:	Words: 26
Sergeant:	Words: 237	FLEANCE:	Words: 17
SIWARD:	Words: 214	Lords:	Words: 16
First Murderer:	Words: 204	First Apparition:	Words: 15
Gentlewoman:	Words: 192	Both Murderers:	Words: 12
Messenger:	Words: 182	ATTENDANT:	Words: 9
Lord:	Words: 159	Soldiers:	Words: 5
Son:	Words: 157		
Second Witch:	Words: 155		

Table A.6: Word-Count table of III.Richard

GLOUCESTER:	Words: 5558	CARDINAL:	Words: 76
KING RICHARD III:	Words: 3937	NORFOLK:	Words: 76
BUCKINGHAM:	Words: 2822	BLUNT:	Words: 70
QUEEN ELIZABETH:	Words: 2329	of young Princes:	Words: 69
QUEEN MARGARET:	Words: 1813	Ghost of CLARENCE:	Words: 68
CLARENCE:	Words: 1324	BISHOP OF ELY:	Words: 65
LADY ANNE:	Words: 1287	Ghost of LADY ANNE:	Words: 64
DUCHESS OF YORK:	Words: 1220	CHRISTOPHER:	Words: 60
HASTINGS:	Words: 1185	of King Henry VI:	Words: 59
RICHMOND:	Words: 1080	Third Messenger:	Words: 55
KING EDWARD IV:	Words: 545	of Prince Edward:	Words: 49
Second Murderer:	Words: 525	Page:	Words: 42
First Murderer:	Words: 496	Girl:	Words: 41
CATESBY:	Words: 480	Ghost of HASTINGS:	Words: 38
DERBY:	Words: 453	Children:	Words: 29
RIVERS:	Words: 435	Second Messenger:	Words: 25
PRINCE EDWARD:	Words: 383	Pursuivant:	Words: 23
YORK:	Words: 351	LOVEL:	Words: 22
BRAKENBURY:	Words: 322	Ghost of RIVERS:	Words: 20
TYRREL:	Words: 278	Ghost of VAUGHAN:	Words: 18
STANLEY:	Words: 252	LORDS:	Words: 17
Third Citizen:	Words: 247	OXFORD:	Words: 14
Messenger:	Words: 231	Sheriff:	Words: 13
RATCLIFF:	Words: 214	Ghost of GREY:	Words: 12
DORSET:	Words: 162	ANOTHER:	Words: 12
Boy:	Words: 158	Both:	Words: 12
Lord Mayor:	Words: 148	Priest:	Words: 12
LORD STANLEY:	Words: 124	HERBERT:	Words: 11
Scrivener:	Words: 118	SURREY:	Words: 10
Second Citizen:	Words: 115	VAUGHAN:	Words: 10
ARCHBISHOP OF YORK:	Words: 103	Gentleman:	Words: 10
GREY:	Words: 89	GENTLEMEN:	Words: 5
of BUCKINGHAM:	Words: 86	Citizens:	Words: 2
First Citizen:	Words: 79		
Fourth Messenger:	Words: 78		

APPENDIX B: PERL IMPLEMENTATIONS

In this chapter, Perl Implementations will be shown which are coded in order to get the documents from internet and to get the document-word matrix.

B.1. Perl Implementations To Get Documents

```
#!/usr/bin/perl -w
use warnings;
use strict;
use LWP::Simple;

my $flag=0; my $i=0; my $j=0; my @categories=(); my $ctgr;
my $url  = 'http://shakespeare.mit.edu/';
my $file  = 'index.html';

my $doc = get($url.$file);
my @categorycontent = split(/\n/,$doc);

foreach my $line (@categorycontent){
  if ($line =~ s/<\/h2>//)
  {$line=Trim($line);
  push(@categories,$line);}
}

foreach my $ctg (@categories){mkdir "$ctg";}

foreach my $line (@categorycontent){
  if ($line =~ m/<td VALIGN/){
  $ctgr=$categories[$i];
  $i++;}
  if ($line =~ m/<a href="/ && $i < 4){
```

Figure B.1: Perl code to get documents from internet.

```
26 ␣␣$line␣=~␣/<a.*href="([\s\S]+?)"/;
27 ␣␣my␣$link␣=␣$1;
28 ␣␣my␣$name␣=␣$link;
29 ␣␣$name␣=~␣s/\/index//;
30 ␣␣$name␣=␣$ctgr.'/'.$name;
31 ␣␣$link␣=␣$url.$link;
32 ␣␣$link␣=~␣s/index.html/full.html/;
33 ␣␣my␣$htmldoc␣=␣mirror(␣$link,␣$name␣);
34 ␣␣print␣$link."\n";
35 ␣␣}
36 }
37 exit␣0;
38
39 sub␣Trim
40 {
41 ␣␣my␣($string)␣=␣@_;
42
43 ␣␣$string␣=~␣s/\n//;
44 ␣␣$string␣=~␣s/^\s+//;
45 ␣␣$string␣=~␣s/\s+$//;
46 ␣␣return␣$string;
47 }
```

Figure B.1: Perl code to get documents from internet. (cont.)

B.2. Perl Implementations To Dissociate Characters

```
1 +#!/usr/bin/perl
2
3 $dir ='History';
4 @dir = <./History/*>;
5
6 foreach $doc (@dir)
7 {
8   $doc = Trim($doc);
```

Figure B.2: Perl code to dissociate characters.

```
  $doc = substr ($doc, length($dir)+3);
  next if ($doc eq "");
  print "$doc\n";
  push(@DocumentList,$doc);
}

foreach $file (@DocumentList)
{
open(HTML,"./".$dir."/".$file)
    || die ("could not open HTML\n");
@lines = <HTML>;
close(HTML,"./".$dir."/".$file);

foreach $line (@lines){
      if ($line =~ s/<title>//)
    {$line =~ s/: Entire Play//;
    $nameoftheplay=Trim($line);
    mkdir $dir.'/'.$nameoftheplay;}
      if ($line =~ m/<A NAME=speech/){
  $line =~ /<b>([\s\S]+?)<\/b>/;
  $character_temp = $1;
  $character_temp = CleanWord($character_temp);
  push(@CharList,$character_temp);
  open (MYFILE, '>>','./'.$dir.'/'
    .$nameoftheplay.'/'.$nameoftheplay
    .$character_temp.'.txt');
  print MYFILE $character_temp."\n";
      }
      elsif ($line =~ m/<A NAME=/){
  $line =~ /<A NAME=.*>([\s\S]+?)<\/A>/;
  my $line = $1;
  if ($line =~ m/\[/){
    $line =~ /\[([\s\S]+?)\]/;
    $line =~ s/$1//;
  }
```

Figure B.2: Perl code to dissociate characters. (cont.)

```
44    print MYFILE $line."\n";
45        }
46        if ($line =~ m/<\/blockquote>/){
47        close (MYFILE);
48      }
49      }
50
51 @CharList = grep { ! $seen{ $_ }++ } @CharList;
52 open(MYFILE,'>','./'.$dir.'/'
53    .$nameoftheplay.'/characterlist.txt');
54 foreach $character (@CharList){
55    print MYFILE $character."\n";
56      }
57 close (MYFILE);
58 }
59 exit 0;
60
61 sub Trim
62 {
63    my ($string) = @_;
64
65    $string =~ s/\n//;
66    $string =~ s/^\s+//;
67    $string =~ s/\s+$//;
68    return $string;
69 }
70
71 sub CleanWord
72 {
73    my ($word) = @_;
74    $word =~ s/[\.\,\?\!\:\;\-\"\(\)\[\]
75 ␣␣␣␣␣␣\^\+\%\&\/\=\#\$\{\}\\\|\*\']+//g;
76 ␣␣return␣$word;
77 }
```

Figure B.2: Perl code to dissociate characters. (cont.)

B.3. Perl Implementations To Get Document-Word Matrices

```
#!/usr/bin/perl
@dir = <./sample/*>;
$dir = './sample';

$docFile = shift(@ARGV);
@DocumentList = ();

foreach $doc (@dir)
{
  $doc = Trim($doc);
  $doc = substr ($doc, length($dir)+1);
  next if ($doc eq "");
  push(@DocumentList,$doc);
}

open(PNC,"<punc.txt")
  || die ("could not open punc.txt\n");
$pncLine = <PNC>;
close PNC;

$pncLine =~ s/\n//;
@pncList = split(/\s+/,$pncLine);

open(STP,"<engstopwords.txt")
  || die ("could not open engstopwords.txt\n");
@lines = <STP>;
close STP;

$stopWordLine = "";

foreach $stpWord (@lines)
{
```

Figure B.3: Perl code to get document-word martix.

```
33
34    $stpWord = Trim($stpWord);
35    $stopWordLine = $stopWordLine."␣".$stpWord;
36 }
37 $stopWordLine = $stopWordLine . "␣";
38
39 %Dictionary = ();
40
41 %word2Doc = ();
42 $prefix = "";
43
44 @hashList = ();
45 foreach $document (@DocumentList)
46 {
47 #  open the document
48    $docFile = $dir . "/" . $document;
49    open(DOC,"<$docFile")
50     || die("could␣not␣open␣$docFile\n");
51    @Lines = <DOC>;
52    close DOC;
53
54    $href = \%{$document};
55 #  we now have a hash array
56    push(@hashList,$href);
57    foreach $line (@Lines)
58    {
59       $line = Trim($line);
60       next if ($line eq "");
61 #     line is not an empty string
62       $line = CleanWord($line);
63       @wordsFound = split(/\s+/,$line);
64       foreach $newWord (@wordsFound)
65       {
66         $newWord = lc($newWord);
67         next if ($newWord eq "");
```

Figure B.3: Perl code to get document-word martix. (cont.)

```
68          next if ($stopWordLine =~ / $newWord /);
69 #        update the dictionary
70    if ($Dictionary{$newWord} eq "")
71    {
72      $Dictionary{$newWord} = 1;
73    }
74    else
75    {
76      $Dictionary{$newWord}++;
77    }
78          if ($href->{$newWord} eq "")
79    {
80      $href->{$newWord} = 1;
81    }
82    else
83    {
84      $href->{$newWord}++;
85    }
86
87        }
88
89     }
90 }
91
92 @Dictionary = sort keys %Dictionary;
93 unlink 'out.txt';
94 open (MYFILE, '>>out.txt');
95 print MYFILE "\t";
96 foreach $document (@DocumentList){
97   print MYFILE "$document\t";
98 }
99 print MYFILE "\n";
100
101 foreach $word (@Dictionary){
102   print MYFILE "$word\t";
```

Figure B.3: Perl code to get document-word martix. (cont.)

```
103   foreach $href (@hashList){
104   if ($href->{$word} eq ""){
105   $href->{$word} = 0;
106   }
107   print MYFILE $href->{$word}."\t";
108   }
109   print MYFILE "\n";
110 }
111 close (MYFILE);
112 exit 0;
113
114 foreach $a (@pncList)
115 {
116   print "$a\n";
117
118 }
119
120 foreach $a (@stopList)
121 {
122   print "$a\n";
123
124 }
125
126 exit 0;
127
128 sub Trim
129 {
130   my ($string) = @_;
131
132   $string =~ s/\n//;
133   $string =~ s/^\s+//;
134   $string =~ s/\s+$//;
135   return $string;
136 }
137
```

Figure B.3: Perl code to get document-word martix. (cont.)

```
138 sub CleanWord
139 {
140   my ($word) = @_;
141   $word =~ s/[\.\,\?\!\:\;\-\"\(\)
142 ␣␣␣␣\[\]\^\+\%\&\/\=\#\$\{\}
143 ␣␣␣␣\\\|\*\']+//g;
144 ␣␣return␣$word;
145 }
```

Figure B.3: Perl code to get document-word martix. (cont.)

B.4. Perl Implementations To Get Document-Word Matrices SVM-Light Standard

```
1  #!/usr/bin/perl
2  # @dir = <./hamlet_sample/*>;
3  # @dir = <./othello_sample/*>;
4  # @dir = <./othello_richardiii_sample/*>;
5  @dir = <./sample/*>;
6  # $dir = './hamlet_sample';
7  # $dir = './othello_sample';
8  # $dir = './othello_richardiii_sample';
9  $dir = './sample';
10
11 $docFile = shift(@ARGV);
12
13 @DocumentList = ();
14
15 # print "scanning files ...\n";
16 foreach $doc (@dir)
17 {
18   $doc = Trim($doc);
19   $doc = substr ($doc, length($dir)+1);
20   next if ($doc eq "");
21 #   print "$doc\n";
```

Figure B.4: Perl code to get document-word martix SVM-Light standard.

```
22    push(@DocumentList,$doc);
23  }
24  # print "done.";
25  # print "------------------------------------------->\n\n\n";
26  open(PNC,"<punc.txt") || die ("could␣not␣open␣punc.txt\n");
27  $pncLine = <PNC>;
28  close PNC;
29
30  $pncLine =~ s/\n//;
31  @pncList = split(/\s+/,$pncLine);
32
33  open(STP,"<engstopwords.txt") || die ("could␣not␣open␣engstopwords.tx
34  @lines = <STP>;
35  close STP;
36
37  $stopWordLine = "";
38
39  foreach $stpWord (@lines){
40    $stpWord = Trim($stpWord);
41    $stopWordLine = $stopWordLine . "␣" . $stpWord;
42  }
43  $stopWordLine = $stopWordLine . "␣";
44
45  %Dictionary = ();
46
47  @hashList = ();
48  foreach $document (@DocumentList){
49  #   open the document
50      $docFile = $dir . "/" . $document;
51      open(DOC,"<$docFile") || die("could␣not␣open␣$docFile\n");
52      @Lines = <DOC>;
53      close DOC;
54
55      $href = \%{$document};
56  #   we now have a hash array %MetamorphosisFK.txt
```

Figure B.4: Perl code to get document-word martix SVM-Light standard. (cont.)

```
57    push(@hashList,$href);
58    foreach $line (@Lines)
59    {
60       $line = Trim($line);
61       next if ($line eq "");
62 #     line is not an empty string
63       $line = CleanWord($line);
64       @wordsFound = split(/\s+/,$line);
65       foreach $newWord (@wordsFound)
66       {
67         $newWord = lc($newWord);
68         next if ($newWord eq "");
69         next if ($stopWordLine =~ / $newWord /);
70 #       update the dictionary
71   if ($Dictionary{$newWord} eq "")
72   {
73     $Dictionary{$newWord} = 1;
74   }
75   else
76   {
77     $Dictionary{$newWord}++;
78   }
79         if ($href->{$newWord} eq "")
80   {
81     $href->{$newWord} = 1;
82   }
83   else
84   {
85     $href->{$newWord}++;
86   }
87
88       }
89
90    }
91 }
```

Figure B.4: Perl code to get document-word martix SVM-Light standard. (cont.)

```
92
93 @Dictionary = sort keys %Dictionary;
94
95 open(MYOUTFILE, ">>train");
96 $i=1;
97 foreach $href (@hashList){
98   $i=0;
99   foreach $word (@Dictionary){
100  $i++;
101  next if ($href->{$word} eq "");
102  print MYOUTFILE $i.":".$href->{$word}." ";
103  }
104  print MYOUTFILE "\n";
105 }
106 close(MYOUTFILE);
107
108 open(MYOUTFILE, ">>wordlist.txt");
109 foreach $word (@Dictionary){
110  print MYOUTFILE "$word\n";
111 }
112 close(MYOUTFILE);
113
114 open(MYOUTFILE, ">>documentlist.txt");
115 foreach $document (@DocumentList){
116  print MYOUTFILE "$document\n";
117 }
118 close(MYOUTFILE);
119 exit 0;
120
121
122 sub Trim
123 {
124  my ($string) = @_;
125  $string =~ s/\n//;
126  $string =~ s/^\s+//;
```

Figure B.4: Perl code to get document-word martix SVM-Light standard. (cont.)

```
127   $string =~ s/\s+$//;
128   return $string;
129 }
130
131 sub CleanWord
132 {
133   my ($word) = @_;
134   $word =~ s/[\.\,\?\!\:\;\-\"\(\)\[\]\^\+\%\&\/\=\#\$\{\}\\\|\*\']+/
135 ␣␣return␣$word;
136 }
```

Figure B.4: Perl code to get document-word martix SVM-Light standard. (cont.)

APPENDIX C: MATLAB IMPLEMENTATION OF pLSA ALGORITHM

C.1. loop.m

```
clc;clear;
tic
%Parameter initialization
maxHigh=10;recur=500;cntl=-inf;
%Import document-word occurence matrix
wd=importdata('./wd_matrix/out.txt','\t');
n=wd.data;
%Get the document list and word list
documentlist=wd.textdata(1,:);
documentlist(1)=[];
documentlist=strrep(documentlist,'.txt','');
wd.textdata(1,:)=[];
wordlist=wd.textdata(:,1);
%Erase single occurrences in a row
take=sum(n~=0,2)==1;
ind=find(take);
n(ind,:)=[];
wordlist(ind)=[];
clear take ind;
%Erase multiple occurrences in a row
%take=sum(n~=0,2)==size(n,2);
%ind=find(take);
%n(ind,:)=[];
%wordlist(ind)=[];
%clear take ind;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
LHlist=zeros(1,recur);
```

Figure C.1: pLSA - Loop Function.

```
for K=3:3
mkdir(num2str(K));
for t=1:recur
  t,
  [pZdw,pWz,pZd,LH]=main(n,K);
  LHlist(t)=LH(end);
  sdLH=LH(end);
  save(sprintf('./%1d/data_%1d.mat',K,t),'pZdw','pWz','pZd',
                    'LH','n','wd','wordlist','documentlist');
  if  sdLH >=cntl,
    pZd_high=pZd; pWz_high=pWz; pZdw_high=pZdw;
    cntl=sdLH;
    LH_high=LH;
  end
  h=figure(3);
  plot(LH);
  hold on;
end
plot(LH_high,'--rs','LineWidth',2, 'MarkerEdgeColor','k'
                    ,'MarkerFaceColor' ,'g','MarkerSize',10);
saveas(h,sprintf('./%1d/img_LH_all.fig',K))
close(h)
save(sprintf('./%1d/data_highest.mat',K),'pZdw_high','pWz_high'
            ,'pZd_high','LHlist','n','wd','wordlist','documentlist');
end
toc
```

Figure C.1: pLSA - Loop Function. (cont.)

C.2. main.m

```
function [pZdw,pWz,pZd,LH]=main(n,K,LH)
    LH=[];
    %Document number - word number
    [M,N]=size(n);
    n_copy=n;
    n=dimduplicator(n,3,K);
    pZdw_raw=rand(M,N,K);
    pZdw_norm = dimduplicator(sum(pZdw_raw,3),3,K);
    pZdw=pZdw_raw./pZdw_norm;
    clear pZdw_raw pZdw_norm;
    for h=1:200
  [pZdw,pWz,pZd]=EM(pZdw,n,M,N,K);
  temp1=permute(pWz,[1 3 2]);
  temp2=permute(pZd,[3 2 1]);
  temp=(temp1*temp2);
  temp=log(temp);
  LH_temp=n_copy.*temp;
  LH_val=sum(sum(LH_temp));
  LH=cat(2,LH,LH_val);
  if h>10 && LH(h)-LH(h-1) < 1, break; end
    end
end
```

Figure C.2: pLSA - Main Function.

C.3. EM.m

```
function [pZdw,pWz,pZd]=EM(pZdw,n,M,N,K)
% "1 ->w","2->d","3->z"
% M-Step Calculations & pWz and pZd
  pWz_raw=sum(n.*pZdw,2);
  pWz_norm = dimduplicator(sum(pWz_raw,1),1,M);
  pWz=pWz_raw./pWz_norm;
  clear pWz_raw pWz_norm;
  pZd_raw=sum(n.*pZdw,1);
  pZd_norm = dimduplicator(sum(pZd_raw,3),3,K);
  pZd=pZd_raw./pZd_norm;
  clear pZd_norm pZd_raw;
%%%%%%%%%%%%%%%%%%%%%%
% E-Step Calculations & pZdw
  temp1=dimduplicator(pZd,1,M);
  temp2=dimduplicator(pWz,2,N);
  pZdw=temp1.*temp2;
  pZdw=pZdw./dimduplicator(sum(pZdw,3),3,K);
  clear temp1 temp2;
end
```

Figure C.3: pLSA - EM Algorithm Function.

C.4. dimduplicator.m

```
function [ np ] = dimduplicator( np, dp , Np)

for i=1:Np-1
    if dp==1
        np=cat(dp,np,np(1,:,:));
    end
    if dp==2
        np=cat(dp,np,np(:,1,:));
    end
    if dp==3
        np=cat(dp,np,np(:,:,1));
    end
end
end
```

Figure C.4: pLSA - Tool Matlab Function.

APPENDIX D: MATLAB IMPLEMENTATION OF VB-LDA

D.1. loop.m

```
clc; clear;
tic;
for k=3:3
  [alpha,beta,gammas,phi,ppl,ppl_list]
            = main('./wd_matrix/train',k,500);
end
toc;
documentlist=importdata('./wd_matrix/documentlist.txt','\t');
docclass_l=cat(2,num2cell(gammas),documentlist);
save(sprintf('./data/data.mat'));
```

Figure D.1: LDA - Loop Function.

D.2. main.m

```
function [alpha_l,beta_l,gammas_l,phi_l,ppl_l,ppl_list]
                                    = main(train,k,it,emmax,demmax)
% Latent Dirichlet Allocation, standard model.
% k      : # of hidden classes assumed
% it    : # of times over the whole LDA calculation
% emmax  : # of maximum VB-EM iteration (default 100)
% demmax : # of maximum VB-EM iteration for a document (default 20)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ungiven parameter initialization
ppl_log=[];
ppl_list=[];
if nargin < 5
  demmax = 40;
  if nargin < 4
    emmax = 150;
```

Figure D.2: LDA - Main Function.

```
16        if nargin < 3
17    it=35;
18        end
19    end
20  end
21  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22  % training data conversion and number of words counter
23  % d : data struct
24  % wnmbr : # of words
25  tempfile = fopen(train);
26    if (tempfile == -1)
27      error(sprintf('I can''t open %s.',train));
28    end
29  d={}; j=0; wnmbr = 0;
30  while ~feof(tempfile)
31    line = fgetl(tempfile);
32    ttemp = sscanf(line,'%d:%g',Inf);
33    nmbrw = length(ttemp) / 2;
34    temp.id  = zeros(1,nmbrw);
35    temp.cnt = zeros(1,nmbrw);
36    for i = 1:nmbrw
37      temp.id(i)  = ttemp(2*i-1);
38      if temp.id(i) > wnmbr %number of words counter
39        wnmbr = temp.id(i); %%%%%%%%%%%%%%%%%%%%%%%%
40      end                   %%%%%%%%%%%%%%%%%%%%%%%%
41      temp.cnt(i) = ttemp(2*i);
42    end
43    j = j + 1;
44    d{j} = temp;
45  end
46  fclose(tempfile);
47  clear j nmbrw ttemp lin temp i tempfile ans;
48  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49  % n : # of documents
50  n = length(d);
```

Figure D.2: LDA - Main Function. (cont.)

```
% parameter decleration
fprintf(1,'number of documents      = %d\n', n);
fprintf(1,'number of words          = %d\n', wnmbr);
fprintf(1,'number of latent classes = %d\n', k);
fprintf(1,'overall LDA iteration step = 1\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initializer calculation
[alpha,beta,gammas,phi,ppl,ppl_log] = lda(d,k,n,wnmbr,emmax,demmax);
ppl_list=cat(2,ppl_list,ppl);
save(sprintf('./data/data_%d_0.mat',k));
alpha_l=alpha; beta_l=beta; gammas_l=gammas; phi_l=phi; ppl_l=ppl;
ppl_log_l=ppl_log;
% LDA loop to find lowest perplexity
for i=1:it-1
fprintf(1,'overall LDA iteration step = %d\n',i+1);
[alpha,beta,gammas,phi,ppl,ppl_log] = lda(d,k,n,wnmbr,emmax,demmax);
ppl_list=cat(2,ppl_list,ppl);
save(sprintf('./data/data_%d_%d.mat',k,i))
  if ppl_l > ppl
    alpha_l=alpha; beta_l=beta; gammas_l=gammas; phi_l=phi;
  ppl_l=ppl; ppl_log_l=ppl_log;
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gammas_l=gammas_l./(sum(gammas_l,2)*ones(1,k));
h=figure(3);
plot(ppl_log_l,'--rs','LineWidth',2, 'MarkerEdgeColor','k'
                              ,'MarkerFaceColor' ,'g','MarkerSize',10);
saveas(h,sprintf('./data/img_PPL_all.fig'))
close(h)
save(sprintf('./data/data_lowestest.mat'))
```

Figure D.2: LDA - Main Function. (cont.)

D.3. lda.m

```
function [alpha,beta,gammas,phi,ppl,ppl_log]
                = lda(d,k,n,wnmbr,emmax,demmax)
% Latent Dirichlet Allocation, Variarional Bayesian EM (VB-EM),
% standard model.
% d      : data of documents
% k      : # of classes to assume
% n      : # of documents
% l      : # of words
% emmax  : # of maximum VB-EM iteration (default 100)
% demmax : # of maximum VB-EM iteration for a document (default 1

% alpha, beta, gamma, perplexity initialization
beta = normalizer(rand(wnmbr,k),1);
alpha = normalizer(rand(1,k));
gammas = zeros(n,k);
ppl = 0; pppl = 0; ppl_log=[];
% loop of EM steps
for j = 1:emmax
  fprintf(1,'iteration %d/%d..\t',j,emmax);
  fprintf('\n');
  % vb-estep
  betas = zeros(wnmbr,k);
  for i = 1:n
    [gamma,phi] = vbEM(d{i},beta,alpha,demmax);
    % gamma of each document separately
    gammas(i,:) = gamma;
    % here program changes each beta value correspondingly
  % to occurence of the words accumulation of each
  % beta value for each document
```

Figure D.3: LDA - LDA Function.

```
35    beta = normalizer(betas,1);
36    % converge?
37    ppl = lda_ppl(d,beta,gammas);
38    ppl_log(j)=ppl;
39    fprintf(1,'PPL = %g\t',ppl);
40    if (j > 1) && (pppl - ppl) < 1.0e-2
41      fprintf(1,'\nconverged.\n');
42    h=figure(3);
43    plot(ppl_log);
44    hold on;
45      return;
46    end
47    pppl = ppl;
48  end
49  fprintf(1,'\n');
```

Figure D.3: LDA - LDA Function. (cont.)

D.4. `lda lik.m`

```
1  function lik = lda_lik(d,beta,gammas)
2  egamma = normalizer(gammas,2);
3  lik = 0;
4  n = length(d);
5  for i = 1:n
6    t = d{i};
7    lik = lik + t.cnt * log (beta(t.id,:) * egamma(i,:)');
8  end
```

Figure D.4: LDA - Likelihood Measuring Function.

D.5. `lda ppl.m`

```
function ppl = lda_ppl(d,beta,gammas)
s = 0;
n = length(d);
for i = 1:n
  s = s + sum(d{i}.cnt);
end
ppl = exp (- lda_lik(d,beta,gammas) / s);
```

Figure D.5: LDA - Perplexity Measuring Function.

D.6. `vbEM.m`

```
function [alpha,phi] = vbEM(d,beta,alpha0,emmax)
% calculates a document and words posterior for a document d.
% alpha  : Dirichlet posterior for a document d
% phi    : (L * K) matrix of word posterior over latent classes
% d      : document data
% alpha0 : Dirichlet prior of alpha
% emmax  : maximum # of VB-EM iteration.
if nargin < 4
  emmax = 50;
end
N = length(d.id);
k = length(alpha0);
phi = zeros(N,k);
gamma = ones(1,k) * N / k;
gamma_temp = gamma;
% The difference in the program with the original paper is
% due to the fact that we are dealing with matrices rather
% than naive for loops scanning  all the elements. To
% explain better, the gamma initialization above does not
% contain alpha in order not to add again. For the posterior
% gamma value it is added when needed as in
% diag(exp(psi(alpha0 + gamma)))
for j = 1:emmax
```

Figure D.6: LDA - Variational Bayesian Expectation-Maximization Function.

```
24    % e-step
25    phi = beta(d.id,:) * diag(exp(psi(alpha0 + gamma)));
26    phi = normalizer(phi,2);
27    % m-step
28    gamma = d.cnt * phi;
29    % convergence test
30    if (j > 1)&&((norm(gamma-gamma_temp)/norm(gamma))<1.0e-4)
31      break;
32    end
33    gamma_temp = gamma;
34  end
35  alpha = alpha0 + gamma;
```

Figure D.6: LDA - Variational Bayesian Expectation-Maximization Function. (cont.)

D.7. `newton alpha.m`

```
1  function alpha = newton_alpha(gammas,maxiter,ini_alpha)
2  % Newton-Raphson iteration of LDA Dirichlet prior.
3  % gammas  : matrix of Dirichlet posteriors (M * k)
4  % maxiter : # of maximum iteration of Newton-Raphson
5
6  [M,K] = size(gammas);
7
8  % if there is only single document, return
9  if (M == 1)
10   alpha = gammas(1,:);
11   return;
12 end
13
14 % ungiven parameter initialization
15 if nargin < 3
16   ini_alpha = mean(gammas) / K; % initial point
17   if nargin < 2
18     maxiter = 50;
19   end
20 end
```

Figure D.7: LDA - Newton Alpha Formula Function.

```
% parameter initialization
% l : loop counter
% g : gradient matrix

l = 0;
g = zeros(1,K);
g_temp = sum(psi(gammas),1)-sum(psi(sum(gammas,2)));
alpha = ini_alpha;
alpha_temp = zeros(1,K);

for t = 1:maxiter
  l = l + 1;
  alpha0 = sum(alpha);
  g = M * (psi(alpha0) - psi(alpha)) + g_temp;
  h = - 1 ./ psi(1,alpha);
  hgz = h * g' / (1 / psi(1,alpha0) + sum(h));

  for i = 1:K
    alpha(i) = alpha(i) - h(i) * (g(i) - hgz) / M;
  end
  if any(alpha < 0)
    alpha = newton_alpha(gammas,maxiter,ini_alpha / 10);
    return;
  end

  if (l>1)&&((norm(alpha - alpha_temp)/norm(alpha))<1.0e-4)
    break;
  end
  alpha_temp = alpha;
end
```

Figure D.7: LDA - Newton Alpha Formula Function. (cont.)

D.8. `normalizer.m`

```
function b = normalizer(a,d)

if nargin < 2
  c=sum(sum(a));
  b=a./c;
  return
end

c = sum(a,d);
s = size(a);
if d == 1
  c = ones(s(1),1)*c;
  b = a./c;
elseif d == 2
  c = c*ones(1,s(2));
  b = a./c;
end
```

Figure D.8: LDA - Tool Function.