Parameter Recovery in Latent Dirichlet Allocation (LDA): Potential Utility of LDA in Formative Constructed Response Assessment

by

Minho Kwak

(Under the Direction of Seock-Ho Kim and Allan S. Cohen)

Abstract

The purpose of this study was to compare the precision and accuracy of estimates of LDA under various conditions. Such conditions included the following factors: 1) sample size, 2) document length, 3) number of unique words, 4) priors, and 5) number of topics. The range of sample size was divided into three levels: 700, 1,500, and 3,000. The range of document length was also divided into three levels: 50, 100, and 300. The range of the number of unique words was, again, divided into three levels: 500, 1,000, and 3,000. For the priors, the informative prior suggested by the previous study and the non-informative prior were applied to the model. For the number of topics, three-topic and five-topic models were used. The simulation results can be summarized as follows. When the sample size was greater than 700, the sample size and the document length had little impact on the precision and accuracy of the estimates. However, the prior had an impact on the accuracy and precision of the estimates. Specifically, the informative prior had a positive effect on the accuracy and precision of the estimates. The informative priors suggested by the previous study were $50/K$ for the α and $200/V$ for the β. This means that the α makes the topic distribution more even, and is more appropriate to the estimates than the non-informative prior. Also, for the β, the prior causes the word distribution to be more sparse, which is appropriate to the estimates. For the number of unique words, the results suggested that when the document length and sample size increased, the most appropriate number of unique words was likewise going to increase. However, when the number of topics increased or the informative prior was applied, the impact of the number of unique words was reduced. Based on the simulation, the empirical results suggested that estimates based on the corpus collected from CR items administered by the GCA can be considered as the appropriate estimates.

Index words: Latent Dirichlet Allocation, Constructed response, Automated-essay scoring, Writing assessment Sample size, Parameter recovery, Dissertation

Parameter Recovery in Latent Dirichlet Allocation (LDA): Potential Utility of LDA in Formative Constructed Response Assessment

by

Minho Kwak

B.S., Seoul National University, 2010

M.A., Seoul National University, 2012

M.S., University of Georgia, 2019

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2019

Parameter Recovery in Latent Dirichlet Allocation (LDA): Potential Utility of LDA in Formative Constructed Response Assessment

by

Minho Kwak

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Statement of Problem

Constructed response (CR) items are used increasingly in assessments, in part, because they are considered useful for assessing certain types of higher order knowledge requiring examinees to show their reasoning in their answers (Attali, 2014; Smith & Tanner, 2010; Weston, Parker, & Urban-Lurain, 2013). The typical method of evaluating CR items is to have human graders compare examinees' responses to one or more rubrics. As a result, responses to CR items are more costly to score and take longer to score than selected response items. Automated essay-grading algorithms are increasingly available that make it possible to grade constructed responses more quickly and less costly (Shermis, 2014). Whether graded by hand or by computer algorithm, the focus is normally only on the rubric-based score. As a result, there is little further attention paid to the actual text of examinees' responses.

Recent developments in analysis of textual data, however, has suggested that statistical topic models may be able to extract potentially useful information from the text of examinees' responses (Kim, Kwak, & Cohen, 2017; Kwak, Kim, & Cohen, 2017). In this regard, Kwak et al. (2017) found that results using the topic model latent Dirichlet allocation (LDA; Blei, Ng, & Jordan, 2003) can detect latent themes in text that are sensitive to instructional intervention. Therefore, this study explores further the use of topic models in the context of examinees' responses to constructed response items.

Topic models are statistical models designed to detect the latent thematic structure in a corpus (i.e., a body) of text (Blei, 2012). These models have been developed to analyze large bodies of text documents, i.e., the corpus of documents, such as medical texts (Yao, Zhang, Wei, Wang, Zhang, Ren, & Bian, 2015), twitter messages (Wang, Gerber, & Brown, 2012), abstracts of scientific journals (Griffiths, & Steyvers, 2004) and blogs (Yano, Cohen, & Smith, 2009). Topic models consider the clusters of words as representing the latent themes in the corpus.

Kim et al. (2017) and Kwak et al. (2017) fit LDA, one of the simpler topic models, to the textual data in examinees' responses to CR items. One difficulty with fitting existing topic models to this kind of text is that topic models were originally developed on large sets of documents, generally numbering over 5,000 (Griffiths and Steyvers, 2004; Griffiths et al., 2007; Thomas et al., 2014). Text data from answers to CR items tend to be somewhat short, generally between 50 and 100 words and more constrained (Kim et al., 2017; Kwak et al., 2017). The text of twitter messages is relatively unconstrained and, as a result, the number of latent topics in twitter data is large (i.e., close to 100). In addition, tests tend to have time limits, a more limited vocabulary that is focused on the prompts (Kwak et al., 2017). Topic modeling of CR responses, on the other hand, has reported far fewer topics, on the order of three to five topics (Kim et al., 2017). These kinds of differences can affect the estimation of the different topic models.

There also is relatively little evidence that LDA precisely estimates the parameters under the constrained conditions of responses to CR items. Kwak et al. (2018) explored effects of different priors given the limited number of words available in CR responses and found some priors to provide more accurate results than others.

1.2 Purpose of the Study

This study focuses on examining parameter recovery in LDA for different practical testing conditions including number of documents, document length, and number of unique words in the corpus.

Chapter 2

Theoretical Framework

2.1 Current Status of Constructed Response Assessment

Currently, scoring of constructed response (CR) items is done either externally with trained graders or locally with teachers (Behizadeh & Pang, 2016; Wolcott & Legg, 1998). With external scoring, school teachers are rarely involved in the scoring process. With internal scoring, school teachers are typically not as well trained as external raters. As a result, most schools tend to choose the external scoring method.

Behizadeh and Pang (2016) note that it is hard to maintain the consistency of local scoring because of the subjectivity of the raters. In order to reduce bias and subjectivity, they suggest that schools ask external raters to evaluate student's response with the rubric, which is one of the tools that guarantees the consistent quality of the evaluation. However, since the external scoring requires great expense in terms of time and money, it might cause the usage of writing assessment to be limited to a specific type of assessment—the summative high-task assessment (Lee, 2011).

With the advent of automated scoring, the cost and time for scoring has been markedly reduced (Berstein, Leacock, & Swartz, 2001). Automated scoring uses software packages that are a combination of lexical and statistical analysis to compare a response with a rubric (Rudner & Gagne, 2001). Each of these software packages employs its own particular algorithms. In 1999, automated scoring was used for the analytical writing assessment portion of the Graduate Management Admissions Test (GMAT). The analytical writing portion consists of two sub-components: 1) Analysis of an argument (Argument essays) and 2) Analysis of an issue (Issue essays). The automatic scoring system determines the level of a responses with a 6-point scale based on a holistic rubric (Burstein, Braden-Harder, Chodorow, Hua, Kaplan, Kukich, & Wolff, 1998). Since 1999, 750,000 GMAT essays have been scored by the program E-rater (Burstein, 2003). Evaluation of E-rater suggests that the agreement rate between human raters and E-rater is close to 97% (Berstein et al., 2001).

Most uses of automated scoring have been in large testing programs although Weston et al. (2013) has reported its use for formative assessment. Weston et al. focused on feedback functions of the software and suggested that the automated text analysis can be a valuable tool as formative feedback to constructed-response item. Dikli (2006) also summarized the characteristics of the feedback from the different programs for the automated scoring. Generally, feedback is mainly focused on refined sentence structure, variety of appropriate word usage, and organizational structure. For example, E-rater focuses on grammatical feedback (e.g., whose instead of who's, should have instead of should of). Intelligent Essay Assessor (IEA; Foltz, Laham, & Landauer, 1999) is another software package for scoring CR item responses. It focuses on general guidance (e.g., ideas and content, organization, sentence fluency, word choice, conventions, and voice) not the lexical structure of students' responses. A third program, Criterion (Burstein, Chodorow, & Leacock, 2004), which provides web-based essay scoring, provides general feedback as an advisory component. It provides three types of advice: 1) The text is too brief to be a complete essay; 2) The essay text does not resemble other essays written about the topic; 3) The essay response is overly repetitive.

The characteristics of the feedback from these programs can be explained from the perspective of language structure. The language structure can be classified into six different categories—phonetics, phonology, morphology, syntax, semantics, and pragmatics (Fromkin, Rodman, & Hyams, 2013). These categories can be interpreted as the flexible hierarchical structure of the language. For example, the set of phonology can be considered as morphology. Each category represents the components of language. Phonetics indicates the physical aspect of speech such as sound and ears. Phonology deals with idealized symbolic units that can be combined according to formal rules. The morphology refers to the grammar along with the syntax. The syntax covers the grammatical arrangements of words within sentences, and how we use speech in communication. The semantics deals with the study of meaning. Pragmatics (as applied to linguistics) is about how we actually use speech in communication, and how context aids the transmission of meaning in utterances.

According to the hierarchy of language structure, the feedback of the programs is mainly concerned with the syntax level (e.g., grammatical errors, appropriate phases, etc.). The topic modeling makes it possible to identify the semantic structure of the student's response, and would be able to produce semantic level feedback. Thus, unlike essay scoring software, the topic modeling provides information about the latent structure of the text.

LDA which is one of the text-mining techniques to extract topics from the corpus has valuable advantage because it reveals the latent semantic structure that is unobservable in a direct way. Based on the results from the corpus, each document can be represented as compound of different topics. For example, a previous empirical study (Kim et al., 2016) showed that the student's constructed response can be represented as the compound of three different topics—general academic words, discipline specific words, and everyday language words. Specifically, the study indicated that frequency of the discipline-specific word has positive moderate correlation with the score. On the other hand, the other two topics showed insignificant relationships with the score. It can produce insight that helps teacher to teach students by understanding latent lexical structure of students.

Topic modeling produces information about the topic proportions, and they can be interpreted as the profile. Based on the profile, teachers might give the instructional guide on the semantic level (Page, Poggio, & Keith, 1997). Also, since it is much less time consuming and expensive, it is likely to encourage the usage of the writing assessment for the various types of assessments such as the classroom test for formative purposes (Chen & Cheng, 2008).

2.2 Latent Dirichlet Allocation

LDA defines the corpus as the set of the documents being analyzed (Blei et al., 2003). Each document also can be viewed as comprised of a set of the latent topics. Thus, the topics, the documents, and the corpus have hierarchical structure within the LDA framework.

The starting point of the modeling process is the detection of the latent topics. The topics are considered the first level of the hierarchy. The topic can be defined as a multinomial distribution of the words. The distribution is the word distribution of topics. The categories of the distribution are the vocabularies in the corpus, and the probabilities corresponding to the categories determine the latent topic. For example, assumed the multinomial distribution is consists of six categories on vocabularies inculding terms such as *fish*, *wave*, *swim*, *tree*, *trail*, and *climb*, the labeling the topics depends on the probabilities corresponding to the categories. Specifically, when the distribution shows the high probabilities of the categories such as *fish*, *wave*, and *swim* with the low probabilities of the other words, the topic can be labeled as *sea*. On the other hand, if another distribution shows the high probabilities of the categories such as *tree*, *trail* and *climb* with the low probabilities of the other words, the topic can be labeled as *mountain*.

When the topics are determined (i.e. extracting a specific number of topics and labeling of the topics), the document, the next level of the hierarchy, can be also defined as a multinomial distribution, and the categories of the distribution are the topics determined in the previous process. The distribution is called as topic distribution of document. Since the categories of the distribution are the topics, the probabilities corresponding to the categories are able to determine a profile of the document. For example, if a multinomial distribution is assumed to have two categories (topics) such as *sea* and *mountain*, the profiles of the documents depend on the probabilities corresponding to the topics. Specifically, when the distribution shows the high probability of the topic *sea* with the low probabilities of topic *mountain*, the profile of the document can be reported as a *sea* dominated document. On the other hand, if another distribution shows the high probabilities of topic *mountain* with the low probability of topic *sea*, the profile of the document can be reported as a *mountain* dominated document. If a distribution is assumed to show similar probabilities for both topics, the document can be reported as a balanced topic document.

When the profiles of the documents are determined, the words are drawn from the word distribution of topic, given the topic. For example, for a specific word, if the topic assignment is determined as topic *mountain*, the one of the categories such as *trail*, *tree*, *climb*, *wave*, *fish*, and *swim* is sampled based on their corresponding probabilities.

The generative process referred above can be rigorously specified as follows:

1. Choose $\gamma_k \sim Dirichlet(\beta)$, where $k = 1, \ldots, K$.

   where a word distribution of topic vector γ_k denotes the set of γ_{vk} which is a probability that a student who is detected in topic k chooses vocabulary v where $v = 1, \ldots, V$. Thus, $\gamma_k = (\gamma_{1k}, \ldots, \gamma_{Vk})$ means the set of probability of V vocabularies in the topic k, and is assumed to have a Dirichlet distribution with parameter vector $\beta = (\beta_1, \beta_2, ..\beta_V)$.

   For each document j in a corpus,

2. Choose $\eta_j \sim Dirichlet(\alpha)$, where $j = 1, \ldots, J$.

   The topic distribution of document vector η_j denotes the set of η_{kj}which is a probability that a topic k occurs in the document j. Thus, $\eta_j = (\eta_{1j}, \eta_{2j}, \ldots, \eta_{Kj})$ means the set of probabilities of the K topics in the document j. η_j is assumed to have a Dirichlet distribution with the parameter vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$.

3. Choose ith word in jth document w_{ij}, where $i = 1, \ldots, N_j$, and:

(a) Choose a topic assignment $z_{ij} \sim Multinomial(\eta_j)$, where $z_{ij} = 1,\ldots,K$, and:

(b) Choose a word $w_{ij} \sim Multinomial(\gamma_k)$ given k=z_{ij}.

When estimating the LDA, the Dirichlet distribution is used in this study as a conjugate prior for the multinomial distribution. The topic assignment is noted as z_{ij}, and corresponding w_{ij} is a specific word i in a particular document j.

The parameters γ_k and η_j determine the shapes of both multinomial distributions. For example, if both distributions are sparse, with high concentration on a few topics or on a few words, it means that the documents are likely easily distinguishable. However, if the shapes of both distributions are very even, with balanced densities, it means that the similarity of the documents is high, and therefore, also likely indistinguishable. The shapes of both γ_k and η_j depend on the parameters, α and β. If these are much smaller than 1 (e.g., 0.01 or 0.001), the multinomial distribution will be sparse. On the other hand, if the parameters are much larger than 1 (e.g., 10 or 20), the multinomial distribution will be more uniform; thus the probability mass of categories (i.e., the words) will be similar to each other.

After the parameters for γ_k and η_j are determined, the topic distributions for document i (i.e., η_j) are sampled, and topics for all words in jth document are sampled conditional on the topic distribution η_j. Finally, ith word in document j is sampled from a multinomial distribution with probability γ_k given topic k.

The joint distribution of w_j, z_j, η_j, and γ_k, given the Dirichlet parameter vectors α and β with document w_j= $(w_{1j}, w_{2j}, \ldots, w_{N_j j})$ as the set of the words w_{ij} and corresponding z_j= $(z_{1j}, z_{2j}, \ldots, z_{N_j j})$ as the set of the topic assignments z_{ij} with the specific number of words in the document j (i.e., N_j) can be written from the generative model (Henrich, 2008):

$$p\left(w_j, z_j, \gamma_k, \eta_j|\alpha, \beta\right) = \left\{\prod_{n=1}^{N_j} p\left(w_{nj}\middle|\gamma_{z_{n,j}}\right) p\left(z_{nj}\middle|\eta_j\right)\right\} p\left(\eta_j\middle|\alpha\right) p\left(\gamma_k\middle|\beta\right). \tag{2.1}$$

The likelihood of a document w_j is obtained by integrating over γ_k and η_j and summing over z_j as follows:

$$p\left(w_j\middle|\alpha,\beta\right)=\int\int p\left(\gamma_k\middle|\beta\right)p\left(\eta_j\middle|\alpha\right)\prod_{n=1}^{N_j}\sum_{z_{nj}}p\left(w_{nj}\middle|\gamma_{z_{n,j}}\right)p\left(z_{n,j}\middle|\eta_j\right)d\eta_j d\gamma_k \tag{2.2}$$

Finally, the likelihood of the total corpus $\boldsymbol{w}=(w_1,w_2,w_3\ldots,w_M)$, can be represented as the product of the likelihood of J documents:

$$p(\boldsymbol{w}|\alpha,\beta)=\prod_{j=1}^{J}p\left(w_j\middle|\alpha,\beta\right) \tag{2.3}$$

There are several existing approaches to estimating the likelihood of $p(\boldsymbol{w}|\alpha,\beta)$, including variational methods (Blei et al., 2003), expectation propagation (Minka & Lafferty, 2002), and Gibbs sampling (Griffiths & Steyvers, 2004). In this study, Gibbs sampling was used to estimate the parameters. Heinrich (2008) notes the target distribution of the Gibbs sampler, $p(\boldsymbol{z}|\boldsymbol{w})$, is directly proportional to the joint distribution $p(\boldsymbol{z},\boldsymbol{w})$ and can be derived as in Equation 2.4:

$$p(\boldsymbol{z}|\boldsymbol{w})=\frac{p(\boldsymbol{z},\boldsymbol{w})}{p(\boldsymbol{w})}=\frac{\prod_{j=1}^{N_j}p(z_j,w_j)}{\prod_{j=1}^{N_j}\sum_{k=1}^{K}p(z_j=k,w_j)} \tag{2.4}$$

The target distribution can be derived and simplified as in Equation 2.5:

$$p\left(z_q=k\middle|z_{\neg q},w\right)\propto\left[\frac{n_{k,\neg q}^{(v)}+\beta_v}{\left[\sum_{v=1}^{V}n_{k,\neg q}^{(v)}+\beta_v\right]}\right]\left[\frac{n_{j,\neg q}^{(v)}+\alpha_k}{\left[\sum_{k=1}^{K}n_{j,\neg q}^{(k)}+\alpha_k\right]-1}\right], \tag{2.5}$$

where q is simplified index for the ith word in the document j (i.e. $q=(i,j)$). $n_{k,\neg q}^{(v)}$ is the count of vocabulary v assigned to the topic k, except for the current qth assignment in the corpus; $\sum_{v=1}^{V}n_{k,\neg q}^{(v)}$ is the total count of vocabularies assigned to topic k, except for the current qth assignment in the corpus; $n_{j,\neg q}^{(k)}$ is the count of words in document j assigned to topic k, except for the current qth assignment in the corpus; and $\sum_{k=1}^{K}n_{j,\neg q}^{(k)}$ is the total number of words in document j, except for the current qth assignment in the corpus.

2.3 Selecting prior

Selecting the appropriate values for α and β parameters for the two Dirichlet distributions is important as these values determine the shapes of the word distributions of topics, γ_ks, and the topic distributions of documents, η_js, respectively. Criteria for selecting the most appropriate values, however, have not yet been reported in the literature. Neither is there any theoretically grounded approach for selecting optimal values for these parameters (Chang, 2010; Thomas, Adams, Hassan, & Blostein, 2014). In this study, suggetsed priors of $50/K$ for α (Griffiths & Steyvers) and $200/V$ for β (Blei et al., 2003) were used. These priors are commonly used in the previous empirical studies (Bíró et al., 2009; Canini, Suh, & Pirolli, 2011; Lu, & Wolfram, 2012; Porteous, Newman, Ihler, Asuncion, Smyth, & Welling, 2008; Rosen-Zvi, Chemudugunta).

2.4 Preprocessing text data

Preprocessing of the text data in the corpus is a necessary first step in application of topic models (Boyd-Graber, Mimno, & Newman, 2014). This step helps prevent spurious extraction of latent topics. The preprocessing step converts the original source data to a form that can be recognized by the LDA. Boyd-Graber et al. note three steps in the preprocessing of the data: 1) tokenization, 2) normalization, and 3) stopword removal.

Tokenization refers to the separating or breaking of a string of text into its constituent words. In this step, the text is decomposed into separate words to construct the term-document matrix. This matrix consists of a row for each document and a column for each word. For example, a sentence from the text, *The lack of multiparameter models permitting easy calculation*, would be decomposed in the tokenization step into the individual words *the*, *lack*, *of*, *multiparameter*, and *calculation*. This preprocessing step focuses the topic model on the individual terms of the document.

The normalization step converts words in the term-document matric that refer to the same underlying concept into the same common term. For example, *Fish*, *Fishes*, *fish*, and *fishes* might be considered as representing the same concept even though some terms are plural, some are singular, some are capitalized and some are not. In that corpus, these four different words could be converted in the normalization process into a single common word such as *fish*. Normalization also corrects typographical errors (e.g., fishs) into the common word *fish*. In this way, the frequency of the underlying concept is amplified, making its contribution to the latent topic *fish* more likely to be recognized than would be the other spellings (or misspellings) of the term.

Stopwords are words that have high frequencies in the document but that contain little useful information with respect to understanding the latent thematic structure. Thus, removal of stopwords is done to improve the information in the term-document matrix. Schofield, Magnusson and Mimno (2017) note that stopwords can make up as much as 40-50% of the words in a corpus documents. Schofield et al. suggest further that stopword frequency is rarely correlated with a given topic, but is very likely to lead to extraction of meaningless latent topics.

Stopword removal is heavily dependent on the context of the corpus. That is, a specific word that might be considered a stopword in one context could be considered a meaningful word in another. For example, *fish* would not generally be treated as a typical stopword in the same way as *a*, *the*, and *I*. This is because, in most contexts the word *fish* would be considered to have an important meaning. If the corpus consists of documents from fishing magazines, then the frequency of *fish* is likely to be very high. As a result, the word might be removed, since the word occurs with high frequency but little unique information across the documents in the corpus. In this way, it could be considered as being similar to more usual stopwords such as *a*, *the*, and *I*, thus being removed.

Selection of stopwords can be done by inspection by a judge or by use of a statistical algorithm. The inspection method would consist of calculating the frequency distribution of all the terms in the term-document matrix following tokenization and normalization. For corpora with large numbers of unique words, this would be a very time-consuming task. Comparison with selection by a second judge would be important in determining the inter-judge consistency of the selection of stop words. Selection using a statistical algorithm, however, would be less subject to inter-judge variability and, at the same time, would be far less time-consuming. In this study, therefore, stopwords were selected using an index called the term frequency-inverse document frequency (TF-IDF; Robertson, 2004).

The TF-IDF produces scores which are assigned to each unique word in the corpus. Words that are less than a given value are considered as potential stopwords. Inspection of the resulting list of terms identified as potential stopwords is necessary before determining whether a given TF-IDF score is appropriate for the given corpus.

The TF-IDF score is calculated by multiplying the term frequency by the inverse document frequency as follows:

$$\text{TF-IDF score} = tf_{vj} \times \log\left(\frac{J}{df_v}\right), \tag{2.6}$$

where tf_{vj} denotes the frequency of the vocabulary v in the document j, J denotes the total number of documents, and df_v is the frequency of documents containing the vocabulary v.

Term tf_{vj} reflects the frequency of a given word in the document. For example, if *fish* occurs in a document frequently, it could be interpreted as an important word. In this way, it could be used as an indication of the importance of the word in the document.

The inverse document frequency, $\log\left(\frac{J}{df_v}\right)$, is the log of the total number of documents divided by the number of documents containing a specific word v. If the word commonly occurs across documents, the value will be zero or close to zero. This is because df_v will be close to J. For example, since *the* tends to occur in all documents regardless of the content, the value of the inverse document frequency is close to 0. For this reason, the TF-IDF scores of stopwords (e.g., *a*, *the*, *she*, and *he*) are likely to be close to 0.

Thus, words with lower TF-IDF scores are more likely to be considered as candidate stopwords. Subsequent judgement of words identified in this way is an important and necessary next step to avoid the possibility of removing words that high importance for the given context.

In selecting stopwords, Hornik and Grün (2011) used a cut-off TF-IDF score as the median of the TF-IDF score distribution over the words in the corpus. In this dissertation, however, if this value had been routinely applied to select stopwords in the empirical example, meaningful academic words such as *beach* and *but* would have been removed.

There do not appear to be clear guidelines as yet in the literature for selecting a TF-IDF cutoff score for a given context. Manning, Raghavan and Schütze (2008) suggest that the 30 most common words in a topic typically can account for roughly 30% of the total unique words in a corpus. Although this is a somewhat ad hoc guideline, a TF-IDF value using this guideline was 0.008 and was used in the simulation study and also in the empirical example in this dissertation. This TF-IDF value produced fewer than 30 stopwords as can be seen in Table 3.3.

The preprocessing step is important, furthermore, in order to evaluate the resulting LDA model, as it is closely related to determining the number of parameters used in the LDA. Specifically, the number of parameters in the LDA can be calculated as Equation 2.7:

$$p = K + K \times V, \tag{2.7}$$

where K is the number of topics and V is the number of unique words in the LDA model. Thus, the preprocessing reduces the number of unique words in the corpus, thereby, resulting in reducing the number of parameter in the model. The process is also closely related to the dimensionality of the posterior distribution of the model, and thus directly affects the estimation of the joint posterior distribution.

2.5 Model Selection

In previous studies, perplexity has been used to calculate the optimal number of topics (Grun & Hornik, 2011). However, Chang (2009) showed that model selection using perplexity is significantly different from results based on trained human judgement. Thus, Chang suggested perplexity was limited in terms of accurately determining model fit of LDA models.

Since LDA was estimated in this study using Bayesian estimation, the deviance information criterion (DIC; Spiegelhalter, Best, Carlin, & Van der Linde, 2002) was used to inform model selection. Bayesian estimation is used in this case, in part because this is the method used in much of the literature on application of LDA (Lauderdale & Clar, 2014; Sizov, 2012) and in part because the number of parameters of the LDA models in this study exceeded the sample size. For this latter reason, indices such as the Bayesian information criterion (BIC; Schwarz, 1978), sample adjusted BIC (ABIC; Sclove, 1987), Akaikes information criterion (AIC; Akaike, 1974), and corrected AIC (AICc; Hurvich & Tsai, 1989), would be inappropriate.

Chapter 3

Simulation and Empirical Studies

3.1 Design of Simulation Study

In this simulation study, the performance of the LDA was evaluated based on two criteria:
1. the accuracy and precision of the word distribution of topic parameter
2. the accuracy and precision of the topic distribution of document parameter

The following paragraphs describe the indices used in evaluating the simulation study and the empirical example. Following that, we describe the selection of factors used in the simulation study and the rationale for their selection.

3.1.1 Convergence of the LDA Sampler

To estimate the parameters in LDA, an R package called *topicmodels* (Hornik, & Grün, 2011) was used. In this package, a Monte Carlo Markov chain (MCMC) algorithm with collapsed Gibbs sampling (Heinrich, 2009) was used. In the Gibbs sampling, a repeated sampling is performed to estimate the posterior distributions of parameters. The number of repetitions is usually referred to as the number of iterations. The set of iterations consists of two parts including burn-in and post-burn-in iterations. First, the early phase, the burn-in iterations, consists of the iterations needed to converge. When the MCMC chain has converged, the burn-in is discarded and the subsequent iterations are retained. It is these later iterations, the post-burn-in iterations, that are used to obtain the posterior estimates of parameters. In this study, the number of burn-in iterations was 5,000, and the number of post-burn-in iterations was 15,000.

3.1.2 Accuracy and Precision of the Word Distribution of Topic Parameter

A recovery analysis was performed to assess the accuracy and the precision of word distribution of topic parameter matrix $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_K)$ where $\gamma_k = (\gamma_{1k}, \gamma_{2k}, \gamma_{3k}, .., \gamma_{Vk})$. in this regard, the size of the matrix is $V \times K$. The result of the recovery analysis was evaluated based on the average bias and root mean square deviation (RMSD) of the estimator. The average bias can be defined as the average of the difference between the true value (i.e., the generating value) of the parameter and the estimated parameter. It can be represented as in Equation 3.1:

$$\text{Averaged bias}\left(\hat{\theta}\right) = E\left(\hat{\theta}\right) - \theta, \tag{3.1}$$

where θ is the true value of the parameter. If $\hat{\theta}$ is the unbiased estimator, then E($\hat{\theta}$) converges to θ resulting in Bias $\left(\hat{\theta}\right)$= 0. If the bias is positive, the estimator has overestimated the parameter and vice versa. The RMSD is the standard deviation of the residuals calculated as the differences between the estimated values and the true values of the parameter θ. It can be represented as Equation 3.2:

$$\text{RMSD}\left(\hat{\theta}\right) = \sqrt{E[(\hat{\theta} - \theta)^2]}. \tag{3.2}$$

If the estimator is unbiased, RMSD can be interpreted as the standard error of the estimator. Thus, while the bias produces information regarding the accuracy of the estimator, the RMSD provides information regarding the precision. In this study, the bias and RMSD of the word distribution of topic parameter were calculated for words and topics, respectively, across replications by the following equations:

$$\text{Averaged bias}\left(\hat{\boldsymbol{\gamma}}\right) = \frac{\sum_{r=1}^{R} \sum_{v=1}^{V} \sum_{k=1}^{K} (\hat{\gamma}_{vkr} - \gamma_{vk})}{R \times V \times K} \tag{3.3}$$

$$\text{RMSD}\left(\hat{\boldsymbol{\gamma}}\right) = \frac{\sum_{r=1}^{R} \sum_{v=1}^{V} \sum_{k=1}^{K} (\hat{\gamma}_{vkr} - \gamma_{vk})^2}{R \times V \times K}, \tag{3.4}$$

where $\hat{\gamma}_{vkr}$ is the estimated words distribution of topic parameter for vocabulary v of topic k in replication r, γ_{vkr} is the true value of the word distribution of topic parameter of vocabulary v of topic k in replication r, R is the number of replications, V is the number of vocabulary, and K is the number of topics.

3.1.3 Accuracy and Precision of the Topic Distribution of Document Parameter

A recovery analysis was performed to assess the accuracy and the precision of the topic distribution of document parameter matrix $\boldsymbol{\eta} = (\eta_1, \eta_2, \eta_3, \ldots, \eta_J)$ where $\eta_j = (\eta_{1j}, \eta_{2j}, \eta_{3j}, .., \eta_{Kj})$. Thus, the size of the matrix is $K \times J$. As noted above, the result of recovery analysis was evaluated based on the averaged bias and RMSD of the estimator. The averaged bias and RMSD of the topic distribution of document parameter was calculated across the topics and documents, respectively, over replications by the following equations:

$$\text{Averaged bias}\,(\hat{\boldsymbol{\eta}}) = \frac{\sum_{r=1}^{R}\sum_{k=1}^{K}\sum_{j=1}^{J}(\hat{\eta}_{kjr} - \eta_{kj})}{R \times K \times J} \tag{3.5}$$

$$\text{RMSD}\,(\hat{\boldsymbol{\eta}}) = \frac{\sum_{r=1}^{R}\sum_{k=1}^{K}\sum_{j=1}^{J}(\hat{\eta}_{kjr} - \eta_{kj})^2}{R \times K \times J}, \tag{3.6}$$

where $\hat{\eta}_{kjr}$ is the estimated topic distribution of document parameter of topic k of document j in replication r, η_{kjr} is the true value of the topic distribution of document parameter of topic k of document j in replication r, R is the number of replications, K is the number of topics, and J is the number of documents.

3.1.4 Document length and Sample Size

LDA was developed to analyze massive corpora of documents, such as might be found on the web. As noted above, the corpus of constructed response answers differs from the texts on the web in that the responses consist of a limited number of words, typically with smaller sample size (i.e., smaller numbers of documents). The effects of document length and sample size on the parameter estimation from this type of corpus have yet to be investigated.

The effects of the ranges of these two factors were determined based on the relatively small sample sizes and more tightly constrained numbers of words that are commonly found in constructed tests. In this way, these factors will reflect practical testing conditions. Therefore, in the simulation study reported here, sample sizes and numbers of unique words were based on empirical data from constructed response (CR) answers obtained from the English and Language Arts (ELA) Assesslet in the online testing program administered by the Georgia Center for Assessment (Assesslet, 2014).

The purpose of the ELA CR items in the Assesslet Testing Program is to evaluate students' reading comprehension knowledge.

Each of the ELA assesslets for each of grades 3 to 11 were designed to measure one of three genres: informative writing, narrative writing, and argumentative writing. Each assesslet is composed of four multiple-choice items, one short answer item and one extended response item in each assesslet. The extended response item is taken as the CR item in this dissertation. Table 1 presents descriptive statistics for sample size and number of unique words for assesslets in each of the grades. Simulation study conditions were based on information in this table.

As suggested in Table 1, the number of unique words appears to increase as grade level increases. Students appear to write more for the argumentative assesslet and less for the informative assesslet.

Selection of Simulation Conditions. The following paragraphs describe the process of selection of the levels of each factor in the simulation study. This process was based on the following preliminary study. The simulation conditions used in this preliminary analysis were as follows: 1) document lengths were 25, 50, 100, 200, and 300 words; 2) numbers of unique words were 150, 300, 500, 1,000, and 3,000 words; 3) sample sizes were 25, 50, and 100 documents. Estimation of the LDA typically is compute intensive, i.e., generally requires a large amount of computing time. Therefore, based on the preliminary analysis, the conditions regarding the number of unique words and document

Table 3.1: *Descriptive Statistics of the CR Corpus from GCA*

Grade & Genre	Number of Documents	Number of Unique words	Average Length of Document
3rd grade			
Informative	3,469	954	42
Narrative	5,559	2,365	45
Argumentative	1,353	622	38
4th grade			
Informative	3,686	1,173	56
Narrative	6,061	2,525	52
Argumentative	835	699	72
5th grade			
Informative	4,319	1,474	81
Narrative	5,350	2,463	81
Argumentative	1,378	1,019	86
6th grade			
Informative	5,080	1,940	105
Narrative	5,520	2,873	123
Argumentative	1,834	1,301	101
7th grade			
Informative	4,639	2,237	146
Narrative	6,594	3,190	98
Argumentative	1,438	1,300	130
8th grade			
Informative	4,751	2,298	162
Narrative	5,860	3,803	166
Argumentative	2,262	1,633	155
9th grade			
Informative	2,341	2,633	227
Narrative	1,590	1,865	121
Argumentative	3,768	3,477	210
11th grade			
Informative	518	1,049	144
Narrative	2,562	5,601	250
Argumentative	1,122	1,394	170

length were reduced to reduce the computing time while at the same time retaining the same levels of accuracy and precision as obtained in the larger preliminary study.

First, with regard to the document length, the analysis suggested that the results were similar for document lengths of 25 and 50 words. Document lengths of 200 and 300 words also yielded similar results.

Thus, the 25 and 200 document length conditions were eliminated from consideration for the simulation study. As a result, the document length conditions of 50, 100, and 300 words were selected as the simulation conditions of the simulation study reported below.

Second, with regard to the number of unique words, the preliminary results showed that results were similar for 150, 300, and 500 unique words. Results obtained from 1,000 and 3,000 words, however, appeared to differ. Therefore, the number of unique words simulated were 500, 1,000, and 3,000 words. With respect to sample size conditions, results in a previous study by Kwak et al. (2018) on the effects of priors suggested that results were similar for sample sizes of 100 and 500 words. Further, results appeared to change only slightly as sample size increased to 1,000. Results for the sample size of 1,000, however, appeared to differ from results for the sample size of 3,000. Thus, a mid-point between 1,000 and 3,000 was selected to determine whether a sample of 1,500 might produce a result closer to 1,000 or 1,500. In addition, as the estimation results appeared to vary only slightly from 500 to 1,000 words, 700 was selected as a minimum sample size. Finally, with a sample size of 3,000, the results appeared to be consistent across levels of the two other other conditions of the preliminary analysis.

3.1.5 Number of Topics

Previous studies applying LDA showed that 20, 50, and 100 topic models were possible (Johri, Wang, Liu and Madhavan, 2011; Canini et al., 2009; Linstead, Lopes, & Baldi, 2008). The results of an LDA analysis of CR data, however, suggested that the number of topics was more limited (Kim et al., 2017). This occurs primarily because the conditions of a test tend to be more constraining with respect to document length and range of words. Kim et al. (2017) and Kwak et al. (2017) both reported that three topic models were a more likely fit to CR data.

Also, the analysis of the CR answers from the GCA Assesslet Testing Program suggested that three-, four-, and five-topic models were good fit to the data (Kwak, Kim, Xiong, Choi, & Cohen, 2018). Specifically, the three-topic model was the best fit for 4th grade narrative, 5th grade informational, 6th grade argumentative, and 8th grade informational assesslets. The four-topic model was the best fit for following the 4th grade argumentative and 3rd grade informational Assesslets. The five-topic model was the best fit for the 9th grade narrative and 9th grade argumentative Assesslets. Based on these results, three-, four-, and five-topic models were used in the simulation study.

3.1.6 Data Generation Process

To reflect the practical setting, the simulated data were generated based on an empirical analysis. The *topicmodels* package (Hornik, & Grün, 2011) was used for the LDA analysis. Data were generated using an R program as outlined below:

1. For a document size, n_j, choose $n_j \sim Poisson(c)$ and generate each document length n_j.
2. For η_j and γ_k,
   (a) Choose a $\eta_j \sim Dirichlet(\alpha)$ and
   (b) Choose a $\gamma_k \sim Dirichlet(\beta)$
3. For each word w_{ij} in document j,
   (a) Choose a topic $z_{ij} \sim Multinomial(\eta_j)$ and
   (b) Choose a word $w_{ij} \sim Multinomial(\gamma_k)$

The process can be explained as follows. The first step is sampling the document lengths from a Poisson distribution. For example, if the document length of the first document (N_1) is sampled as 20, 20 empty elements (i.e., NAs) are assigned to the document. Similarly, if the document length of the second document (N_2) is sampled as 61, 61 empty elements are assigned to the document. Thus, each document can be considered as a vector consisting of a set of empty elements, and the length of the vector is the document length. In the following steps, the topics (z_{ij}) and actual words (w_{ij}) would be assigned to these empty elements. Moreover, the document lengths are sampled as many times as the sample size. For example, if the sample size is 700, 700 sets of empty elements are sampled. Thus, if the condition is that of a sample size of 700 with a document length of 50, 700 vectors consisting of the empty elements will be sampled, and the lengths of the vectors will be determined by the Poisson distribution, with an average of 50.

In the second step, the topic distribution of words of the jth document (η_j) and the word distributions of the kth topic (γ_k) were sampled from Dirichlet distributions, parameterized by α and β, respectively. The α and β were estimated based on the empirical data and using the R package *sirt* (Robitzsch, 2015). In step 2-a), for the empty vectors generated in the first step, the corresponding topic distributions $\eta_j s$ were sampled from $Dirichlet(\alpha)$. For example, if the sample size is 700, $\eta_1, \cdots, \eta_{699}$ and η_{700} are sampled. Also, in step 2-b), γ_ks were sampled from $Dirichletl(\beta)$. For example, if the number of topics is three, and that of unique words 500, γ_1, γ_2 and γ_3 are sampled, and each γ_k is a length of 500, because the length of γ_k implies the number of unique words.

In the third step, the topic assignments and actual words are sampled. In the step 3-a), since every empty vector has its corresponding η_js, the topic assignment can be sampled from $Multinomial(\eta_j)$. For example, for the first document, the topic assignments are sampled from the $Multinomial(\eta_1)$ as many times as N_1. Specifically, if N_1 is 50, the 50 topic assignments would be sampled from $Multinomial(\eta_1)$. Similarly, for the 2nd document, if N_2 is 41, the 41 topic assignments would be sampled from $Multinomial(\eta_2)$.

Likewise, if the 700th document consisted of 31 words, 31 topic assignments would be sampled from $Multinomial(\eta_{700})$. Thus, the results of this step are the sequences of the topic assignments. In the step 3-b), the actual words are sampled from $Multinomial(\gamma_k)$. For example, if a topic assignment is specified as topic 1 in the previous step, the actual words are sampled from the $Multinomial(\gamma_1)$. Likewise, if a topic assignment is specified as topic 2, the actual words are sampled from $Multinomial(\gamma_2)$. This process is applied for all topic assignments from the previous step. Thus, through this process, the simulated data consisting of the actual words are generated.

3.2 Results

3.2.1 Averaged Bias of η

With regard to the averaged bias of η, the results under the different conditions were as follows when the sample size was 700 (see Figure 3.1). The results of the three-topic model with a non-informative prior show the different trends across the different numbers of unique words. When the number of unique words was 500, the magnitude of the averaged bias appears to be consistent over changes in document length. When the numbers of unique words were 1,000 and 3,000, the general tendencies of both conditions appear to be similar. Specifically, the magnitude of the averaged bias appears to decrease, when the document length increased to 100. However, it appears to increase when the document length increased to 300. Under the informative prior condition, the results suggest the magnitude of the averaged bias was consistent across the different document lengths and the different numbers of unique words.

The five-topic model with non-informative prior results appears to show different trends across the different numbers of unique words. When the numbers of unique words were 500 and 1,000, the general tendencies of both conditions appear to be similar. Specifically, the magnitude of the averaged bias appears to decrease as the document length increases. However, when the number of unique words was 3,000, the magnitude of the averaged bias appears to decrease as document length increases to 100 but also appears to remain the same as the document length increased to 300. Under the informative prior condition, the results suggest the magnitude of the averaged bias was consistent across the different document lengths and the numbers of unique words.

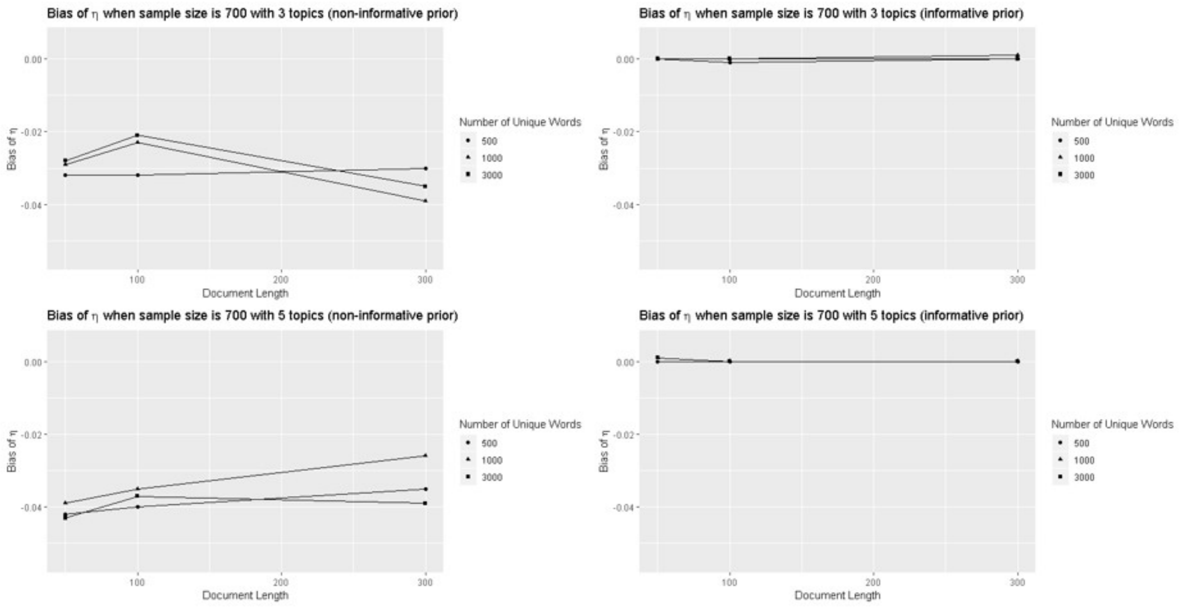

Figure 3.1: The averaged bias of η when sample size is 700.

When the sample size increased to 1,500, the results were as follows (see Figure 3.2). For a three-topic model with a non-informative prior, results appear to suggest different trends across the different numbers of unique words. When the numbers of unique words were 500 and 1,000, the magnitude of the averaged bias appears to be similar as the document length increases. When the number of unique words was 3,000, the magnitude of the averaged bias appears to decrease as the document length increases to 100. However, average bias appears to increase as document length increases to 300. Under the informative prior condition, the results appear to suggest that the magnitude of the averaged bias was consistent across different document lengths and the numbers of unique words.

Results for the five-topic model with a non-informative prior appear to show different trends across the different numbers of unique words. When the numbers of unique words were 500 and 1,000, the magnitude of the averaged bias appears to be similar for different document lengths. However, when the numbers of unique word was 3,000, the magnitude of the averaged bias appears to decrease as the document length increases. Under the informative prior condition, the results appear to show the magnitude of the averaged bias was consistent across different document lengths and numbers of unique words.

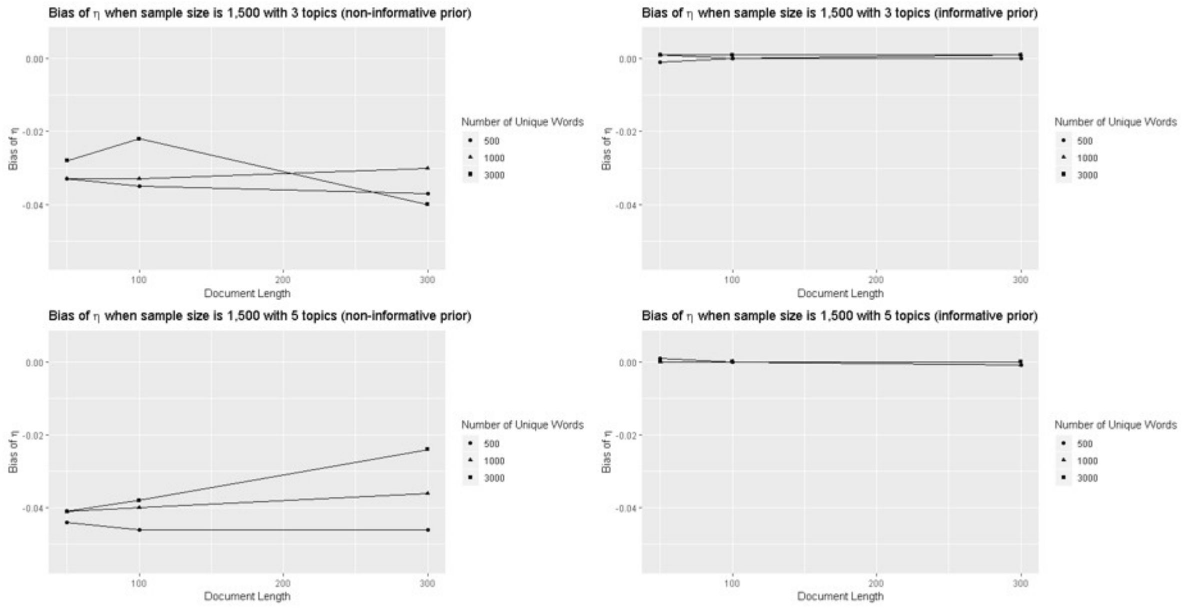

Figure 3.2: The averaged bias of η when sample size 1,500.

When the sample size increased to 3,000, the results were as follows (see Figure 3.3). The three-topic model with non-informative prior results appear to show different trends across the different numbers of unique words. Specifically, when the number of unique words was 500, the magnitude of the averaged bias appears to be the same as document length changes. When the number of unique words was 1,000, the magnitude of the averaged bias appears to decrease as the document length increases up to 100. However, average bias appears to increase as document length increases to 300. When the number of unique words was 3,000, the averaged bias appears to decrease as the document length increases. Under the informative prior condition, the results appear to show the magnitude of the averaged bias is consistent across different document lengths and numbers of unique words.

The five-topic model with non-informative prior results appear to show similar trends across the different numbers of unique words. Specifically, the magnitude of the averaged bias appears to be consistent as document length changes. For the number of unique words, the magnitude of the averaged bias appears to decrease as the number of unique words increases. Under the informative prior condition, the results appear to show that the magnitude of the averaged bias was consistent across the different document lengths and numbers of unique words.

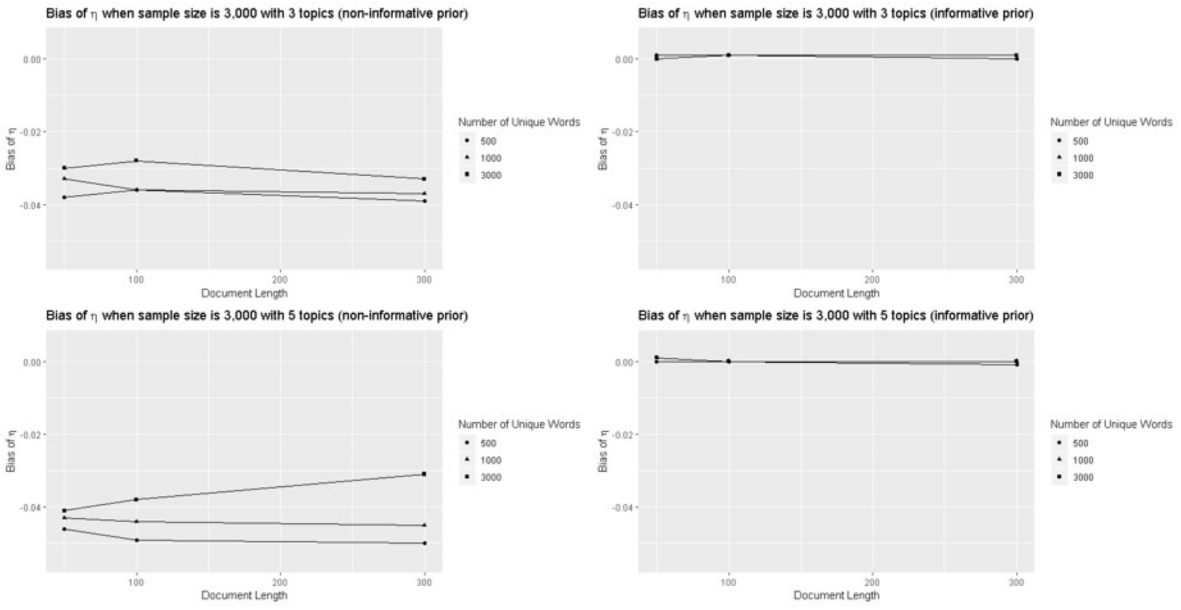

Figure 3.3: The averaged bias of η when sample size 3,000.

3.2.2 RMSD of η

With regard to RMSD, the results under the different conditions when the sample size was 700 are as follows (see Figure 3.4). Results for the three-topic model with non-informative prior appear to show different trends across the different numbers of unique words. When the number of unique words is 500, RMSD appeared to be the same for different document lengths. When the number of unique words was 1,000 and 3,000, the general tendencies of both conditions appeared to be about the same. Specifically, the RMSD appeared to decrease when the document length increased to 100. However, it appeared to increase when the document length increased to 300. Under the informative prior condition, the results appear to show the RMSD was consistent across the different document lengths and the numbers of unique words.

Results for the five-topic model with non-informative priors appear to show different trends across the different numbers of the unique words. Specifically, when the numbers of the unique words were 500 and 3,000, the RMSD appeared to be about the same. For 1,000 unique words, the RMSD appeared to decrease when the document length increased. Under the informative prior condition, the results appear to be similar to those for the three-topic model. Specifically, RMSD appeared to be consistent across the different document lengths and numbers of unique words.

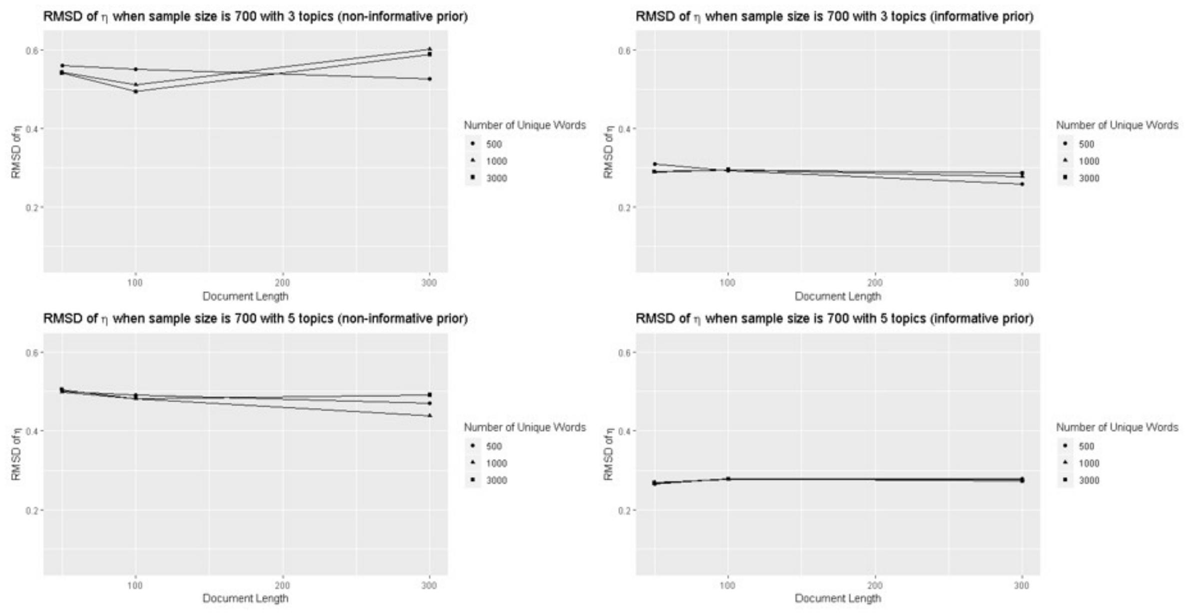

Figure 3.4: RMSD of η when sample size is 700.

When the sample size increased to 1,500, the results were as follows (see Figure 3.5). For the three-topic model with non-informative prior, RMSD results appear to show different trends across the different numbers of unique words. When the numbers of unique words were 500 and 1,000, the general tendencies of both conditions appeared to be similar. Specifically, the RMSD appeared to be the same for different document lengths. When the number of unique words was 3,000, RMSD appeared to decrease when the document length increased to 100. RMSD appeared to increase when the document length increased to 300. Under the informative prior condition, the results appear to show similar trends across the different numbers of unique words. Specifically, the results appear to show that RMSD was consistent across the different document lengths and the numbers of unique words.

The five-topic model with non-informative prior results appeared to show similar trends across the different numbers of unique words. Specifically, when the numbers of the unique words were 500 and 1,000, the RMSD appeared to be about the same. For 3,000 unique words, however, the RMSD appeared to decrease when the document length increased. Under the informative prior condition, the results appear to show that the RMSD was consistent across different document lengths and numbers of unique words.

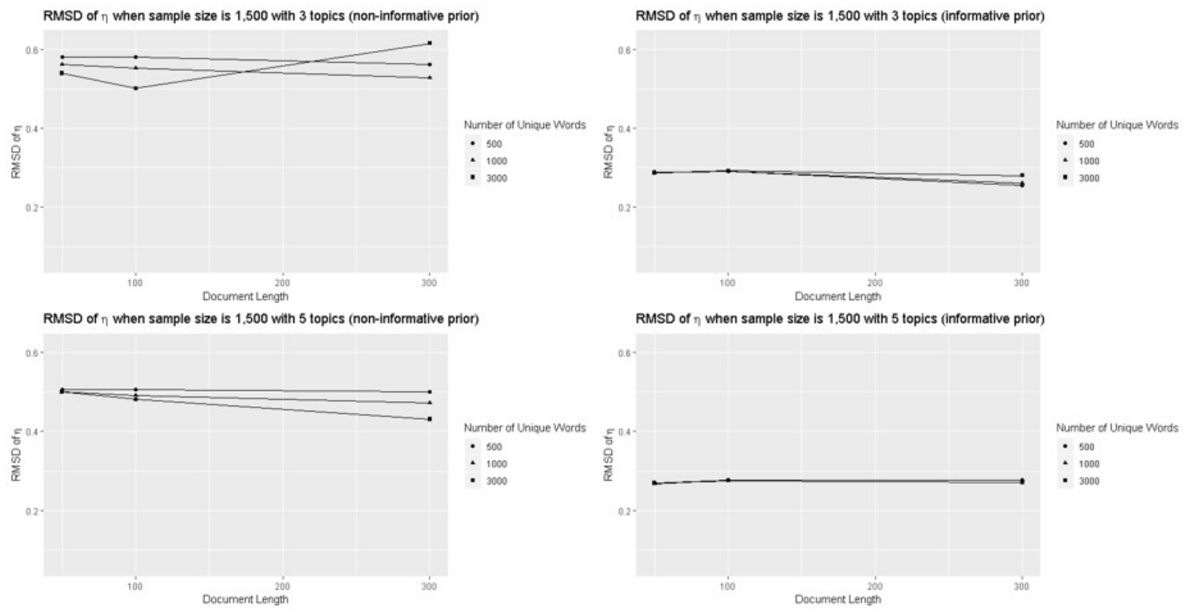

Figure 3.5: RMSD of η when sample size 1,500.

When the sample size increased to 3,000, the RMSD results were as follows (see Figure 3.6). The three-topic model with non-informative prior results appear to show different trends across the different numbers of unique words. When the numbers of unique words were 500 and 3,000, the general tendencies of both conditions appear to be similar. Specifically, RMSD appears to be the same for different document lengths. When the number of unique words was 1,000, RMSD appeared to decrease when the document length was 100. However, it appeared to increase when document length increased to 300. Under the informative prior condition, results appear to suggest that the similar trends existed across different numbers of unique words. Specifically, RMSD appeared to be consistent when document length increase.

For the five-topic model with non-informative priors, results appear to show different trends across the different numbers of unique words. When the numbers of unique words were 500 and 1,000, the general tendencies of both appeared to be similar. That is, RMSD appeared to be consistent across the different document lengths. When the number of unique words was 3,000, RMSD appeared to decrease when the document length increases. Under the informative prior condition, the results appear to show the RMSD was consistent across the different document lengths and the numbers of unique words.

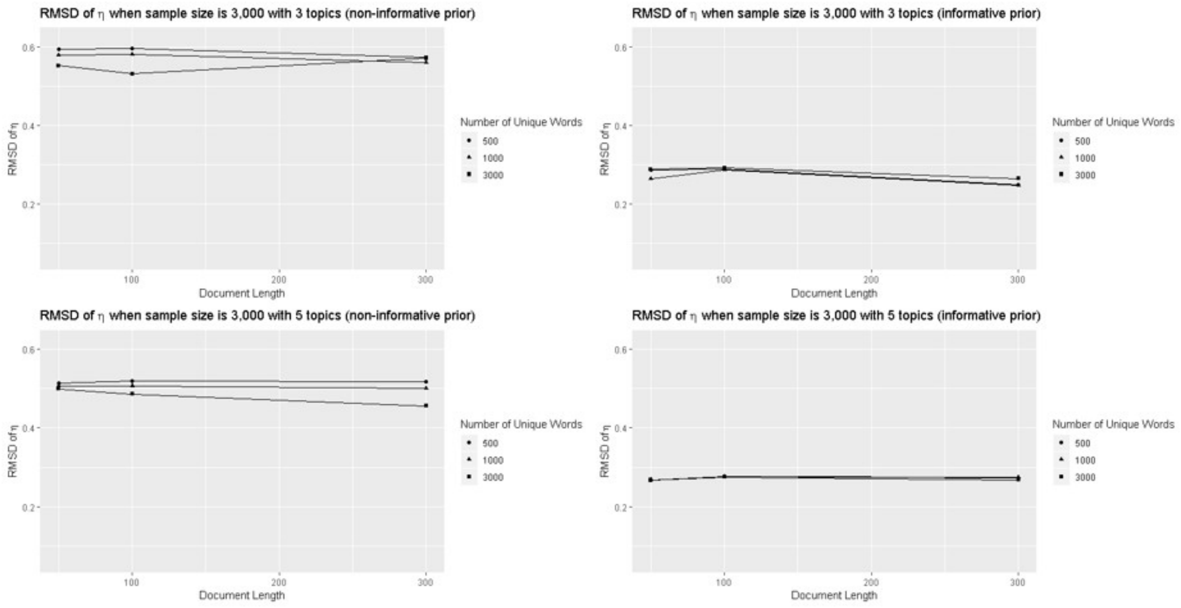

Figure 3.6: RMSD of η when sample size 3,000.

3.2.3 Averaged Bias of γ

With respect to averaged bias of γ, results under the different conditions when the sample size was 700 (see Figure 3.7) for the three-topic model with a non-informative prior appear to show similar RMSDs across different numbers of unique words. The magnitude of the averaged bias of the γ appear to be consistent for document length and the number of unique words. Under the informative prior condition, the results appear to suggest that the magnitude of the averaged bias was consistent across document lengths. For the numbers of unique words, the magnitude of the averaged bias also appears to be consistent over document length.

Results for the five-topic model with non-informative prior appear to suggest similar trends occurred across the different numbers of unique words. The magnitude of the averaged bias of the γ appears to be similar across the different numbers of the unique words. The bias also appears to be consistent for the numbers of unique words. Under the informative prior condition, the results appear to suggest that the magnitude of the averaged bias is consistent across different document lengths and the numbers of unique words.

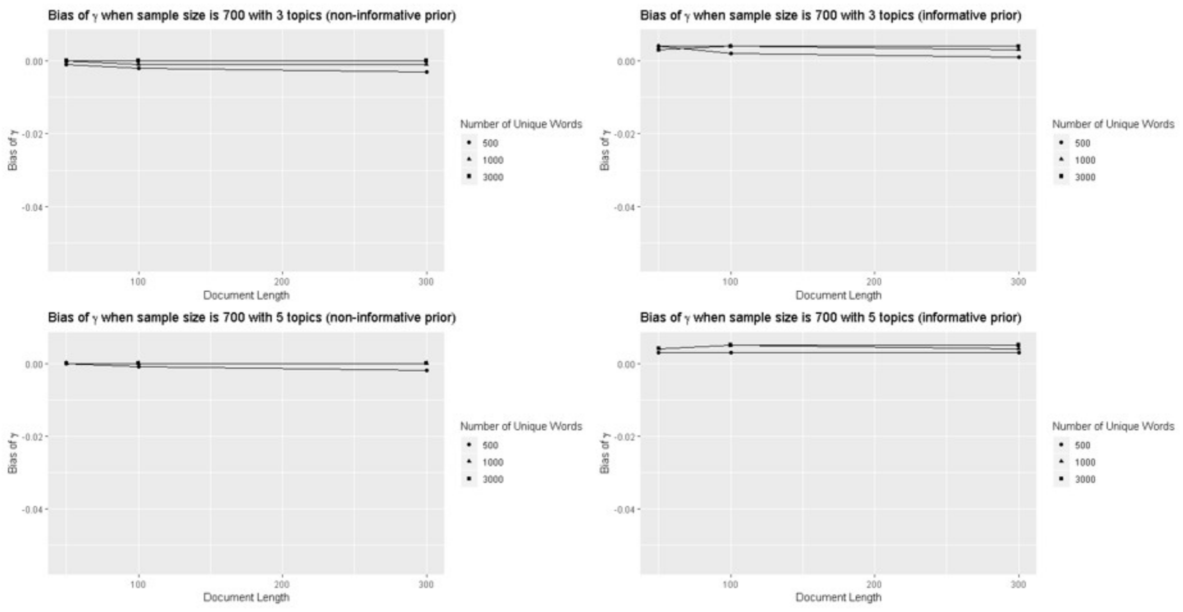

Figure 3.7: The averaged bias of γ when sample size is 700.

With regard to the magnitude of averaged bias of γ, here are the results under the different conditions when the sample size was 1,500 (see Figure 3.8). For the three-topic model with a non-informative prior, results appear to suggest similar trends across different numbers of unique words. The magnitude of the averaged bias of the γ appears to be consistent as the document length and the number of unique words increase. Under the informative prior condition, the results appear to suggest that the magnitude of the averaged bias was consistent across the different document lengths. Also, for the numbers of unique words, the magnitude of the averaged bias appears to be consistent.

Results for the five-topic model with non-informative prior appear to show similar trends across the different numbers of unique words. The magnitude of the averaged bias of the γ appears to be consistent across the different numbers of the unique words. The bias also appears to be consistent for numbers of unique words. Under the informative prior condition, the results show the magnitude of the averaged bias appear to be consistent across different document lengths and the numbers of unique words.

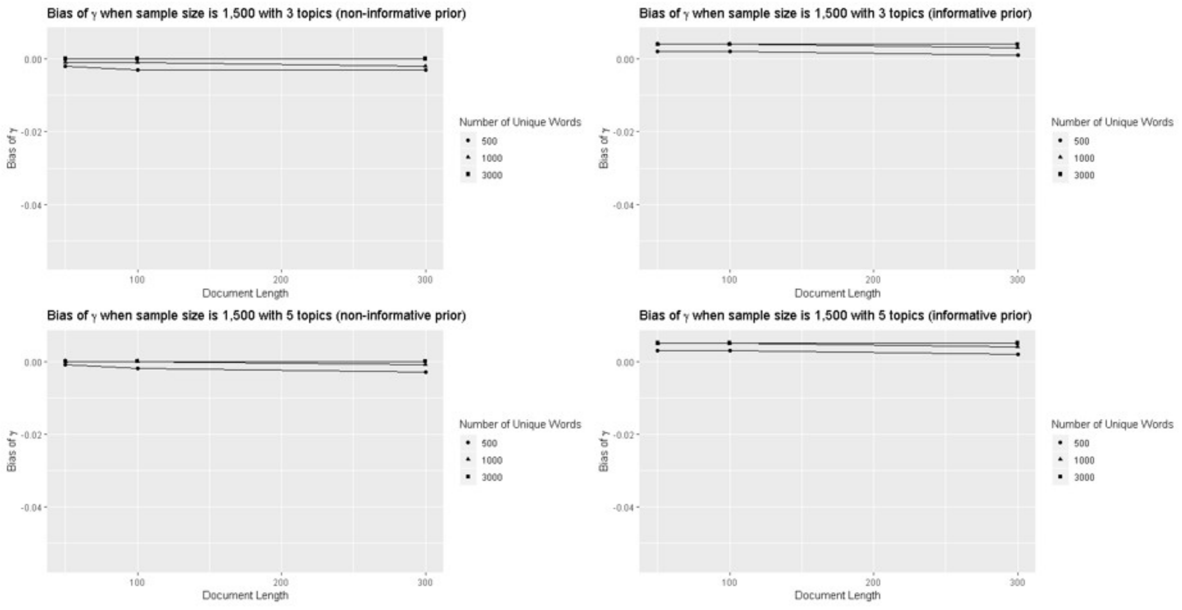

Figure 3.8: The averaged bias of γ when sample size 1,500.

With regard to the averaged bias of γ, the results under the different conditions when the sample size was 3,000 are as follows (see Figure 3.9). For the three-topic model with a non-informative prior, results appear to show similar trends across different numbers of unique words. The magnitude of the averaged bias of the γ appears to be consistent across different numbers of unique words and document lengths. Under the informative prior condition, the results appear to indicate the magnitude of the averaged bias is consistent across the different document lengths and the numbers of the unique words.

The five-topic model with non-informative prior results appear to suggest similar trends across the different numbers of unique words. The magnitude of the averaged bias of the γ appears to be consistent across the different numbers of the unique words and document lengths. Under the informative prior condition, the results appear to show the magnitude of the averaged bias is consistent across different document lengths. Also, for the number of unique words, it appears to be consistent when the number of unique words changes.

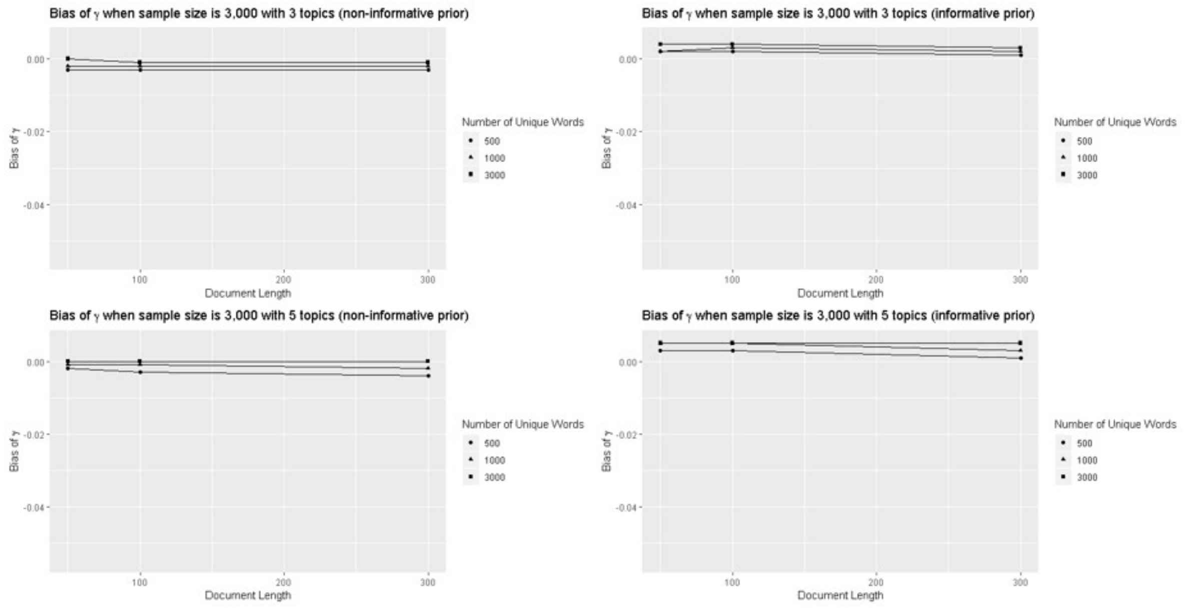

Figure 3.9: The averaged bias of γ when sample size 3,000.

3.2.4 RMSD of γ

With regard to the RMSD of γ, the results under the different conditions when the sample size is 700 are as follows (see Figure 3.10). The results of the three-topic model with non-informative prior appear to show similar trends across the different numbers of unique words. The RMSD of γ appears to be consistent when the document length increases. However, the RMSD appears to decrease when the number of unique words increases. Under the informative prior condition, the results appear to show that the RMSD is consistent across the different document length. Also, for the numbers of unique words, it appears to decrease when the number of unique words increases.

The five-topic model with non-informative prior results appear to show similar trends across the different numbers of unique words. The RMSD of γ appears to be consistent across the different document lengths. However, the RMSD appears to decrease as the number of unique words increases. Under the informative prior condition, the results appear to show that RMSD is consistent across the different document lengths. Also, for the number of unique words, it appears to decrease when the number of unique words increases.

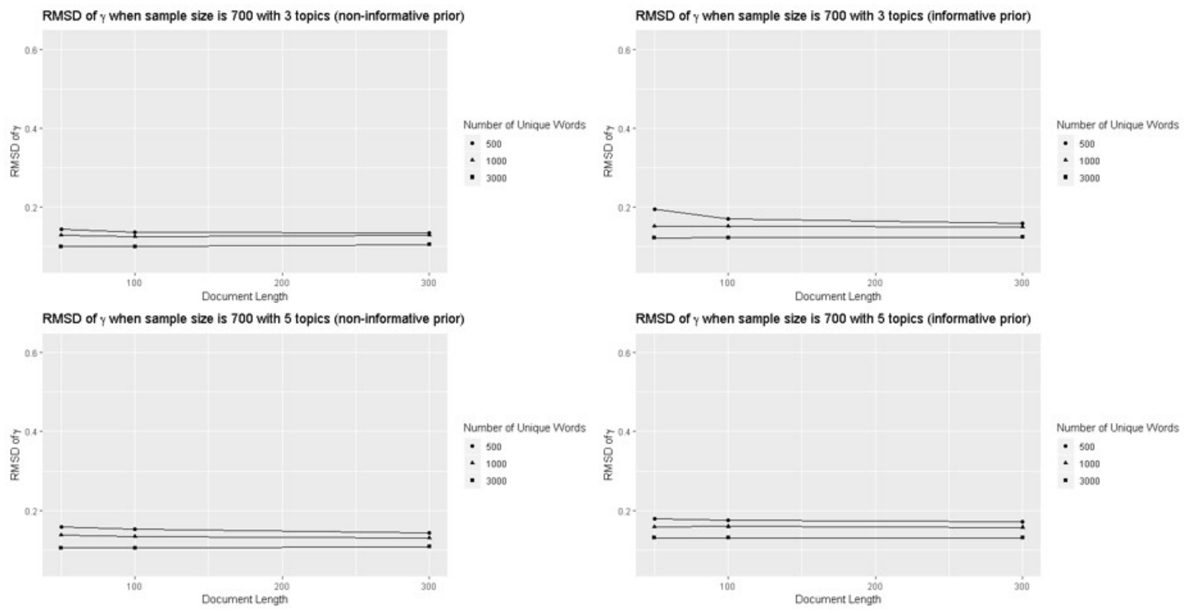

Figure 3.10: RMSD of γ when sample size is 700.

With regard to the RMSD of γ, the results under the different conditions when the sample size is 1,500 are as follows (see Figure 3.11). For the three-topic model with non-informative prior, results appear to show similar trends across the different numbers of unique words. The RMSD of the γ appears to be consistent for document lengths. However, the RMSD appears to decrease when the number of unique words increases. Under the informative prior condition, the results appear to show that the RMSD is consistent across different document lengths. For the numbers of unique words, RMSD appears to decrease as the number of unique words increases.

The five-topic model results with non-informative prior appear to show similar trends across different numbers of unique words. The RMSD of the γ appears to be consistent across different numbers of unique words. The RMSD also appears to decrease as the number of unique words increases. Under the informative prior condition, the results appear to show the RMSD is consistent across different document lengths. Also, for the numbers of unique words, it appears to decrease as the number of unique words increases.

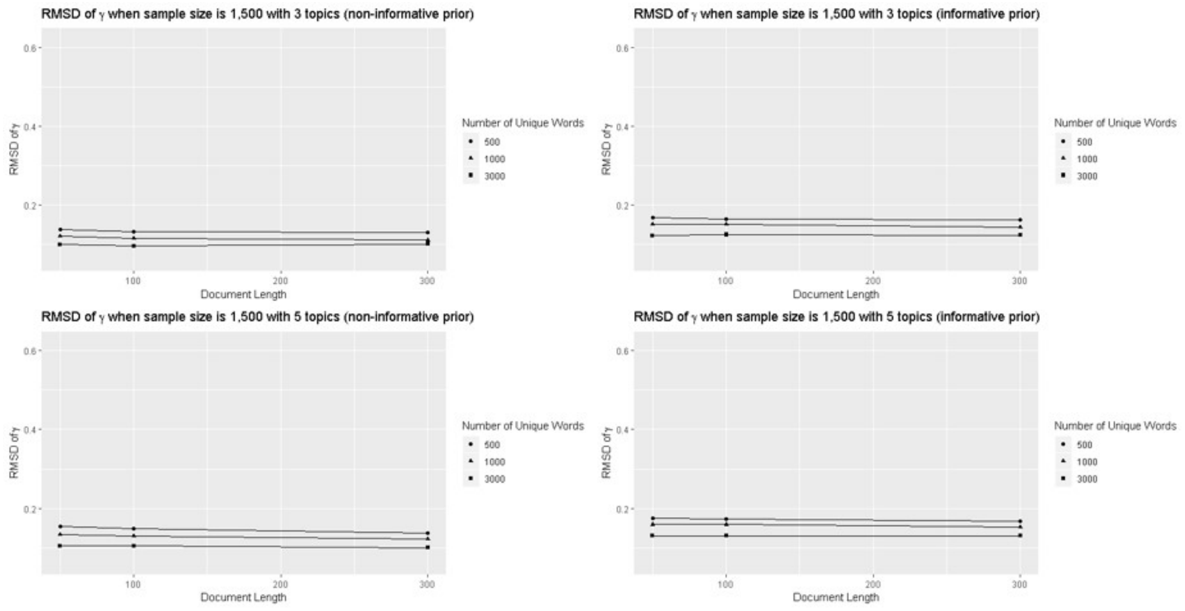

Figure 3.11: RMSD of γ when sample size 1,500.

With regard to the RMSD of γ, the results under the different conditions when the sample size is 3,000 are as follows (see Figure 3.12). For the three-topic model with non-informative prior, results appear to show similar trends across the different numbers of unique words. The RMSD of the γ appears to be consistent when the document length increases. On the other hand, the RMSD appears to decrease as number of unique words increases except for the document length of 100. Under the informative prior condition, the results appear to show that the RMSD is consistent across different document lengths. Also, for the numbers of unique words, it appears to decrease when the number of unique words increases.

The five-topic model with non-informative prior results appear to show similar trends across different numbers of unique words. The RMSD of the γ appears to be consistent across different numbers of unique words. The RMSD also appears to decrease as the number of unique words increases. Under the informative prior condition, the results appear to show that the RMSD is consistent across different document lengths. Also, for the numbers of unique words, the RMSD appears to decrease as the number of unique words increases.

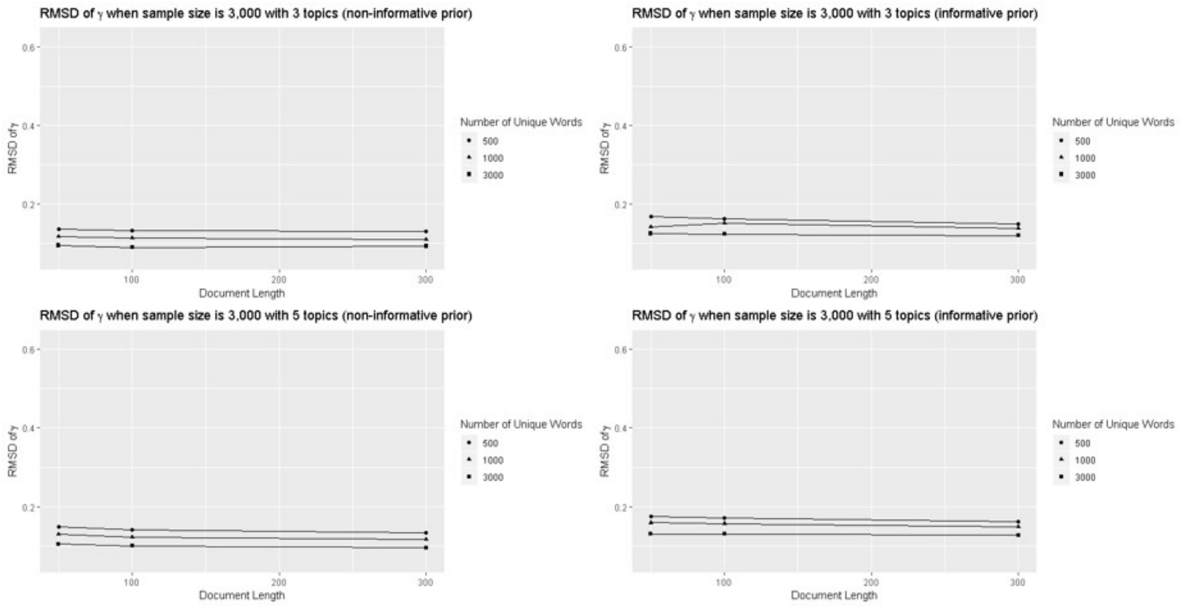

Figure 3.12: RMSD of γ when sample size 3,000.

3.3 Summary and Discussion of the Results

With respect to γ, as the number of unique words increases, the precision also appears to increase. This is expected as the topics become more distinctive as the number of unique words increases. If the topics become distinctive, they overlap less and thus the ambiguity among the topics decreases. Therefore, precision of γ would appear to increase as the number of unique words increases. Similarly, for given number of unique words, the precision appears to slightly decrease as the number of topics increases. This is because the overlap among the topics increases. The two factors, the number of topics and the number of unique words, appear to affect other conditions (i.e., sample size, document length, and prior) in that as the number of topics increases, overlap between topics increases. Also, for a given of topics, as the number of unique words decreases, the overlap between topics appears to increase. Thus, a higher number of topics appears to be associated in the same way as a lower number of unique words. However, the precision appears to be relatively consistent when the number of unique words increases. Also, the γ appears to be robust with regard to other conditions including sample size, document length, and priors.

The ways different conditions affect η is somewhat less straightforward. First, the precision and accuracy of η appear to improve or be consistent as document length increases. This suggests that the precision and accuracy increase as the document length increases. Also, the precision and accuracy did not appear to change even though the length increased from 50 to 300. This suggests that a document length of 50 might be sufficient to obtain stable estimates of the model.

Second, if the sample size is sufficient, the precision and accuracy of η appear to increase as the numbers of unique words and topics increase. In general, accuracy and precision of a model appear to increase when the number of parameters increases.

However, this does not necessarily imply that larger numbers of unique words and topics always yield the most optimal precision or accuracy regardless of the document length. This is because of these appear to interact. That is, if the number of unique words increases, the precision and accuracy of η seem to increase when the document length increased to 100. However, precision and accuracy appear to decrease for document length 300. For example, for the three-topic model with 3,000 unique words, the precision and accuracy appeared to increase up to document length 100. Then a decrease appears to subsequently occur. This suggests the precision and accuracy do not always improve as document length increases. Rather, the precision and accuracy may simply be optimal at a document length of 100. This phenomenon appeared to occur when the number of unique words is large.

The number of the topics and the number of unique words appear to have an interaction effect on other conditions. A larger number of topics appears to have the same effect as the smaller number of unique words on the relationship between the document length and precision and accuracy. That is, when other conditions such as the sample size and the number of unique words are constrained, a smaller number of topics appears likely to yield the pattern as follows: The precision and accuracy increase as document length increases to 100, however, it decreases as document length increases to 300. For example, in the 1,500 sample size, 3,000 unique words, and 3 topic condition, the precision and accuracy become greater as the document length gets longer. However, in the 1,500 sample size, 3,000 unique words, and 5 topic condition, the pattern that precision and accuracy are the best not at the longest but at a certain document length is observed.

To summarize, when the number of unique words is small or the number of the topic is large, the longer a document is the better the precision and accuracy appear to be. On the other hand, the optimal document length seems to be medium when the number of unique words is large or the number of topics is small. Further, if the size of a sample increases, the precision and accuracy of the data seem to be similar to those of a corpus consisting of a smaller number of unique words. Specifically, when the sample size increases, a longer document appears to be better in terms of precision and accuracy even though the number of the unique words is large. For instance, for the three-topic, 1,000 unique words and 700 sample size condition, the medium length document, i.e., 100, appears to show the best precision and accuracy. However, under the same conditions, when the sample size is 1,500, the longer a document is the better the precision and accuracy appear to be. In addition, the prior appears to have an impact on the precision and accuracy of the estimates. Specifically, the informative prior seems to show higher precision and accuracy compared to the estimates from the non-informative priors. Additionally, the precision and accuracy of η seem to be lower than for γ. This latter result is consistent with results from Kwak et al. (2018).

3.4 Empirical Example

The simulation study suggested that LDA results based on the large sample size provided unbiased and precise estimates. In this section, therefore, an example is presented illustrating results of LDA on a large corpus of constructed response test data.

3.4.1 Description of Data

The tests used in this example were designed to provide information on how well students understand concepts and can demonstrate their knowledge and skills in English and Language Arts (ELA). In this regard, the tests were designed as formative measures aligned to the standards for grades 3 to 11 of a large southeastern state.

Items on each test were designed to require extended reasoning and critical thinking beyond basic recall. The tests had three different types of genres: narrative, informational, and argumentative (opinion). The test consisted of three multiple-choice items and two CR items, a short answer item and an extended answer item.

The CR items required integrative writings. That is, the items asked examinees to base their answers on and to integrate words from the passages presented in the item (Weigle & Parker, 2012). The short answer and extended response items differed in what examinees were asked to do. For the short answer, the test asked students to write a description of the main phenomenon or event explained in the passages using specific examples and details from the passages. For the extended response, examinees were asked to rewrite a specific part of the passage using details from the passage.

The text of the answers to the two CR items (i.e., the short answer and the extended answer items) were analyzed for this example. A 3-point rating scale, ranging from 0 to 2 points, was used to score the short answer item. For the extended response item, a 5-point rating scale, ranging from 0 to 4 points, was used. Therefore, the possible range of total scores was from 0 to 6 points.

Tests were administered online through a platform that could be accessed via computers, laptops, tablets, and even smartphones. Tests were untimed but were designed to be administered in a 45-minute class period. The online platform recorded the text of the final answers for each student. Thus, if a student changed an answer, only the final answer was retained as the response. Further, the platform did not provide any syntactical, grammatical or spelling suggestions. Sample size and the scores are presented in Table 3.2.

Table 3.2: *Sample Sizes and Scores for Item and Total Score*

n=5,371	Score (*SD*)
simple CR item score	0.92 (0.622)
extended CR item score	1.54 (0.850)
total score	2.46 (1.245)

3.4.2 Method

Preprocessing of the Textual Data. Stemming is a preprocessing step designed to remove possible ambiguities and to improve the interpretability of the LDA analysis. The process normally results in a reduction of the number of words in the corpus by morphological treatments of the words (Schofield et al., 2017). Typos were also corrected during stemming since different spellings can result in multiple versions of the same word. This would result in each of the different spellings being treated as a unique word and, thus, not being considered in the same term frequency. The TF-IDF value of 0.008 and the resulting words determined to be stopwords in this study are presented in Table 3.3.

Table 3.3: *List of the Stopwords and TF-IDF Score*

TF-IDF score	Stopwords
0.008	and, beach, but, canada, car, for, gabriel, hill, nation, parent, park, parliament, ride, that, the, they, ticket, water, with, come

We provide an example sentence from a student's answer to explain how the three preprocessing steps described in the background section were applied to the data: "*The change throughout Gabriel is that at first for the summer all he wanted to do was surf with his frends every day.*".

First, before applying the preprocessing, all special characters and numbers are removed. Also, in this process, letters are changed to lower-case. The comma was removed, and words such as *The* and *Gabriel* are changed to *the* and *gabriel* respectively. The sentence is converted as follows: "*the change throughout gabriel is that at first for the summer all he wanted to do was surf with his frends every day*".

When tokenization is applied to the sentence, the sentence is broken into individual words as follows: *the, change, throughout, gabriel, is, that, at, first, for, summer, all, he, wanted, to, do, was, surf, with, his, frends, every,* and *day*. Next, the sentence is changed to a set of 23 words.

The normalization converts different forms of a word into a single form if those forms imply the same meaning. The most typical normalization steps include correction of typos, unification of tense, unification of grammatical number (e.g., singular and plural), and unification of pronoun forms (e.g., possessive, objective, and subject). The correction of typos needs to be performed first because if the typos are not corrected, the other three steps cannot be applied. For example, the word *frends* is an explicit typo of a word, *friends*. Thus, if the word is not changed to the correct word, the unification of grammatical number of the correct word cannot be applied. After the correction, a set of words include: *the, change, throughout, gabriel, is, that, at, first, for, the, summer, all, he, wanted, to, do, was, surf, with, his, friend, every,* and *day*. For the unification of tense, the paste tense words are changed to the present tense. In this case, the words such as *wanted* and *was* are changed to *want* and *is*. For the unification of pronoun, the objective or possessive cases are changed to the subject case. For example, the word *his* is changed to *he*. Also, for the unification of grammatical number, the plural words are changed to the singular words. In this case, the word *friends* is changed to *friend*. Thus, after the process, a set of words include: *the, change, throughout, gabriel, is, that, at, first, for, the, summer, all, he, want, to, do, is, surf, with, he, friend, every,* and *day*.

With regard to the stopwords removal, the stopwords (see Table 3.3) are excluded from the set of words. Since the words such as *gabriel* and *the* are excluded, a set of words is as follows: *change, throughout, is, that, at, first, for, the, summer, all, he, want, to, do, is, surf, with, he, friend, every,* and *day*. This set of words is a final preprocessed result of the original sentence *"the change throughout gabriel is that at first for the summer all he wanted to do was surf with his frends every day."*.

3.4.3 Results

Descriptive statistics for the corpus are presented in Table 3.4. Sample sizes of the original corpus and the preprocessed corpus were 5,371 and 5,350, respectively. These consisted of 3,299 and 2,479 unique words, respectively. The average document lengths were 201 and 94, respectively.

Table 3.4: *Descriptive Statistics of the Corpus*

	Number of documents	Number of unique word	Number of total word	Average length
Before preprocessing	5,371	3,299	1,077,327	201
After preprocessing	5,350	2,479	501,104	94

Model selection results are shown in Table 3.5 based on DIC. Specifically, since the four-topic model had the lowest DIC value, the four-topic model was suggested as the best fitting model.

Table 3.5: *DIC Values of the Test by the Number of Topics*

Number of topics	DIC values
2	87416.16
3	85499.20
4	85384.15
5	85847.02
6	86522.31
7	87011.14
8	87313.53
9	87561.40
10	87773.45

The 30 most frequent words for each topic for four-topic model are given in Table 3.6. Topic 1 and Topic 2 contained words such as *have, about, she, see, when, make,* and *people*. These words can be characterized as everyday language. The correlation of Topic 1 with the CR score was very low ($r = -.048$, $p = .018$). Also, the correlation of Topic 2 with the CR score was very low ($r = -.064$, $p < .001$). Topic 3 contained words such as *cruise, canal, boat, kayak,* and *canoe*. This topic also reflects words borrowed from the passages, but the weak negative correlation with score ($r = -.188$, $p < .001$) suggests that these words were simply borrowed and not used appropriately in answering the item. Topic 4 had a moderate positive correlation with the CR score ($r = .391$, $p < .001$). This topic contained words such as *story, change, like, passage, throughout, tell,* and *make*. These words reflect the correct use of integrative borrowing.

Table 3.6: *Top 30 Words and Corresponding Probabilities by Topics with Correlations for Grade 5 Narrative*

	Topic 1 ($r = -.048$, $p = .018$)		Topic 2 ($r = -.064$, $p < .001$)		Topic 3 ($r = -.188$, $p < .001$)		Topic 4 ($r = .391$, $p < .001$)	
1	is	.075	want	.053	canal	.082	story	.034
2	have	.048	say	.049	can	.056	this	.030
3	about	.027	is	.046	late	.049	change	.026
4	boat	.025	dad	.040	boat	.030	you	.025
5	she	.022	get	.036	cruise	.029	like	.022
6	see	.021	there	.035	canoe	.028	passage	.017
7	when	.020	not	.034	back	.027	vacation	.014
8	ottawa	.019	surf	.033	kayak	.027	will	.013
9	new	.018	when	.027	come	.027	throughout	.013
10	friend	.016	mother	.026	know	.026	tell	.012
11	time	.015	then	.024	father	.026	make	.012
12	make	.015	see	.024	bring	.026	best	.012
13	people	.014	because	.023	festivity	.025	say	.012
14	some	.014	friend	.022	down	.025	end	.012
15	age	.014	happy	.019	one	.025	ever	.011
16	plan	.011	he	.018	son	.025	way	.011
17	there	.011	summer	.016	part	.024	what	.011
18	because	.010	have	.015	least	.023	out	.011
19	get	.010	all	.014	head	.022	all	.010
20	also	.009	stay	.013	along	.022	smell	.010
21	lot	.009	can	.012	participate	.021	fun	.010
22	learn	.009	family	.010	surprise	.021	excite	.010
23	food	.008	fun	.010	climb	.021	is	.009
24	family	.008	uncle	.008	notice	.021	how	.009
25	game	.007	aunt	.008	today	.020	think	.009
26	while	.007	day	.007	float	.019	look	.009
27	kid	.007	first	.007	there	.019	not	.009
28	girl	.007	mad	.007	area	.019	trip	.008
29	first	.006	some	.006	view	.019	also	.008
30	tour	.006	boat	.006	exclaim	.017	text	.008

Chapter 4

Conclusion and Discussion

The purpose of this study was to compare the precision and accuracy of estimates of LDA under various conditions. The motivation of this study was to investigate the impact of sample size, document length and informative versus non-informative priors on LDA results from CR responses. LDA was originally developed for analysis of very large corpora collected from a variety of sources. For example, the texts are collected from the web, Twitter messages, and abstracts from journals. Simple extension of use of the LDA based on results from these types of corpora can often provide results that are not useful. As an example, results of previous research suggest that the topic model before and after removal of stopwords can result in different topic model structures (Wilson & Chew, 2010). Specifically, the CR items are relatively constrained because the students answers are limited to the topics related to the contents of the test. This directly impacts the decrease of the number of unique words related to the number of topics. Also, since the test is administered with a time limit, document length is also constrained. These constraints, though appropriate for other corpora, may not be appropriate for a corpus composed of responses to CR items.

It was necessary, therefore, to investigate the precision and accuracy of the estimates from CR items for different factors including the following factors: 1) sample size, 2) document length, 3) number of unique words, 4) priors, and 5) number of topics. The range of sample size was divided into three levels: 700, 1,500, and 3,000. The range of document length was also divided into three levels: 50, 100, and 300. The range of the number of unique words was, again, divided into three levels: 500, 1,000, and 3,000. For the priors, the informative prior suggested by the previous study and the non-informative prior were applied to the model. For the number of topics, three-topic and five-topic models were used.

The simulation results can be summarized as follows. When the sample size was greater than 700, the sample size and the document length appeared to have little impact on the precision and accuracy of the estimates. However, the prior did appear to have an impact on the accuracy and precision of the estimates. Specifically, the informative prior appeared to have a positive effect on the accuracy and precision of the estimates. Informative priors suggested from previous research were $50/K$ for the α (Griffiths & Steyvers, 2004) and $200/V$ for the β (Blei et al., 2003). This α makes the topic distribution more even, and is more appropriate to the estimates than the non-informative prior. Also, for the β, the prior affects the word distribution making it more or less sparse.

For the number of unique words, the results suggested that when the document length and sample size increased, the most appropriate number of unique words was likewise going to increase. However, when the number of topics increased or the informative prior was applied, the impact of the number of unique words appeared to be less.

Thus, from a practical perspective, the simulation results suggested that when the sample size was greater than 700 and the document length exceeded 50 words, the precision and accuracy of the estimates were stable. However, although the number of unique words might have an impact on precision and accuracy of posterior estimates, they appeared to improve as sample size and document length increased. Also, when the informative prior was applied to the model, the impact of the unique words appeared to be less with respect to precision and accuracy. Based on the results from the simulation study, the empirical results suggested that estimates based on the corpus collected from CR items administered by the GCA can be considered as the appropriate estimates. For practical purposes, it appears that the 700 sample size provided similar results to those obtained from the complete data set of more than 3,000 documents. This may be useful when seeking to train a topic model using a small number of documents in order to subsequently analyze large corpora of CR documents.

Bibliography

[1] Assesslet [Computer software]. (2014) , Athens, GA: Georgia center for assessment.

[2] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19,* 716-723.

[3] Attali, Y. (2014). A ranking method for evaluating constructed responses. *Educational and Psychological Measurement, 74,* 795-808.

[4] Behizadeh, N., & Pang, M. E. (2016). Awaiting a new wave: The status of state writing assessment in the United States. *Assessing Writing, 29,* 25-41.

[5] Bíró, I., Siklósi, D., Szabó, J., & Benczór, A. A. (2009). Linked latent dirichlet allocation in web spam filtering. In D. Fetterly, & Z. Gyöngyi (ed), *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web* (pp. 37-40). NY, New York: Association for Computing Machinery.

[6] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research, 3,* 993-1022.

[7] Boyd-Graber, J., Mimno, D., & Newman, D. (2014). Care and feeding of topic models: Problems, diagnostics, and improvements. In Airoldi, Blei, Erosheva, & Stephen (Eds.), *Handbook of mixed membership models and their applications* (pp. 225-255). Retrieved from http://www.people.fas.harvard.edu/~airoldi/pub/books/b02.AiroldiBl eiErosheva Fienberg2014HandbookMMM/Ch12_MMM2014.pdf

[8] Burstein, J. (2003). The e-rater scoring engine: Automated essay scoring with natural language processing. In M. D. Shermis and J. C. Burstein (Eds.), *Automated essay scoring: A cross disciplinary approach* (pp. 113–121). Mahwah, NJ: Lawrence Erlbaum Associates.

[9] Burstein, J., Braden-Harder, L., Chodorow, M., Hua, S., Kaplan, B., Kukich, K., & Wolff, S. (1998). *Computer analysis of essay content for automated score prediction: A prototype automated scoring system for GMAT analytical writing assessment essays*. Princeton, NJ: Educational Testing Service. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/j.2333-8504.1998.tb01764.x/pdf

[10] Burstein, J., Leacock, C., & Swartz, R. (2001). Automated evaluation of essays and short answers, *Proceedings of the 5th International Computer Assisted Assessment Conference*. Loughborough, England: Loughborough University.

[11] Burstein, J., Chodorow, M., & Leacock, C. (2004). Automated essay evaluation: The Criterion online writing service. *Ai Magazine*, *25*(3), 27-27.

[12] Canini, K. R., Suh, B., & Pirolli, P. L. (2011, October). Finding credible information sources in social networks based on content and social structure. In R. Bilof (Eds.), *2011 IEEE International Conference on Privacy, Security, Risk and Trust and 2011 IEEE International Conference on Social Computing* (pp. 1-8). Boston, MA: IEEE.

[13] Chang, J. (2010). Not-so-latent Dirichlet allocation: Collapsed Gibbs sampling using human judgments. In C. Callison-Burch & M. Dredze (Eds.), *Proceedings of NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (pp. 131-138). Stroudsburg, PA: Association for Computational Linguistics.

[14] Chen, C.-F. E., & Cheng, W.-Y. E. (2008). Beyond the design of automated writing evaluation: Pedagogical practices and perceived learning effectiveness in EFL writing classes. *Language Learning & Technology*, *12*(3), 94-112.

[15] Dikli, S. (2006). An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment*, *5*, 4-13.

[16] Foltz, P. W., Laham, D., & Landauer, T. K. (1999). The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 1, 939-944.

[17] Fromkin, V., Rodman, R., & Hyams, N. (2013). *An introduction to language*. MA, Boston: Cengage Learning.

[18] Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*, 5228-5235.

[19] Heinrich, G. (2008). *Parameter estimation for text analysis* (Technical note). Retrived from http://www.arbylon.net/publications/text-est.pdf

[20] Hornik, K., & Grün, B. (2011). topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, *40*, 1-30.

[21] Hurvich, C. M., & Tsai, C. L. (1989). Regression and time series model selection in small samples. *Biometrika*, *76*(2), 297-307.

[22] Kim, S., Kwak, M. & Cohen, A. (2017). A Mixture partial credit model analysis using language-based covariates. In *The annual meeting of the Psychomteric Society* (pp. 321-333). Springer.

[23] Kim, S., Kwak, M., Cardozo-Gaibisso, L., Buxton, C. & Cohen, A.S. Statistical and qualitative analyses of students' answers to a constructed response test of science inquiry knowledge. *Journal of Writing Analytics*, *1*, 82-102.

[24] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In C. S. Mellish (Ed.), *IJCAI*, *Vol. 2*. (pp. 1137-1145). Somerset, NJ: IJCAI

[25] Kwak, M., Kim, S., & Cohen, A. S. (2017, January) Mining students constructed response answers. Paper presented at the *International Conference on Writing Analytics*, Tampa, FL.

[26] Kwak, M., Kim, S., Xiong, J., Choi, H.-J., & Cohen, A. S. (2018, April) Topic model analysis of constructed response items on a formative assessment. Paper presented at the *American Educational Research Association*, Newyork, NY.

[27] Kwak, M., Xiong, J., Kim, S., Choi, H.-J., & Cohen, S. A. (2018, July). Dirichlet priors for latent Dirichlet analysis of constructed response items. Paper presented at the *International Meeting Psychometric Society*, New York, NY.

[28] Lauderdale, B. E., & Clark, T. S. (2014). Scaling politically meaningful dimensions using texts and votes. *American Journal of Political Science*, *58*(3), 754-771.

[29] Lee, I. (2011). Formative assessment in EFL writing: An exploratory case study. *Changing English*, *18*, (1), 99-111.

[30] Lu, K., & Wolfram, D. (2012). Measuring author research relatedness: A comparison of word-based, topic-based, and author co-citation approaches. *Journal of the American Society for Information Science and Technology*, *63*, 1973-1986.

[31] Manning, C.D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

[32] Minka, T., & Lafferty, J. (2002). Expectation-propagation for the generative aspect model. In A. Darwiche, & N. Friedman(Eds.), *Proceedings of the 18th conference on Uncertainty in artificial intelligence* (pp. 352-359). San Francisco, CA: Morgan Kaufmann Publishers.

[33] Page, E. B., Poggio, J. P., & Keith, T. Z. (1997, March). *Computer analysis of student essays: finding trait differences in student profile*. Paper presented at the annual meeting of the American Educational Research Association, Chicago, IL.

[34] Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., & Welling, M. (2008). Fast collapsed Gibbs sampling for latent Dirichlet allocation. In Y. Li, B. Liu & S. Sarawagi (Eds.), *Proceedings of the 14th ACM SIG-KDD International Conference on Knowledge Discovery and Data mining: Vol. 1. Research papers* (pp. 569-577). New York, NY: ACM.

[35] Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for Inverse Document Frequency. *Journal of documentation, 60*, 503-520.

[36] Rosen-Zvi, M., Chemudugunta, C., Griffiths, T., Smyth, P., & Steyvers, M. (2010). Learning author-topic models from text corpora. *ACMTOIS, 28*, 1-33.

[37] Rudner, L., & Gagne, P. (2001). *An overview of three approaches to scoring written essays by computer* (ERIC Digest number ED 458 290).

[38] Schofield, A., Magnusson, M., & Mimno, D. (2017). Pulling out the stops: Rethinking stopword removal for topic models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (Vol. 2, pp. 432-436).

[39] Schwarz, G. (1978), Estimating the dimension of a model, *The annals of statistics, 6*, 461-464.

[40] Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika, 52*(3), 333-343.

[41] Sizov, S. (2012). Latent geospatial semantics of social media. *ACM Transactions on Intelligent Systems and Technology (TIST), 3*(4), 64-84.

[42] Smith, J. I., & Tanner, K. (2010). The problem of revealing how students actually think: Concept inventories and beyond. *Cell Biology Education: Life Science Education, 9*, 1-5.

[43] Spiegelhalter, D., Best, N., Carlin, B., & Van der Linde, A. (2002). Bayesian Measures of Model Complexity and Fit. *Journal of the Royal Statistical Society, 64*(4), 583-639.

[44] Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2014). Studying software evolution using topic models. Science of Computer Programming, 80, 457-479.

[45] Wang, X., Gerber, M. S., & Brown, D. E. (2012, April). Automatic crime prediction using events extracted from twitter posts. In *International conference on social computing, behavioral-cultural modeling, and prediction* (pp. 231-238). Springer, Berlin, Heidelberg.

[46] Weston, M., Parker, J., & Urban-Lurain, M. (2013, April). *Comparing formative feedback reports: Human and automated text analysis of constructed response questions in biology*. Paper presented at the annual conference of the National Association on Research in Science Teaching, Rio Grande, Puerto Rico.

[47] Wilson, A. T., & Chew, P. A. (2010, June). Term weighting schemes for latent Dirichlet allocation. In *human language technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 465-473). Association for Computational Linguistics.

[48] Wolcott, W., & Legg, S. M. (1998). *An overview of writing assessment: Theory, research, and practice.* Urbana, IL: National Council of Teachers of English.

[49] Yano, T., Cohen, W. W., & Smith, N. A. (2009, May). Predicting response to political blog posts with topic models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 477-485). Association for Computational Linguistics.

[50] Yao, L., Zhang, Y., Wei, B., Wang, W., Zhang, Y., Ren, X., & Bian, Y. (2015). Discovering treatment pattern in Traditional Chinese Medicine clinical cases by exploiting

supervised topic model and domain knowledge. *Journal of biomedical informatics, 58,* 260-267.

Appendix A

R code

A.1 Simulation

```
library(tm)
library(MCMCpack)
library(plyr)
library(slam)
library(topicmodels)

sim.cond<-data.frame(doc.length.s=rep(c(50,100,300), 36),
uniq.s=rep(rep(c(500,1000,3000),each=3), 12),
docs.s=rep(rep(c(700,1500,3000),each=9), 4),
prior.s=rep(rep(c(1,2),each=27), 2),
topics.s=rep(c(3,5),each=54)
)
sim.num<-1
current.sim<-sim.cond[sim.num,]

logit<-function(x){asin(sqrt(x))}

k<-3
alpha<-1/2
#delta<-1.0

beta.diri.para<-read.table("beta.diri.para.txt")
gamma.diri.para<-read.table("gamma.diri.para.txt")

simlength<-100
similarity_vector<-rep(c(NA),simlength)
similarity_vector.beta<-rep(c(NA),simlength)
topic1<-matrix(rep(c(NA),30*simlength),nrow=30)
topic2<-matrix(rep(c(NA),30*simlength),nrow=30)
topic3<-matrix(rep(c(NA),30*simlength),nrow=30)
num.doc<-current.sim$docs.s
mean.doclength<-current.sim$doc.length.s
voca.size<-current.sim$uniq.s
voca.id<-seq(1:voca.size)
id<-seq(1:num.doc)

beta.diri.para.fin<-sample(as.vector(unlist(do.call("rbind", rep(list(beta.diri.para),100)))), voca.size)
gamma <- rdirichlet(num.doc, as.numeric(unlist(gamma.diri.para)))
beta <- rdirichlet(3, as.numeric(unlist(beta.diri.para.fin)))
w.beta<-dim(beta)[2]
```

```
bias_vector<-(rep(list(matrix(rep(c(NA),k*num.doc), nrow=num.doc)),simlength))
bias_vector.beta<-(rep(list(matrix(rep(c(NA),k*voca.size), nrow=k)),simlength))
names(beta)<-voca.id
rownames(gamma)<-id

final_rmse_vector.beta<-matrix(rep(c(NA),simlength), nrow=simlength)
final_bias_vector.beta<-matrix(rep(c(NA),simlength), nrow=simlength)

final_rmse_vector.gamma<-matrix(rep(c(NA),simlength), nrow=simlength)
final_bias_vector.gamma<-matrix(rep(c(NA),simlength), nrow=simlength)

for (p in 1:simlength){

colnames(bias_vector.beta[[p]])<-voca.id

doclength<-round(abs(rnorm(num.doc,mean.doclength,20))+1)
m<-length(doclength)

doc.set<-list()
for (j in 1:m){
doc<-rep(NA,doclength[j])
doc<-list(doc)
doc.set[j]<-doc
}

for (j in 1:m){
for (i in 1:doclength[j]){
sel.topic<-which.max(rmultinom(1, 1, gamma[j,]))
sel.word<-which.max(rmultinom(1, 1, beta[sel.topic,]))
doc.set[[j]][i]<-names(beta)[sel.word]
}
}

wftable<-sort(table(unlist(doc.set)), decreasing=T)
mean(sort(table(unlist(doc.set)), decreasing=T))
sd(sort(table(unlist(doc.set)), decreasing=T))
median(sort(table(unlist(doc.set)), decreasing=T))
sum(sort(table(unlist(doc.set)), decreasing=T))
wftable<-sort(table(unlist(doc.set[1])), decreasing=T)
#sum(wftable[1:30])
text00<-doc.set
## convert document term vectors to frequency vectors

text01<-list()
for (i in 1:num.doc){
text01[[i]]<-paste(text00[[i]], collapse=" ")
}

tm0_total<-Corpus(VectorSource(text01))
pstP_dtm0_total<-DocumentTermMatrix(tm0_total)
pstP_dtm0_total
class(tm0_total)
tm0_total[[2]]
pstP_dtm0_total$ncol
pstP_dtm0_total$nrow
```

```
#select words that have more than 2 frequency
two0_total<-findFreqTerms(pstP_dtm0_total,5)
pstP_two.total<-pstP_dtm0_total
term_tfidf <-tapply(pstP_two.total$v/row_sums(pstP_two.total)[pstP_two.total$i], pstP_two.total$j, mean)
*log2(nDocs(pstP_two.total)/col_sums(pstP_two.total > 0))
summary(term_tfidf)

#After removing stop words based on value of tf-idf#
#cv<-as.numeric(term_tfidf[order(term_tfidf)[30]])
cv<-quantile(term_tfidf,0.00)
pstP_two2.total <- pstP_two.total[,term_tfidf >= cv]
pstP_two3.total <- pstP_two2.total[row_sums(pstP_two2.total) > 0,]

empty.doc.list<-which(row_sums(pstP_two2.total)==0, TRUE)
empty.doc.list<-as.numeric(empty.doc.list)
pstP_two3.total <- pstP_two3.total[col_sums(pstP_two3.total) > 0,]
W.total<-pstP_two3.total$ncol
#sum(pstP_two3.total$v)
#W.total

#stopword list
stvoca.list<-pstP_two.total[,term_tfidf < cv]
#stvoca.list$dimnames$Terms

###################################
###########Estimation#############
###################################

# Setting estimation conditions

iter<-20000
keep<-1
burnin<-5000

#set up library
# generate numerous topic models with different numbers of topics in this case a sequence of numbers from 2 to 10, by ones.
#alpha.vec<-c(1.8,1.9,2.0,2.1,2.2,2.3,2.4,2.5)
#alpha<-alpha.vec[j]

#Extracting

k<- 3 #number of topics
#alpha<-17 #dirichlet prior
#delta<-200/W.total #dirichlet prior
delta<-1/2
#alpha<-1/k #dirichlet prior

SEED<-7845652
m <-LDA(pstP_two3.total, k=k, method = "Gibbs", control=list(alpha=alpha, delta=delta, seed=SEED, burnin=burnin, iter=20000))
m@terms
mk<-m@wordassignments
gamma_sim<-m@gamma
beta_sim<-m@beta
colnames(beta_sim)<-m@terms
rownames(gamma_sim)<-rownames(pstP_two3.total)
```

```
n1<-table(m@z)[1]
n2<-table(m@z)[2]
n3<-table(m@z)[3]

##simple one
Topic_k<-topics(m,3)
Terms_k<-terms(m,30)
term.prob<-posterior(m, pstP_two3.total)$terms
data_post<-m@gamma # topic-document distribution
#write.table(data_post, "C:\\2016 AERA\\mydata_post.txt", sep="\t")

#term.prob<-posterior(m, pstP_two.total)$terms

#Support of the phi
list.m.t1<-order(term.prob[1,], decreasing=TRUE)[1:8]
list.m.t2<-order(term.prob[2,], decreasing=TRUE)[1:8]
list.m.t3<-order(term.prob[3,], decreasing=TRUE)[1:8]

#Posterior distribution of phi
m.t1<-term.prob[1,list.m.t1]
m.t2<-term.prob[2,list.m.t2]
m.t3<-term.prob[3,list.m.t3]

data1<-round(as.data.frame(m.t1),3)
data2<-round(as.data.frame(m.t2),3)
data3<-round(as.data.frame(m.t3),3)

name1<-rownames(data1)
name2<-rownames(data2)
name3<-rownames(data3)

## topic assignment arrange ##
beta<-as.data.frame(beta)
beta.t<-as.data.frame(t(beta))
rownames(beta.t)<-voca.id
dim(beta.t)
head(beta.t)
beta_sim.t<-as.data.frame(t(exp(beta_sim)))
dim(beta_sim.t)
head(beta_sim.t)

row.names(beta.t)
# check the merge for beta
beta_merge<-merge(beta.t,beta_sim.t, by=0, all.x=TRUE)
beta_merge[is.na(beta_merge)]<-0
dim(beta_merge)
head(beta_merge)
beta_merge[beta_merge$Row.names=="melt",]
beta_sim.t[rownames(beta_sim.t)=="melt",]
beta.t[rownames(beta.t)=="melt",]

#label sim.1.beta
beta.sim.index.11<-cor(beta_merge[,1+1],beta_merge[,k+2])
beta.sim.index.21<-cor(beta_merge[,1+2],beta_merge[,k+2])
beta.sim.index.31<-cor(beta_merge[,1+3],beta_merge[,k+2])
label.t1.sim<-which.max(c(beta.sim.index.11,beta.sim.index.21,beta.sim.index.31))
```

```
#label sim.2.beta
beta.sim.index.12<-cor(beta_merge[,1+1],beta_merge[,k+3])
beta.sim.index.22<-cor(beta_merge[,1+2],beta_merge[,k+3])
beta.sim.index.32<-cor(beta_merge[,1+3],beta_merge[,k+3])
label.t2.sim<-which.max(c(beta.sim.index.12,beta.sim.index.22,beta.sim.index.32))

#label sim.3.beta
beta.sim.index.13<-cor(beta_merge[,1+1],beta_merge[,k+4])
beta.sim.index.23<-cor(beta_merge[,1+2],beta_merge[,k+4])
beta.sim.index.33<-cor(beta_merge[,1+3],beta_merge[,k+4])
label.t3.sim<-which.max(c(beta.sim.index.13,beta.sim.index.23,beta.sim.index.33))

## rearrange gamma ##
gamma_1<-gamma[,label.t1.sim]
gamma_2<-gamma[,label.t2.sim]
gamma_3<-gamma[,label.t3.sim]
gamma.re<-cbind(gamma_1,gamma_2,gamma_3)

## rearrange beta ##
beta_1<-beta_merge[,1+label.t1.sim]
beta_2<-beta_merge[,1+label.t2.sim]
beta_3<-beta_merge[,1+label.t3.sim]
beta.re<-rbind(beta_1,beta_2,beta_3)

beta.re.size<-dim(beta.re)[2]
colnames(beta.re)<-beta_merge$Row.names
#beta$your

# check the merge for gamma
gamma_merge<-merge(gamma.re,gamma_sim, by=0, all.x=TRUE)
gamma_merge<-gamma_merge[complete.cases(gamma_merge),]

#beta RMSE bias
mean_rmse_vector.beta<-matrix(rep(c(NA),beta.re.size*k), nrow=k)
colnames(mean_rmse_vector.beta)<-beta_merge$Row.names
beta_merge[beta_merge==0] <- NA
beta_merge<-beta_merge[complete.cases(beta_merge),]
rmse_vector.beta..1<-beta_merge[,(1+1):(k+1)]
beta..1<-beta_merge[,(k+1+1):(2*k+1)]
rmse.step<-round(sqrt(sum((logit(rmse_vector.beta..1)-logit(beta..1))^2)/(k*w.beta)),3)
final_rmse_vector.beta[p,]<-rmse.step
mean_bias_vector.beta<-matrix(rep(c(NA),beta.re.size*k), nrow=k)
bias.step<-round((sum((logit(rmse_vector.beta..1)-logit(beta..1)))/(k*w.beta)),3)
final_bias_vector.beta[p,]<-bias.step
final_bias_vector.beta

#gamma RMSE bias
mean_rmse_vector.gamma<-matrix(rep(c(NA),num.doc*k), nrow=num.doc)
rmse_vector.gamma..1<-gamma_merge[,(k+2):(2*k+1)]
gamma..1<-gamma_merge[,(1+1):(k+1)]
rmse.step.g<-round(sqrt(sum((logit(rmse_vector.gamma..1)-logit(gamma..1))^2)/(k*num.doc)),3)
final_rmse_vector.gamma[p,]<-rmse.step.g
mean_bias_vector.gamma<-matrix(rep(c(NA),num.doc*k), nrow=num.doc)
bias.step.g<-round((sum((logit(rmse_vector.gamma..1)-logit(gamma..1)))/(k*num.doc)),3)
final_bias_vector.gamma[p,]<-bias.step.g
```

```
}

rmse.gamma.fin<-round(sqrt(sum(final_rmse_vector.gamma)/simlength),3)
bias.gamma.fin<-round((sum(final_bias_vector.gamma)/simlength),3)
sum(final_bias_vector.beta)
rmse.beta.fin<-round(sqrt(sum(final_rmse_vector.beta)/simlength),3)
bias.beta.fin<-round((sum(final_bias_vector.beta)/simlength),3)
final.table<-cbind(rmse.gamma.fin, bias.gamma.fin, rmse.beta.fin, bias.beta.fin)
write.table(final.table, paste("final.table.",sim.num,".txt", seq=""))
```

A.2 Empirical analysis

```
rm(list=ls()) #remove earlier working space
library(tm)
library(slam)
library(lda)
library(topicmodels)

dat0 = read.csv(file='C:\\Users\\mk59520\\Desktop\\GCA_text_analysis\\Data_0928\\text/ELA_Grade_9_Argumantative_Assesslet_noID_text.csv',
header=T, sep=",", fill=T)
#dat0<-scan("C:\\Users\\mk59520\\Desktop\\GCA text analysis\\lissel\\2_Pink_post_final_wide2.txt", what = "character", sep = ",")
tr0 = read.csv(file='C:\\Users\\mk59520\\Desktop\\GCA_text_analysis\\Data_0928\\text/ELA_Grade_7_Informational_Assesslet_no_ID_text.csv',
header=T, sep=",", fill=T)
na.list = substr(dat0$text,1,2) == "NA" #delete missing data
dat0 = dat0[na.list==F,]
#na.list2 = substr(dat0$text,1,2) == "" #delete missing data
#dat0 = dat0[na.list2==F,]
head(dat0)
#merge part
score_id_dat<-dat0[,2:3]
head(score_id_dat)
names(score_id_dat)
score_id_trait<-tr0[,3:5]
names(score_id_trait)[1]
head(score_id_trait)
total <- merge(score_id_dat,score_id_trait,by="ScoreID")
head(total)
total_1<-total[,1:4]
names(total_1)

#not merge part
head(dat2)
dat1 = dat0[as.character(dat0$text)!="",] #delete blank records
dat2 = dat1[as.character(dat1$text)!=" ",] #delete blank records
score1 = total_1[,3]
score2 = total_1[,4]
score<-score1+score2

#change sample size
max<-dim(dat2)[1]
sample_size<-max
id_doc<-seq(1,sample_size)
id_doc<-sample(id_doc, 700)
start_size<-max-sample_size+1
#text1<-dat2$text
```

```
text1<-dat2$text
text1<-text1[!is.na(text1)]
text1 <- gsub("^[[:space:]]+", "", text1) # remove whitespace at beginning of documents
text2 <- gsub("[[:space:]]+$", "", text1) # remove whitespace at end of documents
text3 =sub("\\.", "␣", text2) #remove periods
text4 = gsub("\\'s", "", text3) #remove "'s"
text5 = gsub("[[:punct:]]", "␣", text4) #remove punctuation characters
text6 = gsub("[[:digit:]]", "␣", text5) #remove digits
text7 <- gsub("^[[:space:]]+", "", text6) # remove whitespace at beginning of documents
text7 = tolower(text7) #to lower case
text7[10]
#text7<-text7[start_size:max]
text7<-text7[id_doc]
length(text7)
text7[2]
#sampling documents
#set.seed(15423)
#text7<-sample(text7,1500, replace=F)
doc.list <- strsplit(text7, "[[:space:]]+")
doc.unlist<-unlist(doc.list)
#delete stop words and words that apprear less than 5 times
term.table = table(doc.unlist)
del  = term.table < 5
new.term.table <- term.table[!del]
vocab <- names(new.term.table)
length(vocab)
new.term.table
#write.table(vocab, file="C:\\Users\\mk59520\\Desktop\\GCA text analysis\\Data_0928/vocab/3narra.txt")
#get document length
d.length = unlist(lapply(doc.list,length))
new.doc.list=list()
s=0
for(l in 1:length(d.length)){
e = s + d.length[l]
new.doc.list[[l]] = doc.unlist[(s+1):e]
s = e
}

# now put the documents into the format required by the lda package:
get.terms <- function(x) {
index <- match(x, vocab)
index <- index[!is.na(index)]
rbind(as.integer(index - 1), as.integer(rep(1, length(index))))
}
documents <- lapply(new.doc.list, get.terms)
del.doc = which(sapply(documents,length)<0) #delete documents that have length less than 10
id_doc_new<-id_doc[-del.doc]
documents.new = documents[-del.doc]
documents.new = documents

# Compute some statistics related to the data set before preprocessing:
D <- length(documents.new)  # number of documents
W <- length(vocab)  # number of terms in the vocab
doc.length <- sapply(documents.new, function(x) sum(x[2, ]))  # number of tokens per document
N <- sum(doc.length)  # total number of tokens in the data
term.frequency <- as.integer(new.term.table)  # frequencies of terms in the corpus
```

```
D
W
N
N/D
sd(doc.length)

#####################delete words using tf-idf##################################

rm(list=ls()) #remove earlier working space
library(tm)
library(slam)
library(lda)
library(topicmodels)

dat0 = read.csv(file='C:\\Users\\mk59520\\Desktop\\GCA_text_analysis\\Data_0928\\text/ELA_Grade_5_Narrative_Assesslet_noID_text.csv',
header=T, sep=",", fill=T)
#dat0<-scan("C:\\Users\\mk59520\\Desktop\\GCA text analysis\\lissel\\2_Pink_post_final_wide2.txt", what = "character", sep = ",")
tr0 = read.csv(file='C:\\Users\\mk59520\\Desktop\\GCA_text_analysis\\Data_0928\\text/ELA_Grade_7_Informational_Assesslet_no_ID_text.csv',
header=T, sep=",", fill=T)
na.list = substr(dat0$text,1,2) == "NA" #delete missing data
dat0 = dat0[na.list==F,]
#na.list2 = substr(dat0$text,1,2) == "" #delete missing data
#dat0 = dat0[na.list2==F,]
head(dat0)
#merge part
score_id_dat<-dat0[,2:3]
head(score_id_dat)
names(score_id_dat)
score_id_trait<-tr0[,3:5]
names(score_id_trait)[1]
head(score_id_trait)
total <- merge(score_id_dat,score_id_trait,by="ScoreID")
head(total)
total_1<-total[,1:4]
names(total_1)

#not merge part
head(dat2)
dat1 = dat0[as.character(dat0$text)!="",] #delete blank records
dat2 = dat1[as.character(dat1$text)!="_",] #delete blank records
score1 = total_1[,3]
score2 = total_1[,4]
score<-score1+score2

#change sample size
max<-dim(dat2)[1]
sample_size<-max
id_doc<-seq(1,sample_size)
id_doc<-sample(id_doc, 700)
start_size<-max-sample_size+1
#text1<-dat2$text
text1<-dat2$text
text1<-text1[!is.na(text1)]
text1 <- gsub("^[[:space:]]+", "", text1) # remove whitespace at beginning of documents
text2 <- gsub("[[:space:]]+$", "", text1) # remove whitespace at end of documents
text3 =sub("\\.", "_", text2) #remove periods
```

```
text4 = gsub("\\'s", "", text3) #remove "'s"
text5 = gsub("[[:punct:]]", " ", text4) #remove punctuation characters
text6 = gsub("[[:digit:]]", " ", text5) #remove digits
text7 <- gsub("^[[:space:]]+", "", text6) # remove whitespace at beginning of documents
text7 = tolower(text7) #to lower case
text7[10]
#text7<-text7[start_size:max]
text7<-text7[id_doc]
length(text7)
text7[2]

#sampling documents
#set.seed(15423)
#text7<-sample(text7,1500, replace=F)
doc.list <- strsplit(text7, "[[:space:]]+")
doc.unlist<-unlist(doc.list)
#delete stop words and words that apprear less than 5 times
term.table = table(doc.unlist)
del  = term.table < 5
new.term.table <- term.table[!del]
vocab <- names(new.term.table)
length(vocab)
new.term.table
#write.table(vocab, file="C:\\Users\\mk59520\\Desktop\\GCA text analysis\\Data_0928/vocab/3narra.txt")
#get document length
d.length = unlist(lapply(doc.list,length))
new.doc.list=list()
s=0
for(l in 1:length(d.length)){
e = s + d.length[l]
new.doc.list[[l]] = doc.unlist[(s+1):e]
s = e
}

# now put the documents into the format required by the lda package:
get.terms <- function(x) {
index <- match(x, vocab)
index <- index[!is.na(index)]
rbind(as.integer(index - 1), as.integer(rep(1, length(index))))
}
documents <- lapply(new.doc.list, get.terms)
del.doc = which(sapply(documents,length)<0) #delete documents that have length less than 10
id_doc_new<-id_doc[-del.doc]
documents.new = documents[-del.doc]
documents.new = documents

# Compute some statistics related to the data set before preprocessing:
D <- length(documents.new)  # number of documents
W <- length(vocab)  # number of terms in the vocab
doc.length <- sapply(documents.new, function(x) sum(x[2, ]))  # number of tokens per document
N <- sum(doc.length)  # total number of tokens in the data
term.frequency <- as.integer(new.term.table)  # frequencies of terms in the corpus
D
W
N
N/D
```

```
sd(doc.length)

######################delete words using tf-idf###################################

#load stemming list
source("C:\\Users\\mk59520\\Desktop\\GCA_text_analysis\\Data_0928\\source\\stemming/5_Narrative_sub_customize_Stemming_lda.R")
#source("C:\\Users\\mk59520\\Desktop\\GCA text analysis\\lissel\\sub_customize_Stemming.R")

#load stopwords
#source("C:\\Users\\mk59520\\Desktop\\GCA text analysis\\Data_0928\\source\\stopword/sub_stopwords_7th_Narrative.R")
d.length = unlist(lapply(doc.list,length))
new.doc.list=list()
s=0
for(l in 1:length(d.length)){
e = s + d.length[l]
new.doc.list[[l]] = doc.unlist[(s+1):e]
s = e
}

# now put the documents into the format required by the lda package:
get.terms <- function(x) {
index <- match(x, vocab)
index <- index[!is.na(index)]
rbind(as.integer(index - 1), as.integer(rep(1, length(index))))
}
documents <- lapply(new.doc.list, get.terms)
del.doc = which(sapply(documents,length)<10) #delete documents that has length less than 10
documents.new_1 = documents[-del.doc]
#problem part: if you got error, correct here
#documents.new_1 = documents
assign<-function(x){vocab[x[1,]+1]}
text7<-lapply(documents.new_1, assign)
text7<-lapply(text7, toString)
text7 <- gsub("^[[:space:]]+", "", text7) # remove whitespace at beginning of documents
text7 <- gsub("[[:space:]]+$", "", text7) # remove whitespace at end of documents
text7 =sub("\\.", "_", text7) #remove periods
text7 = gsub("\\'s", "", text7) #remove "'s"
text7 = gsub("[[:punct:]]", "_", text7) #remove punctuation characters
text7 = gsub("[[:digit:]]", "_", text7) #remove digits
text7 <- gsub("^[[:space:]]+", "", text7) # remove whitespace at beginning of documents
text7 = tolower(text7) #to lower case

#replaceSynonysm function
replaceSynonyms <- content_transformer(function(x, syn=NULL) {
Reduce(function(a,b) {
gsub(paste0("\\b(", paste(b$syns, collapse="|"),")\\b"), b$word, a)}, syn, x)
})
tm0_total<-Corpus(VectorSource(text7))

#Stemming document (customized)
tm0_total <- tm_map(tm0_total, replaceSynonyms, synonyms)
pstP_dtm0_total<-DocumentTermMatrix(tm0_total, control=list(stopwords = FALSE,stemming=F, minWordLength = 1))
#select words that have more than 2 frequency
two0_total<-findFreqTerms(pstP_dtm0_total,5) #
pstP_two.total<-pstP_dtm0_total
term_tfidf <-tapply(pstP_two.total$v/row_sums(pstP_two.total)[pstP_two.total$i], pstP_two.total$j, mean)*log2(nDocs(pstP_two.total)/
col_sums(pstP_two.total > 0))
summary(term_tfidf)
quantile(term_tfidf,0.1)
medi<-unlist(summary(term_tfidf)[2])
a<-sort(term_tfidf)
barplot(a)

#After removing stop words based on value of tf-idf#
#cv<-as.numeric(term_tfidf[order(term_tfidf)[30]])
cv<-quantile(term_tfidf,0.04)
pstP_two2.total <- pstP_two.total[,term_tfidf >= cv]
pstP_two3.total <- pstP_two2.total[row_sums(pstP_two2.total) > 1,]
pstP_two3.total <- pstP_two3.total[col_sums(pstP_two3.total) > 1,]
W.total<-pstP_two3.total$ncol
sum(pstP_two3.total$v)

#stopword list
stvoca.list<-pstP_two.total[,term_tfidf < cv]
stvoca.list$dimnames$Terms

##making training##
tm0_training<-Corpus(VectorSource(text7[1:round(length(text7)*0.6)]))
pstP_dtm0_training<-DocumentTermMatrix(tm0_training, control=list(stopwords = FALSE, stemming=F, minWordLength = 1))
#select words that have more than 2 frequency
two0_training<-findFreqTerms(pstP_dtm0_training,0) #
pstP_two.training<-pstP_dtm0_training[,two0_training]
term_tfidf <-tapply(pstP_two.training$v/row_sums(pstP_two.training)[pstP_two.training$i], pstP_two.training$j, mean)*
log2(nDocs(pstP_two.training)/col_sums(pstP_two.training > 0))
summary(term_tfidf)
medi<-unlist(summary(term_tfidf)[2])
a<-sort(term_tfidf)
barplot(a)

#After removing stop words based on value of tf-idf#
pstP_two2.training <- pstP_two.training[,term_tfidf >= cv]
pstP_two3.training <- pstP_two2.training[row_sums(pstP_two2.training) > 1,]
pstP_two3.training <- pstP_two3.training[col_sums(pstP_two3.training) > 1,]
W<-pstP_two.training$ncol
summary(col_sums(pstP_two3.training))

#making test set
##IDF-TF##
text2_test<-Corpus(VectorSource(text7[round(length(text7)*0.6+2):length(text7)]))

pstP_two.test<-DocumentTermMatrix(text2_test, control=list(stopwords = FALSE, stemming=F, minWordLength = 1))
pstP_two.test2 <- pstP_two.test[,term_tfidf >=cv]
#quantile(term_tfidf, 0.55)

pstP_two.test3 <- pstP_two.test2[row_sums(pstP_two.test2) > 1,]
pstP_two.test3 <- pstP_two.test3[col_sums(pstP_two.test3) > 1,]
W<-pstP_two.test$ncol
summary(col_sums(pstP_two.test3))

####Corpus Statistics after preprocessing###
W.total<-pstP_two3.total$ncol # number of terms in the vocab
D1 <- pstP_two3.total$nrow  # number of documents
```

```
N1 <- sum(pstP_two3.total$v)  # total number of tokens in the data
N1/D1 # average length of document
D1
W.total
N1
N1/D1
sd(row_sums(pstP_two3.total))


dagg<-table(pstP_two3.total$j)
names(dagg)<-pstP_two3.total$dimnames$Terms
sort(dagg, decreasing=T)


###################################
#######choosing best model##########
###################################

library(topicmodels)

# Setting estimation conditions
iter<-10000
keep<-1
burnin<-5000

#set up library
# generate numerous topic models with different numbers of topics
sequ <- seq(2, 7, 1) # in this case a sequence of numbers from 2 to 10, by ones.
#alpha<-30/sequ
#delta<-500/W.total
alpha<-1/2
delta<-1/2
SEED<-3423
train <- lapply(sequ, function(k) LDA(pstP_two3.total, k = k, method = "Gibbs",control=list(alpha=alpha, delta=delta, seed=SEED, burnin=burnin,
iter = iter, keep = keep) ))

library("Rmpfr")
library(psych)
library(car)
library(stringr)
library(ltm)
library(psych)
# extract logliks from each topic
logLiks_many <- lapply(train, function(q) q@logLiks[-c(1:(burnin/keep))])
hm_many2 <- sapply(logLiks_many, function(h) harmonic.mean(h))
average<-function(v){mean(unlist(logLiks_many[v]))}
t<-sequ-1
dbar<--2*sapply(t, function(g) average(g))

# compute Pd
Pd<-dbar-(-2*hm_many2)
dic<--2*hm_many2+2*Pd

# inspect(likelihood & perplexity)
plot(sequ,dic, type="l")
which.min(dic)
```

```
fin.m<-train[[2]]
write.table(fin.m@gamma, "gamma.3topic.txt")
write.table(exp(fin.m@beta), "beta.3topic.txt")
```

A.3 Stemming list

```
doc.unlist<-unlist(doc.list)

doc.unlist[which(doc.unlist=='bout')]='about'
doc.unlist[which(doc.unlist=='absolutely')]='absolute'
doc.unlist[which(doc.unlist=='accommodation')]='accommodate'
doc.unlist[which(doc.unlist=='accommodations')]='accommodate'
doc.unlist[which(doc.unlist=='according')]='accord'
doc.unlist[which(doc.unlist=='acording')]='accord'
doc.unlist[which(doc.unlist=='acted')]='act'
doc.unlist[which(doc.unlist=='acting')]='act'
doc.unlist[which(doc.unlist=='actions')]='action'
doc.unlist[which(doc.unlist=='actives')]='active'
doc.unlist[which(doc.unlist=='activites')]='activity'
doc.unlist[which(doc.unlist=='activities')]='activity'
doc.unlist[which(doc.unlist=='activitys')]='activity'
doc.unlist[which(doc.unlist=='actually')]='actual'
doc.unlist[which(doc.unlist=='added')]='add'
doc.unlist[which(doc.unlist=='admitted')]='admit'
doc.unlist[which(doc.unlist=='admitting')]='admit'
doc.unlist[which(doc.unlist=='adout')]='adult'
doc.unlist[which(doc.unlist=='adults')]='adult'
doc.unlist[which(doc.unlist=='agian')]='again'
doc.unlist[which(doc.unlist=='agin')]='again'
doc.unlist[which(doc.unlist=='agreed')]='agree'
doc.unlist[which(doc.unlist=='alot')]='allot/a_lot'
doc.unlist[which(doc.unlist=='amazed')]='amaze'
doc.unlist[which(doc.unlist=='amazement')]='amaze'
doc.unlist[which(doc.unlist=='amazing')]='amaze'
doc.unlist[which(doc.unlist=='amazingly')]='amaze'
doc.unlist[which(doc.unlist=='american')]='america'
doc.unlist[which(doc.unlist=='analyzed')]='analyze'
doc.unlist[which(doc.unlist=='angrily')]='anger'
doc.unlist[which(doc.unlist=='angry')]='anger'
doc.unlist[which(doc.unlist=='animals')]='animal'
doc.unlist[which(doc.unlist=='annoyed')]='annoy'
doc.unlist[which(doc.unlist=='answered')]='answer'
doc.unlist[which(doc.unlist=='anyways')]='anyway'
doc.unlist[which(doc.unlist=='apoligized')]='apologize'
doc.unlist[which(doc.unlist=='apologized')]='apologize'
doc.unlist[which(doc.unlist=='apologizes')]='apologize'
doc.unlist[which(doc.unlist=='apologizing')]='apologize'
doc.unlist[which(doc.unlist=='appeared')]='appear'
doc.unlist[which(doc.unlist=='approaching')]='approach'
doc.unlist[which(doc.unlist=='aren')]='are_not'
doc.unlist[which(doc.unlist=='areas')]='area'
doc.unlist[which(doc.unlist=='argued')]='argue'
doc.unlist[which(doc.unlist=='argues')]='argue'
doc.unlist[which(doc.unlist=='arguing')]='argue'
doc.unlist[which(doc.unlist=='arived')]='arrive'
doc.unlist[which(doc.unlist=='arrived')]='arrive'
doc.unlist[which(doc.unlist=='arrives')]='arrive'
doc.unlist[which(doc.unlist=='arriving')]='arrive'
doc.unlist[which(doc.unlist=='asked')]='ask'
doc.unlist[which(doc.unlist=='asking')]='ask'
doc.unlist[which(doc.unlist=='astonished')]='astonish'
doc.unlist[which(doc.unlist=='atleast')]='at_least'
doc.unlist[which(doc.unlist=='auntie')]='aunt'
doc.unlist[which(doc.unlist=='aunts')]='aunt'
doc.unlist[which(doc.unlist=='awsome')]='awesome'
doc.unlist[which(doc.unlist=='bags')]='bag'
doc.unlist[which(doc.unlist=='based')]='base'
doc.unlist[which(doc.unlist=='beaches')]='beach'
doc.unlist[which(doc.unlist=='bech')]='beach'
doc.unlist[which(doc.unlist=='beautiful')]='beauty'
doc.unlist[which(doc.unlist=='beutiful')]='beauty'
doc.unlist[which(doc.unlist=='beacause')]='because'
doc.unlist[which(doc.unlist=='becasue')]='because'
doc.unlist[which(doc.unlist=='becuase')]='because'
doc.unlist[which(doc.unlist=='becuse')]='because'
doc.unlist[which(doc.unlist=='became')]='become'
doc.unlist[which(doc.unlist=='becomes')]='become'
doc.unlist[which(doc.unlist=='becoming')]='become'
doc.unlist[which(doc.unlist=='begged')]='beg'
doc.unlist[which(doc.unlist=='began')]='begin'
doc.unlist[which(doc.unlist=='begening')]='begin'
doc.unlist[which(doc.unlist=='begging')]='begin'
doc.unlist[which(doc.unlist=='beggining')]='begin'
doc.unlist[which(doc.unlist=='beging')]='begin'
doc.unlist[which(doc.unlist=='begining')]='begin'
doc.unlist[which(doc.unlist=='beginning')]='begin'
doc.unlist[which(doc.unlist=='begins')]='begin'
doc.unlist[which(doc.unlist=='belive')]='believe'
doc.unlist[which(doc.unlist=='besides')]='beside'
doc.unlist[which(doc.unlist=='bigger')]='big'
doc.unlist[which(doc.unlist=='biggest')]='big'
doc.unlist[which(doc.unlist=='birds')]='bird'
doc.unlist[which(doc.unlist=='blew')]='blow'
doc.unlist[which(doc.unlist=='blowing')]='blow'
doc.unlist[which(doc.unlist=='boarded')]='board'
doc.unlist[which(doc.unlist=='boarding')]='board'
doc.unlist[which(doc.unlist=='boating')]='boat'
doc.unlist[which(doc.unlist=='boats')]='boat'
doc.unlist[which(doc.unlist=='boatcruise')]='boat_cruise'
doc.unlist[which(doc.unlist=='boddy')]='body'
doc.unlist[which(doc.unlist=='booked')]='book'
doc.unlist[which(doc.unlist=='booming')]='boom'
doc.unlist[which(doc.unlist=='bored')]='bored'
doc.unlist[which(doc.unlist=='boys')]='boy'
doc.unlist[which(doc.unlist=='broke')]='break'
doc.unlist[which(doc.unlist=='brightly')]='bright'
doc.unlist[which(doc.unlist=='bought')]='bring'
doc.unlist[which(doc.unlist=='brought')]='bring'
doc.unlist[which(doc.unlist=='buckled')]='buckle'
doc.unlist[which(doc.unlist=='bummed')]='bum'
doc.unlist[which(doc.unlist=='burgers')]='burger'
doc.unlist[which(doc.unlist=='bursting')]='burst'
doc.unlist[which(doc.unlist=='called')]='call'
doc.unlist[which(doc.unlist=='calmly')]='calm'
doc.unlist[which(doc.unlist=='cam')]='calm'
doc.unlist[which(doc.unlist=='canad')]='canada'
doc.unlist[which(doc.unlist=='canada s')]='canada'
doc.unlist[which(doc.unlist=='canadas')]='canada'
doc.unlist[which(doc.unlist=='canadian')]='canada'
doc.unlist[which(doc.unlist=='cananda')]='canada'
doc.unlist[which(doc.unlist=='canda')]='canada'
doc.unlist[which(doc.unlist=='candida')]='canada'
doc.unlist[which(doc.unlist=='canida')]='canada'
doc.unlist[which(doc.unlist=='canidu')]='canada'
doc.unlist[which(doc.unlist=='cannada')]='canada'
doc.unlist[which(doc.unlist=='canal')]='canal'
doc.unlist[which(doc.unlist=='canaling')]='canal'
doc.unlist[which(doc.unlist=='canals')]='canal'
doc.unlist[which(doc.unlist=='canel')]='canal'
doc.unlist[which(doc.unlist=='cannal')]='canal'
doc.unlist[which(doc.unlist=='canoeing')]='canoe'
doc.unlist[which(doc.unlist=='canoes')]='canoe'
doc.unlist[which(doc.unlist=='canoing')]='canoe'
doc.unlist[which(doc.unlist=='conoe')]='canoe'
doc.unlist[which(doc.unlist=='cars')]='car'
doc.unlist[which(doc.unlist=='carefully')]='careful'
doc.unlist[which(doc.unlist=='caught')]='catch'
doc.unlist[which(doc.unlist=='celebrated')]='celebrate'
doc.unlist[which(doc.unlist=='celebrating')]='celebrate'
doc.unlist[which(doc.unlist=='celebration')]='celebrate'
doc.unlist[which(doc.unlist=='changed')]='change'
doc.unlist[which(doc.unlist=='changes')]='change'
doc.unlist[which(doc.unlist=='changing')]='change'
doc.unlist[which(doc.unlist=='cheered')]='cheer'
doc.unlist[which(doc.unlist=='cheerful')]='cheer'
doc.unlist[which(doc.unlist=='cheerfully')]='cheer'
doc.unlist[which(doc.unlist=='cheering')]='cheer'
doc.unlist[which(doc.unlist=='chicken')]='chick'
doc.unlist[which(doc.unlist=='children')]='child'
doc.unlist[which(doc.unlist=='chips')]='chip'
doc.unlist[which(doc.unlist=='chirping')]='chirp'
doc.unlist[which(doc.unlist=='chose')]='choose'
doc.unlist[which(doc.unlist=='cited')]='cite'
doc.unlist[which(doc.unlist=='clearly')]='clear'
doc.unlist[which(doc.unlist=='climbed')]='climb'
doc.unlist[which(doc.unlist=='climed')]='climb'
doc.unlist[which(doc.unlist=='climes')]='climb'
doc.unlist[which(doc.unlist=='closed')]='close'
doc.unlist[which(doc.unlist=='closer')]='close'
doc.unlist[which(doc.unlist=='clouds')]='cloud'
doc.unlist[which(doc.unlist=='coaster')]='coast'
doc.unlist[which(doc.unlist=='cacao')]='cocoa'
doc.unlist[which(doc.unlist=='colonist')]='colony'
doc.unlist[which(doc.unlist=='colonists')]='colony'
doc.unlist[which(doc.unlist=='colonized')]='colony'
doc.unlist[which(doc.unlist=='colored')]='color'
doc.unlist[which(doc.unlist=='colorful')]='color'
doc.unlist[which(doc.unlist=='colors')]='color'
doc.unlist[which(doc.unlist=='com')]='come'
doc.unlist[which(doc.unlist=='comes')]='come'
doc.unlist[which(doc.unlist=='coming')]='come'
doc.unlist[which(doc.unlist=='comeback')]='comeback'
doc.unlist[which(doc.unlist=='complained')]='complain'
doc.unlist[which(doc.unlist=='complaining')]='complain'
doc.unlist[which(doc.unlist=='complains')]='complain'
doc.unlist[which(doc.unlist=='complaints')]='complain'
doc.unlist[which(doc.unlist=='complaning')]='complain'
doc.unlist[which(doc.unlist=='completely')]='complete'
doc.unlist[which(doc.unlist=='conclusion')]='conclusion'
doc.unlist[which(doc.unlist=='confused')]='confuse'
doc.unlist[which(doc.unlist=='connected')]='connect'
doc.unlist[which(doc.unlist=='continued')]='continue'
doc.unlist[which(doc.unlist=='cooking')]='cook'
doc.unlist[which(doc.unlist=='coolest')]='cool'
doc.unlist[which(doc.unlist=='couldn t')]='could_not'
doc.unlist[which(doc.unlist=='crawled')]='craw'
doc.unlist[which(doc.unlist=='cree')]='creep'
doc.unlist[which(doc.unlist=='cried')]='cry'
doc.unlist[which(doc.unlist=='curise')]='curious'
doc.unlist[which(doc.unlist=='curse')]='curious'
doc.unlist[which(doc.unlist=='cruises')]='curise'
doc.unlist[which(doc.unlist=='cruising')]='curise'
doc.unlist[which(doc.unlist=='cruize')]='curise'
doc.unlist[which(doc.unlist=='cruse')]='curise'
doc.unlist[which(doc.unlist=='crusie')]='curise'
doc.unlist[which(doc.unlist=='dancing')]='dance'
doc.unlist[which(doc.unlist=='days')]='day'
doc.unlist[which(doc.unlist=='debated')]='debate'
doc.unlist[which(doc.unlist=='debating')]='debate'
doc.unlist[which(doc.unlist=='decided')]='decide'
doc.unlist[which(doc.unlist=='decides')]='decide'
doc.unlist[which(doc.unlist=='decisions')]='decision'
doc.unlist[which(doc.unlist=='declared')]='declare'
doc.unlist[which(doc.unlist=='definitely')]='definite'
doc.unlist[which(doc.unlist=='depressed')]='depress'
doc.unlist[which(doc.unlist=='destination')]='destinate'
doc.unlist[which(doc.unlist=='details')]='detail'
doc.unlist[which(doc.unlist=='developed')]='develop'
doc.unlist[which(doc.unlist=='develops')]='develop'
doc.unlist[which(doc.unlist=='dident')]='do'
doc.unlist[which(doc.unlist=='didnt')]='do'
doc.unlist[which(doc.unlist=='doing')]='do'
doc.unlist[which(doc.unlist=='dont')]='do'
doc.unlist[which(doc.unlist=='doesn')]='do'
doc.unlist[which(doc.unlist=='does')]='do'
doc.unlist[which(doc.unlist=='didn')]='do'
doc.unlist[which(doc.unlist=='differently')]='different'
doc.unlist[which(doc.unlist=='diffrent')]='different'
doc.unlist[which(doc.unlist=='disapointed')]='disappoint'
doc.unlist[which(doc.unlist=='disappointed')]='disappoint'
doc.unlist[which(doc.unlist=='disappointment')]='disappoint'
doc.unlist[which(doc.unlist=='discovered')]='discover'
doc.unlist[which(doc.unlist=='discovers')]='discover'
doc.unlist[which(doc.unlist=='disrespectful')]='disrespect'
doc.unlist[which(doc.unlist=='distance')]='distant'
doc.unlist[which(doc.unlist=='dogs')]='dog'
doc.unlist[which(doc.unlist=='dragged')]='drag'
doc.unlist[which(doc.unlist=='dreaming')]='dream'
doc.unlist[which(doc.unlist=='drinks')]='drink'
doc.unlist[which(doc.unlist=='driving')]='drive'
doc.unlist[which(doc.unlist=='drove')]='drive'
doc.unlist[which(doc.unlist=='during')]='during'
doc.unlist[which(doc.unlist=='ears')]='ear'
doc.unlist[which(doc.unlist=='earlier')]='early'
doc.unlist[which(doc.unlist=='ate')]='eat'
doc.unlist[which(doc.unlist=='eating')]='eat'
doc.unlist[which(doc.unlist=='emotions')]='emotion'
doc.unlist[which(doc.unlist=='ended')]='end'
doc.unlist[which(doc.unlist=='ending')]='end'
doc.unlist[which(doc.unlist=='ends')]='end'
doc.unlist[which(doc.unlist=='enjoyed')]='enjoy'
doc.unlist[which(doc.unlist=='enjoying')]='enjoy'
doc.unlist[which(doc.unlist=='enjoys')]='enjoy'
doc.unlist[which(doc.unlist=='entered')]='enter'
doc.unlist[which(doc.unlist=='especially')]='especial'
doc.unlist[which(doc.unlist=='europeans')]='european'
doc.unlist[which(doc.unlist=='evening')]='evening'
doc.unlist[which(doc.unlist=='events')]='event'
doc.unlist[which(doc.unlist=='eventually')]='eventual'
doc.unlist[which(doc.unlist=='exactly')]='exact'
doc.unlist[which(doc.unlist=='examples')]='example'
doc.unlist[which(doc.unlist=='excited')]='excite'
doc.unlist[which(doc.unlist=='excitedly')]='excite'
doc.unlist[which(doc.unlist=='excitement')]='excite'
doc.unlist[which(doc.unlist=='exciting')]='excite'
doc.unlist[which(doc.unlist=='excitingly')]='excite'
doc.unlist[which(doc.unlist=='excitment')]='excite'
doc.unlist[which(doc.unlist=='exided')]='excite'
doc.unlist[which(doc.unlist=='exited')]='excite'
doc.unlist[which(doc.unlist=='exiting')]='excite'
doc.unlist[which(doc.unlist=='exitment')]='excite'
doc.unlist[which(doc.unlist=='exsided')]='excite'
doc.unlist[which(doc.unlist=='exsited')]='excite'
doc.unlist[which(doc.unlist=='exclaimed')]='exclaim'
doc.unlist[which(doc.unlist=='exclamed')]='exclaim'
doc.unlist[which(doc.unlist=='exclamied')]='exclaim'
doc.unlist[which(doc.unlist=='expected')]='expect'
doc.unlist[which(doc.unlist=='experiences')]='experience'
doc.unlist[which(doc.unlist=='explained')]='explain'
doc.unlist[which(doc.unlist=='explaining')]='explain'
doc.unlist[which(doc.unlist=='explains')]='explain'
doc.unlist[which(doc.unlist=='exploding')]='explode'
doc.unlist[which(doc.unlist=='explored')]='explore'
doc.unlist[which(doc.unlist=='eyed')]='eye'
doc.unlist[which(doc.unlist=='eyes')]='eye'
doc.unlist[which(doc.unlist=='faces')]='face'
doc.unlist[which(doc.unlist=='facts')]='fact'
doc.unlist[which(doc.unlist=='fallen')]='fall'
doc.unlist[which(doc.unlist=='famly')]='family'
doc.unlist[which(doc.unlist=='fantastic')]='fantasy'
doc.unlist[which(doc.unlist=='farther')]='far'
doc.unlist[which(doc.unlist=='fascinated')]='fascinate'
doc.unlist[which(doc.unlist=='feeling')]='feel'
doc.unlist[which(doc.unlist=='feelings')]='feel'
doc.unlist[which(doc.unlist=='feels')]='feel'
doc.unlist[which(doc.unlist=='felling')]='fell'
doc.unlist[which(doc.unlist=='felt')]='fell'
doc.unlist[which(doc.unlist=='festivals')]='festival'
doc.unlist[which(doc.unlist=='festives')]='festive'
doc.unlist[which(doc.unlist=='festivies')]='festivity'
doc.unlist[which(doc.unlist=='festivites')]='festivity'
doc.unlist[which(doc.unlist=='festivities')]='festivity'
doc.unlist[which(doc.unlist=='fierworks')]='fierwork'
doc.unlist[which(doc.unlist=='fighting')]='fight'
doc.unlist[which(doc.unlist=='figured')]='figure'
doc.unlist[which(doc.unlist=='filled')]='fill'
doc.unlist[which(doc.unlist=='filling')]='fill'
doc.unlist[which(doc.unlist=='finally')]='final'
doc.unlist[which(doc.unlist=='finaly')]='final'
doc.unlist[which(doc.unlist=='finding')]='find'
doc.unlist[which(doc.unlist=='finds')]='find'
doc.unlist[which(doc.unlist=='found')]='find'
doc.unlist[which(doc.unlist=='finished')]='finish'
doc.unlist[which(doc.unlist=='fireworkes')]='firework'
doc.unlist[which(doc.unlist=='fireworks')]='firework'
doc.unlist[which(doc.unlist=='frist')]='first'
doc.unlist[which(doc.unlist=='fishing')]='fish'
doc.unlist[which(doc.unlist=='floated')]='float'
doc.unlist[which(doc.unlist=='floating')]='float'
doc.unlist[which(doc.unlist=='flowing')]='flow'
doc.unlist[which(doc.unlist=='flowers')]='flower'
doc.unlist[which(doc.unlist=='focused')]='focuse'
doc.unlist[which(doc.unlist=='followed')]='follow'
doc.unlist[which(doc.unlist=='following')]='follow'
doc.unlist[which(doc.unlist=='foods')]='food'
doc.unlist[which(doc.unlist=='feet')]='foot'
doc.unlist[which(doc.unlist=='forced')]='force'
doc.unlist[which(doc.unlist=='forgot')]='forget'
doc.unlist[which(doc.unlist=='fourth')]='four'
doc.unlist[which(doc.unlist=='freind')]='friend'
doc.unlist[which(doc.unlist=='frends')]='friend'
doc.unlist[which(doc.unlist=='freinds')]='friend'
doc.unlist[which(doc.unlist=='friends')]='friend'
doc.unlist[which(doc.unlist=='frinds')]='friend'
doc.unlist[which(doc.unlist=='frustrated')]='frustrate'
doc.unlist[which(doc.unlist=='fries')]='fry'
doc.unlist[which(doc.unlist=='gab')]='gabriel'
doc.unlist[which(doc.unlist=='gabe')]='gabriel'
doc.unlist[which(doc.unlist=='gaberiel')]='gabriel'
doc.unlist[which(doc.unlist=='gabirel')]='gabriel'
doc.unlist[which(doc.unlist=='gabreil')]='gabriel'
doc.unlist[which(doc.unlist=='gabrel')]='gabriel'
doc.unlist[which(doc.unlist=='gabrial')]='gabriel'
doc.unlist[which(doc.unlist=='gabriel s')]='gabriel'
doc.unlist[which(doc.unlist=='gabriela')]='gabriel'
doc.unlist[which(doc.unlist=='gabriels')]='gabriel'
doc.unlist[which(doc.unlist=='gabril')]='gabriel'
doc.unlist[which(doc.unlist=='gabrile')]='gabriel'
doc.unlist[which(doc.unlist=='gadriel')]='gabriel'
doc.unlist[which(doc.unlist=='grabriel')]='gabriel'
doc.unlist[which(doc.unlist=='gabrielnoticed')]='gabriel_notice'
doc.unlist[which(doc.unlist=='gained')]='gain'
doc.unlist[which(doc.unlist=='games')]='game'
doc.unlist[which(doc.unlist=='gaming')]='game'
doc.unlist[which(doc.unlist=='gathered')]='gather'
doc.unlist[which(doc.unlist=='gotten')]='get'
doc.unlist[which(doc.unlist=='gets')]='get'
doc.unlist[which(doc.unlist=='getting')]='getting'
doc.unlist[which(doc.unlist=='gave')]='give'
doc.unlist[which(doc.unlist=='gives')]='give'
doc.unlist[which(doc.unlist=='giving')]='give'
doc.unlist[which(doc.unlist=='gleaming')]='gleam'
doc.unlist[which(doc.unlist=='glimmering')]='glimmer'
doc.unlist[which(doc.unlist=='glowing')]='glow'
doc.unlist[which(doc.unlist=='wen')]='go'
doc.unlist[which(doc.unlist=='went')]='go'
doc.unlist[which(doc.unlist=='going')]='go'
doc.unlist[which(doc.unlist=='being')]='be'
doc.unlist[which(doc.unlist=='were')]='be'
doc.unlist[which(doc.unlist=='golden')]='gold'
doc.unlist[which(doc.unlist=='governing')]='govern'
doc.unlist[which(doc.unlist=='grabbed')]='grab'
doc.unlist[which(doc.unlist=='grandparents')]='grandparent'
doc.unlist[which(doc.unlist=='grassy')]='grass'
doc.unlist[which(doc.unlist=='grate')]='great'
doc.unlist[which(doc.unlist=='greatest')]='great'
doc.unlist[which(doc.unlist=='grilling')]='grill'
doc.unlist[which(doc.unlist=='groaned')]='groan'
doc.unlist[which(doc.unlist=='groaning')]='groan'
doc.unlist[which(doc.unlist=='groning')]='groan'
doc.unlist[which(doc.unlist=='grew')]='grow'
doc.unlist[which(doc.unlist=='growing')]='grow'
doc.unlist[which(doc.unlist=='grumbling')]='grumble'
doc.unlist[which(doc.unlist=='grumpily')]='grumpy'
doc.unlist[which(doc.unlist=='guider')]='guide'
doc.unlist[which(doc.unlist=='guides')]='guide'
doc.unlist[which(doc.unlist=='theirs')]='they'
doc.unlist[which(doc.unlist=='guys')]='guy'
doc.unlist[which(doc.unlist=='hadn')]='had_not'
doc.unlist[which(doc.unlist=='hamburgers')]='hamburger'
doc.unlist[which(doc.unlist=='hanging')]='hang'
doc.unlist[which(doc.unlist=='happend')]='happen'
doc.unlist[which(doc.unlist=='happened')]='happen'
doc.unlist[which(doc.unlist=='happening')]='happen'
doc.unlist[which(doc.unlist=='happier')]='happy'
doc.unlist[which(doc.unlist=='happily')]='happy'
doc.unlist[which(doc.unlist=='happiness')]='happy'
doc.unlist[which(doc.unlist=='curise')]='cruise'
doc.unlist[which(doc.unlist=='happly')]='happy'
doc.unlist[which(doc.unlist=='happy')]='happy'
doc.unlist[which(doc.unlist=='harder')]='hard'
doc.unlist[which(doc.unlist=='hes')]='have'
doc.unlist[which(doc.unlist=='has')]='have'
doc.unlist[which(doc.unlist=='hated')]='hate'
doc.unlist[which(doc.unlist=='hates')]='hate'
doc.unlist[which(doc.unlist=='hating')]='hate'
doc.unlist[which(doc.unlist=='having')]='have'
doc.unlist[which(doc.unlist=='haven')]='have_not'
doc.unlist[which(doc.unlist=='headed')]='head'
doc.unlist[which(doc.unlist=='heading')]='head'
doc.unlist[which(doc.unlist=='heads')]='head'
doc.unlist[which(doc.unlist=='heard')]='hear'
doc.unlist[which(doc.unlist=='hearing')]='hear'
doc.unlist[which(doc.unlist=='helped')]='help'
doc.unlist[which(doc.unlist=='helps')]='help'
doc.unlist[which(doc.unlist=='hid')]='hide'
doc.unlist[which(doc.unlist=='hills')]='hill'
doc.unlist[which(doc.unlist=='hitting')]='hit'
doc.unlist[which(doc.unlist=='hopped')]='hop'
doc.unlist[which(doc.unlist=='hop')]='hope'
doc.unlist[which(doc.unlist=='hoped')]='hope'
doc.unlist[which(doc.unlist=='hopes')]='hope'
doc.unlist[which(doc.unlist=='hoping')]='hope'
doc.unlist[which(doc.unlist=='horrible')]='horrible'
doc.unlist[which(doc.unlist=='hotdogs')]='hotdog'
doc.unlist[which(doc.unlist=='hours')]='hour'
doc.unlist[which(doc.unlist=='houses')]='house'
doc.unlist[which(doc.unlist=='hugged')]='hug'
doc.unlist[which(doc.unlist=='hug')]='huge'
doc.unlist[which(doc.unlist=='hung')]='hungry'
doc.unlist[which(doc.unlist=='hurried')]='hurry'
doc.unlist[which(doc.unlist=='ideas')]='idea'
doc.unlist[which(doc.unlist=='imagined')]='imagine'
doc.unlist[which(doc.unlist=='immediately')]='immediate'
doc.unlist[which(doc.unlist=='impressed')]='impress'
doc.unlist[which(doc.unlist=='impressive')]='impress'
doc.unlist[which(doc.unlist=='included')]='include'
doc.unlist[which(doc.unlist=='including')]='include'
doc.unlist[which(doc.unlist=='independence')]='independent'
doc.unlist[which(doc.unlist=='indians')]='indian'
doc.unlist[which(doc.unlist=='insted')]='instead'
doc.unlist[which(doc.unlist=='interested')]='interest'
doc.unlist[which(doc.unlist=='interesting')]='interest'
doc.unlist[which(doc.unlist=='intresting')]='interest'
doc.unlist[which(doc.unlist=='introduced')]='introduce'
doc.unlist[which(doc.unlist=='introducing')]='introduce'
doc.unlist[which(doc.unlist=='invited')]='invite'
doc.unlist[which(doc.unlist=='involved')]='involve'
doc.unlist[which(doc.unlist=='it  s')]='it_is'
doc.unlist[which(doc.unlist=='it  s')]='it_is'
doc.unlist[which(doc.unlist=='joyful')]='joy'
doc.unlist[which(doc.unlist=='joyfully')]='joy'
doc.unlist[which(doc.unlist=='judged')]='judge'
doc.unlist[which(doc.unlist=='judging')]='judge'
doc.unlist[which(doc.unlist=='jumped')]='jump'
doc.unlist[which(doc.unlist=='jumping')]='jump'
doc.unlist[which(doc.unlist=='kayaking')]='kayak'
doc.unlist[which(doc.unlist=='kayaks')]='kayak'
doc.unlist[which(doc.unlist=='kayka')]='kayak'
doc.unlist[which(doc.unlist=='kyak')]='kayak'
doc.unlist[which(doc.unlist=='kept')]='keep'
doc.unlist[which(doc.unlist=='kids')]='kid'
doc.unlist[which(doc.unlist=='kinda')]='kind'
doc.unlist[which(doc.unlist=='kinds')]='kind'
doc.unlist[which(doc.unlist=='knew')]='know'
doc.unlist[which(doc.unlist=='knowing')]='know'
doc.unlist[which(doc.unlist=='known')]='know'
doc.unlist[which(doc.unlist=='knows')]='know'
doc.unlist[which(doc.unlist=='landed')]='land'
doc.unlist[which(doc.unlist=='landlocked')]='landlock'
doc.unlist[which(doc.unlist=='languages')]='language'
doc.unlist[which(doc.unlist=='larger')]='large'
doc.unlist[which(doc.unlist=='lastly')]='last'
doc.unlist[which(doc.unlist=='later')]='late'
doc.unlist[which(doc.unlist=='latter')]='late'
doc.unlist[which(doc.unlist=='laughed')]='laugh'
doc.unlist[which(doc.unlist=='laughing')]='laugh'
doc.unlist[which(doc.unlist=='leaned')]='lean'
doc.unlist[which(doc.unlist=='leaped')]='leap'
doc.unlist[which(doc.unlist=='learnd')]='learn'
doc.unlist[which(doc.unlist=='learned')]='learn'
doc.unlist[which(doc.unlist=='learning')]='learn'
doc.unlist[which(doc.unlist=='learns')]='learn'
doc.unlist[which(doc.unlist=='leaves')]='leave'
doc.unlist[which(doc.unlist=='leaving')]='leave'
doc.unlist[which(doc.unlist=='letting')]='let'
doc.unlist[which(doc.unlist=='lights')]='light'
doc.unlist[which(doc.unlist=='lit')]='light'
doc.unlist[which(doc.unlist=='liked')]='like'
doc.unlist[which(doc.unlist=='likely')]='like'
doc.unlist[which(doc.unlist=='likes')]='like'
doc.unlist[which(doc.unlist=='liking')]='like'
doc.unlist[which(doc.unlist=='listing')]='list'
doc.unlist[which(doc.unlist=='listened')]='listen'
doc.unlist[which(doc.unlist=='listening')]='listen'
doc.unlist[which(doc.unlist=='lived')]='live'
doc.unlist[which(doc.unlist=='lives')]='live'
doc.unlist[which(doc.unlist=='living')]='live'
doc.unlist[which(doc.unlist=='locked')]='lock'
doc.unlist[which(doc.unlist=='longer')]='long'
doc.unlist[which(doc.unlist=='looked')]='look'
doc.unlist[which(doc.unlist=='looking')]='look'
doc.unlist[which(doc.unlist=='looks')]='look'
doc.unlist[which(doc.unlist=='lost')]='lose'
doc.unlist[which(doc.unlist=='lots')]='lot'
doc.unlist[which(doc.unlist=='loudly')]='loud'
doc.unlist[which(doc.unlist=='loved')]='love'
doc.unlist[which(doc.unlist=='lovely')]='love'
doc.unlist[which(doc.unlist=='loves')]='love'
doc.unlist[which(doc.unlist=='loving')]='love'
doc.unlist[which(doc.unlist=='mabe')]='mabe'
doc.unlist[which(doc.unlist=='madly')]='mad'
doc.unlist[which(doc.unlist=='mainly')]='main'
doc.unlist[which(doc.unlist=='made')]='make'
doc.unlist[which(doc.unlist=='makes')]='make'
doc.unlist[which(doc.unlist=='making')]='make'
doc.unlist[which(doc.unlist=='meaning')]='mean'
doc.unlist[which(doc.unlist=='means')]='mean'
doc.unlist[which(doc.unlist=='meant')]='mean'
doc.unlist[which(doc.unlist=='meeting')]='meet'
doc.unlist[which(doc.unlist=='meets')]='meet'
doc.unlist[which(doc.unlist=='ment')]='meet'
doc.unlist[which(doc.unlist=='met')]='meet'
doc.unlist[which(doc.unlist=='mentioned')]='mention'
doc.unlist[which(doc.unlist=='miles')]='mile'
doc.unlist[which(doc.unlist=='minds')]='mind'
doc.unlist[which(doc.unlist=='minutes')]='minute'
doc.unlist[which(doc.unlist=='missed')]='miss'
doc.unlist[which(doc.unlist=='missing')]='miss'
doc.unlist[which(doc.unlist=='moaned')]='moan'
doc.unlist[which(doc.unlist=='moaning')]='moan'
doc.unlist[which(doc.unlist=='moments')]='moment'
doc.unlist[which(doc.unlist=='moods')]='mood'
doc.unlist[which(doc.unlist=='moning')]='morning'
doc.unlist[which(doc.unlist=='mom')]='mother'
doc.unlist[which(doc.unlist=='mom')]='mother'
doc.unlist[which(doc.unlist=='mother')]='mother'
doc.unlist[which(doc.unlist=='mountains')]='mountain'
doc.unlist[which(doc.unlist=='moved')]='move'
doc.unlist[which(doc.unlist=='moving')]='move'
doc.unlist[which(doc.unlist=='mumbled')]='mumble'
doc.unlist[which(doc.unlist=='mumbling')]='mumble'
doc.unlist[which(doc.unlist=='named')]='name'
doc.unlist[which(doc.unlist=='natinal')]='nation'
doc.unlist[which(doc.unlist=='national')]='nation'
doc.unlist[which(doc.unlist=='nationals')]='nation'
doc.unlist[which(doc.unlist=='needed')]='need'
doc.unlist[which(doc.unlist=='neighborhood')]='neighbor'
doc.unlist[which(doc.unlist=='neighborhoods')]='neighbor'
doc.unlist[which(doc.unlist=='neighbors')]='neighbor'
doc.unlist[which(doc.unlist=='nodded')]='nod'
doc.unlist[which(doc.unlist=='noises')]='noise'
doc.unlist[which(doc.unlist=='noticed')]='notice'
doc.unlist[which(doc.unlist=='notices')]='notice'
doc.unlist[which(doc.unlist=='noticing')]='notice'
doc.unlist[which(doc.unlist=='oceans')]='ocean'
doc.unlist[which(doc.unlist=='ones')]='one'
doc.unlist[which(doc.unlist=='oneof')]='one_of'
doc.unlist[which(doc.unlist=='opened')]='open'
doc.unlist[which(doc.unlist=='ottowa')]='ottawa'
doc.unlist[which(doc.unlist=='overjoyed')]='overjoy'
doc.unlist[which(doc.unlist=='packed')]='pack'
doc.unlist[which(doc.unlist=='packing')]='pack'
doc.unlist[which(doc.unlist=='paddling')]='paddle'
doc.unlist[which(doc.unlist=='paddleboating')]='paddleboat'
doc.unlist[which(doc.unlist=='paddleboats')]='paddleboat'
doc.unlist[which(doc.unlist=='paragraphs')]='paragraph'
doc.unlist[which(doc.unlist=='parents')]='parent'
doc.unlist[which(doc.unlist=='parking')]='park'
doc.unlist[which(doc.unlist=='parks')]='park'
doc.unlist[which(doc.unlist=='parlement')]='parliament'
doc.unlist[which(doc.unlist=='parliment')]='parliament'
doc.unlist[which(doc.unlist=='parts')]='part'
doc.unlist[which(doc.unlist=='participated')]='participate'
doc.unlist[which(doc.unlist=='participating')]='participate'
doc.unlist[which(doc.unlist=='particpate')]='participate'
doc.unlist[which(doc.unlist=='passed')]='pass'
doc.unlist[which(doc.unlist=='passage')]='passage'
doc.unlist[which(doc.unlist=='peaceful')]='peace'
doc.unlist[which(doc.unlist=='peacefully')]='peace'
doc.unlist[which(doc.unlist=='picked')]='pick'
doc.unlist[which(doc.unlist=='picky')]='pick'
doc.unlist[which(doc.unlist=='places')]='place'
doc.unlist[which(doc.unlist=='plains')]='plain'
doc.unlist[which(doc.unlist=='planned')]='plan'
doc.unlist[which(doc.unlist=='planning')]='plan'
doc.unlist[which(doc.unlist=='plans')]='plan'
doc.unlist[which(doc.unlist=='planed')]='plan/plane'
doc.unlist[which(doc.unlist=='planes')]='plane'
doc.unlist[which(doc.unlist=='planing')]='plane'
doc.unlist[which(doc.unlist=='played')]='play'
doc.unlist[which(doc.unlist=='playing')]='play'
doc.unlist[which(doc.unlist=='pleas')]='please'
doc.unlist[which(doc.unlist=='pointed')]='point'
doc.unlist[which(doc.unlist=='pointing')]='point'
doc.unlist[which(doc.unlist=='pong')]='pond'
doc.unlist[which(doc.unlist=='popping')]='pop'
doc.unlist[which(doc.unlist=='pouting')]='pout'
doc.unlist[which(doc.unlist=='probably')]='probable'
doc.unlist[which(doc.unlist=='probaly')]='probable'
doc.unlist[which(doc.unlist=='promised')]='promise'
doc.unlist[which(doc.unlist=='proudly')]='proud'
doc.unlist[which(doc.unlist=='proved')]='prove'
doc.unlist[which(doc.unlist=='proves')]='prove'
doc.unlist[which(doc.unlist=='pulled')]='pull'
doc.unlist[which(doc.unlist=='purchased')]='purchase'
doc.unlist[which(doc.unlist=='quickly')]='quick'
doc.unlist[which(doc.unlist=='raced')]='race'
doc.unlist[which(doc.unlist=='reached')]='reach'
doc.unlist[which(doc.unlist=='reading')]='read'
doc.unlist[which(doc.unlist=='realized')]='realize'
doc.unlist[which(doc.unlist=='realizes')]='realize'
doc.unlist[which(doc.unlist=='relized')]='realize'
doc.unlist[which(doc.unlist=='realy')]='really'
doc.unlist[which(doc.unlist=='reasons')]='reason'
doc.unlist[which(doc.unlist=='refused')]='refuse'
doc.unlist[which(doc.unlist=='relaxed')]='relax'
doc.unlist[which(doc.unlist=='relaxing')]='relax'
doc.unlist[which(doc.unlist=='relieved')]='relieve'
doc.unlist[which(doc.unlist=='reluctantly')]='reluctant'
doc.unlist[which(doc.unlist=='reminded')]='remind'
doc.unlist[which(doc.unlist=='rented')]='rent'
doc.unlist[which(doc.unlist=='replied')]='reply'
doc.unlist[which(doc.unlist=='represents')]='represent'
doc.unlist[which(doc.unlist=='researched')]='research'
doc.unlist[which(doc.unlist=='researching')]='research'
doc.unlist[which(doc.unlist=='ridea')]='ride'
doc.unlist[which(doc.unlist=='rideau')]='ride'
doc.unlist[which(doc.unlist=='rides')]='ride'
doc.unlist[which(doc.unlist=='riding')]='ride'
doc.unlist[which(doc.unlist=='rode')]='ride'
doc.unlist[which(doc.unlist=='rocks')]='rock'
doc.unlist[which(doc.unlist=='rolled')]='roll'
doc.unlist[which(doc.unlist=='roller')]='roll'
doc.unlist[which(doc.unlist=='rooms')]='room'
doc.unlist[which(doc.unlist=='running')]='run'
doc.unlist[which(doc.unlist=='rushed')]='rush'
doc.unlist[which(doc.unlist=='rushing')]='rush'
doc.unlist[which(doc.unlist=='sadly')]='sad'
doc.unlist[which(doc.unlist=='sed')]='sad'
doc.unlist[which(doc.unlist=='salty')]='salt'
doc.unlist[which(doc.unlist=='sandy')]='sand'
doc.unlist[which(doc.unlist=='satisfied')]='satisfy'
doc.unlist[which(doc.unlist=='siad')]='say'
doc.unlist[which(doc.unlist=='saying')]='say'
doc.unlist[which(doc.unlist=='says')]='say'
doc.unlist[which(doc.unlist=='scared')]='scare'
doc.unlist[which(doc.unlist=='scheduled')]='schedule'
doc.unlist[which(doc.unlist=='screamed')]='scream'
doc.unlist[which(doc.unlist=='screaming')]='scream'
doc.unlist[which(doc.unlist=='seagulls')]='seagull'
doc.unlist[which(doc.unlist=='seats')]='seat'
doc.unlist[which(doc.unlist=='seconds')]='second'
doc.unlist[which(doc.unlist=='seeing')]='see'
doc.unlist[which(doc.unlist=='seen')]='see'
doc.unlist[which(doc.unlist=='sees')]='see'
doc.unlist[which(doc.unlist=='seemed')]='seem'
doc.unlist[which(doc.unlist=='seems')]='seem'
doc.unlist[which(doc.unlist=='sale')]='sell'
doc.unlist[which(doc.unlist=='sall')]='sell'
doc.unlist[which(doc.unlist=='sent')]='send'
doc.unlist[which(doc.unlist=='separated')]='separate'
doc.unlist[which(doc.unlist=='served')]='serve'
doc.unlist[which(doc.unlist=='setting')]='set'
doc.unlist[which(doc.unlist=='shimmering')]='shimmer'
doc.unlist[which(doc.unlist=='shining')]='shine'
doc.unlist[which(doc.unlist=='shocked')]='shock'
doc.unlist[which(doc.unlist=='shocking')]='shock'
doc.unlist[which(doc.unlist=='shooting')]='shoot'
doc.unlist[which(doc.unlist=='shops')]='shop'
doc.unlist[which(doc.unlist=='chould')]='should'
doc.unlist[which(doc.unlist=='shouldn')]='should_not'
doc.unlist[which(doc.unlist=='shouted')]='shout'
doc.unlist[which(doc.unlist=='shouting')]='shout'
doc.unlist[which(doc.unlist=='showed')]='show'
doc.unlist[which(doc.unlist=='showing')]='show'
doc.unlist[which(doc.unlist=='shown')]='show'
doc.unlist[which(doc.unlist=='shows')]='show'
doc.unlist[which(doc.unlist=='sighed')]='sigh'
doc.unlist[which(doc.unlist=='sights')]='sight'
doc.unlist[which(doc.unlist=='singing')]='sing'
doc.unlist[which(doc.unlist=='sat')]='sit'
doc.unlist[which(doc.unlist=='sitting')]='sit'
doc.unlist[which(doc.unlist=='sized')]='size'
doc.unlist[which(doc.unlist=='slammed')]='slam'
doc.unlist[which(doc.unlist=='slightly')]='slight'
doc.unlist[which(doc.unlist=='slowly')]='slow'
doc.unlist[which(doc.unlist=='smaller')]='small'
doc.unlist[which(doc.unlist=='smelled')]='smell'
doc.unlist[which(doc.unlist=='smelling')]='smell'
doc.unlist[which(doc.unlist=='smells')]='smell'
doc.unlist[which(doc.unlist=='smelly')]='smell'
doc.unlist[which(doc.unlist=='smelt')]='smell'
doc.unlist[which(doc.unlist=='smiled')]='smile'
doc.unlist[which(doc.unlist=='smiling')]='smile'
doc.unlist[which(doc.unlist=='snacks')]='snack'
doc.unlist[which(doc.unlist=='softly')]='soft'
doc.unlist[which(doc.unlist=='somthing')]='something'
doc.unlist[which(doc.unlist=='sons')]='son'
doc.unlist[which(doc.unlist=='soothing')]='soothe'
doc.unlist[which(doc.unlist=='sorts')]='sort'
doc.unlist[which(doc.unlist=='sounded')]='sound'
doc.unlist[which(doc.unlist=='sounds')]='sound'
doc.unlist[which(doc.unlist=='sourounded')]='souround'
doc.unlist[which(doc.unlist=='sparkly')]='spark'
doc.unlist[which(doc.unlist=='sparkled')]='sparkle'
doc.unlist[which(doc.unlist=='sparkling')]='sparkle'
doc.unlist[which(doc.unlist=='speaking')]='speak'
doc.unlist[which(doc.unlist=='spoke')]='speak'
doc.unlist[which(doc.unlist=='spending')]='spend'
doc.unlist[which(doc.unlist=='spent')]='spend'
doc.unlist[which(doc.unlist=='splashed')]='splash'
doc.unlist[which(doc.unlist=='splashing')]='splash'
doc.unlist[which(doc.unlist=='sports')]='sport'
doc.unlist[which(doc.unlist=='spot')]='sport'
doc.unlist[which(doc.unlist=='spots')]='sport'
doc.unlist[which(doc.unlist=='spotted')]='spot'
doc.unlist[which(doc.unlist=='stairs')]='stair'
doc.unlist[which(doc.unlist=='standing')]='stand'
doc.unlist[which(doc.unlist=='stands')]='stand'
doc.unlist[which(doc.unlist=='stood')]='stand'
doc.unlist[which(doc.unlist=='stars')]='star'
doc.unlist[which(doc.unlist=='stared')]='stare'
doc.unlist[which(doc.unlist=='staring')]='stare'
doc.unlist[which(doc.unlist=='started')]='start'
doc.unlist[which(doc.unlist=='starting')]='start'
doc.unlist[which(doc.unlist=='starts')]='start'
doc.unlist[which(doc.unlist=='stats')]='start'
doc.unlist[which(doc.unlist=='stated')]='state'
doc.unlist[which(doc.unlist=='states')]='state'
doc.unlist[which(doc.unlist=='stayed')]='stay'
doc.unlist[which(doc.unlist=='staying')]='stay'
doc.unlist[which(doc.unlist=='stepped')]='steep'
doc.unlist[which(doc.unlist=='stopped')]='stop'
doc.unlist[which(doc.unlist=='streaming')]='stream'
doc.unlist[which(doc.unlist=='stretches')]='stretch'
doc.unlist[which(doc.unlist=='studies')]='study'
doc.unlist[which(doc.unlist=='stuffed')]='stuff'
doc.unlist[which(doc.unlist=='stunning')]='stun'
doc.unlist[which(doc.unlist=='suddenly')]='sudden'
doc.unlist[which(doc.unlist=='sunny')]='sun'
doc.unlist[which(doc.unlist=='supports')]='support'
doc.unlist[which(doc.unlist=='suprised')]='suprise'
doc.unlist[which(doc.unlist=='suprize')]='suprise'
doc.unlist[which(doc.unlist=='serf')]='surf'
doc.unlist[which(doc.unlist=='serfing')]='surf'
doc.unlist[which(doc.unlist=='surffing')]='surf'
doc.unlist[which(doc.unlist=='surfing')]='surf'
doc.unlist[which(doc.unlist=='surpise')]='surprise'
doc.unlist[which(doc.unlist=='surprised')]='surprise'
doc.unlist[which(doc.unlist=='surprises')]='surprise'
doc.unlist[which(doc.unlist=='surprising')]='surprise'
doc.unlist[which(doc.unlist=='surprisingly')]='surprise'
doc.unlist[which(doc.unlist=='surprize')]='surprise'
doc.unlist[which(doc.unlist=='surrounded')]='surround'
doc.unlist[which(doc.unlist=='surrounding')]='surround'
doc.unlist[which(doc.unlist=='suv')]='survey'
doc.unlist[which(doc.unlist=='swam')]='swim'
doc.unlist[which(doc.unlist=='swimming')]='swim'
doc.unlist[which(doc.unlist=='takes')]='take'
doc.unlist[which(doc.unlist=='taking')]='take'
doc.unlist[which(doc.unlist=='took')]='take'
doc.unlist[which(doc.unlist=='talked')]='talk'
doc.unlist[which(doc.unlist=='talking')]='talk'
doc.unlist[which(doc.unlist=='tasted')]='taste'
doc.unlist[which(doc.unlist=='teacher')]='teach'
doc.unlist[which(doc.unlist=='teaches')]='teach'
doc.unlist[which(doc.unlist=='tears')]='tear'
doc.unlist[which(doc.unlist=='tore')]='tear'
doc.unlist[which(doc.unlist=='telling')]='tell'
doc.unlist[which(doc.unlist=='tells')]='tell'
doc.unlist[which(doc.unlist=='told')]='tell'
doc.unlist[which(doc.unlist=='texted')]='text'
doc.unlist[which(doc.unlist=='than')]='thank'
doc.unlist[which(doc.unlist=='thanked')]='thank'
doc.unlist[which(doc.unlist=='thankful')]='thank'
doc.unlist[which(doc.unlist=='thanks')]='thank'
doc.unlist[which(doc.unlist=='that s')]='that_is'
doc.unlist[which(doc.unlist=='that s')]='that_is'
doc.unlist[which(doc.unlist=='thats')]='that_is'
doc.unlist[which(doc.unlist=='thee')]='the'
doc.unlist[which(doc.unlist=='theme')]='the'
doc.unlist[which(doc.unlist=='ther')]='the'
doc.unlist[which(doc.unlist=='thier')]='their'
doc.unlist[which(doc.unlist=='there s')]='there_is'
doc.unlist[which(doc.unlist=='theres')]='there_is'
doc.unlist[which(doc.unlist=='thin')]='thing'
doc.unlist[which(doc.unlist=='things')]='thing'
doc.unlist[which(doc.unlist=='thinking')]='think'
doc.unlist[which(doc.unlist=='thinks')]='think'
doc.unlist[which(doc.unlist=='thot')]='thought'
doc.unlist[which(doc.unlist=='thoughts')]='thought'
doc.unlist[which(doc.unlist=='thout')]='thought'
doc.unlist[which(doc.unlist=='threw')]='throw'
doc.unlist[which(doc.unlist=='tickets')]='ticket'
doc.unlist[which(doc.unlist=='tikets')]='ticket'
doc.unlist[which(doc.unlist=='times')]='time'
doc.unlist[which(doc.unlist=='tired')]='tire'
doc.unlist[which(doc.unlist=='tires')]='tire'
doc.unlist[which(doc.unlist=='tons')]='ton'
doc.unlist[which(doc.unlist=='totally')]='total'
doc.unlist[which(doc.unlist=='touched')]='touch'
doc.unlist[which(doc.unlist=='towards')]='toward'
doc.unlist[which(doc.unlist=='traditional')]='tradition'
doc.unlist[which(doc.unlist=='traveled')]='travel'
doc.unlist[which(doc.unlist=='trees')]='tree'
doc.unlist[which(doc.unlist=='tribes')]='tribe'
doc.unlist[which(doc.unlist=='tried')]='trie'
doc.unlist[which(doc.unlist=='tries')]='trie'
doc.unlist[which(doc.unlist=='trips')]='trip'
doc.unlist[which(doc.unlist=='trying')]='try'
doc.unlist[which(doc.unlist=='turned')]='turn'
doc.unlist[which(doc.unlist=='turning')]='turn'
doc.unlist[which(doc.unlist=='turns')]='turn'
doc.unlist[which(doc.unlist=='types')]='type'
doc.unlist[which(doc.unlist=='uncles')]='uncle'
doc.unlist[which(doc.unlist=='unfortunately')]='unfortunate'
doc.unlist[which(doc.unlist=='united')]='unite'
doc.unlist[which(doc.unlist=='usa')]='use'
doc.unlist[which(doc.unlist=='used')]='use'
doc.unlist[which(doc.unlist=='vacations')]='vacation'
doc.unlist[which(doc.unlist=='vaction')]='vacation'
doc.unlist[which(doc.unlist=='veiw')]='view'
doc.unlist[which(doc.unlist=='viewed')]='view'
doc.unlist[which(doc.unlist=='waited')]='wait'
doc.unlist[which(doc.unlist=='waiting')]='wait'
doc.unlist[which(doc.unlist=='woke')]='wake'
doc.unlist[which(doc.unlist=='walked')]='walk'
doc.unlist[which(doc.unlist=='walking')]='walk'
doc.unlist[which(doc.unlist=='wanna')]='want'
doc.unlist[which(doc.unlist=='wanted')]='want'
doc.unlist[which(doc.unlist=='wanting')]='want'
doc.unlist[which(doc.unlist=='wants')]='want'
doc.unlist[which(doc.unlist=='wonted')]='wanted'
doc.unlist[which(doc.unlist=='wasent')]='was_not'
doc.unlist[which(doc.unlist=='wasnt')]='was_not'
doc.unlist[which(doc.unlist=='wach')]='watch'
doc.unlist[which(doc.unlist=='watch')]='watch'
doc.unlist[which(doc.unlist=='watched')]='watch'
doc.unlist[which(doc.unlist=='watching')]='watch'
doc.unlist[which(doc.unlist=='warter')]='water'
doc.unlist[which(doc.unlist=='waters')]='water'
doc.unlist[which(doc.unlist=='waves')]='wave'
doc.unlist[which(doc.unlist=='ways')]='way'
doc.unlist[which(doc.unlist=='weeks')]='week'
doc.unlist[which(doc.unlist=='wet')]='wet'
doc.unlist[which(doc.unlist=='whats')]='what'
doc.unlist[which(doc.unlist=='whent')]='where'
doc.unlist[which(doc.unlist=='whining')]='whine'
doc.unlist[which(doc.unlist=='whispered')]='whisper'
doc.unlist[which(doc.unlist=='whit')]='white'
doc.unlist[which(doc.unlist=='wined')]='win'
doc.unlist[which(doc.unlist=='wining')]='win'
doc.unlist[which(doc.unlist=='won')]='win'
doc.unlist[which(doc.unlist=='wint')]='winter'
doc.unlist[which(doc.unlist=='wished')]='wish'
doc.unlist[which(doc.unlist=='wishing')]='wish'
doc.unlist[which(doc.unlist=='wit')]='witch'
doc.unlist[which(doc.unlist=='wondered')]='wonder'
doc.unlist[which(doc.unlist=='wonderful')]='wonder'
doc.unlist[which(doc.unlist=='wondering')]='wonder'
doc.unlist[which(doc.unlist=='words')]='word'
doc.unlist[which(doc.unlist=='works')]='work'
doc.unlist[which(doc.unlist=='worried')]='worry'
doc.unlist[which(doc.unlist=='wos')]='worse'
doc.unlist[which(doc.unlist=='woter')]='write'
doc.unlist[which(doc.unlist=='wrote')]='write'
doc.unlist[which(doc.unlist=='yards')]='yard'
doc.unlist[which(doc.unlist=='years')]='year'
doc.unlist[which(doc.unlist=='yelled')]='yell'
doc.unlist[which(doc.unlist=='yelling')]='yell'
doc.unlist[which(doc.unlist=='zoomed')]='zoom'
doc.unlist[which(doc.unlist=='TRUE')]='TRUE'
doc.unlist[which(doc.unlist=='had')]='have'
doc.unlist[which(doc.unlist=='them')]='they'
doc.unlist[which(doc.unlist=='their')]='they'
doc.unlist[which(doc.unlist=='said')]='say'
doc.unlist[which(doc.unlist=='her')]='she'
doc.unlist[which(doc.unlist=='saw')]='see'
doc.unlist[which(doc.unlist=='did')]='do'
doc.unlist[which(doc.unlist=='could')]='can'
doc.unlist[which(doc.unlist=='would')]='will'
doc.unlist[which(doc.unlist=='are')]='be'
doc.unlist[which(doc.unlist=='got')]='get'
doc.unlist[which(doc.unlist=='his')]='he'
doc.unlist[which(doc.unlist=='him')]='he'
doc.unlist[which(doc.unlist=='was')]='is'
```