The University of Akron

College of Business, Department of Management

Advanced Data Analytics Topics (ISM:663)

Project

Topic Modeling of BBC News

By:

Parwan Sharma

Student id – 4808380

MSM Business Analytics

Introduction

Topic modelling is a statistical technique which works to identify underlying or hidden themes, trends, and patterns in a collection of documents. It is usually used to analyze large volumes of unstructured text data, such as news articles or social media posts.

This report is focusing on using the Latent Dirichlet Allocation algorithm(LDA) for topic modelling of some BBC new articles. BBC new news is one of the major international news channels and cover a variety of topics including politics sports science, entertainment, business etc. Using Latent Dirichlet Allocation algorithm for fitting a topic model, we will identify the categories or topics which are covered in these articles and analyze the patterns in these topics.

Chart, histogram

Description automatically generated

LDA algorithm is one of the most widely used model for topic modelling. In LDA, it is assumed that every document consists of a blend of limited number of topics, and each topic can be defined by a set of words(one of the major Assumption). By estimating the mixtures of topics for each document and the distribution of words for each topic, LDA can offer a summary of the important themes and ideas that are prevalent in the BBC news corpus. The field of natural language processing has shown great interest in Topic Modelling, and several algorithms have been created to assist researchers and businesses in discovering the underlying themes within their data.

This report will present an in-depth evaluation of the outcomes of topic modeling, encompassing the recognized topics and their prevalence, the keywords associated with each topic, and the development of topics over a certain period.

Diagram

Description automatically generated

Problem Description

With the growing demand for UpToDate information of the world, major business groups, especially news channels produce a large quantity of text-based data in the form of news articles, media post etc. To be effective with the quality updates of these text based, one should be able to identify the theme on which the topics are based. Due to the large quantity of manually analysing such massive datasets is a laborious task and can lead to inaccuracies.

Hence, the objective of this report is to use the Latent Dirichlet Allocation (LDA) algorithm on BBC news articles and identify the key topics covered by this global news organization. The primary hurdle is to tackle the extensive dimensionality of the data as one article can have mixed reviews and may cover multiple topics at once, while detecting significant topics that represent the content of a particular topic-based news article.

The goal of this report is to offer businesses and researchers a thorough comprehension of the significant topics covered by the BBC news, along with the associated trends and patterns. By achieving this, it is possible to derive valuable insights into the audience's interests and preferences, which can aid in making well-informed decisions.

Objectives:

The primary objective of this report is to identify topics covered by BBC new by applying Latent Dirichlet Allocation(LDA) algorithm. This includes a set of following objectives:

* Gather and process a collection of BBC news articles.
* Utilize the LDA algorithm to examine the pre-processed dataset to find principal topics that the BBC news covers.
* Evaluate topic modeling results.
* Identify the prominent keywords related to each topic and scrutinize the semantic meaning of every topic.
* Analyze the topics over time to identify any trends or patterns in the BBC news corpus.
* Provide recommendations and insights based on the findings of the topic modeling analysis.

Methodology

This report is based on the analysis of the public data set called “BBC”. The literature used is Brett Lantz, Machine Learning with R, 2nd Ed., Packet Publishing, 2015,ISBN: 9781784393908.

Steps taken:

Collecting data

The data set used consist of articles published by the BBC news. The first dataset, named BBC dataset has 2225 articles. These articles are categorized into five topics which are: Technology , Business, Entertainment, Sports and politics. The second dataset is called BBCsports which consist of 737 articles based on five different types of sports, that is, tennis, football, athletics, cricket, rugby.

Text

Description automatically generated with medium confidence

We will assign path to both the folders and used the variable bbc and bbcsports respectively. Each folder includes four file, that is bbc.classes, bbc.terms, bbc.docs, bbc.mtx.

The file bbc.term contain terms, which is one per line and is the row of he matric. Similarly bbc.docs contain the document name which is the column names for the matrix. Bbc.mtx contains a term document matrix. It is a matrix where column represent the terms such as phrases and words, and row represents the article or essays and passages. The cells of the passage indicates the frequency of terms in a specific document.

In this case we are not performing any pre processing such as removing number or unwanted character and punctuation removal etc, as the BBC data set is pre processed and the words were stemmed as well as the words which were used less than three times were removed.

Text

Description automatically generated with medium confidence

In this scenario, the articles are represented by the columns of the matrix, while the terms found in the articles are represented by the rows. The entries in the matrix, denoted by M[i,j], which indicate the number of occurrences of the corresponding term in row i in the document related to column j. The Matrix Market format is used to store the matrix in a compressed and efficient manner, enabling seamless exchange of matrix data between different scientific computing programs. The file format comprises a header section describing the matrix type, size, and structure, followed by the matrix data stored in either coordinate or dense format.

The Matrix Market format has become a standard for exchanging matrix data between various applications in numerical linear algebra and related areas, such as optimization, finite element analysis, and graph theory. Numerous scientific computing software tools and libraries support this format, making it widely used in these fields.

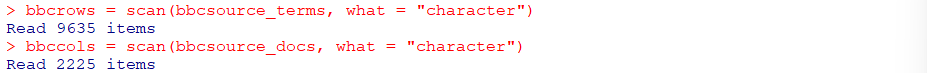
We will use the readMM function to load the data since we cannot load data from a Market Matrix directly. This function will load and store the data into a matrix object.

We use the tm package to convert it into a term document matrix using as.TermDocumentMatrix() function. To distinguish the numbers in the original matrix and their corresponding meaning, a weighting parameter is necessary in this scenario. Given that the term frequencies are in their raw form, we can utilize the weightTf value to indicate this.



We extract the terms and document identifiers from the remaining two files and use them to generate appropriate labels for the rows and columns of the matrices that hold the term and document information.

To read files containing one entry per line and store the data in vectors, we can utilize the standard scan() function. We will use the term vector and document identifier vector to update the names of the rows and columns in the term-document matrix. Finally, we will transpose the matrix into a document-term matrix since this format is required for the subsequent steps.

A picture containing text

Description automatically generatedText

Description automatically generated with medium confidenceA picture containing text

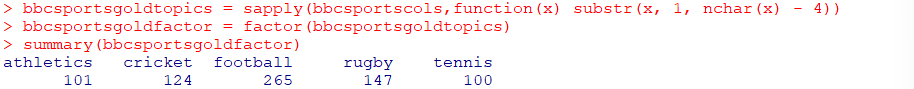
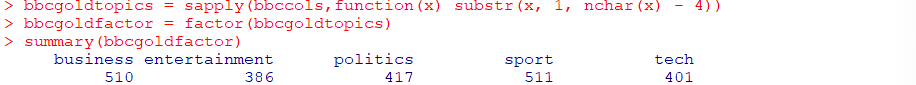
Description automatically generated

Result shows that bbc having 9635 terms and 2225 dcumnets whereas bbcsports has 4613 terms and 737 documents, indicating that dataset bbc is larger than bbcsports. The format of documents is <topic.><counter>.

Text

Description automatically generated with medium confidence

We extract the last four character of each entry to change in into factors.



We observe that there are 510 articles related to business, 386 on entertainment, 417 on politics, 511 on sports, and 401 on tech, respectively. Regarding the BBCsports dataset, football articles are slightly dominant with 265 articles, whereas the other topics contain fewer than 150 articles each.

The topic model package provides a range of tool which includes, Topic model algorithms such as Correlated topic models(CTM), Latent Dirichlet(LDA) and Hierarchical Dirichlet process (HDP). Model Diagnostics tools such as visualization of the distribution of the topics. Model visualization which involves topic-word heatmap, dendrograms and word clouds. It can also classify documents into topics.

In this case we use four different models:

1. LDA\_VEM(Latent Dirichlet Allocation with Variational Expectation Maximization)

It is commonly used algorithm for training LDA topic models, where the topic proportions of each document are assumed to follow a Dirichlet distribution, and the topic-word probabilities follow another Dirichlet distribution. The model parameters are estimated using a variational inference technique, and the algorithm updates the topic proportions and the topic-word probabilities until convergence.

1. LDA\_VEM\_α (Latent Dirichlet Allocation with Variational Expectation Maximization and an asymmetric Dirichlet prior)

It uses asymmetric Dirichlet prior and allow the model to account situations where few topics are present in high number in some documents and hence can be highly useful for handling dataset with imbalances.

1. LDA\_GIB(Latent Dirichlet Allocation with Gibbs Sampling)

It uses Gibbs sampling, a type of Markov Chain Monte Carlo method used to estimate the model parameters. It is typically utilized when the dataset is too large to fit in memory or when the standard LDA\_VEM algorithm fails to converge.

1. CTM\_VEM(Correlated Topic Models with Variational Expectation Maximization)

It is a probabilistic model used for topic modeling, like Latent Dirichlet Allocation (LDA).The main difference between CTM\_VEM and LDA is that CTM models correlations between topics. CTM\_VEM assumes that each document is generated from a mixture of topics, where the topics are drawn from a multivariate normal distribution with a mean and covariance matrix.

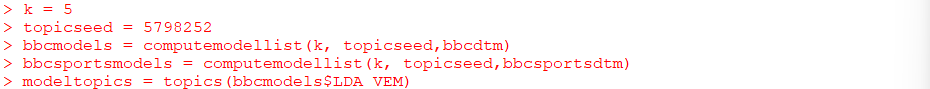
We will be using the LDA() function and we already have the topic model package installed and loaded. The LDA() function require few parameters to function. First is K, which indicates the set number of parameters we want to generate. Second is the specific document term we want to use. Third is the method or the training algorithm we will use.

In this case we will set a seed parameter as the topic model involves rendom elements. At this point, we can indicate if we wish to calculate the α Dirichlet parameter, as well as define the Gibbs sampling parameters, like the number of discarded Gibbs iterations at the beginning of the training process (burnin), the number of omitted intermediate iterations (thin), and the overall number of Gibbs iterations (iter). The CTM() function from the topicmodels package can be used to train a CTM model, which has a syntax similar to that of the LDA() function.

A picture containing text

Description automatically generated

Now that we have the function ready, we can train all the four model with the data set.



We use topic() function to extract a vector with high probability of topic selected for each document. The value of K is set at 1 (by default) , which is showing the top 1 prediction per topic.

Following is the result of a BBC set using LDA\_VEM:

Table

Description automatically generated with low confidence

The result shows that the topics 3,4 and 5 match with business, politics and sports respectively. Whereas there is no correlation between topic 1 and 2. Ideally each model topic should match with each gold topics. Hence this model was not successful in all categories.

We now test model LDA\_GIB :

Table

Description automatically generated with low confidence

We conclude that LDA\_GIB performs better as each topic mostly matches with one of the labelled values. To obtain more precise values of performance, we can calculate accuracy by dividing the maximum row values by the total number of documents.

For LDA\_VEM, this value is (176+202+483+403+507)/2225 = 1771/2225 = 79.6 percent, and for LDA\_GIB, it is (471+506+371+399+364)/2225 = 2111/2225= 94.9 percent.

We can use the following function to automatically compute this value for all models:

Text

Description automatically generated with low confidence

We conclude that for the bbc data set, LDA\_GIB show best performance of 95% followed by LDA\_VEM at 80% and LDA\_VEM\_**α** at 79% and the lowest one is CTM\_VEM at 69%.

For the bbcsports data set, LDA\_VEM showed the highest value at 79% and CTM\_VEM showed the lowest.

Graphical user interface, text, application

Description automatically generated

We use the logLik() function which evaluates the model performance based on loglikelihood. We conclude from the result that the Gibbs sample showed the largest value in both datasets. Similarly, the second-best model for both the dataset is LDA\_VEM.

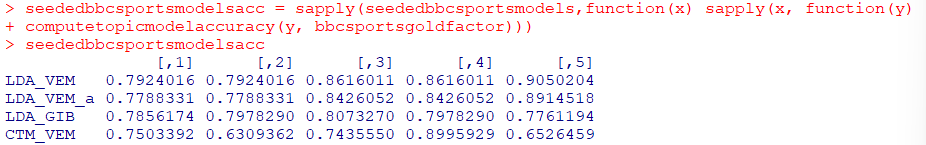
Graphical user interface, text, application

Description automatically generated

Model Stability

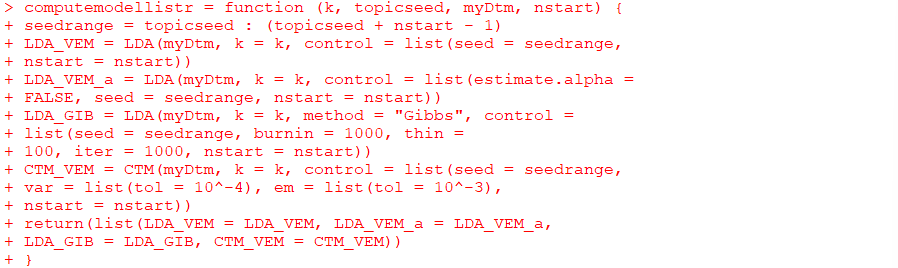
We can also check the accuracy of the model by using different random number seeds which can produce significantly different results at time. We can achieve this by training our model with random seed numbers to see their effects.

Calendar

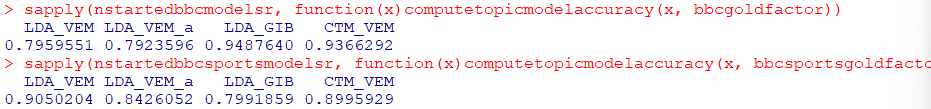
Description automatically generated with medium confidence

Based on the image above, it appears that the LDA\_GIB model is the most consistent, as its values remain relatively unchanged across all five seeds for both the BBC and BBCsports datasets. The accuracy for the BBC dataset is highest for the Gibbs sampling method, while for the BBCsports dataset, the LDA\_VEM model performs slightly better than the other models, including LDA\_VEM\_a. However, the CTM\_VEM model appears to be the least stable, with more variable values across both datasets. It is worth noting that the VEM models are generally faster than the GIB model when working with larger datasets.

We made changes to the computemodellist() function to create a new function called computemodellistr(). This function includes a parameter called nstart. To ensure that the sequence of seeds used for the random restarts is of the same length as nstart, the seed parameter now requires a sequence of seeds. We can generate a sequence of seeds that starts from the initial seed provided by using a range function.



This function is used to create new model list for training.



Finding the number of Topics

In the context of using topic modeling as a form of exploratory analysis to cluster documents based on the similarity of their topics, the number of topics may not be predetermined. This can pose a challenge like selecting the number of clusters when performing clustering. One suggested solution is to conduct cross-validation over a range of topic numbers. However, this method may not be practical for large datasets since training a single topic model is computationally intensive, particularly when considering factors like random restarts.

Topic Distribution

To sample a multinomial distribution of topics, we utilize a Dirichlet distribution. In the LDA\_VEM model, the parameter vector αk is estimated. It's important to note that in our implementation, we use a symmetric distribution. As such, we only estimate the value of α, which is the value that all αk parameters share. When examining the LDA models, we can analyze the value of this parameter.

Background pattern

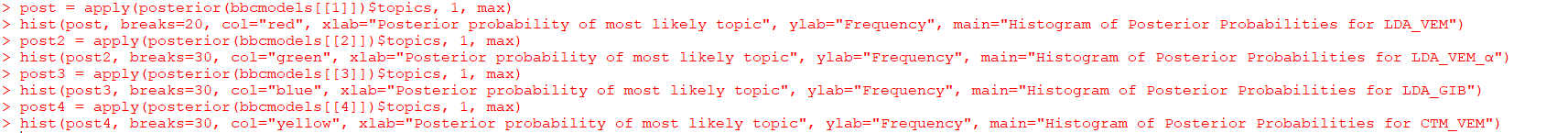
Description automatically generated with low confidence

The estimated value of α was significantly lower than the value we initially used. This indicates that there was a concentrated topic distribution for both datasets. This was further demonstrated by visualizing the posterior probabilities of topics using the posterior() function. For instance, when applying an LDA\_GIB model to the BBC dataset, the posterior probabilities were as follows:

Text, letter

Description automatically generated

The LDA\_VEM\_a and LDA\_GIB models have flatter distributions, with values evenly spread throughout the range without notable peaks. Whereas, both LDA\_VEM and CTM\_VEM models exhibit a concentrated distribution, although the CTM\_VEM model has a wider range.



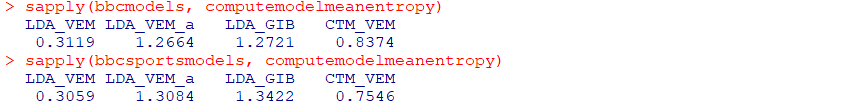
Chart, histogram

Description automatically generated

Another Method to check the quality of distribution is by calculating average entropy. In statistics and information theory, entropy serves as a metric for quantifying the level of uncertainty or randomness in each probability distribution. When a distribution has high entropy, there are many potential outcomes with roughly equal probabilities, while low entropy distributions have fewer outcomes with very distinct probabilities. Mathematically, entropy is frequently expressed as the negative sum of the probabilities of each outcome multiplied by the logarithm of the probability.

We usecomputeentropy() to calculate entropyof specific topic of a document. The we use computemodelmeanentropy() is used to calculate everage entropy in all documents.

A picture containing text

Description automatically generated

Like the graphs, LDA\_VEM has lowest values in both datasets(peakiest distribution), CTM\_VEM is the second peakiest distribution. The other two models showed similar results with smoother distribution.

Word Distribution

In this step we will check the frequent terms for each document and topic. As shown in the picture below, we will first check for the first 10 terms using the function terms().

Text

Description automatically generated

To visualize the above mentioned word and their frequencies =, we will use the wordcloud. To make a word cloud, we will install the wordcloud package. This will plot a cloud of the words in which the size of the word would be directly proportional to the frequency, hence the size would represent the word frequency.

In this case, we had to adjust the DTM to get the list of frequency. We do that using the following function.

Text

Description automatically generated with low confidence

The function works by first identifying the most common terms for each topic based on their most likely assignments. It then selects the corresponding cells in the term-by-document matrix and sums the columns to compute the frequency of each term. Finally, the function generates a word cloud based on the frequency of the top terms. In this instance, we utilized the LDA\_VEM\_a model on the BBC dataset with all 5 topics and 40 most common terms per topic.

Text

Description automatically generated

Topic 1 Topic 2

Text

Description automatically generated Text

Description automatically generated

Topic 3 Topic 4

Text

Description automatically generated Text

Description automatically generated

Topic 5

Text

Description automatically generated

Limitations of LDA

Some of the limitations of the LDA model are

* Limit to number of topics: LDA assumes a fixed number of topics, which needs to be set manually.
* Unable to depict correlations: LDA assumes that each topic is a collection of words that are randomly grouped together. This can lead to the occurrence of uncorrelated topics that do not accurately represent the underlying relationships between words.
* No development of topics over time: LDA assumes that topics are static and do not evolve over time. This can be a limitation when analyzing text data where topics are likely to change as new information is added.
* Exchangeable words, no sentence structure: LDA assumes that words are exchangeable, meaning that the order in which they appear does not matter.
* Unsupervised: LDA is an unsupervised method, meaning that it does not use labelled data to train the model.

Conclusion

Upon performing the topic modeling using the LDA algorithm on two real-time datasets, that is the BBC dataset and the bbcsports dataset. First, we constructed a LDA model utilizing a document term matrix, with the second parameter, k, representing the ideal number of topics for our model. We applied training algorithms to our datasets, utilizing both Gibbs sampling and Variational Expectation Maximization (VEM).

Upon applying these algorithms to our datasets, we observed that the LDA GIB model significantly outperformed the other models on the BBC dataset, while the CTM VEM model has the lowest performance in comparison to the LDA models. Furthermore, all models exhibited comparable performance on the bbcsports dataset with the LDA VEM model narrowly edging out the others.

This report revealed that Gibbs sampling generates a more consistent model for both datasets, with the added benefit of greater accuracy on the bbc datasets.

Coding

> bbc = "C:/Users/Meenu\_Sharma/Downloads/bbc/"

> bbcsports = "C:/Users/Meenu\_Sharma/Downloads/bbcsport/"

> bbcsource = paste(bbc,"bbc.mtx", sep = "")

> bbcsource\_terms = paste(bbc, "bbc.terms", sep = "")

> bbcsource\_docs = paste(bbc, "bbc.docs", sep = "")

> bbcsportssource = paste(bbcsports, "bbcsport.mtx",sep = "")

> bbcsportssource\_terms = paste(bbcsports,"bbcsport.terms", sep = "")

> bbcsportssource\_docs = paste(bbcsports,"bbcsport.docs", sep = "")

> bbcmatrix = readMM(bbcsource)

> bbctdm = as.TermDocumentMatrix(bbcmatrix, weightTf)

> bbcsportsmatrix = readMM(bbcsportssource)

> bbcsportstdm = as.TermDocumentMatrix(bbcsportsmatrix,weightTf)

> bbcrows = scan(bbcsource\_terms, what = "character")

> bbccols = scan(bbcsource\_docs, what = "character")

> bbctdm$dimnames$Terms = bbcrows

> bbctdm$dimnames$Docs = bbccols

> (bbcdtm = t(bbctdm))

> bbcsportsrows = scan(bbcsportssource\_terms, what = "character")

> bbcsportscols = scan(bbcsportssource\_docs, what = "character")

> bbcsportstdm$dimnames$Terms = bbcsportsrows

> bbcsportstdm$dimnames$Docs = bbcsportscols

> (bbcsportsdtm = t(bbcsportstdm))

> bbccols[1:5]

> bbcsportscols[1:5]

> bbcgoldtopics = sapply(bbccols,function(x) substr(x, 1, nchar(x) - 4))

> bbcgoldfactor = factor(bbcgoldtopics)

> summary(bbcgoldfactor)

> bbcsportsgoldtopics = sapply(bbcsportscols,function(x) substr(x, 1, nchar(x) - 4))

> bbcsportsgoldfactor = factor(bbcsportsgoldtopics)

> summary(bbcsportsgoldfactor)

> computemodellist = function (k, topicseed, myDtm){

+ LDA\_VEM = LDA(myDtm, k = k, control = list(seed = topicseed))

+ LDA\_VEM\_a = LDA(myDtm, k = k, control = list(estimate.alpha = FALSE, seed = topicseed))

+ LDA\_GIB = LDA(myDtm, k = k, method = "Gibbs", control = list(seed = topicseed, burnin = 1000, thin =

+ 100, iter = 1000))

+ CTM\_VEM = CTM(myDtm, k = k, control = list(seed = topicseed,

+ var = list(tol = 10^-4), em = list(tol = 10^-3)))

+ return(list(LDA\_VEM = LDA\_VEM, LDA\_VEM\_a = LDA\_VEM\_a,LDA\_GIB = LDA\_GIB, CTM\_VEM = CTM\_VEM))

+ }

> install.packages("topicmodels")

> library("topicmodels")

> k = 5

> topicseed = 5798252

> bbcmodels = computemodellist(k, topicseed,bbcdtm)

> bbcsportsmodels = computemodellist(k, topicseed,bbcsportsdtm)

> modeltopics = topics(bbcmodels$LDA\_VEM)

> table(modeltopics, bbcgoldfactor)

> modeltopics = topics(bbcmodels$LDA\_GIB)

> table(modeltopics, bbcgoldfactor)

> computetopicmodelaccuracy = function(model, goldfactor) {

+ modeltopics = topics(model)

+ modeltable = table(modeltopics, goldfactor)

+ modelmatches = apply(modeltable, 1, max)

+ modelaccuracy = sum(modelmatches) / sum(modeltable)

+ return(modelaccuracy)

+ }

> sapply(bbcmodels, function(x)computetopicmodelaccuracy(x, bbcgoldfactor))

> sapply(bbcsportsmodels, function(x)computetopicmodelaccuracy(x, bbcsportsgoldfactor))

> sapply(bbcmodels, logLik)

> sapply(bbcsportsmodels, logLik)

> sapply(bbcsportsmodels, logLik)

> seededbbcmodels = lapply(5798252 : 5798256,function(x) computemodellist(k, x, bbcdtm))

> seededbbcsportsmodels = lapply(5798252 : 5798256,function(x) computemodellist(k, x,bbcsportsdtm))

> seededbbcmodelsacc = sapply(seededbbcmodels,function(x) sapply(x, function(y)

+ computetopicmodelaccuracy(y, bbcgoldfactor)))

> seededbbcmodelsacc

> seededbbcsportsmodelsacc = sapply(seededbbcsportsmodels,function(x) sapply(x, function(y)

+ computetopicmodelaccuracy(y, bbcsportsgoldfactor)))

> seededbbcsportsmodelsacc

> computemodellistr = function (k, topicseed, myDtm, nstart) {

+ seedrange = topicseed : (topicseed + nstart - 1)

+ LDA\_VEM = LDA(myDtm, k = k, control = list(seed = seedrange,

+ nstart = nstart))

+ LDA\_VEM\_a = LDA(myDtm, k = k, control = list(estimate.alpha =

+ FALSE, seed = seedrange, nstart = nstart))

+ LDA\_GIB = LDA(myDtm, k = k, method = "Gibbs", control =

+ list(seed = seedrange, burnin = 1000, thin =

+ 100, iter = 1000, nstart = nstart))

+ CTM\_VEM = CTM(myDtm, k = k, control = list(seed = seedrange,

+ var = list(tol = 10^-4), em = list(tol = 10^-3),

+ nstart = nstart))

+ return(list(LDA\_VEM = LDA\_VEM, LDA\_VEM\_a = LDA\_VEM\_a,

+ LDA\_GIB = LDA\_GIB, CTM\_VEM = CTM\_VEM))

+ }

> nstart = 5

> topicseed = 5798252

> nstartedbbcmodelsr = computemodellistr(k, topicseed, bbcdtm, nstart)

> nstartedbbcsportsmodelsr = computemodellistr(k, topicseed, bbcsportsdtm,

+ nstart)

> sapply(nstartedbbcmodelsr, function(x)computetopicmodelaccuracy(x, bbcgoldfactor))

> sapply(nstartedbbcsportsmodelsr, function(x)computetopicmodelaccuracy(x, bbcsportsgoldfactor))

> bbcmodels[[1]]@alpha

> bbcmodels[[2]]@alpha

> bbcsportsmodels[[1]]@alpha

> bbcsportsmodels[[2]]@alpha

> options(digits = 4)

> head(posterior(bbcmodels[[1]])$topics)

> computeentropy =function(probs) {return(- sum(probs \* log(probs)))

+ }

> computemodelmeanentropy = function(model) {topics = posterior(model)$topics

+ return(mean(apply(topics, 1, computeentropy)))

+ }

> sapply(bbcmodels, computemodelmeanentropy)

> sapply(bbcsportsmodels, computemodelmeanentropy)

> post = apply(posterior(bbcmodels[[1]])$topics, 1, max)

> hist(post, breaks=20, col="red", xlab="Posterior probability of most likely topic", ylab="Frequency", main="Histogram of Posterior Probabilities for LDA\_VEM")

> post2 = apply(posterior(bbcmodels[[2]])$topics, 1, max)

> hist(post2, breaks=30, col="green", xlab="Posterior probability of most likely topic", ylab="Frequency", main="Histogram of Posterior Probabilities for LDA\_VEM\_α")

> post3 = apply(posterior(bbcmodels[[3]])$topics, 1, max)

> hist(post3, breaks=30, col="blue", xlab="Posterior probability of most likely topic", ylab="Frequency", main="Histogram of Posterior Probabilities for LDA\_GIB")

> post4 = apply(posterior(bbcmodels[[4]])$topics, 1, max)

> hist(post4, breaks=30, col="yellow", xlab="Posterior probability of most likely topic", ylab="Frequency", main="Histogram of Posterior Probabilities for CTM\_VEM")

>GIB\_bbcmodel = bbcmodels[[3]]

> terms(GIB\_bbcmodel,10)

> plotwordcloud = function(model, myDtm, index, numTerms) {

+ modelterms = terms(model,numTerms)

+ modeltopics = topics(model)

+ termsi = modelterms[,index]

+ topici = modeltopics == index

+ dtmi = myDtm[topici, termsi]

+ frequenciesi = colSums(as.matrix(dtmi))

+ wordcloud(termsi, frequenciesi, min.freq = 0)

+ }

> plotwordcloud(GIB\_bbcmodel, bbcdtm,1,32)

> plotwordcloud(GIB\_bbcmodel, bbcdtm,2,32)

> plotwordcloud(GIB\_bbcmodel, bbcdtm,3,32)

> plotwordcloud(GIB\_bbcmodel, bbcdtm,4,32)

> plotwordcloud(GIB\_bbcmodel, bbcdtm,5,32)

>