Statistical Issues in Quantifying Text Mining Performance

by

Christine Peijinn Chai

Department of Statistical Science
Duke University

Date: ____________________________

Approved:

____
David L. Banks, Supervisor

____
Jerome Reiter

____
Sayan Mukherjee

____
James Moody

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Statistical Science
in the Graduate School of Duke University
2017

Abstract

Statistical Issues in Quantifying Text Mining Performance

by

Christine Peijinn Chai

Department of Statistical Science
Duke University

Date: ______________________

Approved:

____
David L. Banks, Supervisor

____
Jerome Reiter

____
Sayan Mukherjee

____
James Moody

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Statistical Science in the Graduate School of Duke University
2017

Abstract

Text mining is an emerging field in data science because text information is ubiquitous, but analyzing text data is much more complicated than analyzing numerical data. Topic modeling is a commonly-used approach to classify text documents into topics and identify key words, so the text information of interest is distilled from the large corpus sea. In this dissertation, I investigate various statistical issues in quantifying text mining performance, and Chapter 1 is a brief introduction.

Chapter 2 is about the adequate pre-processing for text data. For example, words of the same stem (e.g. "study" and "studied") should be assigned the same token because they share the exact same meaning. In addition, specific phrases such as "New York" and "White House" should be retained because many topic classification models focus exclusively on words. Statistical methods, such as conditional probability and p-values, are used as an objective approach to discover these phrases.

Chapter 3 starts the quantification of text mining performance; this measures the improvement of topic modeling results from text pre-processing. Retaining specific phrases increases their distinctivity because the "signal" of the most probable topic becomes stronger (i.e., the maximum probability is higher) than the "signal" generated by any of the two words separately. Therefore, text pre-processing helps recover semantic information at word level.

Chapter 4 quantifies the uncertainty of a widely-used topic model – latent Dirichlet allocation (LDA). A synthetic text dataset was created with known topic proportions, and I tried several methods to determine the appropriate number of topics from the data. Currently, the pre-set number of topics is important to the topic model results because LDA tends to utilize all topics allotted, so that each topic has about equal representation.

Last but not least, Chapter 5 explores a few selected text models as extensions, such as supervised latent Dirichlet allocation (sLDA), survey data application, sentiment analysis, and the infinite Gaussian mixture model.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Symbols

R	The programming language
$\mathbb{R}$	The real line

Abbreviations

EM	Expectation maximization
LDA	Latent Dirichlet allocation
LSI	Latent semantic indexing
PCA	Principal component analysis
sLDA	Supervised latent Dirichlet allocation
SVD	Single value decomposition
TF-IDF	Term frequency – inverse document frequency

Acknowledgements

I would like to thank many people who have involved in my PhD process. First, I am extremely grateful to my advisor, Dr. David Banks, for the guidance and support throughout my research. He is always positive to my ideas and provides timely feedback. I owe a great deal of gratitude to Dr. Jerry Reiter, for offering significant career help and referrals. I would also like to thank Dr. Sayan Mukherjee and Dr. Jim Moody for serving on my committee and for giving valuable advice. For the text datasets, I appreciate MaxPoint Interactive's providing the Trayvon Martin corpus and the Benghazi corpus. Also thanks to Nick Fisher at ValueMetrics, Australia for providing the employee satisfaction dataset.

Next, I would like to thank the whole text mining group for discussing research ideas and mentally support each others. The student members include Brice Acree, Tianlin Duan, Yuan Andrea Gao, Chris Glynn, Sophie Guo, Christoph Hellmayr, Teague Henry, Mengrun Li, Derek Owens-Oas, Min Jung Park, Lin Xiao, and Jun Yoon (in alphabetical order). I give a special mention to the Advanced Machine Learning course taught by Dr. Katherine A. Heller, in which she provided feedback to my research proposal and poster. The discussions with with Peter Shi and Shijia Bian were also helpful.

Third, I have received travel funding from several conferences, and I appreciate their support for enriching my experience. International Conference on Advances in Interdisciplinary Statistics and Combinatorics (AISC) offered me funding as a young researcher in both 2014 and 2016; this could not have been possible without the organizer, Dr. Sat Gupta's help. I also received partial travel support from Women in Statistics and Data Science (WSDS) 2016, and thanks Dr. Mine Cetinkaya-Rundel for reimbursing me the rest of the expenses. In addition, PyData Carolinas 2016 provided me a conference registration waiver, and Industrial Math/Stat Modeling Workshop for Graduate Students 2015 offered me nearly two weeks of housing at North Carolina State University (special thanks to Dr. Ilse Ipsen and Thomas Gehrmann for organizing!). Last but not least, I would like to thank the department administrative employees – Lori, Karen, Nikki, Anne for taking care of the paperwork required.

Furthermore, I am grateful for the supportive fellow department members; they play a significant role in helping me make it through the past four years. The PhD students include and are not limited to: Mary Beth Broadbent, Shih-Han Sean Chang, Xi Chen, Nicole Dalzell, Maryclare Griffin, George Helman, Monika Jingchen Hu, Ksenia Kyzyurova, Maggie Nguyen, Victor Peña, Sayan Patra, Eric Ye Wang, and Lu Wang, and Lan Wei. The MSS and MSEM students include (and are not limited to, either) Radhika Anand, Fan Dai, Dipesh Gautam, Tun-Chieh Eric Hsu, and Jane Yu.

For the campus life outside the department, I definitely would like to thank Duke Grad TSA (Taiwanese Student Association) for hosting celebration and networking activities every semester, which makes me feel like home. I would also like to thank many of my friends for the lovely in-person conversations: Sang Chang, Hsuan-Jung Chen, Lee H. Chen, Rui Chen (Econ), Xing Chen (Nicholas School of the Environment), Serene Cheng, Gordon Fields, Guan-Wun Catherine Hao (best roommate ever!), Li-Feng Benjamin Jiang-Xie, Yuan-Mao Kao, Yi-Hong Ke, Li Liang (Econ), Jin Liang, Song Lin, Jiangang Liu, Chung-Ru Lee, Po-Chen Ellie Lo, Bo Ning, Xiaodan Qiu, Andy Rosko, Ching-Chuan David Wang, Jui Wang, Tai-Lin Wu, Xinyi Cynthia Wu, and Guodong Zhang.

In addition, I am thankful for the summer internship opportunities at the National Mortgage Database, Federal Housing Finance Agency. Thanks to Bob Avery, Forrest Pafenberg, Saty Patrabansh, and Tim Critchfield (joint appointment at the Consumer Financial Protection Bureau) for making this happen. The experience led me to rethink about survey and text mining; I learned how to perform missing data imputation and error correction in big data. During my time at the agency, I had great interactions with the fellow interns – David Agorastos, Andrew Bent, Rachel Ann Brown, Angela Chang, Kristen Cugini, Anna Eapen, Evan Flack, Cami Grimm, Mark Kinkle, and Brock Zeman. Thanks Paul Nguyen for organizing the intern activities and outings. Thanks Allena Johnson and Janice Obeido for being really warm and supportive – "If you have questions and don't know who to contact, ask me." goes a long way. My time as a student trainee was colorful and exciting because of everyone at the Federal Housing Finance Agency.

The summers in Washington DC were part of my best memories. I am grateful to many amazing friends I made there – Yen E. D'Anda (Sunrise Bridge), Andrea Fritz, Nicholas Fritz, Melissa Grimm, Rob Letzler (GAO), Hiroaki Minato (DOE), Trang Nguyen (Census Bureau), and Wen-Bin Yang (NIST). They significantly helped me in the transition between a graduate student and an intern life. I lived in Thompson-Markward Hall[1] in May-August 2016; it is the best place that I have ever stayed at. Warm atmosphere; convenient location; breakfast and dinner provided on weekdays. Thanks for the excellent community of friends for making this happen – Cathi Coridan (Executive Director), Katie Proch (Assistant Director), Megan Lisbeth St (Resident Manager), Brianna Carrier, Holly YF Chung, Dian Maharani, Sherri Sheu, and Patricia Weng.

I would also like to thank the USBF Junior Training Program for providing an online bridge platform with assigned mentors. The mentors I have closely interacted with are Michael Golden, Barry Goren, Michael Rosenberg, Aviv Shahaf, and Joe Stokes. I have had many bridge partners, and I especially would like to mention Sedef Berk (best partner ever!), Ellie Fashingbauer, Hal Jones, Chih-Kuang Kevin Lee, Kai Liu, Henry Meguid, Robert Thorstad, and Phylliss Woody. We have finished at least one major tournament together in the US, and the scores would have been impossible without them.

Moreover, I appreciate many friends who encouraged me online from different

[1]`https://tmhdc.org/`

cities and countries – Evan Ampow, Nan-Chen Chen, Ying-Ju Tessa Chen, Spike Cheng, Ian Chin, Yu-Yun Lavender Dai, Shing Hu, Cindy Szu-ying Huang, Maggie Huang, Angel Li, Frank Meng Lin, Morph Mitmosfet Lin, Lucy Lee, Yoda Lee, Chloe YC Ouyang, Sylvia Tsai, Keng Hua Tu, Charles Wang, Tsui-Wei Lily Weng, and Lin-Chi "507" Wu. We have not seen each other for a long time, and thanks for all friends that think of me.

Last but not least, I am immensely grateful for the support of my parents, Yen-Hsin Chai and Yu-Fan Yang, and my brothers, Eric and Aaron. My family has always been warm and welcome; they provide me unconditional love and wish me the best of success in whatever I pursue.

1

Introduction

Text mining is the automatic process of analyzing text data and classifying them into various topics. My research can be divided into four parts: text data cleaning, topic modeling improvement from text cleaning, topic modeling uncertainty quantification, and exploration of selected text models.

First, text data cleaning is the pre-processing of text datasets in order to be used in analysis phases, such as removing unwanted formatting and preserving semantic information. Next, topic modeling is the methodology of assigning documents to topics, so humans do not have to read through the whole dataset to get the relevant information. The quantification of topic modeling results is divided into two parts: the improvement from text cleaning and the uncertainty within the topic model. Finally, I explore several text models as extensions, such as supervised topic modeling and sentiment analysis.

1.1 Text Data Cleaning

To begin the text mining research, I investigate the text data cleaning methodology because this process is essential to a high-quality analysis (Palace, 1996). Text requires different preparation than numerical variables because it is structured differently – some text contains valuable information and some does not (Chai, 2014).

When web-scraping from the Internet, unwanted formatting is removed (e.g. "<br/>" in HTML) and some non-words (e.g. "&nbsp", "46b08c") because they are unrelated to the text content. Even when all text is retained, semantic information can still be lost because many text classification techniques use bag-of-words models. For example, the phrase "stand your ground law" refers to the right of a person to fight back while in serious danger, but neither of the separate words, "stand", "your", "ground", "law", has this meaning. Therefore, I implement n-gramming by connecting these words with a bottom line, so they will be considered a single token in bag-of-words models.

In addition, many other text data cleaning techniques are also available, such as removing stop words with little semantic meaning and tokenization, i.e. assign words of the same stem to a token (e.g. "study" and "studied" become "studi"). Each step of text data cleaning can be explored, and different methods work best for different types of text data.

1.2 Topic Modeling Improvement from Text Cleaning

Topic modeling is the main part of text analysis, and I create two extensions based on the latent Dirichlet allocation (LDA), which is a widely used topic assignment approach. LDA is a Bayesian data generative process, which draws topics from a Dirichlet distribution as the prior, then updates the probabilities by using the words in the documents (Blei et al., 2003).

The first extension is quantifying the topic modeling results to measure the importance of preserving special phrases. I compare the results for two versions of the dataset, before and after n-gramming. The biggest improvement lies in word distinctivity; i.e., given a certain word, how likely does the document belong to one topic? When two words form a special phrase, the phrase has higher distinctivity than either of the words. For example, the phrase “black panther” has 90.1% probability of distinction in a political blog corpus because this refers to the Black Panther Party. In comparison, “black” has 48.4% and “panther” has 52.5% (Chai, 2016).

The second extension is to quantify the uncertainty within the latent Dirichlet allocation (LDA) results. Hence my research group created a synthetic dataset from Wikipedia articles, and topic proportions are set beforehand. The dataset contains 100 documents with 500 words each, and it is pre-processed using text data cleaning methods.

I attempted to determine the number of topics needed for a corpus based on various criteria, including the number of words in each topic, the number of documents in each topic, and topic weights. However, LDA utilizes all pre-set topics and "spreads out" the topic assignments. As a result, it is hard to remove unnecessary topics in LDA, if not impossible.

To investigate how rare words affect the topic modeling results, I also tried to determine the cutoff frequency for selecting rare words. For the same cleaned synthetic dataset, I removed words that had less than a certain number of appearances and varied the threshold. The result also indicated practical challenges that arise in domain applications.

1.3 Exploration of Selected Text Models

In Chapter 5, I explore several text models, such as supervised topic models, survey data applications, sentiment analysis, and a joint sentiment/topic model. Supervised topic models are used to analyze datasets with both text and numerical information; examples include surveys and movie review datasets. The supervised latent Dirichlet allocation (sLDA) is able to predict ratings given certain topics (Mcauliffe and Blei, 2007). In addition, most survey data contains text responses, so text mining can reveal patterns in the dataset, leading to error correction.

Moreover, I explore sentiment analysis to compare the polarity of documents, using an existing list of positive and negative words from the `R` package `tm.plugin.`⊕ `sentiment`. After working on topic detection and sentiment analysis, I also investigate in a joint sentiment/topic model (Lin and He, 2009), so I implement a hierarchical Bayesian data generative process to simultaneously analyze topics and sentiments.

Finally, I review the infinite Gaussian mixture model (Rasmussen, 1999) as a complicated design, so that the number of topics can be automatically determined by the data.

2 Text Data Cleaning

2.1 Introduction

Since the rise of the Internet, text datasets have been extensively studied, especially in recent years due to the information explosion. Researchers have investigated dynamic text networks, which grow and change with time. Examples include Wikipedia (Warren et al., 2008), blog links (Soriano et al., 2013; Henry et al., 2016), and citation networks (Ding, 2011; Chen, 1999). Nodes are articles; hyperlinks are edges. Since text data are ubiquitous, fields such as sociology and physics can also contribute to the network content. "Merrill Lynch recently estimated that more than 80% of all potentially useful business information is unstructured data." (Gharehchopogh and Khalifelu, 2011)

Creating network models based solely on observed "sufficient" statistics is not sufficient enough, and the need for discovering hidden links among participants (i.e. nodes) of the network is growing. A detailed survey of statistical network models (Goldenberg et al., 2010) reviews most graph modeling structures, such as the latent space model and random graphs. These techniques have been applied in topic modeling (Hu et al., 2014), network modular assignments (Hofman and Wiggins, 2008), and dynamic text networks (Henry et al., 2016).

However, most text analysis methods require adequate data preparation, and text data cleaning is an essential step (Zhang et al., 2003) because text is not as structured as numerical variables. Many papers discuss the text cleaning methodologies, including database integration (Rahm and Do, 2000; Johnson and Dasu, 2003), fuzzy matches for information (Chaudhuri et al., 2003), and even training for text classification (Esuli and Sebastiani, 2009). Removing non-text words from text data online, such as emails, is also an issue (Tang et al., 2005b). For example, an email may contain "news:00a201c2aab2$1215480$d5f82ecf" as part of the header; the random numbers and characters obviously do not form a word.

To divide text into different categories, the bag-of-words model is popular because of its simplicity (Zhang et al., 2010; Wallach, 2006), and each document can be represented by its unordered constituent words. Nevertheless, the bag-of-words model cannot distinguish word order and thus loses semantic information. For example, "Don't buy a PC – buy a Mac" and "Don't buy a Mac – buy a PC" contain the exact same words, but with completely different meanings (Soriano et al., 2013). As another example of lost semantic information, consider negation, where putting a “not” in front of a word can make it mean the opposite (Chai, 2014).

Furthermore, vocabulary size reduction is also important to expedite computation, due to the abundance of natural vocabulary. A variety of methods, such as the PCA (principal component analysis) and TF-IDF (term frequency - inverse document frequency) are available (Fodor, 2002; Tang et al., 2005a) for dimensionality reduction. No matter which method is used, the aim is to filter out words in the corpus of less interest, such as “the” and “for”.

2.1.1 Organization

In this chapter, I propose a text data cleaning methodology to preserve semantic meaning as well as to reduce the vocabulary size. To begin with, remove punctuation and non-words in the corpus. Second, remove words with little meaning or interest from the corpus. Third, stem and tokenize the remaining dataset. Next, replace any word followed by a “not” with its antonym. Then perform n-gramming to keep sets of words with high probability of consecutive co-occurrence.

Preliminary results of topic modeling are presented to show the quality of text data cleaning, and synonym mining is explored using latent semantic indexing, with mixed success. Related issues are also addressed, and different kinds of datasets require different filtering strategies.

2.2 Overview of Datasets

To demonstrate the methods, three datasets are used as examples. The first one is the Trayvon Martin dataset – a distillation of 467 US political blogs sites listed on Technorati[1]. The second one is the Benghazi dataset, whose format is similar to the first one. The third one is the employee satisfaction dataset, which contains 530 employee ratings and comments on their work.

2.2.1 Trayvon Martin Dataset

In this dataset, the articles are related to the fatal shooting of Trayvon Martin in February 2012[2], which triggered a heated debate on the media and many political blogs. Text scraping from blogs was performed by MaxPoint Interactive[3]. If a domain mentions "Trayvon" at least once in a document, then the blog site belongs to the Trayvon Martin corpus. A total of 450 blog posts used the keyword "trayvon", and the collection of these blog posts is the Trayvon Martin dataset.

Most bloggers use a standard protocol for dating their posts, but some use a variation, and some do not seem to have a consistent system. About 80% of the dates are accurate, but the remaining 20% have possible errors due to inconsistency. Therefore, the earliest date mentioned in the posted text was taken, and the dataset consists of posts with dates between January 1, 2012 and December 31, 2012 (Au, 2014). However, the Trayvon Martin incident happened in February 2012, so blog

[1]`http://technorati.com/`

[2]`https://en.wikipedia.org/wiki/Shooting_of_Trayvon_Martin`

[3]`http://maxpoint.com/us`

posts on January 2012 should not be included. In addition, using the latest date for each post is better because this includes related events which took place later, such as the trial of George Zimmerman (after the Trayvon Martin incident).

If I were to web-scrape the datasets, I would like to use the R package `rvest`, the Python library `scrapy`, or the Google Web Scraper.

2.2.2 Benghazi Dataset

This dataset was generated in the same way as the Trayvon Martin dataset, and the blog contents are related to the 2012 Benghazi attack[4], in which a US ambassador and three other Americans were killed by Islamist militants in Benghazi, Libya on September 11, 2012.

Figure 2.1 is a histogram showing how the frequency of blog entries related to Benghazi changed throughout 2012. Before the event, Benghazi was a term rarely mentioned by American bloggers, and those early mentions probably reflect errors in the dating of the posts. After September 11, 2012, suddenly lots of people were talking about it, and the trend gradually decreased. After a few weeks, when the US Congress announced an investigation of the Benghazi incident, more people talked about the Benghazi event again.

[4]`https://en.wikipedia.org/wiki/2012_Benghazi_attack`

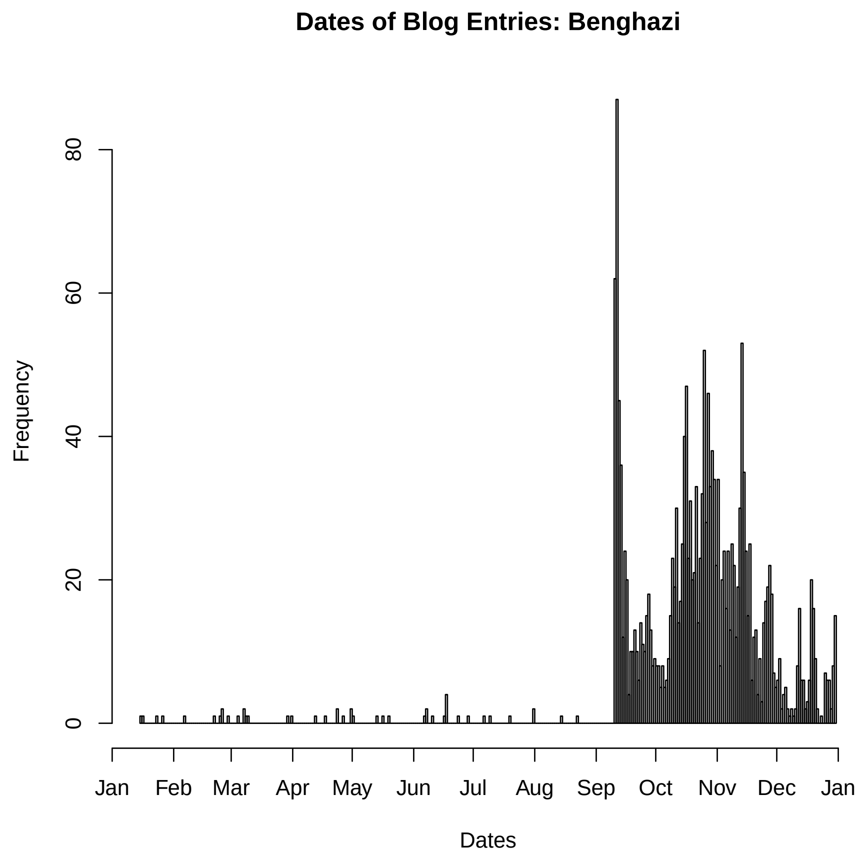


FIGURE 2.1: Histogram of the blog entries related to Benghazi in 2012

2.2.3 Employee Satisfaction Dataset

This dataset is provided by Nick Fisher at ValueMetrics, Australia[5]. In the data collection process, the employees were asked to rate their satisfaction with their company. The ratings range from 1-10, with 1 the least satisfied and 10 the most. Then a free response question was asked for the reason of their ratings. The occurrences for each rating are summarized in Table 2.1; most ratings are between 5 and 9.

[5]`http://www.valuemetrics.com.au/`

Table 2.1: Employee satisfaction dataset: Occurrences for each rating

Rating	1	2	3	4	5	6	7	8	9	10
Occurrence	2	3	14	32	52	57	127	147	72	24

An example of a comment with a high score is "I am very satisfied with the work that I do, and it gives me a sense of achievement." On the contrary, another example associated with a low score is "Minimum support in the work, Poor income." Note that one employee rated his/her company 1 but with the comment "Love my work - very varied", so it is obvious that the person was confused by the rating scale.

2.3 Vocabulary Size Reduction

The vocabulary size is defined as the union of all "words" (actually tokenized stems plus n-grams) in the dataset, and the size is proportional to the dimensionality. To reduce the vocabulary size, the first step is to remove punctuation and non-words in the corpus. Most terms consisting of three letters or less are usually seen as stop words and are not retained, except for common terms such as US and UK. Nevertheless, three-letter acronyms need to be retained and explicitly defined. These acronyms are frequently used, such as NSA (National Security Agency) and DOD (Department of Defense).

Before the stemming and tokenization process, all stop words with little semantic meaning are also removed, such as "for" and "and", but the negation word "not" is an exception. A list of about 300 stop words (including the not-so-obvious ones, e.g. "therefore", "toward") was created for implementation, and the R program removed all words on the list. This method is applied to the Trayvon Martin dataset, and the corpus size is reduced by approximately 12%. On the other hand, the employee satisfaction dataset is a relatively small corpus, so only obvious stop words – “a”, “the”, “in”, “be”, “on”, “my”, “very”,“so”,“been” are removed.

For the stop word list, I considered using `stopwords(‘‘english’’)` from the R package `tm` but eventually decided not to do so. This public word list contains 174 stop words, but too many contractions exist, such as “it’s” and “you’re”. Since punctuation is removed first, these contractions have already disappeared from the corpus, so `stopwords(‘‘english’’)` does not contain enough stop words to filter out most of the “fluff”.

2.3.1 Stemming and Tokenization

To make a vocabulary dataset for the corpus words, my research group implemented stemming and tokenization. Stemming is to remove the suffixes of words, such as “ly” and “ed”. A “token” is the base or root of a word; for example, “carry” and its past tense “carried” are given the same token. After the process, the words may look like typos, e.g. “beauti” means “beauty” or “beautiful”, and “surviv” means “ survive” or “survival”. Some phrases are funny after stemming; for example, “unit state” actually means “United States”.

In the Trayvon Martin dataset, software at MaxPoint is used to tokenize the words, but some issues exist. The corpus and the vocabulary list I received are already tokenized, and some stop words had been removed by MaxPoint. However, I still found more stop words and had to remove them according to the tokenized stop word list, but the point is that the "corpus" was partially cleaned up, and it does not reflect the raw web scraping results.

The employee satisfaction dataset comes directly from text responses in the questionnaire, so it resembles a real-time project. After removing a few stop words, I tokenized the corpus using the function `wordStem` from the R package `SnowballC`. The vocabulary list and counts are saved in an `.RData` file, so the results can be reused later.

2.3.2 Using Public Domain E-Books

I also considered using the e-books in the public domain (Project Gutenberg[6]) as a general stop word list. The reason is that many words or phrases with limited semantic meaning are not considered as stop words, and examples include "years" and "New York". Children's instructional books, such as "Little Busybodies: The Life of Crickets, Ants, Bees, Beetles, and Other Busybodies", are good candidates for the stop word list because the content is general knowledge. On the other hand, some ancient books, such as "Gone with the Wind" and other classics, are not good options for the stop word list because they contain obsolete words such as "shan't" (now people say "shouldn't").

A possible drawback of the e-book method is that many colloquial or easy words will also be removed, even if they have significant semantic meaning. The Trayvon

[6]`https://www.gutenberg.org/`

Martin dataset consists of many basic English words because the political blog posts were written by the general public, so if I remove the words present in a public domain e-book, most of the words in the dataset will be gone, except for a few professional terms. Therefore, it is not a good idea to remove too many potential stop words, otherwise simple words with semantic meaning such as “yell back walk away” will disappear, leaving little content in the political blog dataset. As a result, I decided to remove pre-defined stop words from the Trayvon Martin dataset.

However, this method works better for paper abstract datasets, such as those for presentations at the Joint Statistical Meetings. By comparing the dataset with a public domain e-book, I can retain statistical terms (e.g. “maximum likelihood estimation”, “standard deviation”) because these phrases do not appear in books written for the general public. The cleaned corpus can used for topic modeling to perform session scheduling optimization (Bi, 2016). Sessions of the same topic should be scheduled at different time periods to minimize the overlap, so conference participants can attend more talks in their field of interest.

2.3.3 TF-IDF Scores

Simply removing stop words from a list has limitations; an alternative solution is to look at the TF-IDF (term frequency - inverse document frequency) scores, which is a popular method to filter out stop words. In mathematical terms, the TF-IDF score is $f_{ij} * \log[n/(d_j + 1)]$, where f_{ij} is the term frequency for the ith word in the jth document; d_j is the document frequency, and n is the number of documents in the corpus.

In general, the TF-IDF score balances words being used frequently against the number of documents in which they appear. Stop words such as “the” and “an” appear for many times in almost every document, so they are not distinctive in determining the topic(s) of any document. Term frequency is how many times a specific word appears in each document, so if the word is important or regularly used, the TF is high. On the other hand, inverse document frequency is measured by how many documents the word appears in, so if the word is used in too many documents, IDF weighs the TF-IDF score down.

The R package `textir` is available for computing the TF-IDF scores. The function `tfidf` returns a matrix with the same dimensions as the dataset, but the values are replaced with the TF-IDF scores. Then the function `sdev` returns the column standard deviations from the TF-IDF matrix generated by `tfidf`, obtaining the TF-IDF score for each word in the corpus.

However, TF-IDF scores are inappropriate for cleaning up the Trayvon Martin dataset, because it is difficult to distinguish key words and unimportant words – when they both have semantic meaning and appear across the documents. Since the dataset is about a shooting incident, the victim Trayvon Martin is mentioned by almost every blogger, but the name is a key word. From the standpoint of topic identification, “Trayvon Martin” is not helpful and should be removed. In addition, some words (e.g. “president” or “obama”) have semantic meaning, but they are mentioned in almost all blog posts, so they should also be removed.

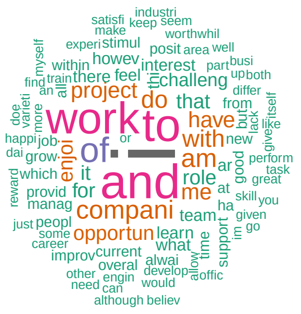
I applied the TF-IDF method to the Benghazi dataset, and the performance was also problematic. Most values in the `tfidf` matrix have low standard deviations, and the histogram of these values is right-skewed. The token "benghazi" has the lowest standard deviation (or TF-IDF score, to be exact), which is 0.00164, but it is an important key word to learn the content of the blog posts. On the other hand, the tokens with high scores (> 0.05) look like random words, such as "croatia", "snipe", and "hammar".

Therefore, I abandoned the TF-IDF score method, except that the structure of the TF-IDF matrix is applied to the employee satisfaction dataset, but it is used for constructing the term frequency matrix for topic modeling.

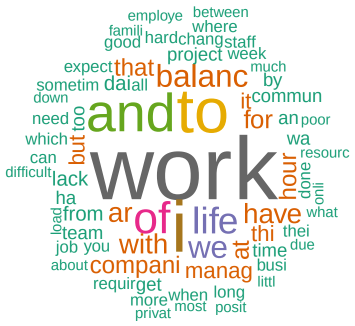
2.3.4 Preliminary Results and Discussion

The word clouds generated from the employee satisfaction dataset is a good demonstration for the difference before and after filtering out stop words. Figure 2.2 shows the topic modeling results by using Latent Dirichlet Allocation (LDA), and the two topics (named as positive and negative words) both show less noise in the word clouds after the filtering. In the bottom row, the key tokens such as "challeng", "opportun", and "balanc" are easier to see than in the upper row. More about topic modeling will be described in Chapter 3.

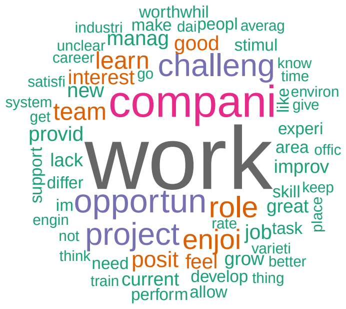
Removing stop words permanently from the corpus is often safe (Yang and Wilbur, 1996), but special attention is required because words filtered out will not come back to the analysis. In some datasets, important semantic meaning can be lost by doing this. For instance, the title “Gone with the Wind” is likely to be present in a movie-related dataset. If “with” and “the” are removed, the title is unrecognizable. In this case, keeping phrases of special meaning (known as “n-gramming”, described in Section 2.5) needs to be done before the stop words removal process.



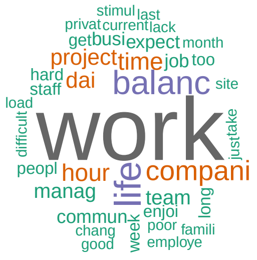
Before – positive words



Before – negative words



After – positive words



After – negative words

FIGURE 2.2: Employee satisfaction dataset: Before and after removing stop words

2.4 Negation Handling

Negation handling is needed for partial recovery of semantic information. Recall the example in Section 2.1 – "Don't buy a PC – buy a Mac" and "Don't buy a Mac – buy a PC" (Soriano et al., 2013). For a simpler example, "not a good idea" means "a bad idea" because "not good" has the same meaning as "bad. Therefore, phrases starting with "not" or its equivalents (e.g. "don't" or "wasn't") should be replaced with the antonym of the second word.

Another reason for handling negation is that words can be rearranged in different orders in a permutation test, with the null hypothesis that all words are randomly distributed. This is good for n-gramming (described in Section 2.5), but a word starting with a negation usually needs to be defined as a token, or putting the negation in front of other words will change the semantic meaning. For instance, if "not a good idea" is changed to "a good not idea", then the object can be something else instead of an idea.

Moreover, I address negation handling before n-gramming because some topic models may not support multi-word expressions (n-grams), so keeping everything at the word level is a constraint worth mentioning in semantic information recovery.

2.4.1 Using a Thesaurus Database

The database (DataStellar Co., Ltd., 2011) serves as the thesaurus for antonyms, and it contains 113,690 English words. However, a complication is that the vocabulary used in the corpus are stemmed tokens, but the word candidates in the thesaurus database are real words. The process of transferring words into tokens is tokenization, and the process of inverting tokens back to words, i.e. inverse tokenization, is needed. The inverse tokenization is not necessarily a function because it maps one token to multiple words, but the pre-image is helpful in identifying root words of tokens. Figure 2.3 is a schematic diagram of the two processes.

$$\begin{tikzcd}
\text{words} \arrow[r, bend left, "f"] & \text{tokens} \arrow[l, bend left, "f^{-1}"]
\end{tikzcd}$$

$$f = \text{tokenization}$$

FIGURE 2.3: Tokenization and inverse tokenization

Word-Token Mapping Method

One solution is to use the word-token mapping method; that is, convert the vocabulary tokens into words, replace the words with their antonyms, and tokenize the replaced words back to tokens. Since the inverse tokenization process is not actually a function, I wrote pseudo tokenization and inverse functions to make the results similar to the real word-token pairs – by changing the suffixes of words/tokens. Using the R package `stringr`, the functions are described as below:

- Pseudo tokenization: Remove "e" at the end of the word, or substitute "y" for "i" at the end.
- Pseudo inverse tokenization: Add "e" at the end of the word, or substitute "i" for "y" at the end.

- Examples of word-token pairs: “believe” $\leftrightarrow$ “believ”, “agree” $\leftrightarrow$ “agre”, “really” $\leftrightarrow$ “realli”, “worry” $\leftrightarrow$ “worri”

This solution is intuitive, but it is not ideal for two reasons. First, constantly switching between words and tokens causes much confusion. What is worse, this method requires three steps, which increases computational difficulty in the implementation.

Thesaurus Database Tokenization

A better solution is to tokenize the antonym database using the R package `SnowballC`. The advantage is that I only need to deal with tokens throughout the text data cleaning, which makes analysis easier. The drawback is that many words remain unchanged after tokenization, such as “mountain” and “old”; since the database is large, it takes a long time to convert all words to tokens. Nevertheless, the pros outweigh the cons because the database tokenization only needs to be done once.

2.4.2 Antonym Replacement

In the corpus, words (tokens, to be exact) starting with negation should be replaced with their antonyms. For instance, “not sure” is replaced with “uncertain”, and “dont know” is replaced with “ignorant”. Negations terms consist of these tokens: **“no”, “not”, “isnt”, “arent”, “wasnt”, “werent”, “dont”, “doesnt”, and “didnt”**. Note that the negation terms should only reverse the meaning of the word after it, so tokens with negation as well as other semantic meanings are not regarded as negation terms; examples are “wont”, “cant”, “hardli”, and “rare” (rarely).

One limitation exist in this approach – the word that should be negated is not always the word immediately following the negation. For example, “not even honest” means “not honest”, instead of “not even” followed by “honest”. Then “not even” can be replaced with a different adjective (more to be discussed in Section 2.4.3). Since “even” is not a stop word, it will not be removed during the vocabulary size reduction phase (Section 2.3).

In general, the antonym rule is helpful although imperfect. Therefore, no automated process exist for a single optimal choice on everything.

Information from the Corpus

Tokens that appear in the corpus many times are usually of interest and high priority. To investigate in the word/phrase frequency of the corpus, I generated frequency lists of single words and two-word phrases, respectively. The word frequency list is used to learn which words are likely to be keywords, and the distribution is extremely right-skewed. For example, the Trayvon Martin dataset has 2,536 tokens, but only 616 of them (24.3%) appear 100 or more times, and 21 (0.82%) of the tokens have 1,000 or higher count.

In the frequency list of two-word phrases, the ones starting with negation terms are retrieved for the purpose of antonyms replacement, but the short list does not always reflect which word is commonly used in the opposite meaning. The reason is due to the frequency calculation of phrases – two phrases are considered the same if and only if they consist of the exact two words and in the exactly same order. As a result, two phrases such as “not know” and “don’t know” have the same meaning, but they are identified to be different phrases.

Therefore, phrases starting with negation terms are combined when the second word (denoted as “not-word”) are the same. For example, if “not know” appears 10 times in the corpus, and “dont know” appears 15 times, the two phrases can both be counted as the not-word “know”, so the frequency count of “know” used in the opposite meaning should add up to 25. In this way, both “not know” and “dont know” will be replaced with “ignorant”.

Methodology

After finding all words preceded by a negation term (i.e. not-words), the antonym replacement process starts by replacing the not-words with their antonyms in the thesaurus database. If more than one antonym candidate is available, one of them is selected randomly. Since the antonym words are in alphabetical order, I should not always select the first.

To expedite the antonym word search, the thesaurus database column of initial letters from A to Z is exploited. I compared the initial letter of the not-word with the column, so the sequential search is only performed on rows with words of the same initial letter as the not-word. In this way, the sample space size is significantly reduced, making the search of antonyms much faster.

Since the Trayvon Martin dataset is large, I used corpus information to determine which antonym to substitute for words preceded by a negation. Instead of deciding on an antonym from the thesaurus completely at random, the candidate with maximum appearance in the corpus is chosen, and then ties are broken by random selection. This is likely because sometimes none of the antonym candidates appear in the corpus. The most common appearance is determined by the word frequency list generated beforehand.

The employee satisfaction dataset is relatively small, so the simplest way is to manually replace all 145 not-words in the corpus. A computational trick worth mentioning is to replace all negation terms (defined in Section 2.4.2) with “not”, so the “not know” vs “dont know” problem will not happen again. I decided not to use the thesaurus database because randomly selecting one antonym candidate leads to instability; that is, the antonym replacement results depends on which random seed is chosen.

2.4.3 Results

The antonym replacement methods are demonstrated by using the Trayvon Martin and the employee satisfaction datasets. The results generally work well for high-frequency words, such as replacing “not agree” with “differ” and replacing “not give” with “take”; further discussion and current limitations will be addressed in Section 2.4.4.

Trayvon Martin Dataset

In this dataset, the antonym replacement list for the top 100 not-words is inspected. The frequency of not-words is right-skewed, 64% of words occur less than 15 times, while the most used ones are “know” (235 times preceded by a negation) and “think” (161 times). A total of 61 not-words are replaced with their antonyms, and the remaining 39 stay in the corpus.

Many words preceded by a negation are replaced with their antonyms when retaining most of the semantic information, but the polysemy issue can cause problems; i.e., one word with more than one meaning. The word “know” has an obvious antonym “ignorant”, but for the word “even”, the official antonym is “uneven”, which is not the colloquial usage. The word “uneven” means unfair or unbalanced, but the word “even” is often used for emphasis, e.g. “Many people do not even know who controls the Congress.” The word “even” can also be used to indicate a number divisible by 2, with the expression “an even number”, which is more likely to occur in a mathematical corpus.

Furthermore, the antonym replacement results depends on the corpus itself because the candidate selected is the one with maximum number of occurrences in the dataset. For example, the phrase “not good” should be replaced with “bad”, but unfortunately the antonym replacement process gives “black” – because “black” appears 1384 times in the Trayvon Martin corpus, while “bad” appears only 229 times. This is not only wrong but also politically incorrect, and the actual context in Table 2.2 also shows that “bad” is a better substitute for “not good”. Therefore, I manually set the code to replace “not good” with “bad”.

Table 2.2: Actual context with the phrase “not good” (or its equivalent) in the Trayvon Martin dataset

“attorney gone flip issu stand ground not good form public relat point”
“not good argument compar usual standard”
“final report probabl make rate jump not good”
“probabl wont good debat money issu directli”

Employee Satisfaction Dataset

To exploit the questionnaire structure, I also summarized the number of negation terms (“nots”) per comment in Table 2.3, and discovered that higher frequency of “nots” is generally associated with lower ratings. Using the number of “nots” per comment or the percentage of comments with at least one “not” shows the same trend in the dataset. Therefore, it appears that people tend to use more negation terms when expressing negative feelings.

Since the employee satisfaction dataset is small, only approximately 150 phrases starting with a negation term are discovered, so I manually replaced them with corresponding antonyms. I selected which antonym to replace which word by hand, but I wrote code when actually putting the antonyms into the corpus. The revised corpus is used for topic modeling applications (further described in Chapter 4).

2.4.4 Discussion and Current Limitations

Antonym replacement is helpful in semantic information recovery, but finding the correct antonym to replace a phrase starting with a negation is challenging because

Table 2.3: Employee satisfaction dataset: Number of negation terms ("nots")

Rating	1	2	3	4	5	6	7	8	9	10
Occurrence	2	3	14	32	52	57	127	147	72	24
# of "nots"	0	0	9	13	34	39	43	20	6	1
# of "nots" per comment	0	0	0.64	0.41	0.65	0.68	0.34	0.14	0.08	0.04
# of comments with a "not"	0	0	6	11	25	26	36	17	6	1
% of comments with a "not"	0	0	3%	34%	48%	46%	28%	12%	8%	4%

of the polysemy issue. One word can have multiple meanings, and which meaning is used depends on the context, so a one-to-one mapping among English words does not exist. Therefore, the antonym selection process does not always give the antonym that best represents the opposite meaning of the not-word. For example, the phrase "not right" can mean going left or doing the wrong thing, and more context is needed to determine which meaning it has.

For the Trayvon Martin dataset, the punctuation removal process during vocabulary size reduction may need to be revised for two reasons. First, some punctuation also contains semantic information. For example, a panda which "eats shoots and leaves" is much friendlier than one which "eats**,** shoots and leaves" (Truss, 2004). Furthermore, quotation marks can make a word mean something different. "Not" with quotes should stay in the corpus, because this kind of "not" may not actually indicate negation. The "nots" without quotes are the ones that should be replaced when followed by another word with semantic meaning.

2.5 N-gramming

The goal of n-gramming is to identify special phrases; in other words, keep phrases with high probability of occurrence. These phrases are regarded as tokens, and are widely used in text data analysis (Marino et al., 2006; Cavnar et al., 1994). N-gramming also contributes to semantic information recovery because words within a preserved special phrase will not be rearranged in a bag-of-words model. For example, the two sentences "The white person lives in the house." and "The person lives in the White House." contain the exact same words but have completely different meaning. The first sentence refers to an ordinary white person living in an ordinary house, but the person mentioned in the second sentence must be related to the US president.

An n-gram is a phrase consisting of n words that should not be separated, such as "new york" and "white house". In particular, when this kind of phrase consists of only one word, it is a uni-gram; two words combined in this way is a bi-gram; three words tied together is a tri-gram.

Searching for n-gram candidates (i.e. phrases consisting of n words) requires a sequential and extensive search in the corpus. If a sentence comprises n words, then there are n uni-grams, $n-1$ bi-gram candidates, and $n-2$ tri-gram candidates within the text. This process is known as "shingling" at the word level (Steorts et al., 2014). For example, "What everyone should know about Trayvon Martin" contains 7 words, 6 two-word phrases, and 5 three-word phrases, listed as below:

- Text: “What everyone should know about Trayvon Martin”
- Two-word phrases: “what everyone”, “everyone should”, “should know”, “know about”, “about trayvon”, and “trayvon martin”
- Three-word phrases: “what everyone should”, “everyone should know”, “should know about”, “about trayvon martin”

Most phrases are simply words that happen to be together and have no special meaning when combined, so the phrases that cannot be separated (a.k.a. n-grams) need to be identified statistically.

2.5.1 Conditional Probability

To identify n-grams, conditional probability is often used (Chater and Manning, 2006; Brown et al., 1992) in natural language processing, and one of the most prominent works is the Turbo Topics software (Blei and Lafferty, 2009) in Python.

The Turbo Topics approach (Blei and Lafferty, 2009) selects n-gram candidates while accounting for other n-grams, using likelihood ratio tests and recursive permutation tests, with the null hypothesis that all words are independent of each other. For example, assume “new” appears for 10,000 times in the corpus, while “york” follows it 6,000 times, and “jersey” 3,000 times. “new york” occurring 6,000 out of 10,000 is obviously a strong signal, which can obscure other significant but not-as-strong ones. When calculating the strength of signals, the 3,000 times of “new jersey” occurrence should be considered out of the remaining 4,000 “non-york” signals, rather than the full 10,000 times because "new york" is already present in the model.

In mathematical terms, assume m words are in the corpus and an n-word phrase is denoted as $k_1k_2\cdots k_n$. The baseline and conditional probabilties are defined as follows:

- Baseline probability: $P(k_a) = \dfrac{\#\text{ of } k_a}{m}$, where $a = 1, 2, \cdots n$
- Conditional probability:
  $$P(k_n|k_1k_2\cdots k_{n-1}) = \frac{P(k_1k_2\cdots k_n)}{P(k_1k_2\cdots k_{n-1})} = \frac{\#\text{ of } k_1k_2\cdots k_n}{\#\text{ of } k_1k_2\cdots k_{n-1}}$$

The baseline probability is directly related to the raw frequency of each word, and the conditional probability means the probability of observing the nth word given the previous $n-1$ words.

2.5.2 Divide by Max

The definition of conditional probability for words in Section 2.5.1 is generally useful, but the "last-word-guessing" problem exist in many cases – if the first two words are known, the third word is obvious. For example, given the first two words "osama bin", the last word should be "laden" because "osama bin laden" was known as a leader of the September 11 attack in the US. For another example, the term "new panther parti" can be viewed as "parti" followed by "new panther" or "panther parti" followed by "new". In the former case, the conditional probability is close to 1, but in the latter one, the conditional probability is low because many words can go after "new", such as "york" or "jersey".

Therefore, I propose the probability calculation method "divide-by-max" by changing the denominator of conditional probability to the maximum frequency of the words involved. In this way, the new definition can be written as

$$P(k_n|k_1k_2\cdots k_{n-1}) = \frac{\#\text{ of } k_1k_2\cdots k_n}{\max\ (\#\text{ of } k_1,\ \#\text{ of } k_2,\ \cdots,\ \#\text{ of } k_n)} \tag{2.1}$$

The occurrence of the n-word phrase is at most the minimum of the word frequency, so the divide-by-max probability is less than or equal to the original conditional probability.

2.5.3 Comparison of Probability Schemes

Figure 2.4, consisting of 2x2 plots, compares the histograms of probabilities for conditional probability (abbreviated to "Cond Prob" in the graph titles) and divide-by-max, using the Trayvon Martin dataset. The first row uses the 425 bi-grams, each of which appears in the corpus 20+ times (i.e. minimum count is 20), and the second row uses the 90 tri-grams with the same requirement.

The bi-grams and tri-grams show different trends in the two methods. In the first row, both distributions of the bi-gram probabilities are right-skewed; more than half of the bi-grams have probability less than 0.1. Interestingly, the trends of the tri-grams' conditional probability and divide-by-max are the opposite as shown in the second row; the one for conditional probability is left-skewed, and the one for divide by max is right-skewed. A left-skewed histogram suggests that most probabilities are close to 1.

Since the goal of the probability methods is to identify the words which have a higher probability to appear together, right-skewed distributions are more reasonable because there are few n-grams in reality, compared with the number of English words. Therefore, the conditional probability approach is inconsistent, due to the "last-word-guessing" problem addressed in Section 2.5.2. On the other hand, the plots of divide-by-max are more reasonable and consistent, because the probabilities of each word group are deflated by the most frequent word in the group.

2.5.4 Simple is Best

For the Trayvon Martin dataset, it turned out that simple is best – good results can be obtained by setting the minumum count of two-word and three-word phrases to be 20 times. Using this rule, less than 5% of phrase combinations will remain in the n-gram selection list. Examples of the top bi-grams are "trayvon martin", "georg zimmerman", and "neighborhood watch"; examples of the top tri-grams include "new panther parti", "stand ground law", and "second degre murder".

The CDF plots of bi-grams and tri-grams are shown in Figure 2.5, and the histograms are too right-skewed to be visualized. In fact, among all 229,245 three-word phrases in the corpus, 215,208 (93.88%) only appear once, which includes many non-useful phrases such as "care reform cost" and "realli bad question".

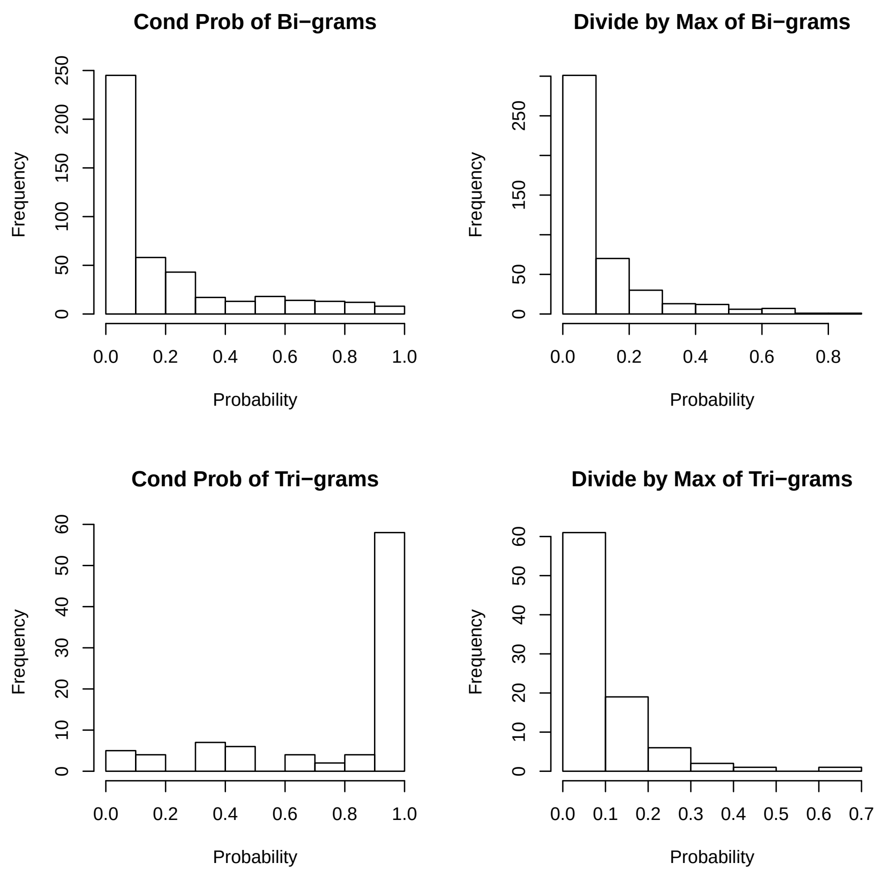


FIGURE 2.4: Histograms of probabilities for bi-grams and tri-grams

2.5.5 Discussion and Known Issues

N-gramming significantly contributes to semantic information recovery because this method retains multi-word expressions such as “trayvon martin” and “georg zimmerman”, and the divide-by-max method of calculating conditional probability also removes a lot of unwanted phrases.

To further improve the quality of retained information, the problem of signal split

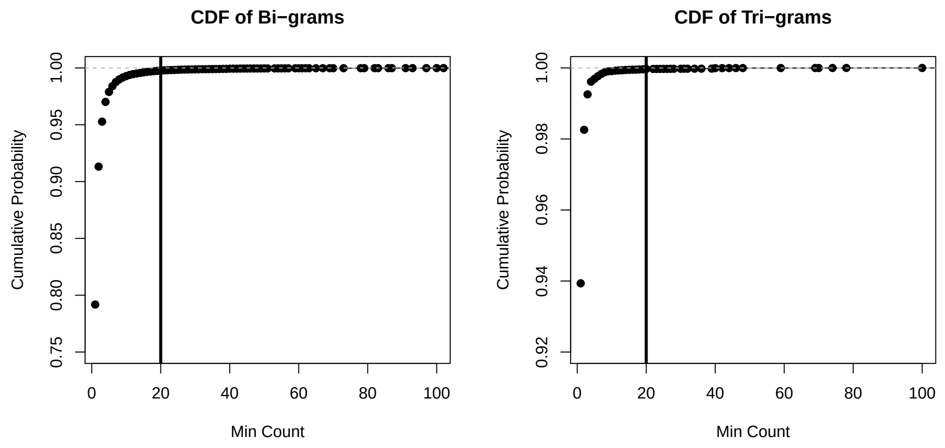


FIGURE 2.5: The CDF of bi-grams and tri-grams

needs to be resolved. One reason for signal split is misspelling because typos exist everywhere and many blog sites do not perform spell checking. Another reason is that a few terms need to be combined. In the Benghazi dataset, the terms “obama”, “presid obama”, “barack obama”, and “presid barack obama” all refer to the same person. For another example, “hillari clinton”, “secretari state hillari clinton”, ”secretari clinton”, and “secretari state” also mean the same person, but it is difficult to determine whether “clinton” indicates “hillari clinton” or “bill clinton” without the help of nearby words in the corpus. This leads to the synonym mining problem, which is addressed in Section 2.6.

Other possible improvements also exist. Modeling skip-grams makes n-grams more flexible because this approach allows some words to be in between the words comprising the n-gram (Guthrie et al., 2006). Removing stop words after the n-gramming can also improve the text data quality, as mentioned in Section 2.3.4 –

some key phrases contain stop words (such as the movie title “Gone with the Wind” and the statistics term “converge in probability”), so the phrases should be n-grams. Note that the longer stop words are kept in the corpus, the more expensive is the computation process.

2.6 Synonym Mining

Synonym discovery is one of the most difficult fields in text mining (Berry and Castellanos, 2004). Most researchers simply generate synonyms of words from text data (Meyer et al., 2008), which results in something similar to the thesaurus database such as the R package `wordnet`, instead of showing interesting results from the corpus. For example, the words “car” and “limousine” (luxury car) are synonyms; however, in terms of semantic meaning, “car” includes “limousine” but not the opposite. In text data cleaning, I also explored synonym mining in the corpus using latent semantic indexing, with mixed success.

2.6.1 Methodology

The idea of latent semantic indexing (Deerwester et al., 1990) is that two words used in similar documents can share the same meaning; for example, green and emerald both mean the same color. Therefore, a latent space is created to store the information of which terms appear in which documents. Every word (token, to be exact) is defined as a binary vector, with at least a “1” present in each vector (i.e. each word appears in at least one document in the corpus), and rare words usually have a short vector Euclidean length (close to 1). Next, the reduced-rank singular value decomposition (SVD) (Berry et al., 1995) and principal component analysis (PCA) are performed to reduce the dimensionality of the latent space; the goal is to retain the high-frequency and meaningful words in the documents.

The similarity between two words is defined by the cosine function:

$$\cos(\text{“word1”}, \text{“word2”}) = \frac{\overrightarrow{\text{“word1”}} \blacksquare \overrightarrow{\text{“word2”}}}{|\overrightarrow{\text{“word1”}}||\overrightarrow{\text{“word2”}}|} \tag{2.2}$$

Note that $\cos(\text{“word1”}, \text{“word2”}) \approx 1$ implies the two words are similar.

The method of latent semantic analysis is implemented in R using the package `lsa` and the Trayvon Martin dataset (the first 1,000 documents only). To reduce the dimensionality, singular value decomposition (SVD) and principal component analysis (PCA) are used. Since the histogram of the 1,000 singular values is right-skewed, I selected the singular values such that the sum of selection equals to half of the sum of all singular values using the R function `dimcalc_share`. Even though large singular values are rare, 192 elements are still needed to sum up to half of the total.

In mathematical terms, the new latent space can be defined as $M = TSD$, and the reduced-rank version with k dimensions is $M_k = T_k S_k D_k$ (Wild, 2015). The original dataset contains 22,278 words and 1,000 documents, so each component in SVD has dimensionality as below:

- $M, M_k : 22278 \times 1000$
- $T_k : 22278 \times 192$
- $S_k : 192 \times 192$

- D_k : 192 × 1000

2.6.2 Results and Discussion

In terms of synonym discovery, the latent semantic indexing (LSI) technique results in mixed success – sometimes it works and sometimes it does not. For example, the cosine similarity between “obama” and “barack” is as high as 0.733, so LSI correctly identifies the two words to have similar meaning. However, the similarity between “obama” and “bipartisanship” is 0.726; the two words are related but do not mean the same thing. What is worse, the words with closest distance to “abandon” are “learnt” (0.775) and “austrian” (0.770), and this is unreasonable. The top five words with highest cosine similarity to the words “obama” and “abandon” are listed in Tables 2.4 and 2.5, respectively.

Table 2.4: Top five words with highest cosine similarity to “obama”

“obama”	“barack”	“presid”	“bipartisanship”	“lofti”	“excurs’
cosine similarity	0.733	0.726	0.719	0.714	0.710

Table 2.5: Top five words with highest cosine similarity to “abandon”

“abandon”	“learnt”	“austrian”	“therebi”	“burglari”	“robert”
cosine distance	0.775	0.770	0.709	0.702	0.693

The main reason why latent semantic indexing does not work well is that rare words result in short vector lengths and high cosine values, according to the definition of similarity in Equation 2.2. If both words are rare but happen to appear in one or two documents at the same time, it can still result in a high cosine value for the pair. The two words can be completely unrelated, such as “abandon” and “austrian”.

For another reason, latent semantic indexing has the bag-of-words model problem because each text document is represented by an unordered collection of words.. Revisit the example in Section 2.1 – “Don’t buy a PC – buy a Mac” and “Don’t buy a Mac – buy a PC”. Latent semantic indexing regards “PC” and “Mac” as synonyms because they are used in the exact same set of documents; the result is undesirable.

2.7 Further Extensions

This research can be extended to identify what text data cleaning methods are appropriate for which types of datasets. Since different filtering strategies best apply to different kinds of datasets, there is no uniformly most appropriate text data cleaning method. The investigation will involve multiple types of datasets, such as blog text, questionnaire data, Twitter, and even journal databases.

In addition, the order of text cleaning steps should also be investigated to determine which best suits the dataset. It is standard practice to remove stop words prior to n-gramming the dataset, but n-grams containing stop words will also be removed, as described in Section 2.3.4.

A solution is to reverse the steps, i.e. n-gram the dataset before removing stop words. For example, (Bi, 2016) uses the JSM (Joint Statistical Meetings) 2015 abstract dataset:

1. N-gram the JSM 2015 dataset

2. Remove the words in the dataset present in a Project Gutenberg e-book

Further details of the second step are described in Section 2.3.

When n-gramming is done first, statistics phrases are preserved, such as “maximum likelihood estimation” and “converge in probability”. This demonstrates the importance of n-gramming. On the contrary, if the two steps are reversed, almost everything is removed, except for statistics jargon such as “Bayesian”, “posterior”, and “likelihood”. Then some statistics words such as “expectation” and “impossible” are removed because they are also used in the realm of general discourse.

2.8 Conclusion

Text data cleaning is an important aspect of data mining, and it needs to be done before implementing most analysis methods. In general, the key words are the ones that appear in the document for 20+ times, excluding stop words, and I note that the definition of stop words varies from corpus to corpus. Replacing words followed by negation with their antonyms and n-gramming (“binding” a few words together) mitigate the loss of semantic information due to bag-of-words models, which ignores word order.

3
Quantifying Improvement of Topic Modeling Results by N-Gramming

3.1 Introduction

The importance of n-gramming in topic modeling seems obvious, but the improvement needs to be quantified because “people don’t ask how; they ask how much.” N-gramming is the process of combining multiple words into a token to preserve semantic information, so the words will not be separated. For example, “New York” is a city in the US, but “new” is an adjective and “York” is another city in England. These phrases are called n-grams.

The n-gramming step in text data cleaning is ubiquitous because it retains word order for semantic information, and many applications of n-gramming in natural language processing exist, such as text categorization (Cavnar et al., 1994), machine translation (Marino et al., 2006), and even music retrieval (Doraisamy and Rüger, 2003). Some literature also takes a further step to address the conditional probability (Brown et al., 1992) or generalize the n-gram model (Guthrie et al., 2006).

However, few previous studies (Wallach, 2006) actually evaluate the information gain from n-gramming. Therefore, the problem is defined as performance quality comparison of topic modeling results on different versions of the dataset, so the contribution of n-gramming can be quantified. The approach I take to solve the problem is latent Dirichlet allocation (LDA), and the parameters to be estimated are the topics and weights of topics for each word in each document. Improvement of topic modeling results can be measured at the corpus level and/or the word level.

The dataset, consisting of 450 blog posts, is a political blog corpus related to the Trayvon Martin shooting incident in 2012[1], and it will be referred as the "Trayvon Martin dataset". When a blog post talks about the victim Trayvon Martin, sometimes only the word "Trayvon" or "Martin" is used, but they all refer to the same person. Since most topic modeling methods in text mining use bag-of-words models (Zhang et al., 2010), all words in a corpus are rearranged for a permutation test, but some words, such as "Trayvon Martin" or "New York" have special meaning. Note that separate words do not retain that meaning (Blei and Lafferty, 2009).

[1]`https://en.wikipedia.org/wiki/Shooting_of_Trayvon_Martin`

3.1.1 Organization

In this chapter, n-gramming and topic modeling are addressed in the methods section. Next, the implementation consists of posterior probability and word distinctivity results generated by LDA. After the initial setup, I investigated in the improvement of topic modeling results after n-gramming by examining the top 10 words for each topic, and discovered the contribution of n-gramming is the increase in word distinctivity. In a previous conference talk, I emphasized "When two words form a bi-gram with special meaning, the maximum posterior probability of being in a certain topic given the bi-gram is higher than the probability for each word separately." (Chai, 2016)

3.2 Methods

The data cleaning process for the Trayvon Martin dataset (details described in Chapter 2) mainly consists of removing stop words and n-gramming. In the beginning, pre-defined stop words (approximately 300) with little semantic meaning (e.g. "for", "and") are removed from the tokenized corpus, and this is the first version of the dataset, i.e. without n-gramming. After the n-gramming step is performed, the second version of the dataset is created.

For topic modeling, latent Dirichlet allocation (LDA) is performed on both versions of the Trayvon Martin dataset, one with n-gramming and one without. The goal of topic modeling is to statistically classify text documents into different topics, so the relationship between documents can be discovered. LDA is selected for text data analysis because it is a standard approach (Blei et al., 2003) and has been widely applied in topic modeling, such as in (Bi, 2016; Blei, 2012; Wei and Croft, 2006). However, the bag-of-words model assumption in LDA clearly suggests the need for n-gramming.

3.2.1 N-Gramming

N-gramming is performed on the Trayvon Martin dataset in R using functions from an existing GitHub repository (Henry, 2016). The function `textToBigrams` generates bi-grams from the corpus, i.e., the n-grams with only two words. The default cutoff frequency is 100, so any two-word phrase has to appear in the corpus at least 100 times to be considered a potential bi-gram. In addition, the p-value calculation is based on the probability $P(\text{word } 2 \mid \text{word } 1)$, given the null hypothesis that words are independent. The required level of significance (p-value cutoff value) is 0.05.

Since cutoff thresholds exist for bi-grams, the vocabulary set induced by n-gramming has limited size. The Trayvon Martin dataset consists of 2428 unique words (tokens, to be exact), and the number of generated bi-grams is 22, which are:

```
barack_obama, black_panther, civil_right, comment_dave
dave_surl, dont_know, dont_think, fox_news
georg_zimmerman, gregori_william, look_like, mitt_romney
neighborhood_watch, new_york, presid_obama, right_wing
self_defens, stand_ground, trayvon_martin, unit_state
white_hous, year_old
```

The bi-grams can be divided into three categories: people names, special phrases, and common expressions. The second category, special phrases, are of the most interest because the semantic information increase is highest when the two words are regarded as a single token.

Reasoning

The minimum number of occurrences and p-value cutoffs are both indispensable. First, the occurrence cutoff removes rare words with little semantic meaning. In addition, the cutoff for phrase counts needs to scale according to corpus size. If the corpus is as large as Google Scholar, many typos can appear 100+ times but should still be considered rare. If the corpus is small (e.g. our employee satisfaction dataset, with only 530 ratings), setting the minimum frequency to 10 can be enough.

Note that some bi-grams that appear only once or twice can be meaningful to humans, but since the bi-gram contains little semantic information, almost any text data cleaning procedure will remove it. One example is the name of the police officer who arrested George Zimmerman.

On the other hand, the p-value cutoff is also essential because this removes most words which just happen to be together. For example, "said obama" is a common phrase, but not a meaningful bi-gram because "said" has a high marginal probability. Therefore, conditional probabilities are needed to filter this out, and p-values are used as a testing threshold.

3.2.2 Topic Modeling

Topic modeling is a part of natural language processing because this allows computers to perform automated text analysis, i.e. to statistically classify text documents into different topics. In this way, human brains can focus on key articles and need not to read through too much irrelevant information. The ultimate goal of topic modeling is to discover the relationship between documents, and a common approach is latent Dirichlet allocation (LDA).

For instance, a document about cats may contain “kitten” and“meow”; a document about dogs may contain “puppy” and “bone”. On the other hand, stopwords such as “the” and “with” appear in most documents. In some cases, non-stopwords are also prevalent across the corpus; for example, “A collection of documents on the auto industry is likely to have the term auto in almost every document” (Manning et al., 2008).

Latent Dirichlet Allocation (LDA)

Latent Dirichlet allocation (LDA) is based upon a Bayesian generative model, which draws topics from a Dirichlet distribution as the prior, and then updates the probabilities by using the words in the documents.

Instead of assigning a document to a single topic, LDA outputs a topic assignment vector for each document, and the probabilities for topics are defined using word counts. For example, (0.4, 0.2, 0.4) means the document has topic proportions 40% in Topic 1, 20% in Topic 2, and 40% in Topic 3. Therefore, this document contains 40% words that belong to Topic 1, 20% words that belong to Topic 2, and 40% words that belong to Topic 3.

However, to avoid overfitting, the statistical value of parsimony and regularization means that small probabilities should be shrunk to zero. Therefore, if a document has 99% probability in one topic, it is reasonable to assign the document to that topic.

The entire corpus is a set of documents $\{D_1, \cdots, D_m\}$. The words within a document are denoted as $D_j = \{d_1, \cdots, d_{n_j}\}$, where $d_i \in W$, and $W = \{w_1, \cdots, w_V\}$ is a finite vocabulary set.

The data generative process of LDA is illustrated in Figure 3.1 and defined as below:

- For each topic k
  - Draw a word proportion vector: $\phi_k \sim \text{Dirichlet}_V(\alpha)$
- For each document D_j
  - Draw a topic proportion vector: $\theta_j \sim \text{Dirichlet}_K(\beta)$
  - For each word w_i in document D_j
    * Draw a topic assignment $z_i \sim \text{Mult}(\theta_j)$
    * Draw a word from the topic $w_i \sim \text{Mult}(\phi_{z_i})$

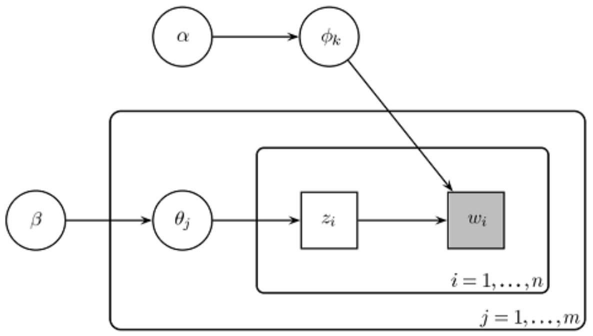


FIGURE 3.1: Plate diagram for the LDA process (reproduced from (Blei et al., 2003))

Properties

The Dirichlet prior conjugates with the data likelihood which follows a multinomial distribution (Thulin, 2012).

- Prior probabilities:

  $(p_1, \cdots, p_K) \sim \text{Dirichlet}(\alpha_1, \cdots, \alpha_K)$ with $\sum p_i = 1$ and $p_i \in (0, 1)$

  pdf: $f(p_1, \cdots, p_K) = \dfrac{1}{B(\alpha)} \prod_{i=1}^{K} p_i^{\alpha_i - 1}$, where $B(\alpha) = B(\alpha_1, \cdots, \alpha_K) = \text{constant}$

- Likelihood:

  Data $(x_1, \cdots, x_K)$ modeled as multinomial,

  with $x_i \in \{0, 1, \cdots, n\}$ and $\sum x_i = n$

- Posterior probabilities:

  $(p_1, \cdots, p_K)|(x_1, \cdots, x_K) \sim \text{Dirichlet}(\alpha_1 + x_1, \cdots, \alpha_K + x_K)$

The priors for each word are the same because they come from the same Dirichlet distribution. The priors are set for the word proportion vector ($\phi_k \sim \text{Dirichlet}_V(\alpha)$

for each topic k) and the topic proportion vector ($\theta_j \sim \text{Dirichlet}_K(\beta)$ for each document D_j). The parameters I need to pre-assign are only α and β.

Variational Inference

Even though the Dirichlet distribution prior is conjugate with the multinomial distribution, the joint posterior distribution for θ (topic proportion vector) and z (topic assignment vector) contains a computationally intractable likelihood component (Blei et al., 2003):

$$p(\theta, z|D, \alpha, \beta) = \frac{p(\theta, z, D|\alpha, \beta)}{p(D|\alpha, \beta)} \tag{3.1}$$

To approximate the likelihood $p(D|\alpha, \beta)$, I need a lower bound for $\log[p(D|\alpha, \beta)]$, where D denotes the documents (data). Note that additional variational parameters γ, ψ are introduced.

$$\log[p(D|\alpha, \beta)] = \mathbb{L}(\gamma, \psi|\alpha, \beta) + D_{KL}(q(\theta, z|\gamma, \psi)||p(\theta, z|D, \alpha, \beta)) \tag{3.2}$$

This is the mean-field variational method (Moghaddam, 2009). The target distribution $p(z|x)$ can be approximated by a factorized distribution $q(z)$, and the "distance" between the two is measured by the Kullback-Leibler (KL) divergence:

$$D_{KL}(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z|x)} \tag{3.3}$$

Since KL divergence $\geqslant 0$, the lower bound of $\log[p(D|\alpha, \beta)]$ is $\mathbb{L}(\gamma, \psi|\alpha, \beta)$.

Then

$$\mathbb{L}(\gamma, \psi|\alpha, \beta) = E_q[\log p(\theta|\beta)] + E_q[\log p(z|\theta)] + E_q[\log p(w|z, \beta)] \tag{3.4}$$

$$- E_q[\log q(\theta)] - E_q[\log q(z)], \tag{3.5}$$

where $q(\theta, z|\gamma, \psi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\psi_n)$. γ and $\psi = (\psi_1, \cdots, \psi_N)$ are free variational parameters.

To minimize the KL divergence, I need to get

$$(\gamma^*, \psi^*) = \arg\min_{(\gamma,\psi)} D_{KL}(q(\theta, z|\gamma, \psi)||p(\theta, z|D, \alpha, \beta)) \tag{3.6}$$

This can be done using iterative updating:

$$\psi_{n,i} \propto \alpha_{i,D_n} \exp\{E_q[\log(\theta_i)|\gamma]\} \tag{3.7}$$

$$\gamma_i = \beta_i + \sum_{n=1}^{N} \psi_{n,i} \ . \tag{3.8}$$

Variational inference can also be applied to Bayesian optimization (Ranganath et al., 2016; Blei et al., 2016).

Parameter Estimation

Variational expectation-maximization (EM) is done to find the parameters α, β, in order to maximize the log likelihood of data:

$$l(\alpha, \beta) = \sum_{d=1}^{M} \log(p(D_d|\alpha, \beta)). \tag{3.9}$$

The E-step (expectation) and M-step (maximization) are iterated repeatedly until convergence:

- E-step: maximize the bound w.r.t. the variational parameters
- M-step: maximize the bound w.r.t. the model parameters

Note that EM may converge to a local maximum, but it has been proven that variational inference for LDA-like methods generally converges to the global maximum (Awasthi and Risteski, 2015).

3.3 Implementation

Available R packages for LDA include `lda` and `topicmodels`. The package `lda` is easier to use, while `topicmodels` supports more functions.

For the parameters, the number of topics is set to $K = 5$ because previous researchers used this value (Soriano et al., 2013; Au, 2014) on the same dataset, and other parameter values are set to $\alpha = 0.1, \beta = 0.1$.

3.3.1 Posterior Probability

The preliminary results are generated by `lda.collapsed.gibbs.sampler` from the R package `lda`. Tables 3.1 and 3.2 list the top 10 words for each topic generated by the LDA function `top.topic.words`; the former is before n-gramming, and the latter is after. Four bi-grams, "fox_news", "trayvon_martin", "georg_zimmerman", and "dave_surl" appear in Table 3.2.

To show the difference between topics, I named all five categories in Table 3.1 based on the vocabulary, and I used the same labels in other tables. The first topic is “Election” because it contains words such as “obama”, “presid”, “romney”, and “democrat”, which normally appear in the 2012 US presidential election. The second topic is obviously “Incident” because it includes both “martin” and “zimmerman”. The third topic is “News Coverage” due to the words “fox” (Fox News), “malkin” (Michelle Malkin, an American political commentator), and “msnbc” (an American television network). The fourth topic is “Gun Law” because the top words are “law”, “gun”, “ground”, and “stand”. Last but not least, the fifth topic is named as “General” because of the words “comment”, “know”, and “think”.

The function `top.topic.words` selects words for each topic distribution based on $P($ word $j|$ topic $i,$ data $)$, which is the probability of getting word j given that the document belongs to topic i. Hence this function returns words that are most likely to appear in each topic. However, some words are actually common across topics, and these are (unfortunately) not stop words. For example, “trayvon” and “martin” belong to Topic 2 in Table 3.2, but they are prevalent across documents because the whole dataset is related to the Trayvon Martin shooting incident, so the two words have low distinctivity in terms of topic selection. This also explains why one topic in both tables is labeled as “General”, since words with limited semantic meaning can be in the same topic.

3.3.2 Word Distinctivity

Word distinctivity calculates the probability that the document belongs to each topic, given a certain word. I select the top 10 most distinctive words for each topic, which is measured by $P($ topic $i|$ word $j,$ data $)$, i.e. how likely a specific topic i is, given

Table 3.1: **Before** n-gramming: top 10 words for each topic generated by LDA

Topic 1 Election	Topic 2 Incident	Topic 3 News Coverage	Topic 4 Gun Law	Topic 5 General
obama	zimmerman	anonym	law	like
presid	martin	fox	gun	peopl
year	said	news	ground	dont
romney	trayvon	liber	stand	comment
american	polic	tommi	forc	get
democrat	georg	malkin	alec	know
said	call	msnbc	mar	think
nation	black	show	defend	right
women	prosecutor	conserv	reason	make
govern	charg	gregori	shoot	white

Table 3.2: **After** n-gramming: top 10 words for each topic generated by LDA

Topic 1 News Coverage	Topic 2 Incident	Topic 3 General	Topic 4 Zimmerman Trial	Topic 5 Election
anonym	zimmerman	like	comment	obama
liber	martin	peopl	spokesmancom	presid
fox	case	get	perjuri	year
tommi	polic	make	**dave_surl**	american
show	law	know	bond	romney
conserv	**trayvon_martin**	think	free	republican
news	said	right	access	govern
fox_news	**georg_zimmerman**	say	view	law
msnbc	trayvon	dont	surl	women
hanniti	shoot	see	account	countri

a specific word j.

Therefore, the word distinctivity is calculated by the Bayes' theorem:

$$P(\text{ topic } i|\text{ word } j,\text{ data }) = \frac{P(\text{ word } j|\text{ topic } i,\text{ data })P(\text{ topic } i)}{\sum_{\text{topic k}} P(\text{ word } j|\text{ topic } k,\text{ data })P(\text{ topic } k)} \quad (3.10)$$

Tables 3.3, 3.4 list the top 10 distinctive words for each topic generated by LDA; the former is before n-gramming, and the latter is after. In Table 3.3, Topic 1 is about racism because a hoody is a fashionable were among African American teenagers, and Al Sharpton is an American civil rights activist. Topic 2 is about gun laws (i.e. whether people are allowed to have their own guns); Topic 3 is about the 2012 presidential campaign (Barack Obama and Mitt Romney); Topic 4 is about the news, and Topic 5 is about the trial of George Zimmerman.

In Table 3.4, some bi-grams make the topics more specific. For example, bi-grams such as "stand_ground", "neighborhood_watch", and "self_defense" make it clear that the topic is about the facts of the Trayvon Martin shooting itself. Note that the bi-grams "trayvon_martin" and "georg_zimmerman" are not included in the top 10 distinctive words because the whole corpus is related to the Trayvon Martin incident.

Table 3.3: **Before** n-gramming: top 10 distinctive words for each topic

Topic 1 Racism	Topic 2 Gun Law	Topic 3 Election	Topic 4 News Coverage	Topic 5 Zimmerman Trial
hoodi	alec	barack	tommi	spokesmancom
dispatch	gun	mitt	anonym	surl
mar	moral	presid	tue	perjuri
sharpton	legisl	obama	malkin	corey
minut	group	administr	idiot	dave
polic	individu	candid	stupid	bond
martin	weapon	tax	gregori	expert
trayvon	ground	health	palin	access
walk	violenc	congress	rich	prosecutor
unarm	violat	romney	liber	comment

Table 3.4: **After** n-gramming: top 10 distinctive words for each topic

Topic 1 Zimmerman Trial	Topic 2 Gun Law	Topic 3 News Coverage	Topic 4 Racism	Topic 5 Election
comment_dave	spokesmancom	malkin	**black_panther**	**mitt_romney**
dave_surl	york	palin	sharpton	**barack_obama**
perjuri	retreat	tue	panther	alec
surl	**stand_ground**	fox	young	congress
dave	access	hanniti	white	administr
bond	**neighborhood_watch**	tommi	race	econom
expert	hoodi	msnbc	black	tax
comment	sanford	dog	trayvon	economi
voic	**self_defens**	anonym	racism	senat
corey	florida	mitt	drug	**unit_state**

3.3.3 Remarks

In the tables, the words are truncated due to stemming and tokenization (Section 2.3.1), so they are actually tokens of the base form. Currently all word stem programs do this, or the program will have to automatically identify what is the most common word form, which greatly increases the implementation difficulty.

Moreover, the LDA algorithm is "unstable" because it does not know what topic the words actually belong to; it simply determines which words belong to the same topic. In other words, which topic is labeled as "Topic 1" in the LDA output is arbitrary. For example, Topic 1 in Table 3.1 is about the 2012 US presidential election, but it becomes Topic 5 in Table 3.2.

One solution to the instability (labeling) problem is using seeded topic models. I can identify topics that are similar with respect to a probability metric between MCMC iterations (Stephens, 2000). Available probability metrics include total variation distance and the Kolmogorov-Smirnov (K-S) test. For implementation, Topic 1 can be set on the 2012 election with words "barack_obama" and "mitt_romney"; Topic 2 can be set on individual gun rights with words "stand_ground" and "self_defens".

3.4 Comparison of Results

The topic modeling results before and after n-gramming are compared in two ways. One is the increase of word distinctivity, and the other is the perplexity, which measures the predictive power of topic models. The paper "Topic Modeling: Beyond Bag-of-Words" (Wallach, 2006) also measures the predictive performance (but it uses text compressibility), so the increase of word distinctivity is interesting because it directly compares posterior probabilities at the word level. On the other hand, the perplexity has a long history of being used as a measure in natural language processing (Jelinek et al., 1977).

3.4.1 Increase of Word Distinctivity

To compare the topic modeling results before and after n-gramming, I define the change of word distinctivity for uni-grams (single words) and bi-grams as below:

- Uni-grams: $\max_i[P(\text{ topic } i| \text{ word } j, \text{ data after }) - P(\text{ topic } i| \text{ word } j, \text{ data before })]$
- Bi-grams: $\max_i[P(\text{ topic } i| \text{ bi-gram } b, \text{ data after}) - \text{max.prob}]$,
  where $\text{max.prob} = \max(P(\text{ topic } i| \text{ word } 1, \text{ data before}), P(\text{ topic } i| \text{ word } 2, \text{ data before}))$

The null and alternative hypotheses are:

- H_0: After n-gramming, the results are no better. $\Leftrightarrow$ Probability change $\leqslant 0$
- H_1: After n-gramming, the results are better. $\Leftrightarrow$ Probability change > 0

A t-test is performed to compare the overall probability change for uni-grams with all zeros. The 95% confidence interval is [-0.0081, 0.0025], and the p-value is 0.303. Accordingly, I fail to reject H_0 and conclude that the overall change is insignificant.

On the other hand, eight individual bi-grams show $> 10\%$ probability change, as listed in Table 3.5. For these bi-grams, $P($ topic $i|$ word 1, data before$)$ ("first prob.") and $P($ topic $i|$ word 2, data before$)$ ("second prob.") are low, but the bi-gram's posterior probability $P($ topic $i|$ bi-gram $b,$ data after$)$ is high. As a result, the bi-grams have higher distinctivity, so the uncertainty in topic identification decreases. In plain English, the phrase almost always appears in a certain topic, and examples include "black_panther" and "stand_ground".

Table 3.5: The posterior probability (abbreviated as "prob.") of bi-grams

	Bi-grams	Bi-gram prob.	First prob.	Second prob.	Prob. change
1	black_panther	0.901	0.484	0.525	0.376
2	unit_state	0.848	0.479	0.475	0.369
3	self_defens	0.798	0.370	0.476	0.322
4	neighborhood_watch	0.852	0.584	0.463	0.269
5	white_hous	0.793	0.354	0.528	0.265
6	stand_ground	0.910	0.632	0.687	0.222
7	year_old	0.574	0.385	0.391	0.183
8	new_york	0.491	0.339	0.314	0.152

3.4.2 Perplexity

Perplexity is a single number summarizing how well the topic models predict the remaining words, conditional on the rest of the document. This is the effective number of equally likely words based on the model, so the lower the perplexity, the better (higher precision). The mathematical definition is "the inverse of the per-word geometric average of the probability of the observations" (Blei and Lafferty, 2006). Given that P words are used for training in each document,

$$\text{Perp}(\Phi) = (\prod_{d=1}^{D} \prod_{i=P+1}^{N_d} p(w_i|\Phi, w_{1:P}))^{\frac{-1}{\sum_{d=1}^{D}(N_d-P)}}, \tag{3.11}$$

where Φ is the fitted model, D is the total number of documents, N_d is the number of words in document d, and w_i indicates a token in a document.

The goal of perplexity is to compare the average negative log-likelihoods, and the definition of perplexity involves the geometric average because the geometric mean of likelihoods is equivalent to the exponential of the arithmetic average log-likelihoods. The proof is relatively long, so it is included in Appendix D). Here is an example for the geometric mean $\bar{x}_{GM}$ of x_1, x_2, x_3 (Cardinal on Stack Exchange, 2012):

$$y_1 = \log x_1, y_2 = \log x_2, y_3 = \log x_3 \tag{3.12}$$

$$\bar{x}_{GM} = (x_1 x_2 x_3)^{1/3} = \exp(\frac{y_1 + y_2 + y_3}{3}) = \exp(\bar{y}) \tag{3.13}$$

To obtain the perplexity scores, I performed ten-fold cross validation on both versions of the Trayvon Martin dataset (Soriano et al., 2013; Au, 2014). The corpus was randomly divided into ten equal-sized parts – nine parts are the training data, and one part is for testing (hold-out sample). For each combination, I trained the topic model using the `LDA` function in the `R` package `topicmodels`, and calculated the perplexity for the testing data. Each corpus version gets ten perplexity scores, as shown in Table 3.6.

After the perplexity scores of before vs after n-gramming are available, hypothesis testing is done using a t-test. Similar to the previous section, the null and alternative hypotheses are:

- H_0: After n-gramming, the results are no better.
  $\Leftrightarrow$ The perplexity is not lower.
- H_1: After n-gramming, the results are better.
  $\Leftrightarrow$ The perplexity is lower.

The t-test shows p-value > 0.05, so the improvement of results from n-gramming is insignificant. Hence I fail to reject H_0 and conclude that the perplexity does not improve after n-gramming. Since the significant bi-grams account for only a small part of the corpus, it is currently unclear how n-gramming helps at the corpus level.

Table 3.6: Perplexity scores of before and after n-gramming – (rounded to the nearest integer).

Before	1121	1164	1159	1060	1169	1161	1126	1165	1028	1116
After	1167	1250	1210	1121	1243	1236	1193	1213	1081	1161

3.4.3 Current Limitations

In using the metrics to compare topic modeling results, current limitations include issues with the *t*-test and label switching in MCMC.

First, the *t*-test computes statistical, not practical, significance. For small sample sizes, the *t*-test is under-powered, i.e. a large difference is required to show statistical significance. Therefore, I set the cutoff to require any two-word phrases appear at least 100 times (Section 3.2.1), but better methods may exist. On the other hand, the t-test is also susceptible to multiple testing, which leads to a high false discovery rate (Benjamini and Hochberg, 1995). However, the t-test is the standard testing method to compare the mean of two groups, and since my research is not focused on multiple testing, I use the default methodology.

On the other hand, label switching is a common issue in the MCMC chain (Jasra et al., 2005). Word-topic classification is a form of Bayesian mixture models, and if the prior distribution is the same for each component of the mixture, the posterior will be nearly symmetric. This symmetry is problematic when researchers try to estimate quantities with respect to individual components. For example, the marginal classification probabilities for each cluster are the same, leading to poor estimates of the parameters (Stephens, 2000).

3.5 Conclusion

The contribution of n-gramming to topic models is the increase in word distinctivity, measured by the posterior probability $\max_i P(\text{ topic } i | \text{ word or bi-gram } j, \text{ data})$. Without n-gramming, stemmed words (such as "self" and "defens") have low posterior probability, but when the words are combined, the distinctivity is high, so the uncertainty in topic identification decreases.

4

Quantifying the Uncertainty of Latent Dirichlet Allocation (LDA)

4.1 Introduction

George Box famously remarked that "All models are wrong, but some are useful" (Box et al., 1987). Every statistical model is subject to variability; the usefulness of any statistical output depends on both the point estimate (accuracy) and the probabilistic region (precision) (Loucks and van Beek, 2005). Therefore, the measurement of uncertainty is important (Kirkup and Frenkel, 2006; Bell, 2001); good examples include frequentist confidence intervals and Bayesian credible intervals. Latent Dirichlet allocation (LDA), a widely-used topic modeling method described in Section 3.2.2, is no exception in the measurement of uncertainty.

However, many researchers take the latent Dirichlet allocation (LDA) model for granted because numerous applications exist (Wei and Croft, 2006; Hu, 2009; Agarwal and Chen, 2010). But there is virtually no literature addressing the uncertainty within LDA. Accordingly, our group created synthetic datasets from Wikipedia articles for which the ground truth topics were known, and tested the LDA performance on several synthetic datasets with different combinations of topics. In comparison to Chapter 3, which is about quantifying the improvement of topic modeling results by n-gramming, this chapter is focused on how to measure the uncertainty of LDA itself.

To measure the uncertainty in LDA results, I investigated in two aspects: dropping excessive topics and determining the cutoff frequency. In LDA practice, the number of topics is a pre-defined constant, so it would be better if the text data can automatically determine the best number of topics. Due to the model constraints, we want to start with a large number of topics and then develop a methodology to remove unnecessary topics.

Additionally, for almost any document corpus, many words appear only a few times (Goldwater et al., 2011), and words that are very rare need to be removed. The reason for this is that their signal is low in determining a topic, so their removal is a form of regularization. Additionally, reducing the vocabulary greatly accelerates the computation. The cutoff on the number of occurrences below which a word is removed is usually set arbitrarily, so we shall also explore how this pre-set threshold affects the quality of LDA results. These two explorations can provide guidance to the practical application of LDA by other researchers.

4.2 The Synthetic Datasets

Two synthetic datasets were created by cutting and pasting text from Wikipedia articles. Since Wikipedia article are classified in a taxonomy of topics, we have the ground truth; i.e., the underlying topics in each document. The first dataset, denoted as "far", contains 50 documents from two Wikipedia categories that are intellectually distant from each other (statistics and dance). The second dataset is denoted as "similar" because it contains 50 documents created from two similar Wikipedia categories (statistics and calculus). These two datasets serve as the benchmarks for testing topic models.

Each synthetic document contains 500 words and exactly two topics. The topic of any text is determined by the topic of its source Wikipedia page, and the topic proportions in each document are determined by the number of words drawn from each of those parent pages. The proportions can be either 50%/50%, 25%/75%, or 12.5%/87.5%. Therefore, for example, a document with topic proportions 25%/75% contains 125 words from one topic and 375 words from the other.

4.2.1 Scraping Wikipedia Pages

We used the `R` package `rvest` for webscraping. The script is reproducible given a random seed, hence it can be generalized and replicated for any topic in Wikipedia and for any length of the synthetic documents. For each topic (statistics, dance, and calculus), we obtained a list of relevant Wikipedia pages. Then a random sample of those pages was drawn.

For each chosen page, a segment of text was scraped, with a random starting point. The rest of the webpage needed to contain enough words to scrape, otherwise the starting point needs to be resampled. Various starting points help the chosen text become more "representative" of the Wikipedia articles, because the first few paragraphs are often introductory and contain less technical language.

To create the 100 synthetic documents with 500 words each, we generated the text with 10 different article combinations and 5 different proportions. The corresponding number of words in topics for each proportion is listed in Table 4.1.

Table 4.1: Number of words in topics for each proportion

Proportions	# of Words in Topic 1	# of Words in Topic 2
1/8, 7/8	62	438
1/4, 3/4	125	375
1/2, 1/2	250	250
3/4, 1/4	375	125
7/8, 1/8	438	62

The file name for each synthetic document has three components: The first number is the ordering of the Wikipedia article combination (from 1 to 10 in both "far" and "similar" corpuses). In each 50-document corpus, the same first number means the same combination of article sublinks. The second number is the number of words in Topic 1, and the third number is the number of words in Topic 2. For example, the file `1_62_438.txt` in "far" contains 62 words in statistics and 438 words in dance.

4.2.2 Cleaning Raw Text

After the 100 documents are scraped, the raw text is cleaned using regular expressions to remove citations and HTML tags. In this way, only alphabetic characters remain in the synthetic corpus, and we converted all text to lower case for simplicity.

Next, stop words are removed from the corpus, using the list `stopwords(kind = ‘‘english’’)` in the R package `quanteda`. Note that the publicly available stop word list include contractions (e.g. “we’d”) and negations (e.g. “no”, “isn’t”), but we used the default settings of the package and did nothing special to handle them.

After removing stop words, my research group stemmed the corpus using the the `wordStem` function from the R package `SnowBallC`. The input to `wordStem` must be a vector of words, so it cannot be directly applied to the entire corpus. Finally, unnecessary white spaces in the text documents are trimmed, to ensure the documents are in `PlainTextDocument` format for further processing.

4.2.3 N-Gramming Methodology

In text mining, n-gramming (described in Section 3.2.1) is the process of retaining special phrases, in order to recover semantic information that would otherwise be lost. LDA uses the bag-of-words model, so word order is not retained unless n-gramming is used (Soriano et al., 2013). For example, “white” and “house” are common words, but the phrase “White House” refers to the official residence of the US president. The n-gramming discovers the improbably frequent co-occurence of this pair, and assigns them a unique joint identifier.

To perform n-gramming, we also used the functions from an existing GitHub repository (Henry, 2016). We used a p-value of 0.01 as the threshold for determining that a word pair, or bi-gram, co-occurs with sufficiently high frequency to have distinct semantic meaning, and this is a standard choice in the n-gramming literature (Blei and Lafferty, 2009).

We also chose to discard bi-grams with counts less than 10. As previously discussed, rare words have low signal and increase computing time. The count cutoff was empirically determined. The bi-gram "unbiased estimate" was selected as the benchmark (which appears 11 times in the corpus), because every document in the synthetic corpus contains statistical content.

The code written by Henry (2016) uses exact binomial probabilities to compute the p-value, and my research group tried to use a normal distribution to approximate the binomial one. A binomial distribution $\mathrm{Bin}(n, p)$ can be approximated by $\mathrm{N}(np, np(1-p))$, especially when $np \geqslant 10$ or $n(1-p) \geqslant 10$.

Nevertheless, the p-values computed by the normal approximation are much larger than the ones calculated using the exact binomial distribution. When the success probability parameter p is small, the coverage of the 95% confidence interval using a normal approximation (a.k.a. Wald interval) is much less than 95%.

Therefore, the data is more likely to fall outside the 95% Wald interval than the exact 95% confidence interval (calculated using the binomial distribution). For example, the p-value of the benchmark bi-gram “unbiased estimate” is 1.768×10^{-23} using exact confidence intervals and 4.959×10^{-4} using the approximation. More details about the coverage probabilities for binomial confidence intervals are in Appendix C.

4.2.4 N-Gramming Results

A total of 52 bi-grams were identified from the synthetic corpus. These can be divided into three types: statistics, calculus, and dance. The classified bi-grams are listed below. Note that the truncated words result from the stemming done in the pre-processing step (so that, for example, distribution and distributed and distributions are all identified by the same token).

- Statistics

```
standard_deviat, random_variabl, normal_distribut
null_hypothesi, time_seri, row_vector, sampl_mean
converg_probabl, control_group, fit_data, unbias_estim,
column_vector, meanunbias_estim, sampl_size, vector_space
popul_mean, simpl_random, hypothesi_test
probabl_distribut, data_set, random_sampl, converg_distribut
```

- Calculus

```
analyt_function, inflect_point, finit_differ, academ_disciplin
linear_map, point_inflect, function_f, symplect_manifold
complex_analyt, x_y, partial_deriv,  real_number
```

`differenti_function, exist_continu, function_differenti`

`deriv_function, variable_x, point_x`

- Dance

  `lindy_hop, perform_art, intellectu_rigour, class_c`

  `jazz_dance, danc_form, unit_state, th_centuri, can_also`

  `can_made, can_us, on_can`

To identify tri-grams and even longer n-grams, we repeatedly applied the same function in the existing R script (Henry, 2016) until no more n-gram is identified with the p-value and count cutoff.

Note that different concatenation symbols are used for bi-grams and tri-grams, in order to avoid confusion. For example, a bi-gram is "analyt_function", where the "_" symbol connects the two words. Then the tri-gram generated from it is "complex.analyt_function", with a different punctuation "." between "complex" and "analyt_function".

In the Wikipedia corpus, we did not identify any tri-gram with at least 10 counts. The close candiates are "complex.analyt_function" with 9 appearances and "simpl_random.sampl" with 8.

4.2.5 Discussion

The synthetic datasets ("far" and "similar") serve as a benchmark to assess topic modeling because the topic proportions are pre-set beforehand. The two datasets are used in my investigation to drop excessive topics (Section 4.3) and determine the cutoff frequency for removing rare words (Section 4.4).

Developing a methodology to determine the count cutoff of n-grams is also a potential next step. It is obvious that the count cutoff should be determined by the corpus size, but the cutoff threshold also depends on the topic proportion of the dataset. For example, if the corpus is the Joint Statistical Meetings (JSM) abstracts, the cutoff for statistical n-grams should be much higher than the case of a mixed-discipline corpus (e.g. a random sample of Wikipedia pages).

Last but not least, the text cleaning process can be improved by preserving inline equations, especially in a mathematical or statistical corpus. When all punctuation and non-alphabetic symbols are removed, the math equations look odd after they have been converted to text. For example, the z-score $Z = (x - \mu)/\sigma$ becomes "zx", and $\tan S = \sqrt{(\dfrac{\partial z}{\partial x})^2 + (\dfrac{\partial z}{\partial y})^2}$ becomes "tanSzxzy". Converting these equations to LaTeX-style text is a possible option, but the converted long "n-grams" are difficult to read by humans and require more insight. For instance, the LaTeX formula of the z-score is "Z = (x - \mu) / \sigma" and the tangent-related equation looks like "\tan S = \sqrt{(\dfrac{\partial z}{\partial x})^2 + (\dfrac{\partial z}{\partial y})^2}".

4.3 Determining the Number of Topics

A drawback of latent Dirichlet allocation (LDA) is the need to pre-set the number of topics. LDA does not specify it automatically, even though each topic is an infinite mixture model over a latent set of topic probabilities (Blei et al., 2003). In principle, one might use a Poisson prior on the number of topics, but this has proven unwieldy (Henry et al., 2016).

Thus, I tested various numbers of topics on the same corpus to compare sensitivity of the results. I developed three measures to assess the quality of LDA outputs as a function of the number of topics allowed: the number of words in each topic, the topic weights, and the number of documents in each topic.

Because text data pre-processing can be done to different degrees and in different ways, I needed to make critical decisions prior to implementing LDA. To determine which methods work best for the synthetic datasets and whether to use the "far" and/or "similar" datasets, I started by using all 100 documents without removing stop words, and then modified the cleaning process based on the output.

4.3.1 Without Removing Stop Words (Raw Data)

In this portion of the study, I experimented with the combined Wikipedia dataset using the raw data, without removing stop words or stemming. When the stop words are not removed, the topic modeling results are poor. In this case, even n-gramming does not help much, since bi-grams such as "as_the" and "of_this" exist.

I tried to remove these misleading signals by retaining only the words in each topic that had minimum frequency 20, but this did not provide much improvement. For example, I pre-set two topics for LDA, and words are assigned to a topic if and only if $P(\text{topic}|\text{word, data}) > 0.9$. Then for each topic, I selected only the words with minimum frequency 20 to avoid rare words with high conditional probability. The two topics are described in Tables 4.2 and 4.3. Topic 1 contains "convergence" and "political"; Topic 2 contains "statistical" and "dance". Therefore, under this protocol, the results do not clearly distinguish topics.

In conclusion, removing stop words is essential because unremoved stop words can dominate the topics, making important words such as "statistics" and "dance" insignificant.

Table 4.2: Topic 1 in the Wikipedia corpus (min frequency 20)

after	almost	audience	became	century
convergence	derivative	disciplines	events	exercise
he	including	inflection	manifold	notation
political	research	science	social	studies
very	was_a	without		

Table 4.3: Topic 2 in the Wikipedia corpus (min frequency 20)

age	analytic	arts	average	below
c	close	coding	complex	**dance**
data	distance	equal	error	estimate
estimator	from_a	group	mean	median
model	much	observed	population	power
sample	samples	sampling	series	simple
size	standard	standard_deviation	states	**statistical**
test	text	the_data	variable	when

4.3.2 After Removing Stop Words

Removing stop words is obviously the right way to go, but the presence of both “far” (statistics+dance) and “similar” (statistics+calculus) documents can interfere with the latent Dirichlet allocation (LDA) results. Hence I used the perplexity metric (described in Section 3.4.2) to compare the corpus with far topics and the other corpus with similar topics.

The null and alternative hypotheses are:

- H_0: The “far” dataset generates no better results than the “similar” one.
  $\Leftrightarrow$ The perplexity of “far” is not lower than “similar”.
- H_1: The “far” dataset generates better results than the “similar” one.
  $\Leftrightarrow$ The perplexity of “far” is lower than “similar”.

Ten-fold cross validation was performed on each corpus, and the perplexity of far is significantly higher than the perplexity of similar, with the one-sided p-value 0.999. Since lower perplexity is better, the far corpus performs no better than the similar corpus, which is contradictory to intuition.The perplexity values are listed in Tables 4.4 and 4.5. Therefore, it is desirable to use all 100 documents because the two datasets do not show significant distinction.

Table 4.4: Perplexity for far corpus: mean = 1633.41.

1657.537	1518.041	1723.257	1437.822	1762.265
1739.266	1452.458	1467.279	1750.059	1823.425

Table 4.5: Perplexity for similar corpus: mean = 1187.651.

1253.083	1258.688	1080.575	1248.421	1151.776
1179.671	1147.829	1201.601	1112.587	1242.273

I originally assumed that words appearing fewer than 10 times would contribute to the noise. The vocabulary size for the Wikipedia corpus is 4673, and only 722 (15.5%) of the words appear at least 10 times (only 17 words have frequency at least 100, so setting the cutoff to 100 would be much too high.)

I then performed the same steps using both the far and similar corpora but with only the words that appear at least 10 times. The computation is faster because the datasets have been greatly reduced.

However, Tables 4.6 and 4.7 show almost identical results as the previous corpus versions (only with stop words removed). Moreover, the mean perplexity does not change much for both far and similar datasets, so I conclude that setting a frequency cutoff at the corpus stage does not improve the results.

Table 4.6: Perplexity for far corpus (min frequency 10): mean = 1644.795.

1672.202	1539.568	1744.917	1367.133	1778.632
1766.468	1487.579	1573.029	1665.245	1853.182

Table 4.7: Perplexity for similar corpus (min frequency 10): mean = 1183.127.

1297.297	1313.053	1060.982	1236.473	1128.139
1157.816	1132.066	1189.442	1178.170	1137.835

4.3.3 Determine the Number of Topics

To determine the appropriate number of topics for LDA, I set a large number and see how unneeded topics can be dropped according to the results. The three measures I developed are: the number of words in each topic, the topic weights, and the number of documents in each topic.

The version of the synthetic dataset I used is all 100 documents, with stop words removed, tokenized, and n-grammed. Since the corpus contains only 100 documents, the number of topics is initially set to 50 (two documents in one topic on average).

Number of Words in Each Topic

The first measure is the number of words in each topic, and the motivation is that any topic containing very few words should be removed. In LDA, each word gets a topic assignment vector, and if one topic is dominant (i.e. assignment probability > 0.5), the word will always gets assigned to that topic.

This is the same concept as word distinctivity: Given word j and the data, which topic is the most likely? So one calculates

$$P(\text{ topic } i | \text{ word } j, \text{ data }) = \frac{P(\text{ word } j | \text{ topic } i, \text{ data })P(\text{ topic } i)}{\sum_{\text{topic k}} P(\text{ word } j | \text{ topic } k, \text{ data })P(\text{ topic } k)} \tag{4.1}$$

and word distinctivity is defined as $\max_i P(\text{ topic } i | \text{ word } j, \text{ data })$.

In the implementation, I set the word distinctivity threshold to 0.9; i.e., a word is assigned to topic i if and only if the corresponding word-topic assignment vector $(x_1, \cdots, x_K)$ has $x_i \geqslant 0.9$ for some $i \in \{1, \cdots, K\}$. The candidate values for the number of topics (K) are 2, 3, 5, 10, 20, and 50. Here $K = 2$ means statistics and dance; $K = 3$ means statistics, calculus, and dance. As K increases, I use the same threshold, but the number of words assigned to each topic remains large.

Table 4.8 shows the number of words for each topic when $K = 2, 3, 5, 10$. Figures 4.1 and 4.2 shows the histograms for 20 and 50 topics, respectively; they are both right-skewed.

Table 4.8: Number of **words** in each topic for various pre-set numbers of topics

Number of Topics	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
2	1594	1750								
3	804	1038	1225							
5	545	554	618	690	478					
10	273	181	331	350	364	295	204	203	194	395

What I had expected to see is that a few topics contain the majority of the words, and that many topics contain only one or two words. If a topic contains less than ten words, then the topic should probably be combined with another topic.

However, LDA tends to "spread out" word-topic assignments, making it difficult to drop excessive topics. In Table 4.8, the number of words in each topic are close to each other. Even though the histograms for 20 and 50 topics are right-skewed, very few topics are rare enough (i.e. contains less than 10 words) to be considered

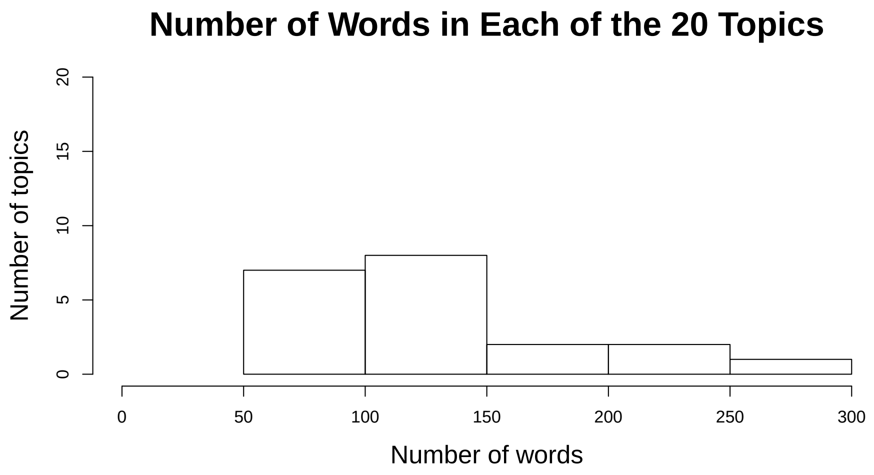


FIGURE 4.1: Histogram for number of words in each of the 20 topics

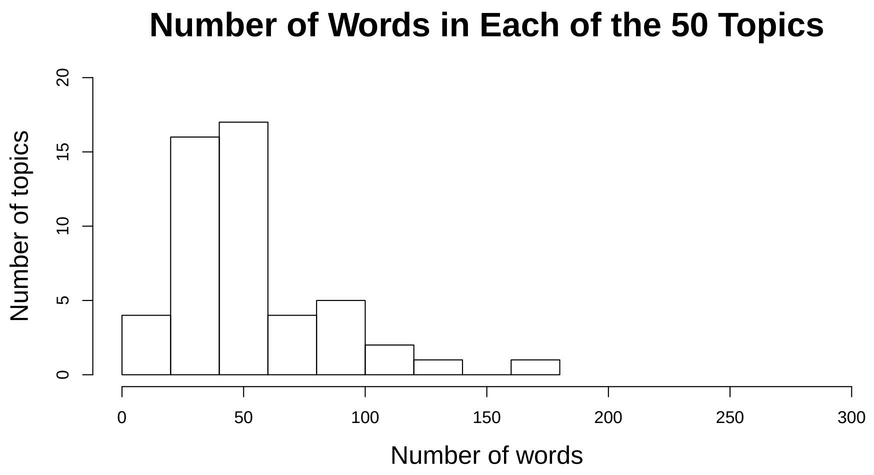


FIGURE 4.2: Histogram for number of words in each of the 50 topics

for removal.

Topic Weights

As an alternative approach, the topic weight is defined as probability of the most predominant topic in a document, compared with other topics. The motivation is that if no document has a high probability to be assigned to a certain topic, then the topic should be removed. In plain English, a topic should exist only when at least one document is "loyal" to it. In mathematical terms, topic weights for each topic i are computed by

$$\mathrm{max}_k P(\text{topic } i|\text{ document } k,\ \text{data }) \tag{4.2}$$

Topic weights are called "column max" or "max probability". Since the topic assignment matrix (dimension: documents$\times$topics) records all posterior probability $P(\text{topic } i|\text{ document } k,\ \text{data })$ values, max_k returns the max value of each column. Note that column max values actually belong to documents (rows), and the code implementation is described in Appendix B.

The results show that each topic has max probability > 0.99, regardless of the number of topics. Therefore, the topic weights cannot be distinguished from each other, so this is not a good method to use in order to winnow topics.

Number of Documents in Each Topic

A third approach considers the number of documents in each topic as a measure to determine the number of topics. Similarly to the description in the number of words in each topic, a topic should be removed if it contains very few documents. For topic assignments, a document k is assigned to a topic i if and only if $P(\text{topic } i|\text{ document } k,\ \text{data }) > 0.8$ (a threshold chosen after some exploratory analysis).

Table 4.9 shows the number of documents in each topic when $K = 2, 3, 5, 10$; Table 4.10 summarizes the distribution for 20 topics, and Table 4.11 lists the results of 50 topics. The distribution of number of documents in each topic is spread-out and right-skewed, so it is not a uniform distribution. For statistical evidence, I performed the Kolmogorov-Smirnov (KS) test in R on the distributions of 20 and 50 topics against a uniform distribution, but both p-value are less than 0.05. However, since most topics contain few documents, I have insufficient information to determine which topic(s) to remove.

Table 4.9: Number of **documents** in each topic for various pre-set numbers of topics

Number of Topics	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
2	42	45								
3	27	26	35							
5	16	18	19	16	20					
10	11	8	9	12	8	9	13	9	6	7

Table 4.10: Number of **documents** for each of the 20 topics

# of documents in the topic	3	4	5	6	7	8
# of topics of this type	3	7	6	1	2	1

Table 4.11: Number of **documents** for each of the 50 topics

# of documents in the topic	1	2	3	4
# of topics of this type	15	25	7	3

Discussion

These findings indicate that LDA tends to "spread out" word and topic assignments, making it difficult to identify and drop unnecessary topics. One can argue that the synthetic dataset is small, but I do not think using a larger corpus will substantially change these conclusions. In my opinion, more complicated topic models are needed to determine the appropriate number of topics for a given dataset. For example, a penalty function can be added to avoid generating too many topics, just like the case of ridge regression and lasso (least absolute shrinkage and selection operator).

Moreover, the "appropriate" number of topics depends on how coarse or how refined each topic need to be, and no unique best answer exists. In a political blog post dataset, topics can be as general as "sensational crimes", and they can also be as specific as "racism on Trayvon Martin shooting". An alternative is to model the text topics using dendrograms (tree diagrams), so the smaller topics are nested within the larger ones.

4.4 Determine the Cutoff Frequency

In addition to stop words, rare words in the corpus also interfere with the topic modeling results, so these words should be removed. Nevertheless, most researchers set the minimum cutoff frequency based on empirical evidence or exploratory data analysis, without using any mathematical principle. Thus, I explored various minimum cutoff frequency values for latent Dirichlet allocation (LDA) to see how the topic modeling results change.

4.4.1 Setup

The minimum cutoff frequency (abbreviated as "min freq") is the minimum number of occurrences for each word to remain in the text dataset for topic modeling. If the min freq is 1, the whole corpus is retained. If the min freq is 2, all words that appear only once in the corpus are removed. If the min freq is m, all words that appear at most $m - 1$ times are removed from the dataset.

To control for the number of topics, I use the "far" synthetic dataset, whose documents contain only statistics and dance related contents. I inspected the documents in the "far" corpus and ensured that the topics can be distinguished by human eye. The "similar" dataset (statistics+calculus) is not included because I would like to make the distance between the two topics large.

4.4.2 Error Definition for Topic Assignment Vectors

The topic assignment vectors for each document are used to quantify the "distance" between the LDA results and the true topic proportions of each document. The topic model error for each document is defined as the absolute value of the difference between the first components of the assignment vector and the true proportion. Then the error of the topic model is the average of error across all documents in the corpus.

For example, the true topic proportion of a given document is $(0.5, 0.5)$, and the output topic assignment vector is $(0.6, 0.4)$. The error is $|0.5 - 0.6| = 0.1$. In an extreme case, the ground truth $(0.5, 0.5)$ is estimated as $(0.0, 1.0)$, and the error is 0.5, which is high. Since only two topics (statistics and dance) exist in the "far" corpus, it is equivalent to examine the first component and the second one of the topic assignment vector.

Nevertheless, since LDA does not indicate which topic is the Statistics one, the ground truth generated from the corpus (filename list) may have switched labels. For example, the true proportion is $(0.3, 0.7)$, while the estimate is $(0.7, 0.3)$, resulting in an error of 0.4, which is incorrect.

Fortunately, this can be solved by choosing the minimum error for the two orderings. For every document, I computed the difference between the first component and the first component of the two topic vectors, and also the difference between the first component and the second component. Then the averages are calculated for the 50 documents, and the minimum value is selected.

4.4.3 Results and Discussion

The results in Table 4.12 underscore how extremely inaccurate LDA is when only stop words are removed. All mean absolute errors are more than 40%, no matter whether the minimum cutoff is 2, 3, 5, 10, or 15.

Table 4.12: The mean error of topic assignment vectors for different cutoff frequencies

Min Freq	1	2	3	5	10	15
Mean Error	46.59%	46.48%	41.92%	48.22%	46.00%	45.35%

I identified two potential reasons:

1. Too few words exist in each document. After I removed stop words and performed n-gramming on the corpus, each document contains only 275.72 words on average, compared with the original 500 words.

2. The minimum cutoff frequency does not matter much in LDA.

4.5 Conclusion

"Negative findings are a valuable component of the scientific literature because they force us to critically evaluate and validate our current thinking, and fundamentally move us towards unabridged science." (Matosin et al., 2014)

Currently, the pre-set number of topics in LDA is empirically determined, and it plays an important role in the topic model because LDA tends to "spread out" topic assignments, so that all topics have about equal representation in the corpus.

Neither dropping unneeded topics nor determining the minimum cutoff frequency is a success, but the methods can be helpful to future researchers who want to explore more deeply the uncertainty quantification of topic models.

5

Exploration of Selected Text Models

5.1 Introduction

Text mining is an extremely broad field with lots of extensions and applications (Hotho et al., 2005; Srivastava and Sahami, 2009). In addition to text data cleaning and topic model quantification methodologies in the previous chapters, I also explored five text model applications, which are built on top of existing methods and can be useful in future research.

The first one is the supervised latent Dirichlet allocation (Mcauliffe and Blei, 2007), a.k.a. sLDA or supervised LDA. This topic model combines text and numeric data; the numbers serve as the response or the dependent variable. In comparison, the ordinary LDA (latent Dirichlet allocation) simply assigns documents to topic probability vectors without any pre-known labels, so it is unsupervised.

The second text application is on surveys, based on my summer internships at the Federal Housing Finance Agency, in which I worked on the National Mortgage Survey Origination project. The mortgage survey questionnaire contains an "Other (specify)" blank for the survey participants to write a text response, and these text responses can be utilized to understand what the respondents actually mean.

The third text model extension is sentiment analysis, in which the polarity of each document is determined. Previous work (Au, 2014) used the Travyon Martin dataset, so I implemented similar methods on the Benghazi dataset to see whether polarity methods are corpus-independent. Both datasets are described in Section 2.2. I also generated word clouds for positive, negative, and netural terms in the Benghazi dataset, and the results are reasonable.

The fourth one is a joint model for sentiment and topics (Lin and He, 2009). This is an unsupervised model which simultaneously detects topic and sentiment from the documents; most importantly, the algorithm is relatively easy and straightforward to implement from scratch in R. The current biggest drawback is the slow computation speed due to the inevitable nested `for` loops.

Last but not least, the infinite Gaussian mixture model (Rasmussen, 1999) allows the number of topics to be automatically determined by the text data. By initializing the number of topics to be large, I can get the number of topics by choosing the components whose probabilities add up to almost 1.

5.2 Supervised Latent Dirichlet Allocation (sLDA)

Text data with feedback is ubiquitous, e.g. movie ratings and Youtube comments. This motivated me to use the supervised latent Dirichlet allocation (sLDA), in order to analyze a survey with both numerical and text responses. In the employee satisfaction dataset (described in Section 2.2.3), I should make good use of the the text comments associated with the ratings for two reasons.

First, when employees talk about their work in real life, they do not rate their satisfaction on a 1-10 scale unless being specifically asked to do so. Therefore, if the rating given the comment is predicted, the satisfaction level of that employee can be estimated. This can extend the use of free-form responses on surveys, or perhaps increase response rates.

Second, exploiting text data helps with error correction – the data contain errors because some people were confused by the rating scale. In fact, one employee wrote "Love my work - very varied." but rated his/her company 1 (least satisfied), and the rating should be corrected to 10 (most satisfied). If I did not read the text comment, I would not know that this employee actually loves his/her work.

In survey data analysis, it is implicitly assumed that everyone rates on the same scale. In reality, the same experience can result in different feelings for different people. Even with the same extent of feeling, one person may rate it 9 out of 10, while another may rate it only 7.

5.2.1 Data Generative Process

The sLDA is similar to LDA, but sLDA regards the numbers as response variables. The first two steps of sLDA are the same as LDA – The algorithm draws topics from a Dirichlet distribution as the prior, then updates the probabilities by using the words in the documents. Finally, sLDA draws the numerical response variable for each document from a normal distribution using the posterior topic assignments.

For notation, assume K topics are denoted as a set of vectors $\beta_{1:K}$, the response variable $y \in \mathbb{R}$ (the whole real line), and η, σ^2 are pre-set constants for the normal distribution.

The data generative process is illustrated in Figure 5.1 and defined as below:

- Draw topic proportions $\theta|\alpha \sim \text{Dirichlet}(\alpha)$.
- For each word w_n
  - Draw topic assignment $z_n|\theta \sim \text{Mult}(\theta)$
  - Draw word $w_n|z_n, \beta_{1:K} \sim \text{Mult}(\beta_{z_n})$.
- Draw response variable $y|z_{1:N}, \eta, \sigma^2 \sim N(\eta\bar{z}, \sigma^2)$
  - Define $\bar{z} = (1/N)\sum_{n=1}^{N} z_n$.
- Transform the Gaussian response variable $Y_d \in \mathbb{R}$ (for each document d) to J categories
  - $-\infty = \tau_0 \leqslant \tau_1 \leqslant \tau_2 \leqslant \cdots \leqslant \tau_{J-1} \leqslant \tau_J = \infty$

- In the jth category, $\tau_{j-1} \leqslant Y \leqslant \tau_j$.

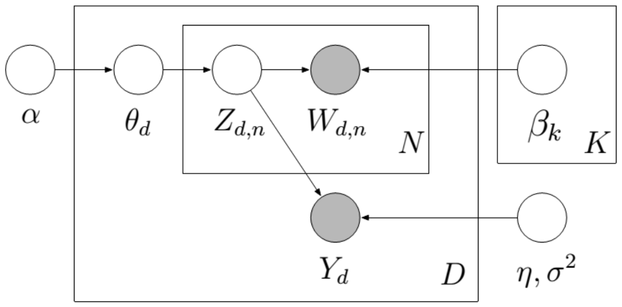


FIGURE 5.1: Plate Diagram for sLDA
(reproduced from (Mcauliffe and Blei, 2007))

Note that the joint posterior $p(\theta, z_{1:N}|w_{1:N}, y, \alpha, \beta_{1:K}, \eta, \sigma^2)$ has an intractable likelihood, so variational inference is performed and the parameters are estimated by expectation-maximization (EM) (Mcauliffe and Blei, 2007). A common problem of EM is convergence to a local maximum. Nevertheless, as described in Section 3.2.2, LDA-like methods, when combined with variational inference, generally ensure EM to converge to the global maximum (Awasthi and Risteski, 2015).

5.2.2 sLDA Prediction

The goal of prediction is to get the expected response value, given a new document (text response) $w_{1:N}$ and a fitted model $\{\alpha, \beta_{1:K}, \eta, \sigma^2\}$.

In the data generation process, the response variable y follows a normal distribution:

$$y|z_{1:N}, \eta, \sigma^2 \sim N(\eta\bar{z}, \sigma^2). \tag{5.1}$$

Then the expectation of y is calculated as

$$E(y|w_{1:N}, \alpha, \beta_{1:K}, \eta, \sigma^2) = \eta E[\bar{z}|w_{1:N}, \alpha, \beta_{1:K}] \approx \eta E_q[\bar{z}] = \eta\bar{\psi}, \tag{5.2}$$

where $\bar{\psi} = (1/N)\sum_{n=1}^{N} \psi_n$, $\bar{z} = (1/N)\sum_{n=1}^{N} z_n$, and q is a variational distribution.

Since the range of y is the whole real line, y can be mapped to a value in $(0, 1)$ using the cumulative distribution function through the probability integral transformation. In this way, the y is transformed into a probability, thus it can be used in generalized linear models such as logistic regression.

5.2.3 Implementation

The supervised latent Dirichlet allocation (sLDA) is implemented using the R package `lda`, and sample code is available in `demo(slda)`. The demo code starts with expectation-maximization using the function `slda.em`, then creates credible intervals for each topic, visualizes the intervals, and plots the predictive distributions of each rating.

The employee satisfaction dataset contains 530 employee ratings about their work, with comments to free response questions. The ratings range from 1 (least satisfied) to 10 (most satisfied), and the histogram is shown in Figure 5.2. Note that the peak occurs in 6-8, so the histogram is left-skewed.

In the `slda` code, the parameters are set as $\alpha = 1, \eta = 0.1, \sigma^2 = 0.25$. For the number of iterations, I set 10 for E-step and 4 for M-step. Moreover, the number of topics is pre-set to 10 to reflect the 10 ratings, although the number of topics and the number of discrete ratings do not need to be the same.

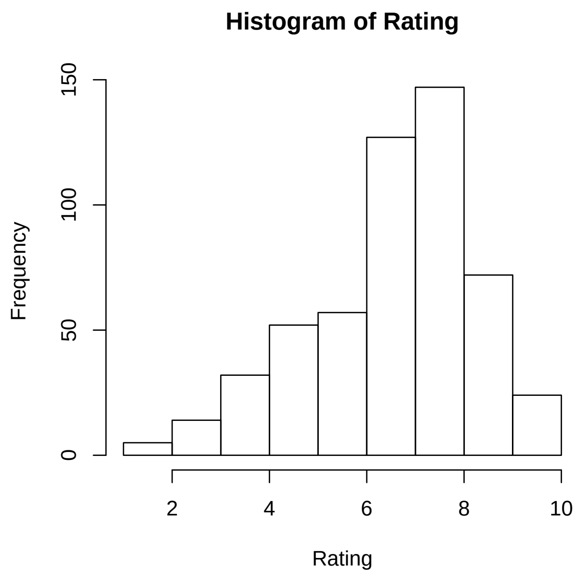


FIGURE 5.2: Histogram of ratings in the employee satisfaction dataset

5.2.4 Topic Modeling Results

The implementation results contain three items: a table for `top.topic.words`, a graph of credible intervals for the scores associated with each topic, and a density plot of predictive distribution for each rating.

Table 5.1 lists the tokenized words for each topic and rating. The topic modeling results are generated from `slda.em`, and the words are selected by the function `top.topic.words`. Lower ratings are associated with topics such as lack of interest and work-life balance, while higher ratings are associated with positive words such as "challenge" and "opportunity".

Ratings 2, 5, and 7 all contain the word "enjoy" because this word is widely used

Table 5.1: Selected words (tokens) for each topic and rating

Rating (Topic)	Selected Words
10	challeng opportun learn dai new
9	team work great staff make
8	role feel current perform career
7	job work enjoi too project
6	manag work happi last month
5	work get enjoi chang need
4	compani project skill resourc engin
3	life balanc compani work peopl
2	lack interest differ enjoi requir
1	time hour lot week take

(77 out of 530 ratings) in describing people's satisfaction extent of the company. Therefore, "enjoy" is used in a wide range of ratings and contain limited useful information, so word distinctivity (described in Section 3.3.2) is a better approach than the standard `top.topic.words`.

An example of a rating 10 comment is:

I **enjoy** the work I do and the people I work with. My management team are responsive and caring and help me be flexible with my work/life balance. I love my job.

An example of a rating 4 comment is:

The current work volume is extremely high, with inadequate resources to share the load. My team is full of great individuals that I **enjoy** working with. But the work load is wearing down physically, emotionally, not a healthy environment at all. Long days sleepless nights.

5.2.5 *Rating Estimation Results*

Figure 5.3 plots the 68% credible intervals (point estimate ± standard error) of the rating for each topic. For example, given the topic "challeng opportun learn dai new", the credible interval of the score is approximately between 8 and 9. The t-values (interval thickness) determine statistical significance. The lighter the color of the interval, the higher the estimate score is.

In Figure 5.3, the estimated ratings are mostly between 6 and 8, because this is the interval to which the majority of ratings in the data belong. The interval for the topic with rating 1 is wider than the others because only two responses in the dataset have rating 1.

Furthermore, Figure 5.4 plots the probability density of the predictive distribution for each rating using in-sample prediction, and the predicted ratings are still mostly between 6 and 8.

5.2.6 *Discussion*

sLDA can predict ratings based on the text response, and analyzing the employee satisfaction data is a starting point. The work can be extended to measure the predictive power of word sequences, which is also related to the quantification of

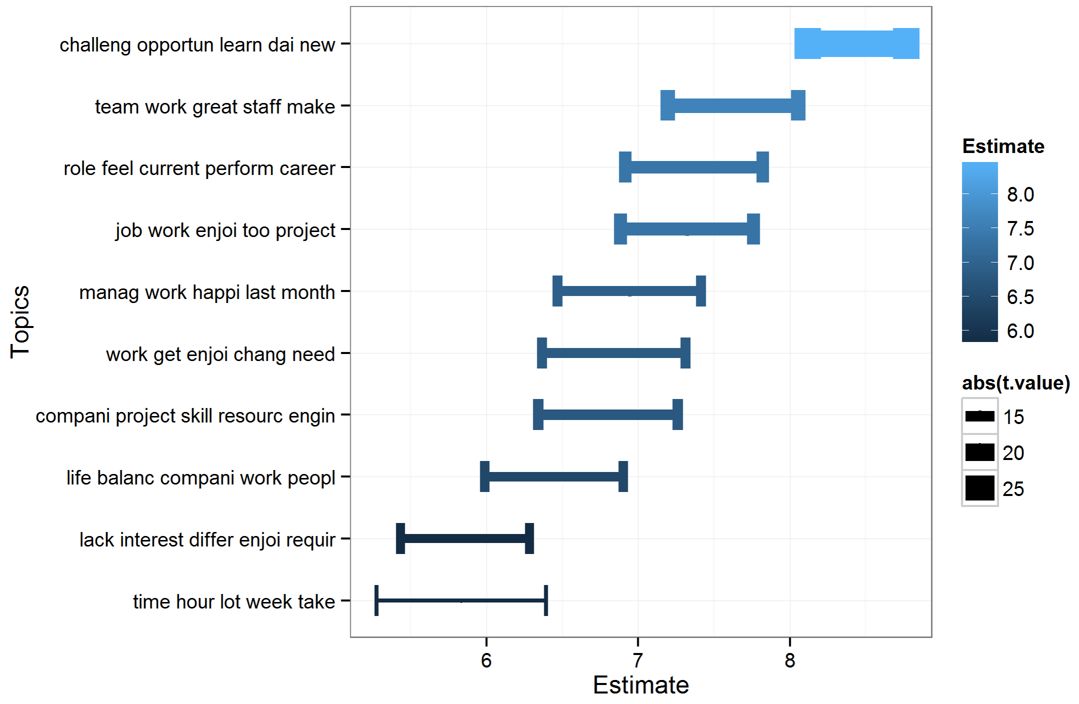


FIGURE 5.3: 68% credible intervals for each topic

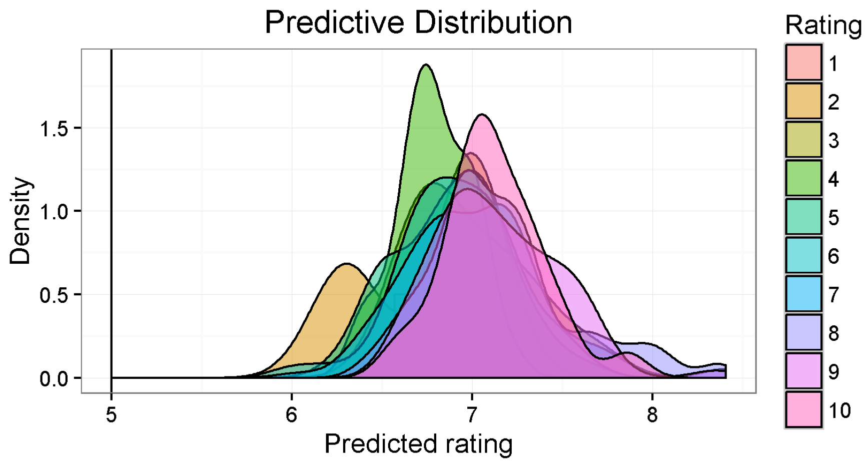


FIGURE 5.4: Predictive distribution for each rating

improvement through n-gramming (described in Chapter 3). Another extension is to explore the linear combination of multiple topics in a single comment, since some topics are related to each other.

5.3 Text Mining in Survey Data

Most questionnaires contain free response fields, and it is useful to analyze the open answers in text data (Yamanishi and Li, 2002; Hirasawa et al., 2007). One text mining application is to use the text in “Other (specify)” portion of a questionnaire, because this may reveal important information. The National Mortgage Survey Originations project at the Federal Housing Finance Agency samples 6000 borrowers per quarter who had taken out or co-signed for a mortgage loan within the past 18 months (Federal Housing Finance Agency, 2016), including any refinancing of a previous mortgage. I will discuss some examples of how the text answers can be used in the next few paragraphs, and the survey will be referred as “the mortgage survey questionnaire.”

First, the questionnaire contains the categorical question: Any unpleasant “surprises” at loan closing? A few respondents wrote “delayed closing” or “additional paperwork” in the “Other (specify)” blank, so a new option should be added if a significant amount of people answered this. Another respondent wrote “wanted 30-year term but got 15-year term”, and this response should be recoded to the existing option “different loan terms”.

For a second example, another question asks how the respondent acquired the property. The options include “purchased an existing home” and “purchased land and built a house”, but some respondents answered “refinance” in the “Other (specify)”. It is obvious that they answered a different question – the primary purpose for their most recent mortgage (e.g. house purchase, refinance, mortgage modification), so the answer to “How did you acquire this property” reflects ambiguity in the question.

Since manually inspecting the text values is tedious, topic modeling that automatically clusters the text records is a better approach. For instance, any text in the second example which contains “refinance” should be automatically flagged as “missing” in that question. For text data cleaning, a challenge is that n-gramming (discussed in Section 2.5) may not work for short text responses, so computational linguistics can be useful to identify key words. For government agencies that are concerned about using open-source software such as R, an alternative is to use the SAS Text Miner to perform text analysis.

5.4 Sentiment Analysis

Sentiment analysis is a broad field, and a sea of literature exists on the Internet (Pang et al., 2002; Mei et al., 2007; Kouloumpis et al., 2011; Taddy, 2013). Supervised datasets such as movie reviews are extremely common in the implementation of sentiment analysis methods (Vishwanathan, 2010; Goyal and Parulekar, 2015; Thet et al., 2010; Zhuang et al., 2006) because the ratings serve as the “ground truth” of the text mining. However, each movie review comprises at most several sentences, so the text is short and not difficult to analyze.

Therefore, I explore the subset of the 2012 political blog database for which the posts include the word "Benghazi". For pre-processing, stop words are removed first, and the Benghazi corpus is tokenized, but not n-grammed.

5.4.1 Methodology

The polarity of each document is determined by the relative number of positive and negative terms, and this method is similar to the one described in (Au, 2014). Moreover, positive and negative terms can be determined by how often a word is used in documents. The methodology is divided into several steps:

1. Use the R package `tm.plugin.sentiment` and the dataset `dic_gi` to get the positive and negative word lists. In comparison, (Au, 2014) used the AFINN list (Nielsen, 2011) with each word rated on a $[-5, 5]$ scale.

2. Remove some negative terms from the word list because they are about the incident itself, so they do not necessarily mean something negative. The removed terms (tokens) are: arrest, assault, bullet, confront, crime, dead, death, defens, die, gun, fatal, ignor, injuri, killer, manslaught, murder, punch, shoot, shot.

3. Create positive and negative term matrices from the sparse tf-idf matrix of the Benghazi dataset, which is generated by using the R package `tm`.

4. For each document, calculate the polarity score as $\frac{(p-n)}{(p+n)} \in [-1, 1]$, where p is the number of positive words and n is the number of negative words in that

document.

5. Plot the histogram of document polarity in Figure 5.5. The threshold is set at ± 0.15, so positive documents have scores > 0.15; negative documents have scores < -0.15, and neutral documents have scores in between.

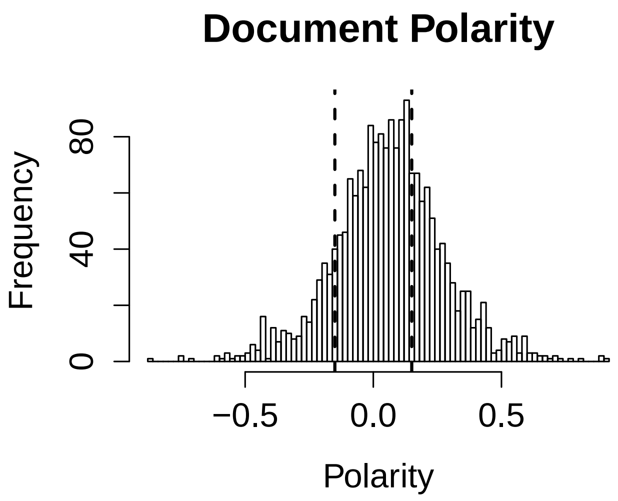


FIGURE 5.5: Polarity of documents in the Benghazi dataset

6. A two-proportion z-test is implemented to identify frequently used terms in positive/negative documents. For each token, let $\hat{p}_+$ be the proportion of positive documents containing it, $\hat{p}_-$ be the proportion of negative documents containing it, and $\hat{p}$ be the pooled proportion of documents containing it. The null hypothesis is $p_+ = p_-$; the alternative hypothesis is $p_+ \neq p_-$. The z-statistic is calculated as below:

$$z = \frac{\hat{p}_+ - \hat{p}_-}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_+} + \frac{1}{n_-})}}, \tag{5.3}$$

where n_+ and n_- denote the number of positive and negative documents containing that token, respectively.

7. Define positive terms as the ones with $z > 1.96$, negative terms as the ones with $z < -1.96$, and neutral terms as the remaining terms, i.e. $z \in [-1.96, 1.96]$.

8. Figures 5.6, 5.7, and 5.8 show the word clouds for positive, negative, and neutral terms, with minimum word frequency 200. According to Figure 5.5, there are more positive documents than negative documents, so the word cloud of positive terms is larger than the one of negative terms.

An example of a positive blog post looks like:

"This blog is looking for wisdom, to have and to share. It is also looking for other rare character traits like good humor, courage, and honor. It is not an easy road, because all of us fall short. But God is love, forgiveness and grace. Those who believe in Him and repent of their sins have the promise of His Holy Spirit to guide us and show us the Way."

An example of a negative blog post looks like:

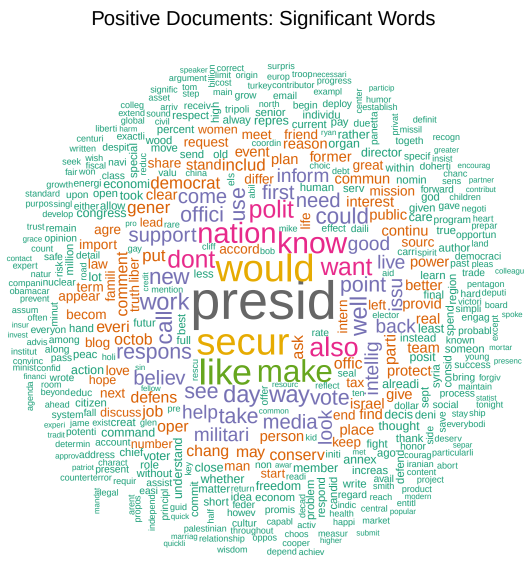


Figure 5.6: Positive word cloud for the Benghazi dataset

> "There's more indication of that today with this ugly situation described by Doctors Without Borders: Doctors Without Borders has suspended its work in prisons in the Libyan city of Misrata because it said torture was so rampant that some detainees were brought for care only to make them fit for further interrogation, the group said Thursday."

5.4.2 Discussion

This is a preliminary exploration of sentiment analysis for the Benghazi corpus, and the methodology can be improved through a number of ways. First, the threshold of polarity is arbitrarily defined as ±0.15, and the word clouds will be much smaller if

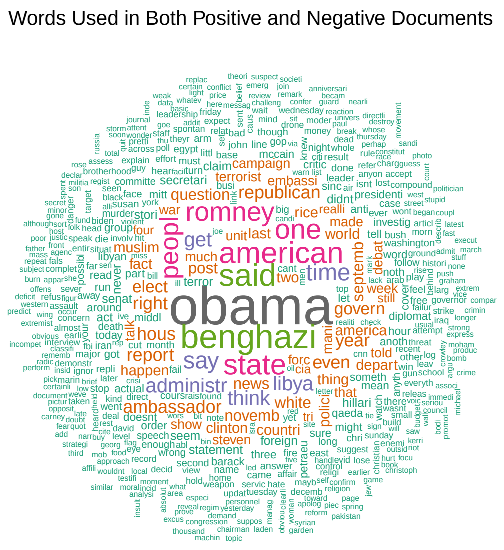


FIGURE 5.7: Neutral word clouds for the Benghazi dataset

the threshold is changed to ±0.5. Accordingly, a more sophisticated process is needed to determine what is the best document classification. An alternative method to obtain the "baseline truth" is asking people to rate the blog posts from 1 (negative) to 10 (positive), then perform cross validation. However, getting humans to read through each blog post is costly and may not be worthwhile for this research.

Furthermore, better data visualizations should also be explored. Word clouds are attractive to the human eye, but the size of each word does not accurately reflect the frequency of occurrence (Hein, 2012). For example, longer words take up more space than shorter words, so the former seems to have higher frequency based on the word cloud.

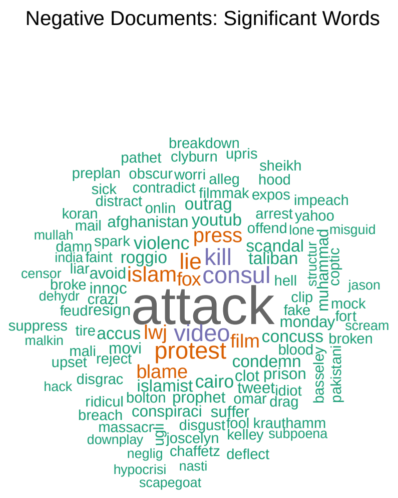


FIGURE 5.8: Negative word clouds for the Benghazi dataset

5.5 Joint Sentiment/Topic Model (JST)

Most work done in the text mining field has focused on either finding topics (Blei et al., 2003; Wallach, 2006; Blei, 2012; Griffiths and Steyvers, 2004; Titov and McDonald, 2008a) or sentiment analysis (Pang et al., 2002; Taddy, 2013; Melville et al., 2009; Mohammad et al., 2013). However, topic detection and sentiment analysis are related because the polarity of words depends on topics. For example, the word "black" is neutral (a color) in an art topic, but negative in electrical power failure or drunkenness ("black out"). Therefore, some existing literature attempted to simultaneously model topics and sentiments (Titov and McDonald, 2008b; Mei et al., 2007; Lin and He, 2009), and I decided to explore the joint sentiment/topic model

(JST) (Lin and He, 2009).

5.5.1 Algorithm Description

The JST (joint sentiment/topic) model (Lin and He, 2009) is a hierarchical Bayesian data generative process, and it is quoted as below:

- For each document d, choose a distribution $\pi_d \sim$ Dirichlet(γ).
- For each sentiment label l under document d, choose a distribution $\theta_{d,l} \sim$ Dirichlet(α).
- For each word w_i in document d
  - Choose a sentiment label $l_i \sim \pi_d$
  - Choose a topic $z_i \sim \theta_{d,l_i}$
  - Choose a word w_i from the distribution over words defined by the topic z_i and sentiment label l_i, $\phi_{z_i}^{l_i}$.

For the Gibbs sampler, the matrices for words, topic, and (sentiment) labels have sizes:

- Words $\Phi : V \times T \times S$
- Topics $\Theta : T \times S \times D$
- Labels $\Pi : S \times D$

V is the number of distinct terms in the vocabulary; T is the total number of topics; S is the number of distinct sentiment labels, and D is the number of documents in the corpus.

Hence the pseudo code for the JST Gibbs sampling process is:

```
1. Initialize prior parameters $\alpha, \beta, \gamma$.
2. Initialize word matrix $\Phi$, topic matrix $\Theta$, and sentiment label matrix $\Pi$.
3. for $M$ Gibbs sampling iterations:
    • for each document $d$
        – for each word $i$ in document $d$
            * $\phi_{i,j,k} = \frac{N_{i,j,k} + \beta}{N_{j,k} + V\beta}$
            * $\theta_{j,k,d} = \frac{N_{j,k,d} + \alpha}{N_{k,d} + T\alpha}$
            * $\pi_{k,d} = \frac{N_{k,d} + \gamma}{N_d + S\gamma}$
            * Sample a sentiment label $l_i \sim \pi_d$
            * Sample a topic $z_i \sim \theta_{d,l_i}$
            * Update the word count: $N_{i,j,k} \leftarrow N_{i,j,k} + 1$.
        – Update $N_{j,k,d}$ using $z$ and $l$.
4. Output word matrix $\Phi$, topic matrix $\Theta$, and sentiment label matrix $\Pi$.
```

5.5.2 Implementation

I implemented the JST (joint sentiment/topic) model in R from scratch, and the code is in Appendix A. I created five short documents as the toy dataset, and they are:

- Document 1: The quick brown fox is the winner
- Document 2: I love you my dog Jimmy
- Document 3: I hate this guy because he stole money from my parents
- Document 4: The economics is very bad recently
- Document 5: The weather is so good and warm

To run JST on this collection of documents, I initialized the parameters below:

- $\alpha = (1, \cdots, 1)_T$
- $\beta = 0.01$
- $\gamma = (2, 4)$ (2 for positive, 4 for negative sentiment labels)
- $V = 30$ (total words in the vocabulary set)
- $D = 5$ (total number of documents)
- $S = 2$ (total number of sentiment labels – positive and negative)

- $T = 3$ (total number of topics)

It took about 8.5 seconds to run 1000 Gibbs iterations.

Table 5.2 is the output of Θ, and Table 5.3 is the output of Π. Note that each column for each document in Θ sums to 1, while each column in Π is not normalized.

Table 5.2: Topics $\Theta : T \times S \times D$ ($3 \times 2 \times 5$) from the toy dataset

Document 1	Label 1	Label 2	Document 2	Label 1	Label 2
Topic 1	0.25	0.50	Topic 1	0.50	0.25
Topic 2	0.37	0.25	Topic 2	0.25	0.37
Topic 3	0.38	0.25	Topic 3	0.25	0.38
Document 3	Label 1	Label 2	Document 4	Label 1	Label 2
Topic 1	0.50	0.61	Topic 1	0.67	0.33
Topic 2	0.25	0.08	Topic 2	0.16	0.50
Topic 3	0.25	0.31	Topic 3	0.17	0.17
Document 5	Label 1	Label 2			
Topic 1	0.16	0.28			
Topic 2	0.17	0.43			
Topic 3	0.67	0.29			

Table 5.3: Labels $\Pi : S \times D$ (2×5) from the toy dataset

Documents	1	2	3	4	5
Label 1	0.700	0.300	0.200	0.500	0.454
Label 2	0.35	0.643	0.737	0.500	0.533

Next, I scaled the code to the Trayvon Martin dataset, and the data parameters are defined as below:

- $V = 2548$ (total words in the vocabulary set)

- $D = 450$ (total number of documents)

- $S = 2$ (total number of sentiment labels – positive and negative)

- $T = 3$ (total number of topics)

However, 10 Gibbs sampling iterations take approximately 30 minutes to run, so 100 iterations will take 50 hours, which is extremely slow. The computational complexity for each iteration is $O(DVTS) = O(n^4)$, and unfortunately `for` loops are inevitable. Asymptotically, this is slower than a quadratic time algorithm.

In conclusion, it is nearly impossible to code up a topic model completely in R because it is too slow. In fact, most functions in R packages run much faster because C/C++ code is called.

5.6 Infinite Gaussian Mixture Model

Topic modeling of text is a type of clustering because this method classifies text documents into topics. The number of topics (clusters) is often pre-defined for convenience of implementation. When analyzing the Travyon Martin dataset, I also followed previous researchers (Soriano et al., 2013; Au, 2014) to pre-set the number of topics to 5, because my research focus is on text data cleaning and topic modeling methodologies.

However, it is usually difficult to estimate the number of clusters required, so people either arbitrarily set the number or rely on expert knowledge – both methods are subjective. Obviously, it is better to allow the text data to automatically determine the best number of topics.

Therefore, the infinite Gaussian mixture model (Rasmussen, 1999) starts with an infinite number of components, so this removes the need of prescribing a fixed number of Gaussian classes. Then a "good" number can be determined by adding up the probabilities of the components so that the sum is close to 1.

In mathematical terms, a finite Gaussian mixture model can be defined as

$$p(y|\mu_1, \cdots, \mu_k, s_1, \cdots, s_k, \pi_1, \cdots, \pi_k) = \sum_{j=1}^{k} \pi_j N(\mu_j, s_j^{-1}). \tag{5.4}$$

Each component j is a normal distribution with mean μ_j and precision s_j (inverse of variance). The training dataset is $y = \{y_1, \cdots, y_n\}$, containing n datapoints. Each mixing proportion π_j are the expected probability of a datapoint to be generated from the Gaussian class j; note that $\pi_j > 0$ and $\sum \pi_j = 1$.

The mixing proportions π_j are assigned to a symmetric Dirichlet prior.

$$p(\pi_1, \cdots, \pi_k|\alpha) \sim \text{Dirichlet}(\frac{\alpha}{k}, \cdots, \frac{\alpha}{k}). \tag{5.5}$$

Then set $k \to \infty$, so Equation 5.5 becomes a Dirichlet process.

In practice, we may set k to be a large number (e.g. $2^{31} - 1 = 2,147,483,647$). Since it is extremely unlikely that any text dataset will contain so many topics, this is almost equivalent to $k \to \infty$.

The concept of infinite latent features and the Dirichlet process can be applied to complicated models, such as hierarchical Dirichlet process (Teh et al., 2012) and Bayesian hierarchical clustering (Heller and Ghahramani, 2005). For text model applications, Yin and Wang (2014) implemented a Dirichlet multinomial mixture model for short text clustering.

5.7 Conclusion

Since text mining is a new and emerging field, many analysis methods are under development. Text mining can be extended to lots of applications, and the ones described in this chapter are just a few.

Appendix A

Joint Sentiment/Topic Model Code

This is the code for the joint sentiment/topic model (JST) (Lin and He, 2009).

```
# File: Coding_JST_v4_thesis_appendix.R


# Goal: Implement the JST (Joint Sentiment Topic model)
# Paper: Joint Sentiment/Topic Model for Sentiment Analysis (2009)


# Citation: Lin, Chenghua, and Yulan He.
# "Joint sentiment/topic model for sentiment analysis."
# Proceedings of the 18th ACM conference on Information
# and knowledge management. ACM, 2009.


# Fourth version, toy dataset to verify correctness
# WITH PRIOR INFORMATION ADDED
# Code updated by Christine Chai on November 4, 2016

# Sample documents
doc1 = c("the","quick","brown","fox","is","the","winner")
doc2 = c("i","love","you","my","dog","jimmy")
doc3 = c("i","hate","this","guy","because","he","stole",
"money","from","my","parents")
doc4 = c("the","economics","is","very","bad","recently")
doc5 = c("the","weather","is","so","good","and","warm")

docs = list(doc1,doc2,doc3,doc4,doc5)
vocab = sort(unique(c(doc1,doc2,doc3,doc4,doc5)))

positive = c("good","love","quick","warm","winner")
negative = c("bad","hate","stole")

V = length(vocab) # total vocabulary 30
D = length(docs)  # total number of documents
S = 2   # total number of sentiment labels (positive,negative)
T = 3   # total number of topics
M = 1000 # total number of iterations in Gibbs sampling

# Convert words into numbers
# First version
# doc1.num = c()
# for (ii in 1:length(doc1)){
#  doc1.num[ii] = match(doc1[ii],vocab) # match(x, table)
# }
# doc1.num = c()
# Second version
# doc1.num = sapply(doc1,match,table=vocab)
docs.num = list(rep(-1,D))
for (dd in 1:D){
  docs.num[[dd]] = sapply(docs[[dd]],match,table=vocab)
}

# Matrices to store the sampling results (should be numbers)
# Phi:   V*T*S -> predictive distribution of words  (ii)
# Theta: T*S*D -> predictive distribution of topics (jj)
# Pi:    S*D   -> predictive distribution of labels (kk)
Phi   = array(0,dim=c(V,T,S))
Theta = array(0,dim=c(T,S,D))
Pi    = array(0,dim=c(S,D))

# -------------------------------------------------------------

set.seed(21)
```

```
# Initialization

library(gtools)

alpha = rep(1,T)

beta = 0.01

gamma = c(2,4) # 2 for positive, 4 for negative sentiment labels


# pi.d ~ dirichlet(gamma)

Pi = t(rdirichlet(D,gamma)) # each column = one document


# theta.dl ~ dirichlet(alpha)

Theta = array(0,dim=c(T,S,D))

for (dd in 1:D) {

  for (kk in 1:S) {

    Theta[,kk,dd] = rdirichlet(1,alpha)

  }

}


# for each word w.i in document d

N.ijk = array(0,dim=c(V,T,S))

N.jkd = array(0,dim=c(T,S,D))

l.i = rep(0,V)

z.i = rep(0,V)
```

```
for (dd in 1:D) {
  l.i = rep(0,V)
  z.i = rep(0,V)
  for (ii in docs.num[[dd]]) {
    # Add prior information: pre-defined positive/negative words
    if (vocab[ii] %in% positive) {
      l.i[ii] = 1
    }
    else if (vocab[ii] %in% negative) {
      l.i[ii] = 2
    }
    else {
      l.i[ii] = sample(x=1:S,size=1,prob=Pi[,dd])
      # sentiment label l.i is a number
    }

    z.i[ii] = sample(x=1:T,size=1,prob=Theta[,l.i[ii],dd])
    # topic is also a number
    Phi[ii,,] = 1/V
    # Phi records the probability (uniform prior)
    N.ijk[ii,z.i[ii],l.i[ii]] = N.ijk[ii,z.i[ii],l.i[ii]] + 1
    # j*k (T*S) matrix
  }
  N.jkd[,,dd] = table(factor(z.i,levels=1:T),factor(l.i,levels=1:S))
  # j*k (T*S) matrix
  # don't care about words
}

# ----------------------------------------------------------------


# Gibbs Sampler
# phi   = array(0,dim=c(V,T,S)) # probability
# theta = array(0,dim=c(T,S,D))
# pi    = array(0,dim=c(S,D))

# Start the clock!
ptm <- proc.time()

for (mm in 1:M) {
    for (dd in 1:D) {
      l.i = rep(0,V)
      z.i = rep(0,V)
      for (ii in docs.num[[dd]]) {
        # Calculate the probability of assigning word i
        # to topic and sentiment label
        for (jj in 1:T) {
          for (kk in 1:S){
      Phi_num = N.ijk[ii,jj,kk]+beta
      Phi_denom = sum(N.ijk[,jj,kk])+V*beta
            Phi[ii,jj,kk] = Phi_num/Phi_denom


      Theta_num = N.jkd[jj,kk,dd]+alpha[jj]
      Theta_denom = sum(N.jkd[,kk,dd])+T*alpha[jj]
            Theta[jj,kk,dd] = Theta_num/Theta_denom


Pi_num = um(N.jkd[,kk,dd])+gamma[kk]
Pi_denom = sum(N.jkd[,,dd])+S*gamma[kk]
            Pi[kk,dd] = Pi_num/Pi_denom
          }
        }


        # Sample a sentiment k and a topic j
        l.i[ii] = sample(x=1:S,size=1,prob=Pi[,dd])
        # sentiment label l.i is a number
        z.i[ii] = sample(x=1:T,size=1,prob=Theta[,l.i[ii],dd])
        # topic is also a number
        N.ijk[ii,z.i[ii],l.i[ii]] = N.ijk[ii,z.i[ii],l.i[ii]] + 1
        # j*k (T*S) matrix
      }
      N.jkd[,,dd] = table(factor(z.i,levels=1:T),factor(l.i,levels=1:S))
      # j*k (T*S) matrix
    }
```

```
}

# Stop the clock
proc.time() - ptm  # 8.57 seconds elapsed

# ----------------------------------------------------------------

# How to get the results

vocab # all 30 words

# Matrices to store the sampling results (should be numbers)
# Phi:   V*T*S -> predictive distribution of words  (ii)
# Theta: T*S*D -> predictive distribution of topics (jj)
# Pi:    S*D   -> predictive distribution of labels (kk)

Phi # the probability of each word
    # sum(Phi[,1,1]) == 1
Theta # the topic vs sentiment tables for each document
      # Probability: topic1 + topic2 + topic3 = 1 per sentiment label
Pi # the polarity (sentiment) of each document
#           [,1]      [,2]      [,3]      [,4]      [,5]
# [1,] 0.7000000 0.8000000 0.5333333 0.7000000 0.2727273
# [2,] 0.3571429 0.2857143 0.4736842 0.3571429 0.6666667
```

Appendix B

Topic Weights Implementation

Sample code from the $\mathbb{R}$ package `topicmodels`:

```
library(topicmodels)
data("AssociatedPress", package = "topicmodels")


lda <- LDA(AssociatedPress[1:20,], control = list(alpha = 0.1), k = 2)
lda_inf <- posterior(lda, AssociatedPress[21:30,])
dim(lda_inf$terms) # 2 10473 (topics*words)
dim(lda_inf$topics) # 10 2 (documents*topics)
```

The number of topics (originally set to 2) can be modified to 100, so we have excessive topics.

Step-by-step guidelines to compute topic weights:

1. Use the matrix `lda_inf$topics` (documents*topics), which computes $Pr(\text{topic } i|\text{ document } k,\ \text{data })$

2. **Row max**: Highest probability of a document to be classified into that topic, compared with other **topics**

   - $P(\text{topic } i|\text{ document } k,\ \text{data })$,
     compared with $P(\text{topic } j|\text{ document } k,\ \text{data })$, topics $i \neq j$
   - $\max_i P(\text{topic } i|\text{ document } k,\ \text{data })$ for each document k
   - **Row max values actually belong to documents.**
   - So it is not good for removing unnecessary topics.

3. **Column max**: Max probability of a document to be classified into that topic, compared with other **documents**

   - $P(\text{topic } i|\text{ document } k,\ \text{data })$,
     compared with $P(\text{topic } i|\text{ document } l,\ \text{data })$, documents $k \neq l$
   - $\max_k P(\text{topic } i|\text{ document } k,\ \text{data })$ for each topic i
   - **Column max values actually belong to topics.**
   - If a topic has very low maximum posterior probability, the topic is unlikely to contain any documents.
   - Because each document is unlikely to be assigned to that topic.

4. Decision: Use **column max** to get the weights of each topic (choose max)
   Threshold examples: 0.01 / 0.1 / 0.5

Appendix C

Coverage Probabilities for Binomial Confidence Intervals

Figure C.1 compares the exact coverage probabilities of 95% confidence intervals for binomial distributions $\text{Bin}(n, p)$ with $n = 25$, and the three methods illustrated are Clopper-Pearson, Score, and Wald. $n = 25$ is selected because this is a large enough sample size for the central limit theorem (Hogg and Tanis, 1977). In fact, this is reproduced from a figure in Agresti and Kateri (2011).

The exact coverage probabilities of 95% confidence intervals are closer to 95% when $p \approx 0.5$. The Clopper-Pearson interval is an over-coverage, and the coverage probability is close to 1 when $p \approx 0$ or $p \approx 1$. At other values of p, the interval coverage probability fluctuates. On the other hand, the score interval coverage also fluctuates, but the coverage probability plummets for $p \approx 1$. Last but not least, the Wald interval performs well with a slight under-coverage at $p \approx 0.5$, and the coverage probability significantly decreases at both extreme values of p.

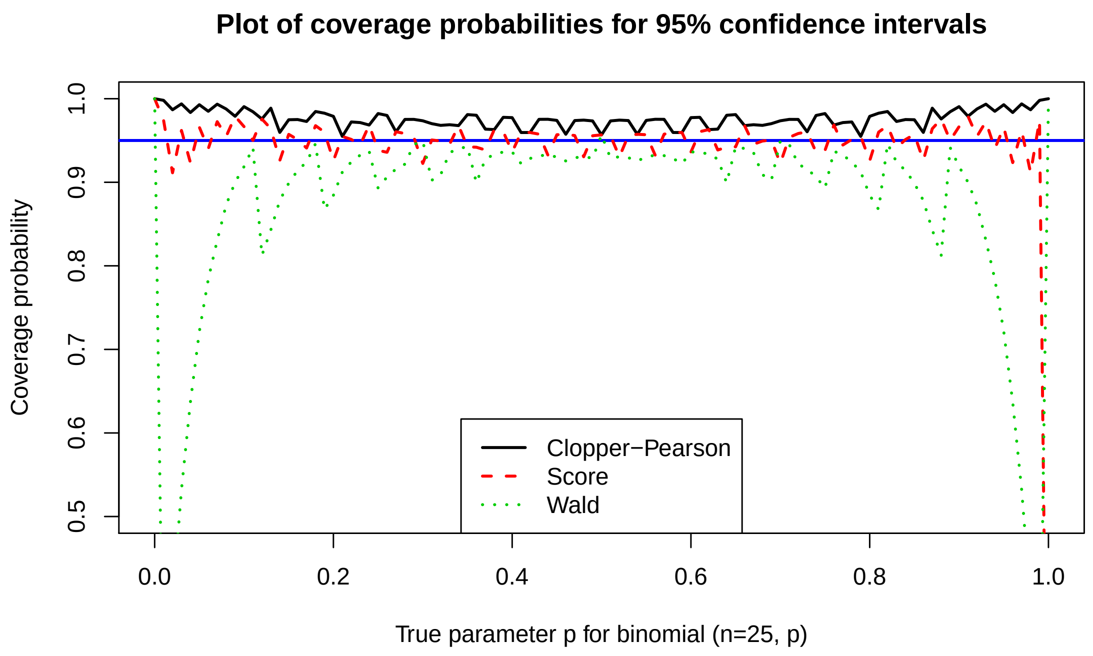


FIGURE C.1: Exact coverage probability for binomial distributions with $n = 25$

C.1 Exact Coverage Probability of Confidence Intervals (CI)

A 95% confidence interval means that we expect 95 out of 100 intervals of this kind to include the true parameter (constant but unknown). An interval can either contain or not contain the true parameter p. The formula of calculating the exact coverage probability is:

$$P(p \text{ in CI} \mid p) = \sum P(p \text{ in CI} \mid x, p) \times P(\text{probability of getting } X = x \mid p) \quad \text{(C.1)}$$

Given a binomial random variable $X \sim \text{Bin}(n, p)$, the observed outcome is $X = x$ and the estimated parameter is $\hat{p} = x/n$. The three methods of calculating $(1 - \alpha)$ confidence intervals are explained as below:

C.2 Clopper-Pearson Interval

The Clopper-Pearson interval is the intersection of $S_{\leqslant}$ and $S_{\geqslant}$, where

$$S_{\leqslant} = \{p | P(\text{Bin}(n, p) \leqslant X) > \frac{\alpha}{2}\} \text{ and } S_{\geqslant} = \{p | P(\text{Bin}(n, p) \geqslant X) > \frac{\alpha}{2}\} \tag{C.2}$$

In terms of the beta distribution, the interval becomes

$$\text{Beta}(\frac{\alpha}{2}; x, n - x + 1) < p < \text{Beta}(1 - \frac{\alpha}{2}; x + 1, n - x), \tag{C.3}$$

where $\text{Beta}(q; a, b)$ represents the qth quantile of $\text{Beta}(a, b)$.

Using the F distribution and similar notations, we get (Agresti and Kateri, 2011)

$$[1 + \frac{n - x + 1}{xF(\frac{\alpha}{2}; 2x, 2(n - x + 1))}]^{-1} < p < [1 + \frac{n - x}{(x + 1)F(1 - \frac{\alpha}{2}; 2(x + 1), 2(n - x))}]^{-1} \tag{C.4}$$

C.3 Score Interval

The score interval is a quadratic region:

$$\{p : |\hat{p} - p| \leqslant z_{1-\frac{\alpha}{2}} \sqrt{\frac{\mathbf{p}(1 - \mathbf{p})}{n}}\} \tag{C.5}$$

Solving the equation above, we get the endpoints of the confidence interval:

$$(1+\frac{z^2_{1-\frac{\alpha}{2}}}{n})^{-1}\times[\hat{p}+\frac{z^2_{1-\frac{\alpha}{2}}}{2n}\pm z_{1-\frac{\alpha}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}+\frac{z^2_{1-\frac{\alpha}{2}}}{4n^2}}] \tag{C.6}$$

The parameter p is unknown (fixed), and the estimate $\hat{p}$ is observed (random). The p and $\hat{p}$ in bold are simply highlighted; they are not vectors.

C.4 Wald Interval

This is the normal approximation for binomial confidence intervals in Section 4.2.3, written as

$$p \in (\hat{p}-z_{1-\frac{\alpha}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \hat{p}+z_{1-\frac{\alpha}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}) \tag{C.7}$$

C.5 Code

My code is revised from Horton (2012).

```
# n = 25 and X ~ Binomial(n,theta)

# Clopper-Pearson

# Score method

# Wald CI's


# set.seed(1327) # analytical solution, no need to set random seed


library(Hmisc) # for comparison purposes only
```

```
CICoverage = function(n, p, alpha=0.05){
  Cover = matrix(0, nrow=n+1, ncol=4)
  # put the results in this table/matrix
  colnames(Cover) = c("X","ClopperPearson","Score","Wald")
  Cover[,1] = 0:n

  zvalue = qnorm(1-alpha/2) # 1.96 for alpha = 0.05
  aa=1; bb=0; cc=0;
  left=10; right=10;
  for(i in 0:n){
    # Clopper-Pearson
    # CPInt = binom.test(i,n)$conf.int
    left=10; right=10;

    if (i == 0){ left = 0 }
    else {
       temp = qf(p=alpha/2,df1=2*i,df2=2*(n-i+1))
       left = (1 + (n-i+1)/(i*temp))^(-1)
    }

    if (i == n){ right = 1 }
    else {
       temp = qf(p=1-alpha/2,df1=2*(i+1),df2=2*(n-i))
       right = (1 + (n-i)/((i+1)*temp))^(-1)
    }

    CPInt = c(left,right)

    # Score method
    # SInt  = prop.test(i,n)$conf.int
    phat = i/n
    aa=1; bb=0; cc=0;
    aa = 1 + (zvalue^2)/n
    bb = -(2*phat + (zvalue^2)/n)
    cc = phat^2
    delta = sqrt(bb^2 - 4*aa*cc)
    SInt = (-bb + c(-delta,delta))/(2*aa)
    # Quadratic equation:
    # ax^2 + bx + c = 0, then x = (-b +/- sqrt(b^2 - 4ac))/(2a)

    # Wald CI
    # phat = i/n
    WInt  = phat + c(-1,1)*zvalue*sqrt(phat*(1-phat)/n)

    # Compare my answers with the packages
    #CPInt = binom.test(i,n)$conf.int
    #SInt  = prop.test(i,n)$conf.int
    #CPInt = c(binconf(i,n,method="exact")[2:3])
```

```
    #SInt  = c(binconf(i,n,method="wilson")[2:3])
    #WInt  = c(binconf(i,n,method="asymptotic")[2:3])

    # Check: whether the true parameter p is in the CI (yes/no)
    Cover[i+1,2] = InInt(p, CPInt)
    Cover[i+1,3] = InInt(p, SInt)
    Cover[i+1,4] = InInt(p, WInt)
  }

  p = dbinom(0:n, n, p) # pmf = Pr(X = x), make p an array
  ProbCover = rep(0,3)
  names(ProbCover) = c("ClopperPearson","Score","Wald")

  # sum Pr(X=x) * I(p in CI from x)
  for (i in 1:3){
    ProbCover[i] = sum(p*Cover[,i+1])
  }
  list(n=n, p=p, Cover=Cover, PC=ProbCover)
}

InInt = function(p,interval){
  # whether p is within the interval or not
  if (interval[1] <= p && interval[2] >= p)
    return(TRUE)
  else
    return(FALSE)
}

CISummary = function(n, p) {
  M = matrix(0,nrow=1*length(p),ncol=5)
  colnames(M) = c("n","p","ClopperPearson","Score","Wald")

  k = 1
  for (pp in p) {
    M[k,] = c(n, pp, CICoverage(n,pp)$PC)
    k = k+1 # k++;
  }
  data.frame(M)
}

n = 25
probs = seq(from=0,to=1,by=0.01)
results = CISummary(n, probs)

pdf('coverage_probability_2.pdf',height=5,width=8)
plot(c(0,1),c(0.5,1), type="n",
  main="Plot of coverage probabilities for 95% confidence intervals",
  xlab="True parameter p for binomial (n=25, p)",
  ylab="Coverage probability")
abline(h=0.95, col="blue",lwd=2)
lines(results$p, results$ClopperPearson, col=1, lty=1, lwd=2)
lines(results$p, results$Score,          col=2, lty=2, lwd=2)
lines(results$p, results$Wald,           col=3, lty=3, lwd=2)
tests = c("Clopper-Pearson","Score","Wald")
legend("bottom", legend=tests, lwd=2,lty=1:3,col=1:3)
       #col=1:4, lwd=lwdval, cex=0.70)
dev.off()
```

Appendix D

Perplexity and Geometric Mean

Section 3.4.2 describes the perplexity measure to compare topic model results, and the goal of perplexity is to compare the average negative log-likelihoods. The definition of perplexity involves the geometric average because the geometric mean of likelihoods is equivalent to the exponential of the arithmetic average log-likelihoods.

The mathematical definition of perplexity is "the inverse of the per-word geometric average of the probability of the observations" (Blei and Lafferty, 2006). Given that P words are used for training in each document,

$$\text{Perp}(\Phi) = \left(\prod_{d=1}^{D} \prod_{i=P+1}^{N_d} p(w_i|\Phi, w_{1:P})\right)^{\frac{-1}{\sum_{d=1}^{D}(N_d - P)}}, \tag{D.1}$$

where Φ is the fitted model, D is the total number of documents, N_d is the number of words in document d, and w_i indicates a token in a document.

The proof of the relationship between the geometric mean and the average log-likelihood is derived as below:

1. For each word i and document d, the likelihood is

$$L_{i,d} = p(w_{i,d}|\Phi, w_{1:P}). \tag{D.2}$$

2. Each document d contains $N_d - P$ words that need to be predicted, so the number of all "unknown" words (to be predicted) is $\sum_{d=1}^{D}(N_d - P)$.

3. The negative log-likelihood for each word i and document d is

$$l_{i,d} = -\log p(w_{i,d}|\Phi, w_{1:P}). \tag{D.3}$$

   The average negative log-likelihood is

$$\bar{l}_{i,d} = [\sum_{d=1}^{D}\sum_{i=P+1}^{N_d}(-\log p(w_{i,d}|\Phi, w_{1:P}))]/[\sum_{d=1}^{D}(N_d - P)]. \tag{D.4}$$

4. The geometric-log-arithmetic formula is (Cardinal on Stack Exchange, 2012):

$$y_1 = \log x_1, y_2 = \log x_2, y_3 = \log x_3 \tag{D.5}$$

$$\bar{x}_{GM} = (x_1 x_2 x_3)^{1/3} = \exp(\frac{y_1 + y_2 + y_3}{3}) = \exp(\bar{y}). \tag{D.6}$$

5. Using the formula described in Step 4, I get

$$\exp(\bar{l}_{i,d}) = \exp(\sum_{d=1}^{D}\sum_{i=P+1}^{N_d}(\log p(w_{i,d}|\Phi, w_{1:P})) \cdot \frac{-1}{\sum_{d=1}^{D}(N_d - P)}) \tag{D.7}$$

$$= [\prod_{d=1}^{D}\prod_{i=P+1}^{N_d} p(w_i|\Phi, w_{1:P})]^{[\frac{-1}{\sum_{d=1}^{D}(N_d-P)}]} = \bar{L}_{i,d}, \tag{D.8}$$

where $\bar{L}_{i,d}$ is the geometric mean of the exponential of the arithmetic average negative log-likelihoods.

Bibliography

Agarwal, D. and Chen, B.-C. (2010), "fLDA: matrix factorization through latent dirichlet allocation," in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 91–100, ACM.

Agresti, A. and Kateri, M. (2011), *Categorical data analysis*, Springer.

Au, T. C. (2014), "Topics in Computational Advertising," Ph.D. thesis, Duke University.

Awasthi, P. and Risteski, A. (2015), "On some provably correct cases of variational inference for topic models," in *Advances in Neural Information Processing Systems*, pp. 2098–2106.

Bell, S. (2001), *A beginner's guide to uncertainty of measurement*, National Physical Laboratory Teddington, Middlesex.

Benjamini, Y. and Hochberg, Y. (1995), "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300.

Berry, M. W. and Castellanos, M. (2004), "Survey of text mining," *Computing Reviews*, 45, 548.

Berry, M. W., Dumais, S. T., and O'Brien, G. W. (1995), "Using linear algebra for intelligent information retrieval," *SIAM (Society for Industrial and Applied Mathematics) review*, 37, 573–595.

Bi, Y. (2016), "Scheduling Optimization with LDA and Greedy Algorithm," Master's thesis, Duke University.

Blei, D. and Lafferty, J. (2006), "Correlated topic models," *Advances in neural information processing systems*, 18, 147.

Blei, D. M. (2012), "Probabilistic topic models," *Communications of the ACM*, 55, 77–84.

Blei, D. M. and Lafferty, J. D. (2009), "Visualizing topics with multi-word expressions," *arXiv preprint arXiv:0907.1013*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003), "Latent dirichlet allocation," *the Journal of machine Learning research*, 3, 993–1022.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2016), "Variational inference: A review for statisticians," *arXiv preprint arXiv:1601.00670*.

Box, G. E., Draper, N. R., et al. (1987), *Empirical model-building and response surfaces*, vol. 424, Wiley New York.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992), "Class-based n-gram models of natural language," *Computational linguistics*, 18, 467–479.

Cardinal on Stack Exchange (2012), "Which 'mean' to use and when?"
`http://stats.stackexchange.com/questions/23117/`⊕
`which-mean-to-use-and-when`.

Cavnar, W. B., Trenkle, J. M., et al. (1994), "N-gram-based text categorization," *Ann Arbor MI*, 48113, 161–175.

Chai, C. (2014), "Pre-analysis of Text Data," International Conference on Advances in Interdisciplinary Statistics and Combinatorics,
`http://at.yorku.ca/c/b/i/z/41.htm`.

Chai, C. (2016), "Quantifying Improvement of Topic Modeling Results by N-Gramming," International Conference on Advances in Interdisciplinary Statistics and Combinatorics, `http://at.yorku.ca/cgi-bin/abstract/cbmf-06`.

Chai, C. P. (2013), "Facebook Account Misuse Detection – A Statistical Approach," Master's thesis, National Taiwan University.

Chater, N. and Manning, C. D. (2006), "Probabilistic models of language processing and acquisition," *Trends in cognitive sciences*, 10, 335–344.

Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R. (2003), "Robust and efficient fuzzy match for online data cleaning," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 313–324, ACM.

Chen, C. (1999), "Visualising semantic spaces and author co-citation networks in digital libraries," *Information processing & management*, 35, 401–420.

DataStellar Co., Ltd. (2011), "Thesaurus / Synonyms Database of 113,690 English Words," `http://www.thedataplanet.com/databases/synonyms-database.⊕ html`, Online database; accessed 2014.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990), "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, 41, 391–407.

Ding, Y. (2011), "Scientific collaboration and endorsement: Network analysis of coauthorship and citation networks," *Journal of informetrics*, 5, 187–203.

Doraisamy, S. and Rüger, S. (2003), "Robust polyphonic music retrieval with n-grams," *Journal of Intelligent Information Systems*, 21, 53–70.

Esuli, A. and Sebastiani, F. (2009), *Training data cleaning for text classification*, Springer.

Federal Housing Finance Agency (2016), "National Mortgage Survey Originations," `https://www.fhfa.gov/PolicyProgramsResearch/Programs/⊕ Pages/National-Survey-of-Mortgage-Originations.aspx`.

Fodor, I. K. (2002), "A survey of dimension reduction techniques," Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory.

Gharehchopogh, F. S. and Khalifelu, Z. A. (2011), "Analysis and evaluation of unstructured data: text mining versus natural language processing," in *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pp. 1–4, IEEE.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M. (2010), "A survey of statistical network models," *Foundations and Trends® in Machine Learning*, 2, 129–233.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2011), "Producing power-law distributions and damping word frequencies with two-stage language models," *Journal of Machine Learning Research*, 12, 2335–2382.

Goyal, A. and Parulekar, A. (2015), "Sentiment Analysis for Movie Reviews," `http://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/003.pdf`.

Griffiths, T. L. and Steyvers, M. (2004), "Finding scientific topics," *Proceedings of the National academy of Sciences*, 101, 5228–5235.

Guthrie, D., Allison, B., Liu, W., Guthrie, L., and Wilks, Y. (2006), "A closer look at skip-gram modelling," in *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pp. 1–4.

Hein, R. (2012), "The pros and cons of word clouds as visualizations," `https://www.visioncritical.com/pros-and-cons-word-clouds-⊕ visualizations/`.

Heller, K. A. and Ghahramani, Z. (2005), "Bayesian hierarchical clustering," in *Proceedings of the 22nd international conference on Machine learning*, pp. 297–304, ACM.

Henry, T. (2016), "Quick NGram Processing Script," `https://github.com/trh3/NGramProcessing`.

Henry, T., Banks, D., Chai, C., and Owens-Oas, D. (2016), "Modeling community structure and topics in dynamic text networks," *arXiv preprint arXiv:1610.05756*.

Hirasawa, S., Shih, F.-Y., and Yang, W.-T. (2007), "Student questionnaire analyses for class management by text mining both in Japanese and in Chinese," in *2007 IEEE International Conference on Systems, Man and Cybernetics*, pp. 398–405, IEEE.

Hofman, J. M. and Wiggins, C. H. (2008), "Bayesian approach to network modularity," *Physical review letters*, 100, 258701.

Hogg, R. V. and Tanis, E. A. (1977), *Probability and statistical inference*, vol. 993, Macmillan New York.

Horton, N. (2012), "Example 9.37: (Mis)behavior of binomial confidence intervals," `https://www.r-bloggers.com/example-9-37-misbehavior-of-binomial-⊕ confidence-intervals/`.

Hotho, A., Nürnberger, A., and Paaß, G. (2005), "A brief survey of text mining." in *Ldv Forum*, vol. 20, pp. 19–62.

Hu, C., Ryu, E., Carlson, D., Wang, Y., and Carin, L. (2014), "Latent Gaussian Models for Topic Modeling," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pp. 393–401.

Hu, D. J. (2009), "Latent dirichlet allocation for text, images, and music," *University of California, San Diego. Retrieved April*, 26, 2013.

Jasra, A., Holmes, C., and Stephens, D. (2005), "Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling," *Statistical Science*, pp. 50–67.

Jelinek, F., Mercer, R. L., Bahl, L. R., and Baker, J. K. (1977), "Perplexity – a measure of the difficulty of speech recognition tasks," *The Journal of the Acoustical Society of America*, 62, S63–S63.

Johnson, T. and Dasu, T. (2003), "Data quality and data cleaning: An overview," in *SIGMOD Conference*, vol. 681.

Kirkup, L. and Frenkel, R. B. (2006), *An introduction to uncertainty in measurement: using the GUM (guide to the expression of uncertainty in measurement)*, Cambridge University Press.

Kouloumpis, E., Wilson, T., and Moore, J. D. (2011), "Twitter sentiment analysis: The good the bad and the omg!" *Icwsm*, 11, 538–541.

Lin, C. and He, Y. (2009), "Joint sentiment/topic model for sentiment analysis," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 375–384, ACM.

Loucks, D. and van Beek, E. (2005), "Model sensitivity and uncertainty analysis," *Water resources systems planning and management*, pp. 255–290.

Manning, C. D., Raghavan, P., and Schtze, H. (2008), *Introduction to Information Retrieval*, Cambridge University Press, `http://nlp.stanford.edu/IR-book/⊕ html/htmledition/inverse-document-frequency-1.html`.

Marino, J. B., Banchs, R. E., Crego, J. M., de Gispert, A., Lambert, P., Fonollosa, J. A., and Costa-Jussà, M. R. (2006), "N-gram-based machine translation," *Computational Linguistics*, 32, 527–549.

Matosin, N., Frank, E., Engel, M., Lum, J. S., and Newell, K. A. (2014), "Negativity towards negative results: a discussion of the disconnect between scientific worth and scientific culture," *Disease Models and Mechanisms*, 7, 171–173.

Mcauliffe, J. D. and Blei, D. M. (2007), "Supervised Topic Models," in *Advances in Neural Information Processing Systems*, pp. 121–128.

Mei, Q., Ling, X., Wondra, M., Su, H., and Zhai, C. (2007), “Topic sentiment mixture: modeling facets and opinions in weblogs,” in *Proceedings of the 16th international conference on World Wide Web*, pp. 171–180, ACM.

Melville, P., Gryc, W., and Lawrence, R. D. (2009), “Sentiment analysis of blogs by combining lexical knowledge with text classification,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1275–1284, ACM.

Meyer, D., Hornik, K., and Feinerer, I. (2008), “Text mining infrastructure in R,” *Journal of Statistical Software*, 25, 1–54.

Moghaddam, B. (2009), “Variational Mean Field for Graphical Models,” `http://courses.cms.caltech.edu/cs155/slides/⊕ cs155-14-variational.pdf`.

Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013), “NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets,” *arXiv preprint arXiv:1308.6242*.

Nielsen, F. A. (2011), “AFINN,” `http://www2.imm.dtu.dk/pubdb/p.php?6010`.

Palace, B. (1996), “Data Mining: What is Data Mining?” `http://www.anderson.ucla.edu/faculty/jason.frand/teacher/⊕ technologies/palace/datamining.htm`.

Pang, B., Lee, L., and Vaithyanathan, S. (2002), “Thumbs up?: sentiment classification using machine learning techniques,” in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics.

Rahm, E. and Do, H. H. (2000), “Data cleaning: Problems and current approaches,” *IEEE Data Eng. Bull.*, 23, 3–13.

Ranganath, R., Altosaar, J., Tran, D., and Blei, D. M. (2016), “Operator Variational Inference,” *arXiv preprint arXiv:1610.09033*, 1604.

Rasmussen, C. E. (1999), “The infinite Gaussian mixture model,” in *NIPS*, vol. 12, pp. 554–560.

Soriano, J., Au, T., and Banks, D. (2013), “Text mining in computational advertising,” *Statistical Analysis and Data Mining*, 6, 273–285.

Srivastava, A. N. and Sahami, M. (2009), *Text mining: Classification, clustering, and applications*, CRC Press.

Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014), "A comparison of blocking methods for record linkage," in *International Conference on Privacy in Statistical Databases*, pp. 253–268, Springer.

Stephens, M. (2000), "Dealing with label switching in mixture models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62, 795–809.

Taddy, M. (2013), "Measuring political sentiment on Twitter: Factor optimal design for multinomial inverse regression," *Technometrics*, 55, 415–425.

Tang, B., Shepherd, M., Milios, E., and Heywood, M. I. (2005a), "Comparing and combining dimension reduction techniques for efficient text clustering," in *Proceeding of SIAM International Workshop on Feature Selection for Data Mining*, pp. 17–26, Citeseer.

Tang, J., Li, H., Cao, Y., and Tang, Z. (2005b), "Email data cleaning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 489–498, ACM.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2012), "Hierarchical Dirichlet processes," *Journal of the american statistical association.*

Thet, T. T., Na, J.-C., and Khoo, C. S. (2010), "Aspect-based sentiment analysis of movie reviews on discussion boards," *Journal of information science*, pp. 1–26.

Thulin, M. (2012), "Why is the Dirichlet distribution the prior for the multinomial distribution?" `http://stats.stackexchange.com/questions/44494/⊕ why-is-the-dirichlet-distribution-the-prior-for-the-multinomial-⊕ distribution`.

Titov, I. and McDonald, R. (2008a), "Modeling online reviews with multi-grain topic models," in *Proceedings of the 17th international conference on World Wide Web*, pp. 111–120, ACM.

Titov, I. and McDonald, R. T. (2008b), "A Joint Model of Text and Aspect Ratings for Sentiment Summarization." *ACL*, 8, 308–316.

Truss, L. (2004), *Eats, shoots & leaves: The zero tolerance approach to punctuation*, Penguin.

Vishwanathan, S. (2010), “Sentiment Analysis for Movie Reviews,” *Proceedings of 3rd IRF International Conference.*

Wallach, H. M. (2006), “Topic modeling: beyond bag-of-words,” in *Proceedings of the 23rd international conference on Machine learning*, pp. 977–984, ACM.

Warren, R., Airoldi, E., and Banks, D. (2008), “Network Analysis of Wikipedia,” *Statistical Methods in eCommerce Research*, pp. 81–102.

Wei, X. and Croft, W. B. (2006), “LDA-based document models for ad-hoc retrieval,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 178–185, ACM.

Wild, F. (2015), *lsa: Latent Semantic Analysis*, R package version 0.73.1.

Yamanishi, K. and Li, H. (2002), “Mining open answers in questionnaire data,” *IEEE Intelligent Systems*, 17, 58–63.

Yang, Y. and Wilbur, J. (1996), “Using corpus statistics to remove redundant words in text categorization,” *Journal of the American Society for Information Science*, 47, 357–369.

Yin, J. and Wang, J. (2014), “A Dirichlet multinomial mixture model-based approach for short text clustering,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 233–242, ACM.

Zhang, S., Zhang, C., and Yang, Q. (2003), “Data preparation for data mining,” *Applied Artificial Intelligence*, 17, 375–381.

Zhang, Y., Jin, R., and Zhou, Z.-H. (2010), “Understanding bag-of-words model: a statistical framework,” *International Journal of Machine Learning and Cybernetics*, 1, 43–52.

Zhuang, L., Jing, F., and Zhu, X.-Y. (2006), “Movie review mining and summarization,” in *Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 43–50, ACM.

Biography

Christine Peijinn Chai was born on August 14, 1991 in Massachusetts, USA. She started college at National Taiwan University in 2007 and received her undergraduate degree in electrical engineering in 2011. Then she continued her studies at National Taiwan University and got a Master's in electrical engineering in 2013. Her Master's thesis is "Facebook Account Misuse Detection – A Statistical Approach" (Chai, 2013), under the supervision of Dr. Chin-Laung Lei. After that, Christine moved back to the United States to pursue a PhD in statistical science at Duke University. She earned her Master's en route to PhD in statistical science in 2016, and expects to finish her doctorate degree in 2017. Her advisor is Dr. David L. Banks.