UNIVERSITY OF CALIFORNIA

Los Angeles

A Data Augmentation Approach to Short Text Classification

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

Ryan Robert Rosario

2017

ABSTRACT OF THE DISSERTATION

A Data Augmentation Approach to Short Text Classification

by

Ryan Robert Rosario
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2017
Professor Yingnian Wu, Chair

Text classification typically performs best with large training sets, but short texts are very common on the World Wide Web. Can we use resampling and data augmentation to construct larger texts using similar terms? Several current methods exist for working with short text that rely on using exte2rnal data and contexts, or workarounds.

Our focus is to test a new preprocessing approach that uses resampling, inspired by the bootstrap, combined with data augmentation, by treating each short text as a population and sampling similar words from a semantic space to create a longer text. We use blog post titles collected from the Technorati blog aggregator as experimental data with each title appearing in one of ten categories. We first test how well the raw short texts are classified using a variant of SVM designed specifically for short texts as well as a supervised topic model and an SVM model that uses semantic vectors as features. We then build a semantic space and augment each short text with related terms under a variety of experimental conditions. We test the classifiers on the augmented data and compare performance to the aforementioned baselines. The classifier performance on augmented test sets outperformed the baseline classifiers in most cases.

The dissertation of Ryan Robert Rosario is approved.

Junghoo Cho

Jan de Leeuw

Arash Ali Amini

Yingnian Wu, Committee Chair

University of California, Los Angeles

2017

To my dear parents. . .

who supported me, encouraged me and believed in me
every step of the way even when I may
not have believed in myself.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I have many people to thank for their assistance on my journey – from the conception of the idea to the final stages, but I also have several people to thank that had a significant impact on my broader graduate education.

I first thank Yingnian Wu for agreeing to serve as my doctoral committee chair and advisor, after my original chair retired. In a short period of time Yingnian took me under his wing and quickly got up to speed on my research topic. His feedback was crucial not only for the successful completion of this process, but also for my confidence as I had widely performed my research and writing myself at a remote location in the Eastern Sierra, somewhat isolated from academia. Yingnian provided much useful feedback on my manuscript and also my defense slides and often gave me advice that I did not think of myself. His position as chair yielded some great feedback from the committee and made for a much stronger manuscript. I also thank Arash Amini for serving on my committee under a short timeline due to similar circumstances. I greatly appreciate his time and his feedback on my research and manuscript.

My original advisor and chair, Jan de Leeuw, served a very broad and significant role in my graduate education. One of the things I admire most about Jan is his dedication to spreading awareness of the open-source scientific community, reproducible research and practicing the philosophy of "don't just do it, share it" – something that I intend to emulate throughout my career. Jan helped me get my first internship which ended up leading to the idea for my research. He was also very supportive of my other educational endeavors with the Department of Computer Science. Jan's work exposed me to many different aspects of the field of statistics. Through his labs and centers, I served as a web developer, a statistical consultant, got to investigate geographical information systems, served as a student editor for the Journal of Statistical Software and gained significant public speaking experience via the Los Angeles R Users' Meetup which he co-founded. Most of these activities also served as a source of funding during my education. I feel that the variety of Jan's interests is difficult to replicate, and I am very thankful for the diverse experiences he provided me. I am grateful that Jan agreed to remain on my committee even after retirement. I hope that

Jan continues doing what he is doing, and also enjoys his well-deserved retirement!

Throughout the years, Glenda Jones has been my rock within the Department of Statistics. She has been supportive of all of my educational and professional endeavors. She was a consistent smile and friendly face that celebrated my wins and listened to me during the low times. I am grateful for the level of confidence and trust she had in me and for her constant encouragement to follow through with my goals. Glenda spent hours helping me with the logistics of pursuing a graduate degree in the Department of Computer Science, another degree that has proven very lucrative for my career. Her connections and knowledge of University policies is one-of-a-kind and I am grateful for all of her assistance and all of her work helping me with enrollment. I also thank her for organizing countless quarters of teaching assistantships and readerships that were pivotal to funding my education. Glenda *always* goes above and beyond for her students, and she should truly be commended for her dedication and service to the Department.

The Department of Computer Science played a large role in my studies and also my professional career. I first want to thank Junghoo "John" Cho for getting me hooked on computer science after taking his class in web information management. His class was the first formal setting where I saw how important statistics is to computer science and vice versa. He later gave me the opportunity to attend his research seminars in social networks and web mining where I met many friends and colleagues that share my interest and still communicate with to this day. I also thank him for serving on my doctoral committee and for providing so much useful feedback. I thank Michael Dyer who gave me the courage to learn more about natural language processing, a field that I believed was a pipe dream until I took his sequence of courses and again realized how important statistics is to computer science. He encouraged me to continue doing research and even encouraged me to write a book, which I plan to do eventually.

While my education at UCLA provided me with the mechanics for my research, it was actually industry that inspired the idea for my research. I thank Ted Ives for his mentorship during my first internship where he approached me with a problem – there needs to be a way to cluster and classify ad creatives which are usually *short*. This ultimately lead me into my own research in machine learning and natural language processing particularly for short texts. I originally wanted to use tweets for my research, but tweets exhibit so many syntactical challenges above and beyond other user generated content without any labels, I would never finish! Neal Richter, my manager at another internship as well as my first full-time position gave me the idea to use the Technorati site to collect labeled data. Without his suggestion, I would have spent a lot more time looking for data... or still working on cleaning tweets!

I thank Vladislavs Dovgalecs for reading my dissertation cover to cover and providing thorough feedback on each chapter and pointing out typos and inconsistencies that I missed several times. I am grateful for his time because he brought up many points that my committee also brought up, and was the only person outside of my committee and myself that read my dissertation prior to submission.

I saved the best for last – the most important people in my life: my mother and father, who deserve more thanks and gratitude than I could ever put into words. My parents have encouraged me at every step in my life and always believed in me. They celebrated with me and when I fell down, they were the ones that always helped me back up with love, encouragement and wisdom. They taught me the importance of hard work and being true to myself. They taught me to be curious, always ask questions and never take "no" for an answer regarding my own ambitions. I thank them for the countless hours and days they spent listening to me complain about everything involving school and my research, and tolerated me when I would defiantly ignore my work out of frustration. My mother was a teacher and when I was very young I remember asking her about educational assessment, something everybody that went to school in California is well aware of. I was curious about all of the scores and numbers and what they said about me. I found it fascinating and she told me a person that works with data (test scores in this case) is a "statistician" and I knew at that moment that I wanted to be a statistician when I grew up. We frequently talked about college and education in general, and I asked about continuing school after college. She explained to me the concept of graduate school and earning a masters degree and the fact that the Ph.D. was the highest academic degree. From that moment on, at the ripe know-it-all age of ten, I knew I wanted to get a Ph.D. – a Ph.D. in statistics. Sure enough, I originally entered the Ph.D. program to study psychometrics applied to educational assessment. Although I ended up specializing in machine learning, psychometrics is a field that is still very interesting to me. My dad is my rock. His experience and wisdom come from experiences different from my mother and provide me with a different perspective. There were times when I was so frustrated with research that I wanted to give up, especially once I started working. My dad reminded me daily that I should be working on my dissertation and that completing my research was not only worth it for my career, but also to finish what I started to fulfill a commitment I accepted. My father's mantra is "winners never quit and quitters never win" and this has become part of my DNA via his beliefs. It is customary to thank funding sources, and my dad was one of them. During the two years where I worked remotely, my dad funded part of my education as well as other expenses I would usually cover when working. His generosity is one-of-a-kind and he did not need to assist me, but he did everything he could to help me finish. I would not be the person I am today without the encouragement and support of my parents. I am very fortunate to have a strong relationship with them, and for that I am very grateful.

I also owe thanks to entities other than humans. I thank Dropbox for their data storage and synchronization product. Throughout the years, I had stored my thoughts, research and code in so many different places that I often lost it. Even standard version control tools proved problematic for one reason or another. A few years ago I began judiciously storing all of my code, data, research materials and this manuscript in my Dropbox where it was always kept up-to-date regardless which machine I was on, or where in the world I was. I also appreciate the version control capability as there were many times I had an "oops" moment and lost files. This manuscript will essentially be etched in stone for all time, and while Dropbox may not outlast it, I feel that it was a crucial part of my productivity and worthy of thanks. Speaking of productivity, I owe a lot of gratitude to my Mammoth Lakes sanctuary. The solitude and calm provided by the Eastern Sierra gave me a wealth of time to concentrate on research, writing code, and writing my dissertation without many distractions. I spent the days hiking and mountain biking which allowed me to burn off stress and anxiety caused by the ups and downs of working on a dissertation. Interacting with the flora and the fauna gave me a different perspective on life and its challenges and improved who I am as a person. John Muir was correct when he said in 1901 that “Going to the mountains is going home.”

VITA

2006	B.S. (Mathematics of Computation, Statistics) UCLA, Los Angeles, California.
2007–2017	Teaching Fellow Department of Statistics UCLA, Los Angeles, California.
2008	M.S. (Statistics) UCLA, Los Angeles, California.
2010	M.S. (Computer Science) UCLA, Los Angeles, California.
2010	Research Mentor Research in Industrial Projects for Students (RIPS) Institute for Pure and Applied Mathematics UCLA, Los Angeles, California.
2013–2015	Quantitative Engineer, Facebook Menlo Park, California.

PUBLICATIONS AND PRESENTATIONS

N. Christou, R. R. Rosario, and I. Dinov. *Technology-Enhanced Probability and Statistics Education Using the Statistics Online Computational Resource.* Emerging Technologies for Online Learning Symposium 2010. Fairmont Hotel San Jose. 20 Jul. 2010.

R. R. Rosario. Review of Text Mining with Perl, by Roger Bilisoly. In *Journal of Statistical*

Software, Book Reviews. 14 Jan. 2009. Vol. 29, Book Review 9. http://www.jstatsoft.org/v29/b09

E. Agapie, G. Chen, D. Houston, E. Howard, J. Kim, M. Y. Mun, A. Mondschein, S. Reddy, R. Rosario, J. Ryder, A. Steiner, J. Burke, E. Estrin, M. Hansen and M. Rahimi. *Seeing Our Signals: Combining location traces and web-based models for personal discovery*, in *Proceedings of Hot Mobile*, 2007

CHAPTER 1

Introduction

1.1 Motivation and Objectives

The World Wide Web (WWW, or “Web”) has greatly changed the way humans consume, produce, process and think about information. Since its humble beginnings in 1989 and its advent as the major Internet protocol in the mid 1990s, the Web has become a virtual world where users not only consume data, but also produce their own data[11]. In turn, statisticians, computer scientists, and data scientists mine through all of this user generated data and analyze it to build models to make decisions based on the data. Such local decisions influence the content the user sees on subsequent visits and allows for a more personal, interactive experience local to the user. Some example usages of such data are personalized newsfeeds, recommendation systems for e-commerce[1] and music listening[2], targeted advertising and tailored search engine results[3]. Much of these systems rely on two specific types of information: network/graph data, and text data. This manuscript will focus heavily on working with text data, and only mention network/graph data in passing.

Statistics is the study of uncertainty and random variation[40]. Humans are not perfect, and thus they produce data that contain a lot of noise. Statistical methods assist in filtering noise out of data, but the ubiquity, massiveness, and uncertainty presented by data produced by humans creates many statistical problems, many of which have yet to be studied thoroughly. Blogs and social networking sites produce several challenges since the users are

[1] `http://www.amazon.com`

[2] `http://www.pandora.com`

[3] `http://www.google.com`

permitted to freely express themselves[134][110][10]. First, social network users typically produce text that is timely, but does not necessarily represent meaningful content. Second, users are prone to express emotions and sentiment rather than content. Third, users usually do not respect proper English grammar or spelling in the content that they provide. Most importantly, much text provided by users is *short*. Recall from Statistics 101 that large sample sizes typically yield the best results in statistical significance testing and other statistical models. This manuscript attempts to improve upon existing methods for extracting meaning from texts by extending NLP and statistical methods into the domain of short texts.

The Web is full of short texts. Some examples of short texts include Facebook posts and comments, web forum posts, instant and private messages, ad creatives, and much more. Additionally, user generated content on the Web typically follows a structure imposed by the entity providing the service; for example, a blog post contains a short title, a short description and then a longer post body. Twitter is perhaps the most well-known repository of short texts because each "tweet" or "status update" contains no more than 140 characters[4]. Twitter also provides several interesting features to the NLP researcher that other services do not offer. Starting in 2009, Twitter allowed users to create "lists" of users — groups of users that a human labels as discussing a particular topic. Naturally, Twitter lists provide an ad-hoc labeling mechanism for Twitter users[92]. Users can also tag their tweets with *hashtags*, which is the combination of a hash symbol (`#`) and a word, thus providing a human induced labeling of the tweet's content[41]. Since their inception on Twitter, hashtags have spread to other services such as Facebook[5], Instagram[6] and others

Tweets and other short texts from user generated content sites pose significant interesting challenges to researchers. Many users post updates that do not contain any meaningful content. They may post mundane content (i.e. what they are currently doing) or they may be spamming. Due to character limits on services like Twitter and the "quick" and informal nature of online communication, analyzing content on the individual level is difficult. The

[4] `https://support.twitter.com/articles/15367`

[5] `http://www.facebook.com`

[6] `http://www.instagram.com`

limitations complicate the user's message by requiring them to use contractions, drop vowels, or use abbreviations to make their point[147]. Additionally, users do not use perfect English grammar, often misspell words, or use the wrong form of a homophone (i.e. they're vs. their vs. there)[50]. Some users may not even make any sense to other human users. All of these issues must be resolved by the researcher prior to attempting to train and test a classifier.

The early 1990s gave rise to the first so-called topic model. From the 1990s to the present, several topic models have been developed that can mathematically model text and classify a text into one or sometimes multiple topics based on the words that appear in the text. As with any statistical model however, the assumption is that there is large enough sample size – enough text – to present in the training phase. With tweets and other short texts, small sample sizes pose a large challenge that will be discussed in this manuscript. One common method used to estimate model parameters in statistics is the bootstrap[48]. In this manuscript, a data augmentation method inspired by the bootstrap will be applied to simultaneously attempt to solve the short text problem and also to create a framework to improve classification.

1.2 Manuscript Outline

This manuscript is organized as follows. Topic models and classification methods for written text are introduced in Chapter 2 and will provide the motivation for this work. In Chapter 3 we review several existing methods for short text augmentation, query expansion and data augmentation. In Chapter 4 the statistical bootstrap is discussed for its use with samples of small size. In Chapter 5 a data augmentation method motivated by the bootstrap is proposed as a method for augmenting short texts and several experiments are introduced. The results of the experiments are presented in Chapter 6. Finally, future work is discussed in Chapter 7.

1.3 Individual Contributions

The major individual contributions to the field described in this manuscript are discussed in Chapters 4, 5 and 6, namely a method for augmenting short texts by resampling terms from the semantic space computed from a coarse topic model. This proposed method is applied to short texts for the later purpose of classification in this research.

CHAPTER 2

Advances in Topic Modeling and Classification

Topic modeling refers to algorithms that attempt to extract the theme of textual content. Topic modeling algorithms are useful for document retrieval, identification of trending topics, novel event detection, document classification, language modeling and machine translation[61]. This manuscript focuses on document classification. In this chapter, we discuss the history of topic modeling, beginning with elementary principles.

2.1 Naïve Bayes Classifiers

The Naïve Bayes classifier has demonstrated great success in NLP domains such as language identification[66], sentiment analysis[130], and even topic identification [118]. According to [73], the Naïve Bayes model is useful when the feature space is high, and when the features F_d are independent conditioned on a particular class c_k. The fact that Naïve Bayes has been so successful in NLP is curious because words in the English language are not independent given the structure of a sentence[45]. For example, a grammatically correct English sentence containing an adjective will have a noun following the adjective[113]. Thus, there is clear dependence among words in grammatically correct English sentences. Still, Naïve Bayes has received appreciable consideration in the NLP community and has served as a baseline classification algorithm for research in the field[113].

Given features $f_1, f_2, \ldots, f_{|V|}$, the Naïve Bayes classifier[73] selects the most likely class label c^* as follows:

$$\begin{aligned} c^* &= \mathrm{argmax}_{c_k \in C} P(c_k|f_i) \\ &= \mathrm{argmax}_{c_k \in C} P(c_k) \prod_{i=1}^{|V|} P(f_i|c_k) \end{aligned} \tag{2.1}$$

where $P(c_k)$ is the prior probability of class c_k and $|V|$ is the number of features (the size of the vocabulary). Then the posterior probability, as estimated from feature presence indicators, that a particular document belongs to class c_k is

$$\tilde{P}(c_k|f_i) = \frac{\tilde{P}(c_k) \prod_{i=1}^{|V|} \tilde{P}(f_i|c_k)}{\sum_{k'} \tilde{P}(c_{k'}) \prod_{i=1}^{|V|} \tilde{P}(f_i|c_{k'})}$$

In the *standard* incarnation of Naïve Bayes, sometimes referred to as Bernoulli Naïve Bayes[120], the probabilities are computed with respect to the number of documents in a class c_k that contain feature f_i. *The number of times f_i appears in the document is not taken into account – only its binary presence.* Then the maximum likelihood estimate for $P(f_i|c_k)$ is given as

$$\tilde{P}(f_i|c_k) = \frac{n_d(f_i, c_k)}{|D|} \tag{2.2}$$

where $n_d(f_i, c_k)$ is the number of *documents* in class c_k containing the feature f_i and $|D|$ is the total number of documents in the corpus. One issue that arises when using Naïve Bayes with text is sparsity in the feature set which causes two problems[131]:

1. computing posterior probabilities can cause underflows due to the vast number of multiplications and,

2. if a feature never appears in a document within a certain class in the training data, probability computations yield meaningless results.

To resolve the underflow issue, it is customary to take the logarithm of the product in (2.1) instead, yielding the following decision rule

$$\begin{aligned} c^* &= \text{argmax}_{c_k \in C} \log P(c_k|f_i) \\ &= \text{argmax}_{c_k \in C} \left[\log P(c_k) + \sum_{i=1}^{|V|} \log P(f_i|c_k) \right] \end{aligned}$$

To resolve the problem with zeros in the computation of probabilities where a feature f_i never appears in a document in class c_k, the common approach is to use Laplace smoothing[113] by which a count of zero is replaced by a slightly larger but negligible number α_{f_i} for each feature, usually 1. The smoothed version of (2.2) now becomes:.

$$\tilde{P}(f_i|c_k) = \frac{n_d(f_i, c_k) + \alpha_{f_i}}{|D| + \sum_i \alpha_{f_i}}$$

2.1.1 Multinomial Naïve Bayes Classifier

The standard incarnation of Naïve Bayes treats a feature f_i as either appearing in a document in class c_k or not. Multinomial Naïve Bayes (MNB) models the terms in a document d as coming from a multinomial distribution, rather than a multivariate Bernoulli distribution and *includes the number of times* a feature appears in a document rather than simply its presence/absence. In their paper [118], McCallum, Nigam et al. showed that MNB performed better in terms of accuracy due to the inclusion of frequency information. The probability of seeing a document d, consisting of features F_d, in class c_k is

$$P(F_d|c_k) = \frac{(\sum_i n(f_i, d))!}{\prod_i n(f_i, d)!} \prod_i P(f_i|c_k)^{n(f_i,d)}$$

where $n(f_i, d)$ is the number of times feature f_i appears in document d. The classification rule is now

$$c^* = \text{argmax}_c \left[\log P(c_k) + \sum_{i=1}^{|V|} n(f_i, d) \log P(f_i|c_k) \right]$$

where $\tilde{P}(f_i|c_k)$ is estimated from the expectation of a Dirichlet prior as

$$\tilde{P}(f_i|c_k) = \frac{n(f_i, c_k) + \alpha_{f_i}}{N_{w_{c_k}} + \sum_i \alpha_i} \tag{2.3}$$

and $n(f_i, c_k)$ is the number of occurrences of f_i in documents in topic c_k, α_{f_i} is a smoothing parameter and $N_{w_{c_k}}$ is the total number of word occurrences in topic c_k. Finally, $P(c_k)$ is estimated as

$$\tilde{P}(c_k) = \frac{N_{d_{c_k}}}{|D|}$$

leading to the final form for the classification rule

$$c^* = \text{argmax}_c \left[\log \frac{N_{d_{c_k}}}{|D|} + \sum_{i=1}^{|V|} n(f_i, d) \log P(f_i|c_k) \right]$$

2.1.2 Other Variations of the Naïve Bayes Classifier

Researchers have developed finer variations of Naïve Bayes to attempt to resolve some of the failed assumptions involved in using Naïve Bayes with text data.

In [149], Rennie et al. discuss a major problem with Naïve Bayes namely the systemic bias caused by having an unequal number of training samples in each class. When the number of samples in one class is much greater than the others, predictions tend to be skewed towards the larger class. The researchers made several marked improvements to Naïve Bayes in a method called Term Weighted Complementary Naïve Bayes (TWCNB). First, the authors change the prediction problem to focus on all classes *excluding the class of interest* – the inverse problem. They replace (2.3) with

$$P(\overline{c_k}|F) = \frac{n(f_i, \overline{c_k}) + \alpha_{f_i}}{N_{w_{\overline{c_k}}} + \sum_i \alpha_i} \tag{2.4}$$

where $\overline{c_k}$ refers to the set of classes *excluding* k. In their paper, this variation is referred to as Complementary Naïve Bayes (CNB). Second, the researchers noted that the distribution of term counts exhibited a power law and they proposed a transformation to coerce the data so that it would exhibit more of a multinomial behavior. Finally, they normalized by document length and frequency. All of these transformations yield what the authors called Term-Weighted Complementary Naïve Bayes (TWCNB) and is presented as Algorithm 1. Term Weighted Complementary Naïve Bayes is implemented in Mahout[55] as `cbayes` and also in scikit-learn[132] as `MultinomialNB`.

One other variation of Naïve Bayes for text classification is Poisson Naïve Bayes from [93] where the researchers use the Poisson distribution as the generative distribution of the text. If X is a random variable corresponding to the number of times a particular feature appears then for each document

$$P(F_d) = \prod_{i=1}^{|V|} P(X_i = f_i) = \prod_{i=1}^{|V|} \frac{e^{-\lambda_i}\lambda_i^{f_i}}{f_i!}$$

The authors also derive the Naïve Bayes problem to use the new distribution.

$$\begin{aligned} P(c_k|F) &= \frac{P(F|c_k)P(c_k)}{P(F)} \\ &= \frac{P(F|c_k)P(c_k)}{P(F|c_k)P(c_k) + P(F|\overline{c})P(\overline{c})} \\ &= \frac{\frac{P(F|c_k)}{P(F|\overline{c_k})} \cdot P(c_k)}{\frac{P(F|c_k)}{P(F|\overline{c_k})} \cdot P(c_k) + P(\overline{c_k})} \end{aligned} \tag{2.5}$$

Then by taking

$$\begin{aligned} z_{c_k} &= \sum_{i=1}^{|V|} \log\left(\frac{P(X_i = f_i|c_k)}{P(X_i = f_i|\bar{c}_k)}\right) \\ &= \sum_{i=1}^{|V|} \log\left(\frac{e^{-\lambda_i}\lambda_i^{f_i}}{e^{-\mu_i}\mu_i^{f_i}}\right) \end{aligned}$$

Equation 2.5 becomes

$$P(c_k|F) = \frac{e^{z_{c_k}}P(c_k)}{e^{z_{c_k}}P(c_k) + P(\bar{c}_k)}$$

Algorithm 1: Steps for Document Classification with Term Weighted Complementary Naïve Bayes (TWCNB)

Note that this computation is dependent on the entire corpus, thus there is a small change in notation to take both features and documents into account.

For some set of training documents D:

1. Compute the TF-IDF transform, denoted t_{ij} for each feature.

$$t_{ij} = \log\left(f_{ij} + 1\right) \times \frac{|D|}{|d \in D : f_{ij} \in d|}$$

   where $|d \in D : f_{ij} \in d|$ is the number of documents containing f_{ij}.

2. Normalize by document length.

$$\tilde{t}_{ij} = \frac{t_{ij}}{\sqrt{\sum_{i'} \left(t_{i'j}\right)^2}}$$

3. Use the complement classes.

$$\omega_{ik} = \log \frac{\sum_{j:d_j \notin c_k} \tilde{t}_{ij} + \alpha_i}{\sum_{j:d_j \notin c_k} \sum_{i'} \tilde{t}_{i'j} + \alpha}$$

   where α_i is a smoothing parameter and α is the sum. The condition $j : d_j \notin c_k$ denotes the set of documents that are not in class c_k.

4. Normalize these weights and store.

$$\tilde{\omega}_{ik} = \frac{\omega_{ik}}{\sum_{i'} \omega_{i'k}}$$

Then for some unseen testing document d'_j, let f_{ij} be the usual frequency of word i in document j. Then the predicted class for d'_j is given by

$$c^* = \arg\min_{c} \sum_{i} f_{ij} \tilde{\omega}_{ic}$$

Similar to Multinomial Naïve Bayes, the researchers avoid the zero-count problem problem using a modified count.

$$\tilde{f}_i = \frac{f_i + \alpha_{f_i}}{N_{w_d} + \sum_i \alpha_i} \cdot \tau$$

where α_{f_i} serves the same purpose as in (2.4) and τ is a large integer that makes $\tilde{f}_i$ an integer. λ_i is the Poisson mean for f_i in class c_k and μ_i is the Poisson mean for f_i in class $\overline{c_k}$ and are estimated from modified weights $\tilde{f}_i$.

$$\tilde{\lambda}_i = \sum_{d \in D_{c_k}} \tilde{P}(F_d | c_k) \cdot \tilde{f}_{id} \qquad \tilde{\mu}_i = \sum_{d \in D_{\overline{c_k}}} \tilde{P}(F_d | \overline{c_k}) \cdot \tilde{f}_{id}$$

where

$$\tilde{P}(F_d|c_k) = \frac{\beta}{N_{d_{c_k}}} + (1-\beta)\frac{N_{w_{d_k}}}{\sum_{d' \in D_{c_k}} N_{w_{d'}}}$$

That is, λ_i and μ_i are the average number of the counts for classes c_k and $\overline{c_k}$ respectively. In the same work, Kim, Seo et al. also studied the effect of β[7] and introduced a special feature weighting method similar to that in [149]; for more information, see [93]. The researchers found that combining Poisson Naïve Bayes with their feature weighting method yielded a modest 4.5% improvement in MicroF1 performance – from 72% for Multinomial Naïve Bayes to approximately 76.5%.

2.2 Latent Semantic Analysis (LSA)

The first sophisticated method for topic modeling that was designed specifically for text was Latent Semantic Analysis, sometimes instead called Latent Semantic Indexing (LSI)[43]. For the purpose of this manuscript, LSA and LSI will be used interchangeably – though LSI is typically the acronym used for information retrieval discussion, the terms are used interchangeably in most literature. The authors discuss that their motivation for developing LSA was that typical keyword matching systems do not take synonymy and polysemy into account when retrieving or modeling documents.

The authors use the term synonymy to refer to the case where many words are used to describe a particular object or concept. Human raters use the same words to describe a particular object less than 20% of the time [57], which Deerwester et al. states reduces recall performance in a document retrieval system. The word polysemy is used in their work to describe the inverse phenomenon – the case of words having multiple meanings depending on context. The researchers state that polysemy reduces precision performance in a document retrieval system.

Latent Semantic Analysis (LSA) works as follows. Let $\mathbf{X}$ be a $|V| \times |D|$ matrix where $|V|$ is the number of terms in a previously curated vocabulary and $|D|$ is the number of

[7] β is denoted as α in Kim, Seo et. al. but modified for this manuscript to avoid confusion with other uses.

documents in the corpus. This matrix is referred to as a *term-document matrix* in the topic modeling literature. Each element X_{ij} is some measure that relates term i to document j such as word frequency as in [43] or TF-IDF[154] as in [186]. Using the singular value decomposition[63], the term-document matrix is decomposed as

$$\mathbf{X} \approx \hat{\mathbf{X}} = \mathbf{W}\mathbf{S}\mathbf{D}'$$

where **W** and **D** contain the left and right singular vectors respectively and **S** is a diagonal matrix containing the singular values of **X**. It is important to recognize that **W** and **D** can be switched in the equation depending on how rows and columns are defined in **X**. Note that **W** contains the eigenvectors of $\mathbf{X}\mathbf{X}'$ and **D** contains the eigenvectors of $\mathbf{X}'\mathbf{X}$. It is important to note that $\mathbf{X}\mathbf{X}'$ and $\mathbf{X}'\mathbf{X}$ contain, as elements, a measure of similarity between terms and between documents respectively in the original vector space. To reduce the dimensionality of the original data and to introduce generality into the data we only retain the top k singular values and set the rest to zero. The result is a reduced rank matrix $\hat{\mathbf{X}}_\mathbf{k}$ that encapsulates the relationship among terms and documents.

$$\hat{\mathbf{X}}_\mathbf{k} = \mathbf{W}_\mathbf{k}\mathbf{S}_\mathbf{k}\mathbf{D}'_\mathbf{k}$$

Then one can compare terms and documents respectively in this space using

$$\hat{\mathbf{X}}_\mathbf{k}\hat{\mathbf{X}}'_\mathbf{k} = \mathbf{W}_\mathbf{k}\mathbf{S}_\mathbf{k}{}^\mathbf{2}\mathbf{W}'_\mathbf{k}$$

$$\hat{\mathbf{X}}'\hat{\mathbf{X}} = \mathbf{D}_\mathbf{k}\mathbf{S}_\mathbf{k}{}^\mathbf{2}\mathbf{D}'_\mathbf{k}$$

Given a query, a set of words, one can retrieve similar documents to the query by first constructing what the authors call a pseuodocument $\mathbf{D}_\mathbf{q}$– a vector of terms corresponding to the text in the query. It is important to note that only terms in the query that also appear in V may be included.

$$\mathbf{D}_\mathbf{q} = \mathbf{X}'_\mathbf{q}\mathbf{W}\mathbf{S}_\mathbf{k}^{-\mathbf{1}}$$

Then, relevant documents can be discovered by computing a similarity metric such as the cosine similarity[114] defined as

$$\cos\theta = \frac{v_i \cdot v_j}{||v_i||||v_j||}$$

where v_i and v_j represent the row or column corresponding to two terms or two documents respectively.

Latent Semantic Analysis is an important component to this manuscript because LSA is used as the population from which we resample words to create augmented bags-of-words and will be discussed later. The researcher refers to this as a semantic space. It preserves the relationship between terms and documents in such a way that differences between terms, between documents and between term/document pairs can be measured. One major pitfall of LSA and algorithms that rely on singular value decomposition is computational complexity; SVD has a runtime of $O\left((|V| + |D|)^2 k^3\right)$ where k is the number of dimensions in the reduced semantic space[100]. Instead, optimizations using algorithms such as Lanczos method[152] provide a reduced runtime of $O(nk)$ where n is the number non-zero entries in the sparse term/document matrix $\mathbf{X}$.

2.3 Probabilistic Latent Semantic Analysis (PLSA)

As an attempt to correct some of the deficiencies involving LSA, particularly the computation complexity of SVD, Hofmann developed a stochastic version of LSA called Probabilistic Latent Semantic Analysis (PLSA)[78]. Based on the review of the literature, PLSA seemed to be the first widely accepted *generative* model for topic modeling.

PLSA is based on the aspect model[155] and has the following form

$$
\begin{aligned}
P(d,w) &= P(d)P(w|d) \\
&= P(d)\left[\sum_{k\in\{1,\ldots,K\}} P(w|z=k)P(z=k|d)\right] \\
&= P(d)\left[\sum_{k\in\{1,\ldots,K\}} P(w|z=k)\frac{P(z=k,d)}{P(d)}\right] \\
&= P(d)\left[\sum_{k\in\{1,\ldots,K\}} P(w|z=k)\frac{P(d|z=k)P(z=k)}{P(d)}\right] \\
&= \sum_{k\in\{1,\ldots,K\}} P(w|z=k)P(d|z=k)P(z=k)
\end{aligned}
$$

where z is a random variable denoting a topic assignment and k now refers to a particular topic in a set of topics. The topic assignments are estimated using the Expectation Maximization (EM) algorithm as follows in Algorithm 2.

Algorithm 2: Expectation Maximization (EM) Algorithm for PLSA

1. Make an initial guess of $P(z|d,w)$, $P(w|z)$, $P(d|z)$, $P(z)$.
2. *Expectation Step (E Step)*: for all w, d, z compute topic probabilities.

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

3. *Maximization Step (M Step)*: for all w, d, z compute new values for the parameters.

$$
\begin{aligned}
P(w|z) &\propto \sum_{d\in D} n(d,w)P(z|d,w) \\
P(d|z) &\propto \sum_{w\in V} n(d,w)P(z|d,w) \\
P(z) &\propto \sum_{d\in D}\sum_{w\in V} n(d,w)P(z|d,w)
\end{aligned}
$$

4. Repeat steps 2 and 3 until convergence.

Note that $n(d,w)$ is the number of times term w is used in document d.

For a full derivation of the likelihood for PLSA, the ensuing EM algorithm for parameter estimation and improvements using tempered EM, see [79]. The graphical model for PLSA is shown as Figure 2.1. Hofmann compared PLSA with LSA and also an ensemble of PLSA models. On various corpora, PLSA saw an average increase in precision of 34% with precision improvement across the board, and the ensemble of PLSA models yielded an average increase in precision of 46% over LSA with precision improvement across the board.

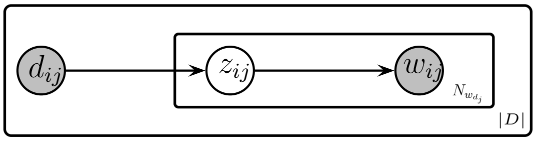


Figure 2.1: Graphical model depicting Probabilistic Latent Semantic Analysis (PLSA).

PLSA has been used in a variety of different topic modeling and information retrieval contexts outside of standard text. In [23], researchers used PLSA for object and scene categorization. PLSA and other topic models have also found a loyal home in the field of digital advertising. In [170], Wang et al. used PLSA to categorize concepts presented in video advertisements by dividing scenes and using particular keyframes to generate a "video-based bag-of-words model." Wu et al. used PLSA to improve user segmentation for behavioral targeting[178]. In their work, the researchers found that including PLSA in user segmentation yielded higher precision (clickthrough-rate) while other traditional methods such as k-means and CLUTO yielded higher recall. This finding lead to the conclusion that the precision of traditional methods could be increased by including more users into each segment, but that with PLSA the extra users were not necessary and thus PLSA provided a more efficient method. In [160], Son et. al. used a variation of PLSA for comparing locations based on geographic topics and features. The most popular PLSA implementation appears in `Lemur`[8] (C++).

2.4 Latent Dirichlet Allocation

A method that has gained and maintained a lot of traction within the natural language processing community is Latent Dirichlet Allocation[19]. Latent Dirichlet Allocation, or

[8] `http://www.lemurproject.org/`

LDA as it will be referred to in this manuscript, is a generative Bayesian graphical model for topic discovery. Theoretically, LDA and PLSA are very similar and they are equivalent in the sense that PLSA is LDA with a uniform Dirichlet prior[62]. In [161], Song notes that LDA is more robust on sparse data and a much larger feature space suggesting it performs better for text data. In Blei et al.'s original paper [19], they note that the number of parameters to estimate in PLSA increases linearly with the number of training documents and can lead to severe overfitting. It is also difficult to generalize PLSA to new test documents because such requires an expensive process called "folding-in"[174]. LDA fixes these issues by establishing distributions over topics rather than requiring a matrix of probabilities.

The goal in Latent Dirichlet Allocation is to estimate the posterior distribution $P(\mathbf{z}|\mathbf{w})$ for a set of documents where $\mathbf{z}$ is a vector of topic assignments and $\mathbf{w}$ is a matrix or words in documents. The graphical model from [19] appears in Figure 2.2 and LDA is described as follows.

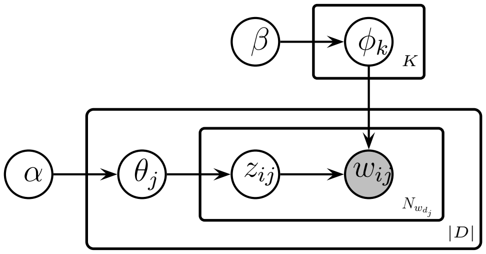


Figure 2.2: Graphical model depicting Latent Dirichlet Allocation (LDA).

We initially have a corpus D containing documents $d_j, 1 \leq j \leq |D|$ where each document contains $N_{w_{d_j}}$ word occurrences from a fixed vocabulary V of size $|V|$. Let K be the number of topics we wish to discover. $z_{ij} \in \{1, \ldots, K\}$ is the topic to which word occurrence i in document d_j is currently assigned. The parameter θ_j is a vector containing, for document d_j, the probability distribution over the K topics. Over the entire corpus, we then have the model parameter θ which is a matrix containing topic distributions for each of the $|D|$ documents. That is, θ is the probability that a particular document d is assigned to topic k. The parameter ϕ_k is the vector containing the probability distribution over the $|V|$ words within topic k and the model parameter ϕ is a matrix containing probability distributions over all $|V|$ words, conditioned on topic k. Finally, the hyperparameters α and β control the nature of the priors of the LDA model and will be discussed in more detail later; for now we can assume they are arbitrary and fixed. LDA assumes that documents are generated by the process denoted in Algorithm 3, though in practice the sampling procedure proceeds as described in Algorithm 4 after first randomly assigning all words in all documents to arbitrary topics.

Algorithm 3: Generative process for Latent Dirichlet Allocation (LDA).

```
1. Draw the topic distribution $\theta_j \sim \text{Dir}(\alpha)$ for each document $j$.
2. Draw the word distribution $\phi_k \sim \text{Dir}(\beta)$ for each topic $k$.
3. For each word $w_{ij}$ such that $i \in \left\{1, \ldots, N_{w_{d_j}}\right\}$
    (a) Sample a topic $z_{ij} \sim \text{Mult}(\theta_j)$.
    (b) Sample a word $w_{ij} \sim \text{Mult}(\phi_{z_{ij}})$.
```

The parameters for the LDA model can be derived using Gibbs sampling [69], variational Bayesian inference[19], or expectation propagation[124]. This manuscript will focus on the Gibbs sampling approach because it is one the most popular methods used in the literature, is the simplest to implement and in [6] has been shown to perform just as well, and sometimes better than variational Bayes estimation especially for large K. In the same work, Asuncion found that Variational Bayes and its collapsed variant are sometimes not as accurate as collapsed Gibbs sampling, but they are deterministic and faster to converge.

2.4.1 Parameter Estimation with the Collapsed Gibbs Sampler

In this section, we derive the collapsed Gibbs sampler for the LDA model. A complete step-by-step derivation is difficult to find in the literature, thus it is provided here for readers. What follows is adapted from the derivation provided in [28]. With Gibbs sampling, we do not explicitly estimate the model parameters ϕ or θ. Instead, we consider the posterior probability distribution of the assignment of words to topics, denoted $P(\mathbf{z}|\mathbf{w})$ and estimate the parameters from it.

From the definition of conditional probability,

$$P(\mathbf{z}|\mathbf{w}) = \frac{P(\mathbf{w},\mathbf{z})}{P(\mathbf{w})}$$

Note that ϕ is a collection of K multinomial distributions over the $|V|$ words. By placing a Dirichlet prior on ϕ, which is conjugate to the multinomial distributions ϕ and θ, we can compute the joint distribution $P(\mathbf{w},\mathbf{z})$ by integrating out the parameters ϕ and θ. To construct a collapsed Gibbs sampler, we want to compute the probability of assigning topic z_{ij} to w_{ij} given all of the other existing assignments of topics to word occurrences except the one at the moment of sampling.

$$\begin{aligned} P(z_{ij}|z_{\overline{ij}},\mathbf{w},\alpha,\beta) &= \frac{P(z_{ij},z_{\overline{ij}},\mathbf{w}|\alpha,\beta)}{P(z_{\overline{ij}},\mathbf{w}|\alpha,\beta)} \\ &\propto P(z_{ij},z_{\overline{ij}},\mathbf{w}|\alpha,\beta) \end{aligned}$$

but note that $z_{ij}, z_{\overline{ij}} = \mathbf{z}$ so we get

$$= P(\mathbf{w},\mathbf{z}|\alpha,\beta)$$

Now we must find the form of $P(\mathbf{w},\mathbf{z}|\alpha,\beta)$ to complete the sampler. Based on conjugacy, we have that

$$\begin{aligned} P(\mathbf{w},\mathbf{z}) &= \int\int P(\mathbf{w},\mathbf{z},\theta,\phi|\alpha,\beta)d\theta d\phi \\ &= \int\int P(\phi|\beta)P(\theta|\alpha)P(\mathbf{z}|\theta)P(\mathbf{w}|\phi,\mathbf{z})d\theta d\phi \end{aligned}$$

then from the independence suggested by the graphical model,

$$= \int P(\mathbf{z}|\theta)P(\theta|\alpha)d\theta \times \int P(\mathbf{w}|\phi,\mathbf{z})P(\phi|\beta)d\phi$$

which from the definition of the distributions is,

$$= \int \prod_{j=1}^{|D|} P(z_{\cdot j}|\theta_j)P(\theta_j|\alpha)d\theta \times \int \prod_{k=1}^{K} P(\phi_k|\beta) \prod_{j=1}^{|D|} \prod_{i=1}^{N_{w_{d_j}}} P(w_{ij}|\phi_{z_{ij}})d\phi_k$$

Now we can pull some of the products outside of the integrals,

$$= \prod_{j=1}^{|D|} \int P(z_{\cdot j}|\theta_j)P(\theta_j|\alpha)d\theta_j \times \prod_{k=1}^{K} \int P(\phi_k|\beta) \prod_{j=1}^{|D|} \prod_{i=1}^{N_{w_{d_j}}} P(z_{ij}|\phi_{z_{ij}})d\phi_k$$

Now substitute in the PDFs for the distributions defined in the LDA model specification presented in Algorithm 3:

$$\begin{aligned}
= \prod_{j=1}^{|D|} \int & \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{jk}^{\alpha_k - 1} \prod_{i=1}^{N_{w_{d_j}}} \theta_{j z_{ij}} d\theta_j \times \\
& \prod_{k=1}^{K} \int \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\sum_{i=1}^{|V|} \Gamma(\beta_i)} \prod_{i=1}^{|V|} \phi_{ik}^{\beta_i - 1} \prod_{j=1}^{|D|} \prod_{i=1}^{N_{w_{d_j}}} \phi_{w_{ij} z_{ij}} d\phi_k
\end{aligned}$$

Next, we can remove the dependency on $N_{w_{d_j}}$ by recognizing that z_{ij} is simply an integer representing a topic assignment. If we "regroup" terms based on these topics and let $n_{\cdot jk}$ be the number of word occurrences in d_j has been assigned to topic k, and let $n_{i \cdot k}$ be the number of times the word w_i has been assigned to topic k regardless of document then we can reduce the expression further.

$$\begin{aligned}
&= \prod_{j=1}^{|D|} \int \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{jk}^{\alpha_k - 1} \prod_{k=1}^{K} \theta_{jk}^{n_{\cdot jk}} d\theta_j \times \prod_{k=1}^{K} \int \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\sum_{i=1}^{|V|} \Gamma(\beta_i)} \prod_{i=1}^{|V|} \phi_{ik}^{\beta_i - 1} \prod_{i=1}^{|V|} \phi_{ik}^{n_{i \cdot k}} d\phi_k \\
&= \prod_{j=1}^{|D|} \int \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{jk}^{\alpha_k - 1 + n_{\cdot jk}} d\theta_j \times \prod_{k=1}^{K} \int \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\sum_{i=1}^{|V|} \Gamma(\beta_i)} \prod_{i=1}^{|V|} \phi_{ik}^{\beta_i - 1 + n_{i \cdot k}} d\phi_k
\end{aligned}$$

To cancel out the integrals (by getting them to sum to one), we must reorganize the inside of the integrals so that they match known probability distributions. At the same time, we cannot change the value of the expression. We do the following:

$$= \prod_{j=1}^{|D|} \int \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)}{\prod\limits_{k=1}^{K} \Gamma\left(\alpha_k + n_{\cdot jk}\right)} \frac{\prod\limits_{k=1}^{K} \Gamma\left(\alpha_k + n_{\cdot jk}\right)}{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)} \prod_{k=1}^{K} \theta_{jk}^{\alpha_k - 1 + n_{\cdot jk}} d\theta_j \times$$

$$\prod_{k=1}^{K} \int \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\sum_{i=1}^{|V|} \Gamma(\beta_i)} \frac{\Gamma\left(\sum\limits_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)}{\prod\limits_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})} \frac{\prod\limits_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})}{\Gamma\left(\sum\limits_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)} \prod_{i=1}^{|V|} \phi_{ik}^{\beta_i - 1 + n_{i\cdot k}} d\phi_k$$

$$= \prod_{j=1}^{|D|} \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\prod\limits_{k=1}^{K} \Gamma\left(\alpha_k + n_{\cdot jk}\right)}{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)} \int \frac{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)}{\prod\limits_{k=1}^{K} \Gamma\left(\alpha_k + n_{\cdot jk}\right)} \prod_{k=1}^{K} \theta_{jk}^{\alpha_k - 1 + n_{\cdot jk}} d\theta_j \times$$

$$\prod_{=1}^{K} \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\prod_{i=1}^{|V|} \Gamma(\beta_i)} \frac{\prod\limits_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})}{\Gamma\left(\sum\limits_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)} \int \frac{\Gamma\left(\sum\limits_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)}{\prod\limits_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})} \prod_{i=1}^{|V|} \phi_{ik}^{\beta_i - 1 + n_{i\cdot k}} d\phi_k$$

$$= \prod_{j=1}^{|D|} \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\prod\limits_{k=1}^{K} \Gamma\left(\alpha_k + n_{\cdot jk}\right)}{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)} \underbrace{\int \mathrm{Dir}\left(\theta_j | \alpha_k + n_{\cdot jk}\right) d\theta_j}_{=1} \times$$

$$\prod_{k=1}^{K} \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\prod_{i=1}^{|V|} \Gamma(\beta_i)} \frac{\prod\limits_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})}{\Gamma\left(\sum\limits_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)} \underbrace{\int \mathrm{Dir}\left(\phi_k | \beta_i + n_{i\cdot k}\right) d\phi_k}_{=1}$$

$$= \prod_{j=1}^{|D|} \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\prod\limits_{k=1}^{K} \Gamma\left(\alpha_k + n_{\cdot jk}\right)}{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)} \times \prod_{k=1}^{K} \frac{\Gamma\left(\sum_{i=1}^{|V|} \beta_i\right)}{\prod_{i=1}^{|V|} \Gamma(\beta_i)} \frac{\prod\limits_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})}{\Gamma\left(\sum\limits_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)}$$

Now we can eliminate terms that involve only the hyperparameters and not the data, to get the following:

$$\propto \prod_{j=1}^{|D|} \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + n_{\cdot jk})}{\Gamma\left(\sum_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)} \times \prod_{k=1}^{K} \frac{\prod_{i=1}^{|V|} \Gamma(\beta_i + n_{i\cdot k})}{\Gamma\left(\sum_{i=1}^{|V|} \beta_i + n_{i\cdot k}\right)}$$

Recall that the collapsed Gibbs sampler chooses a topic for some sample point (a word and document occurrence), say x and y. Then we split up the first product over documents into two factors – one relating to the sampling position x and one relating to all sampling positions not involving x.

$$\begin{aligned} = \prod_{j \neq y} \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + n_{\cdot jk})}{\Gamma\left(\sum_{k=1}^{K} \alpha_k + n_{\cdot jk}\right)} \times \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + n_{\cdot yk})}{\Gamma\left(\sum_{k=1}^{K} \alpha_k + n_{\cdot yk}\right)} \\ \times \prod_{k=1}^{K} \frac{\prod_{i \neq w_{xy}} \Gamma(n_{i \cdot k} + \beta_i) \Gamma(n_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n_{i \cdot k} + \beta_i)} \end{aligned}$$

Since the collapsed Gibbs sampler only depends on sampling point (x, y), we can remove all terms that are not directly dependent on the sampling point.

$$= \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + n_{\cdot yk})}{\Gamma\left(\sum_{k=1}^{K} \alpha_k + n_{\cdot yk}\right)} \times \prod_{k=1}^{K} \frac{\Gamma(n_{w_{xy} \cdot k} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n_{i \cdot k} + \beta_i)}$$

The final "trick" to the derivation is to introduce a new count variable n'_{ijk} similar to n_{ijk} but *excluding the current word/document pair sampling point* (x, y). Then for arbitrary i, j, k we have that $n_{ijk} = n'_{ijk} + 1$. Note that we break up the products to account for the fact that we are analyzing the current sample point separate from the others.

$$\begin{aligned} &= \frac{\prod_{k \neq z_{xy}} \Gamma(\alpha_k + n'_{\cdot yk}) \Gamma(\alpha_{z_{xy}} + n'_{\cdot yk} + 1)}{\Gamma(1 + \sum_{k=1}^{K} \alpha_k + n'_{\cdot yk})} \\ &\qquad \times \prod_{k \neq z_{xy}} \frac{\Gamma(n'_{w_{xy} \cdot k} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n'_{i \cdot k} + \beta_i)} \frac{\Gamma(n'_{\cdot w_{xy} \cdot z_{xy}} + \beta_{w_{xy}} + 1)}{\Gamma(1 + \sum_{i=1}^{|V|} n'_{i \cdot z_{xy}} + \beta_i)} \\ &= \frac{\prod_{k \neq z_{xy}} \Gamma(\alpha_k + n'_{\cdot yk}) \Gamma(\alpha_{z_{xy}} + n'_{\cdot yk})(\alpha_{z_{xy}} + n'_{\cdot yk})}{\Gamma(1 + \sum_{k=1}^{K} \alpha_k + n'_{\cdot yk})} \\ &\quad \times \prod_{k \neq z_{xy}} \frac{\Gamma(n'_{w_{xy} \cdot k} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n'_{i \cdot k} + \beta_i)} \frac{\Gamma(n'_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})(n'_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n'_{i \cdot z_{xy}} + \beta_i) \left(\sum_{i=1}^{|V|} n'_{i \cdot z_{xy}} + \beta_i\right)} \end{aligned}$$

Then we wrap some of the Gamma terms back into their original products:

$$
\begin{aligned}
&= \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + n'_{\cdot yk})(\alpha_{z_{xy}} + n'_{\cdot y z_{xy}})}{\Gamma(1 + \sum_{k=1}^{K} \alpha_k + n'_{\cdot yk})} \\
&\qquad \times \prod_{k=1}^{K} \frac{\Gamma(n'_{w_{xy} \cdot k} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n'_{i \cdot k} + \beta_i)} \frac{(n'_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})}{\left(\sum_{i=1}^{|V|} n'_{i \cdot z_{xy}} + \beta_i\right)} \\
&= \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + n'_{\cdot yk})(\alpha_{z_{xy}} + n'_{\cdot y z_{xy}})}{\Gamma(\sum_{k=1}^{K} \alpha_k + n'_{\cdot yk})(\sum_{k=1}^{K} \alpha_k + n'_{\cdot yk})} \\
&\qquad \times \prod_{k=1}^{K} \frac{\Gamma(n'_{w_{xy} \cdot k} + \beta_{w_{xy}})}{\Gamma(\sum_{i=1}^{|V|} n'_{i \cdot k} + \beta_i)} \frac{(n'_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})}{\left(\sum_{i=1}^{|V|} n'_{i \cdot z_{xy}} + \beta_i\right)}
\end{aligned}
$$

Finally, note that most of the remaining terms are constant over all z_{xy} and can be dropped yielding:

$$
\begin{aligned}
&\propto \frac{(\alpha_{z_{xy}} + n'_{\cdot yk})(n'_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})}{\left(\sum_{i=1}^{|V|} n'_{i \cdot z_{xy}} + \beta_i\right)\left(\sum_{k=1}^{K} \alpha_k + n'_{\cdot yk}\right)} \\
&= \frac{(\alpha_{z_{xy}} + n'_{\cdot yk})(n'_{w_{xy} \cdot z_{xy}} + \beta_{w_{xy}})}{\left(n'_{\cdot \cdot z_{xy}} + \sum_{i=1}^{|V|} \beta_i\right)\left(n'_{\cdot y \cdot} + \sum_{k=1}^{K} \alpha_k\right)}
\end{aligned}
\tag{2.6}
$$

Recall that the hyperparameters α and β can take on a single uniform value, or can be represented as a K-vector and a $|V|$-vector respectively. The previous derivation assumed differing values for different topics and words respectively. If we restrict α and β to be a value such that $\alpha = \alpha_1 = \alpha_2 = \ldots = \alpha_K$ and $\beta = \beta_1 = \beta_2 = \ldots = \beta_V$ then the conditional probability reduces to

$$
\propto \frac{(\alpha + n'_{\cdot yk})(n'_{w_{xy} \cdot z_{xy}} + \beta)}{\left(|V|\beta + \sum_{i=1}^{|V|} n'_{i \cdot z_{xy}}\right)\left(K\alpha + \sum_{k=1}^{K} n'_{\cdot yk}\right)} = \frac{(\alpha + n'_{\cdot yk})(n'_{w_{xy} \cdot z_{xy}} + \beta)}{\left(|V|\beta + n'_{\cdot \cdot z_{xy}}\right)\left(K\alpha + n'_{\cdot y \cdot}\right)} \tag{2.7}
$$

where

$$\hat{\theta}_{jk} = \frac{\alpha + n'_{\cdot jk}}{K\alpha + n_{\cdot jk}}$$

and

$$\hat{\phi}_{ik} = \frac{n'_{i\cdot k} + \beta}{n'_{\cdot\cdot k} + |V|\beta}$$

The selection of the hyperparameters for LDA has not yielded much practical discussion in the literature. Blei explains in [18] that for a symmetric Dirichlet distribution, lower values of α yield sparser topics such that documents contain a small number of topics, or possibly even only one. Higher values of α encourage more mixing with each document containing a large proportion of the topic space. For the asymmetric Dirichlet distribution, higher values of α impose a more specific topic distribution for each document. Likewise, Blei explains that higher values for β place more words into a topic making each topic perhaps more vague; lower values for β cluster less words into a topic. In the asymmetric case, higher values for β impose a more specific collection of words into a specific topic. In summary, in the asymmetric case, certain topics and words are more likely a priori. Most of the literature discusses the symmetric Dirichlet case, something that was also noticed and discussed in [168]. In [6], Asuncion at al. discuss several data-driven methods for estimating α and β including placing a Gamma prior on the hyperparameters and using Minka's fixed-point iterations[125], Newton-Raphson[169], sampling[163] and using grid search with a validation set. Asuncion et al. also found that hyperparameter selection depends on the method used to estimate the parameters for LDA, with Collapsed Variational Bayes and MAP performing best when the hyperparameters are offset by 0.5 for α and 1 for β.

It is common knowledge within the topic modeling community that the number of topics K is typically specified a priori and is a value chosen based on the context of the data, or the desired granularity of topics in the LDA model. Researchers have developed several methods for stochastically choosing K mostly using Bayesian nonparametric algorithms. Griffiths and Steyvers[69] fit several independent models using different values of K and then choose the value of K that yields the highest log-likelihood. Heinrich[76] discusses Infinite LDA, the use of hierarchical Dirichlet processes (HDP) to estimate K. Finally, Griffiths and Ghahramani use the Indian Buffet Process in [68] to estimate K for latent feature models.

2.4.2 Parameter Estimation via Collapsed Variational Inference

As mentioned previously, this manuscript will focus on the collapsed Gibbs sampling approach for estimating the LDA model parameters. This section provides a high-level explanation of collapsed variational Bayes which is a variation of the original method provided in [19].

With variational inference[56], one estimates the posterior, $P(\theta, \phi | \mathbf{w}, \mathbf{z}, \alpha, \beta)$, in this case, with a distribution $\hat{Q}(\mathbf{z}, \theta, \phi)$, such that $\hat{Q}$ is "close" to the posterior. Collapsed Variational Bayes, presented by Teh in [163], first marginalizes out the model parameters θ and ϕ, so an appropriate choice for $\hat{Q}$ is

$$\hat{Q}(\mathbf{z}, \theta, \phi) = \hat{Q}(\theta, \phi | \mathbf{z}) \prod_{ij} \hat{Q}\left(z_{ij} | \gamma_{ij}\right) \tag{2.8}$$

where the terms $\hat{Q}(z_{ij}|\gamma_{ij})$ are multinomial with parameter γ_{ij}. The main restriction with the choice of $\hat{Q}$ is that it must come from a family of distributions that is similar to P, but simpler in structure. The goal in variational inference is to choose $\hat{Q}$ as close to P as possible using Kullback-Leibler divergence as the distance function

$$\log P(\mathbf{w}, \mathbf{z}, \theta, \phi | \alpha, \beta) = D_{KL}\left(Q || P\right) + \mathscr{L}(Q)$$

where $\log P$ is the so-called evidence and $\mathscr{L}(Q)$ is the variational free energy. Given the approximate posterior in (2.8), the variational free energy is computed as

$$\begin{aligned}
\mathscr{L}\left(\hat{Q}(\mathbf{z})\hat{Q}(\theta, \phi|\mathbf{z})\right) &= E_{\hat{Q}(\mathbf{z})\hat{Q}(\theta,\phi|\mathbf{z})}\left[-\log P(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta)\right] - \mathscr{H}\left(\hat{Q}(\mathbf{z})\hat{Q}(\theta, \phi|\mathbf{z})\right) \\
&= E_{\hat{Q}(\mathbf{z})}\left[E_{\hat{Q}(\mathbf{z})}\left[-\log P(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta)\right] - \mathscr{H}\left(\hat{Q}(\theta, \phi|\mathbf{z})\right)\right] \\
&\qquad - \mathscr{H}\left(\hat{Q}(\mathbf{z})\right)
\end{aligned}$$

Then we minimize $\mathscr{L}$ with respect to $\hat{Q}(\theta, \phi|\mathbf{z})$ followed by $\hat{Q}(\mathbf{z})$ with the minimum being achieved when $\hat{Q}(\theta, \phi|\mathbf{z}) = P(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta)$. Teh argues that

$$\begin{aligned}\mathscr{L}(\hat{Q}(\mathbf{z})) &\equiv \min_{\hat{Q}(\theta,\phi|\mathbf{z})} \mathscr{L}(\hat{Q}(\mathbf{z})\hat{Q}(\theta, \phi|\mathbf{z})) \\ &= E_{\hat{Q}(\mathbf{z})}\left[-\log P(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta)\right] - \mathscr{H}(\hat{Q}(\mathbf{z})) \qquad (2.9)\end{aligned}$$

Then by minimizing (2.9) with respect to γ and performing some algebra the update equation for the variational parameters γ_{ijk} can be estimated as

$$\begin{aligned}\gamma_{ijk} &= \hat{Q}(z_{ij} = k) \\ &\propto \exp\left(E_{\hat{Q}(\mathbf{z}_{\overline{ij}})}\left[\log\left(\alpha + n'_{\cdot jk}\right) + \log\left(\beta + n'_{i\cdot k}\right) - \log\left(|V|\beta + n'_{\cdot\cdot k}\right)\right]\right)\end{aligned}$$

Teh then computes the expectations using a Gaussian approximation. The mechanics of the approximation are beyond the scope of this manuscript, but when taken into account yield the following update equation for γ_{ijk}

$$\begin{aligned}\gamma_{ijk} \propto\ & \frac{\left(\alpha + E_{\hat{Q}}(n'_{\cdot jk})\right)\left(\beta + E_{\hat{Q}}(n'_{i\cdot k})\right)}{|V|\beta + E_{\hat{Q}}(n'_{\cdot\cdot k})} \times \\ & \exp\left(-\frac{\mathrm{Var}_{\hat{Q}}(n'_{\cdot jk})}{2(\alpha + E_{\hat{Q}}(n'_{\cdot jk}))^2} - \frac{\mathrm{Var}_{\hat{Q}}(n'_{i\cdot k})}{2(\beta + E_{\hat{Q}}(n'_{i\cdot k}))^2} + \frac{\mathrm{Var}_{\hat{Q}}(n'_{\cdot\cdot k})}{2(|V|\beta + E_{\hat{Q}}(n'_{\cdot\cdot k}))^2}\right)\end{aligned}$$

where the fragment outside of the exponentiation looks very similar to (2.7) and the fragment inside the exponentiation is a small correction for variance. For more details on Variational Bayes and collapsed Variational Bayes, see [19], [163], [6] and [56].

2.4.3 A Latent Dirichlet Allocation Algorithm

Now that we have discussed the estimation of the posterior probability using the collapsed Gibbs sampler, we can discuss how to use the LDA algorithm.

As input we have a set of word vectors across the corpus, one for each document, the hyperparameters α and β and the number of topics K. Throughout the process, we keep counters for the number of word occurrences in document d_j that have been assigned to topic k, $n_{\cdot jk}$, and the number of times word w_{ij} has been assigned to topic k, $n_{i \cdot k}$. For efficiency, we also keep a counter for their sums $N_{w_{d_j}}$ – the number of word occurrences in document d_j, and $n_{\cdot \cdot k}$ the total number of word occurrences assigned to topic k. The output is the topic association for every word in every document, denoted $\mathbf{z}$, and also the multinomial parameters ϕ and θ.

First, we initialize all of the global counters to zero. Then, for every word occurrence in every document, we randomly sample an initial topic and increment the appropriate counters for each selection. Then we run the actual collapsed Gibbs sampler until convergence, recording the results of the final M iterations – those iterations that are after burn-in. For every word occurrence in every document, we first decrement all of our counters to take the current sampling position into account. Then, we sample a new topic for the word occurrence according to (2.6). Finally we increment our counters to take into account this new sampling. We repeat this process until the parameters do not change, or the associated topic labels become stable. Upon convergence, we output the model parameters ϕ and θ as well as the topic assignments $\mathbf{z}$. The full algorithm[75] is provided as Algorithm 4.

In Blei, Ng and Jordan's original work, they compared LDA with PLSA, mixture of unigrams and unigram model using the perplexity[113] measure on two different corpora: TREC AP which consisted of 2,500 news articles with 37,871 unique terms, and CRAN consisting of 1,400 technical abstracts with 7,747 unique terms. They found that LDA consistently had significantly lower perplexity with $K > 20$ and all methods performed similarly for $K \leq 20$. Additionally, the perplexity measure for LDA decreased monotonically, whereas the other methods remained more or less constant across K. In a separate analysis in the same work, the researches showed that classification error for LDA was consistently lower than the mixture of unigrams model and Naïve Bayes with both LDA and the mixture of unigrams model slowly increasing in error as $K \to \infty$ whereas Naïve Bayes remained constant at around 0.13. The classification error for LDA varied from around 0.1 to 0.12 and for the mixture of unigrams it varied from 0.11 to 0.13.

Algorithm 4: Latent Dirichlet Allocation via collapsed Gibbs sampling.

```
Data: counts $n_{i\cdot k}, n_{\cdot jk}, n_{i\cdot\cdot}, n_{\cdot\cdot k}$
Input  : word vectors for documents $\mathbf{w}$, number of topics $K$, model hyperparameters $\alpha, \beta$.
Output: model parameter estimates $\phi, \theta$ and document/word occurrence topic assignments $\mathbf{z}$
set $n_{i\cdot k} = n_{\cdot jk} = n_{i\cdot\cdot} = n_{\cdot\cdot k} = 0$;
forall documents $i \in [1, D]$ do
    forall word occurrences $j \in [1, N_i]$ in document $i$ do
        draw topic assignment $k \leftarrow z_{ij} \sim \text{Mult}\left(\frac{1}{K}\right)$ ;
        //increment counters ;
        $n_{i\cdot k}$++; $n_{\cdot jk}$++; $n_{i\cdot\cdot}$++; $n_{\cdot\cdot k}$++;
    end
end
repeat
    forall documents $i \in [1, D]$ do
        forall word occurrences $j \in [1, N_i]$ in document $i$ do
            //Decrement counters to account for the current topic assignments. ;
            $n_{i\cdot k}$--; $n_{\cdot jk}$--; $n_{i\cdot\cdot}$--; $n_{\cdot\cdot k}$--;
            //sampling a topic index according to (2.6) ;
            select a topic $k$ from $P\left(z_{ij} | z_{\overline{ij}}, \mathbf{w}, \alpha, \beta\right)$ ;
            //increment counters according to this new assignment ;
            $n_{i\cdot k}$++; $n_{\cdot jk}$++; $n_{i\cdot\cdot}$++; $n_{\cdot\cdot k}$++;
        end
    end
    if converged then
        output $\phi$ ;
        output $\theta$ ;
    end
until convergence;
```

Several implementations of classic Latent Dirichlet Allocation exist in a variety of languages. `gensim`[148] is a Python package that provides an implementation of LDA and multiple pre-processing features including the Hierarchical Dirichlet Process. Several implementations of LDA exist for Java: JGibbLDA[136] a simple no-frills implementation, MALLET[119] a fully featured NLP toolkit, and Mahout[55] a scalable machine learning library that can interoperate with Hadoop. The authors of JGibbLDA also provide a C++ implementation called GibbsLDA++[135]. The R package `topicmodels`[70] provides an interface to the original C implementation from Blei[17]. Jonathan Chang also developed the `lda`[31] package for R based on Blei's original code and has migrated it to Julia[9] as `TopicModels.jl`; only the Julia version is now maintained. Chang's package also has implementations for several variations of LDA discussed in this chapter as well.

[9] `https://github.com/slycoder/TopicModels.jl`

Latent Dirichlet Allocation has been extended and modified to cover various other use cases. The remainder of this chapter focuses on introducing these use cases and the generative processes associated with each of them. Natural language processing with Latent Dirichlet Allocation is a continuously growing field, and the methods mentioned throughout the rest of this chapter only touch the surface of what is possible.

2.5 Correlated Topic Model

Classic Latent Dirichlet Allocation models words as being generated from topics – the correlation between words and a document is not strongly considered. One major limitation with classic Latent Dirichlet Allocation is that all topics are assumed to be *independent* of each other due to the reliance on the Dirichlet distribution to explain the variance among topic proportions[15]. For example, in classic LDA, if a document is about statistics, this gives us no other information about what other topics may be in the document. In reality, a document containing words about machine learning is also likely to contain words about statistics. That is, the topics "machine learning" and "statistics" are correlated and not independent. In Blei and Lafferty's *Correlated Topic Model*[21], the correlation among topic proportions is modeled using the logistic normal distribution[4]. That is, the authors replace the Dirichlet distribution that models the topic proportions with a logistic normal distribution which models their proportions and correlations by introducing a covariance structure. Similar to classic LDA, θ_j represents the probability distribution of topics in document d_j. In CTM, we instead consider a parameterization of this multinomial distribution $\eta = \log\left(\frac{\theta_k}{\theta_K}\right)$ and

$$\theta = f(\eta) = \frac{\exp\{\eta\}}{\sum_k \eta_k}$$

CTM uses the generative process presented in Algorithm 5.

Algorithm 5: Generative model for CTM.

```
1. Sample $\eta_j | \{\mu, \Sigma\} \sim N(\mu, \Sigma)$.

2. For $1 \leq i \leq N_{w_{d_j}}$:
    (a) Sample topic assignment $z_{ij} | \eta_j \sim \text{Mult}\left(f\left(\eta_j\right)\right)$.
    (b) Sample word $w_{ij} | \{z_{ij}, \beta\} \sim \text{Mult}\left(\beta_{z_{ij}}\right)$.
```

Unfortunately, one of the aspects of classic Latent Dirichlet Allocation is missing with the Correlated Topic Model – we lose conjugacy. In classic LDA, we can work with the conjugacy between the multinomial and Dirichlet distributions, but by replacing the Dirichlet distribution with the logistic normal distribution, conjugacy is no longer present[21]. For this reason, Gibbs sampling is said to be not possible for CTM, and instead a variational approach is preferred. However, [123] provides a Gibbs sampling algorithm that uses block Gibbs sampling where the topic assignments z and the logistic-normal parameters β are estimated using all other variables in the data. The graphical model for CTM is displayed as Figure 2.3.

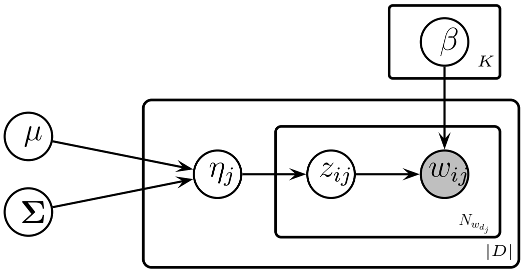


Figure 2.3: Graphical model depicting Correlated Topic Model (CTM).

Blei and Lafferty[21] performed an experiment using a correlated topic model with 100 topics on 16,351 articles from *Science* and compared the results to classic Latent Dirichlet Allocation. By analyzing the held-out log-likelihood, they found that both LDA and CTM performed similarly for up to $K = 30$ topics. When they used more than 30 topics, CTM outperformed LDA. Blei and Lafferty also showed that CTM required about 10% less words to be analyzed than LDA for CTM to accurately predict the topic assignments of the remaining words. In [123], Minmo and Wallach compared their blocked Gibbs sampler implementation to the standard LDA model estimated by collapsed Gibbs sampling and found the results to be indistinguishable. One would argue that CTM should exhibit *better* predictive performance than classic LDA, so these results should be taken lightly.

Few implementations for CTM exist currently. The `topicmodels` package[71] for R provides an implementation using the original variational EM method presented in [21]. The R `lda` package[31] provides an implementation using collapsed Gibbs sampling and is a wrapper around Blei's original C implementation [16] `ctm-c`.

2.6 Relational Topic Model

In classic LDA the only connections among documents are implicitly defined by topics, but there are many cases in which text documents are connected to each other explicitly. For example, Facebook status updates may be linked to one another via the friend graph where an edge between two friends denotes a possible connection in idea or interest[97]. The relational topic model (RTM)[30] models not only the words in a set of documents, but the explicit relationships among documents such as co-authorship[184]. Much literature exists analyzing only the structure of networks, for example [173], but the purpose of RTM is to augment this network structure with the content of each node in the network.

As with CTM, RTM begins with an LDA recipe by generating documents from a finite set of topics. Then, RTM proceeds by modeling the link between a pair of documents as a binary variable denoting whether or not there is a connection between the pair of documents. Similar to LDA, the hyperparameters of RTM are K-distributions of terms β, and a K-dimensional parameter α. RTM assumes a function Ψ that yields link presence probabilities based on the topics assigned to words in every pair of documents, and assumes that words $\mathbf{w}$ and binary links $\mathbf{y}$ are generated by the process described in Algorithm 6; the graphical model for RTM is displayed as Figure 2.4.

Algorithm 6: Generative model for Relational Topic Model (RTM).

```
1. For every document $d_j$,
    (a) Draw the topic proportions $\theta_j|\alpha \sim \mathrm{Dir}(\alpha)$.
    (b) For every word occurrence $w_{ij}$ in the document:
        i. Sample a topic assignment $z_{ij}|\theta_j \sim \mathrm{Mult}(\theta_j)$.
        ii. Sample a word $w_{ij}|z_{ij}, \beta \sim \mathrm{Mult}(\beta_{z_{ij}})$.
2. For every pair of documents $d_j$ and $d_{j\prime}$,
    (a) Sample an indicator of whether or not a link exists between $d$ and $d'$: $y|z_{\cdot j}, z_{\cdot j'} \sim \Psi(\cdot|z_{\cdot j}, z_{\cdot j'})$.
```

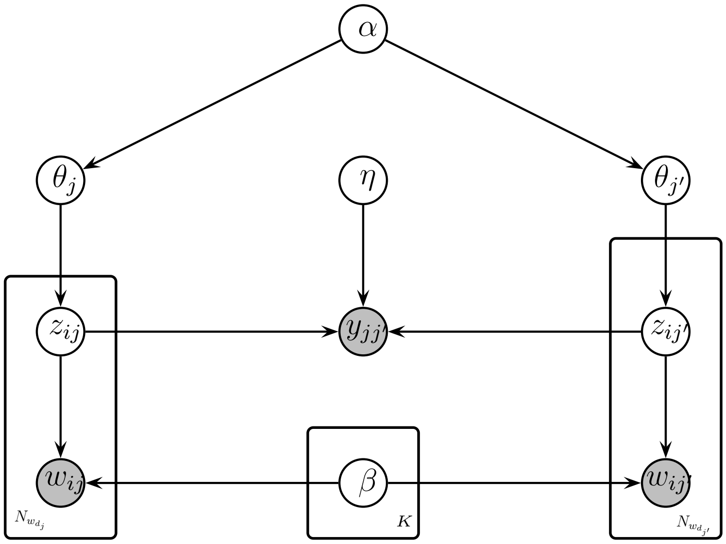


Figure 2.4: Graphical model depicting Relational Topic Model (RTM).

Like with CTM, the relational topic model requires variational methods to estimate the model parameters; for information on deriving the estimates for the model parameters, see [30]. Chang and Blei studied the use of RTM on three textual datasets: abstracts from the Cora[156] research paper database, WebKB[156] and PNAS[165] as well as the links among the documents in the each dataset. The authors evaluated the predictive distribution using two different link functions Ψ – logistic and exponential, and compared the performance to three alternatives: a baseline model where words and links are independent, a "Mixed-Membership"[127] model which extends the mixed membership stochastic block model[3], and finally a combination of classic LDA and logistic regression. Without surprise, RTM performed better on held-out log-likelihood than all three alternative models on all three datasets. RTM performed on average 5% better than the baseline and the LDA/logistic regression hybrid. The exponential link outperforms the logistic link for WebKD and PNAS but is approximately equivalent for the Cora dataset. While RTM appears to be a very powerful mechanism for modeling words and topics as well as the links among documents, the author of this manuscript does not feel that it would perform well for social media in general because there is an innate reliance on network theory and not solely on text.

2.7 Hierarchical Latent Dirichlet Allocation (hLDA)

Classic LDA has also been extended to fit topics from text into hierarchies and taxonomies. LDA treats topics as a set where each topic is a distribution over words. A hierarchical topic model treats a topic space as a tree with a root node containing all topics, and leaf nodes containing very precise topic descriptions. Interior nodes in such a hierarchy represent varying levels of granularity in topics[176]. As an example, one such topic hierarchy may have an interior node that quantitatively defines *general sports* and several leaf descendants representing specific sports like *baseball*, *football* and others. One of the daunting aspects of fitting hierarchies to text is that the problem is so open-ended and there are an infinite number of such hierarchies that can fit the data[60][67]. Hierarchical Latent Dirichlet Allocation (hLDA) is proposed in [67] and forms the backbone for most of the modern literature and research into topic modeling with hierarchies. As a generative model, a large space of hierarchical models that fit the text data are considered. The caveat though is that such a space of all possible hierarchies is infinite. The researchers use the *Chinese Restaurant Process* to search the space of all possible hierarchies.

2.7.1 Chinese Restaurant Process

Jim Pitman and Lester Dubins coined the phrase Chinese Restaurant Process[137] in the early eighties as a reference to the many Chinese restaurants in San Francisco, all of which appeared (to them) to have infinite seating. The Chinese Restaurant Process attempts to model how the restaurants seat patrons and has been used in mixture models to assist in determining the number of mixture components as in [177]. Imagine M patrons at a Chinese restaurant. The first patron sits at the first table and then the mth patron chooses a table according to the process in Algorithm 7 where γ is a parameter to be optimized and m_k is the number of patrons already sitting at table k.

Algorithm 7: Chinese Restaurant Process: Selecting a Table for mth Patron

$$
\begin{aligned}
P(\text{choose occupied table } i|\text{existing seated customers}) &= \frac{m_k}{\gamma + m - 1} \\
P(\text{choose next unoccupied table}|\text{existing seated customers}) &= \frac{\gamma}{\gamma + m - 1}
\end{aligned}
$$

Griffiths et al. state that CRP is useful for problems involving unknown numbers of components because it places a one-to-one correspondence between tables and components (or topics), and a one-to-many relationship between components and texts (tables and patrons seated at the table). For purposes of fitting a hierarchy, the authors use an altered formulation of CRP, the Nested Chinese Restaurant Process (nCRP), where every data point is associated with multiple components/topics along a path in such a hierarchy. From [20], nCRP realizes the following scenario. Suppose that there are an infinite number of infinite seating Chinese restaurants. Designate one restaurant as the “root” restaurant. Each table in the root restaurant contains a card on it that points to another unvisited Chinese restaurant in the city. The tourist then goes to that restaurant, gets assigned a table (using CRP), and that table has a card on it specifying yet another Chinese restaurant. Imagine this process continues infinitely and each Chinese restaurant can only be visited once. One can see that this traversal of Chinese restaurants forms a tree with infinite branching factor and infinite depth with each node in the tree representing one restaurant. After M tourists have participated, we have one subtree out of the infinite set of all trees. nCRP provides a method for determining a prior on the infinite space of trees and is used as a component of hLDA. The generative process for hLDA is presented in Algorithm 8 and illustrated as Figure 2.5. First, each document is constructed by sampling a length L path through the infinite tree. Then, draw an L-vector of topic proportions θ from a Dirichlet distribution with corpus hyperparameter α. Finally, sample words for the document from the topics associated with the nodes along the length L path.

Algorithm 8: Generative model for Hierarchical Latent Dirichlet Allocation (hLDA)

```
1. Define $c_1$ as the root restaurant.
2. For each level in the path $l \in \{2, \ldots, L\}$.
    (a) Sample a table from restaurant $c_{l-1}$ using Algorithm 7 and set it as $c_l$,
3. Draw an $L$-vector $\theta_l$ of topic proportions from $\mathrm{Dir}(\alpha)$.
4. For each word occurrence $i \in \{1, \ldots, N_{w_l}\}$:
    (a) Draw $z_l \in \{1, \ldots L\}$ from $\mathrm{Mult}(\theta_l)$.
    (b) Draw $w_{il}$ from the topic associated with restaurant $c_{z_l}$.
```

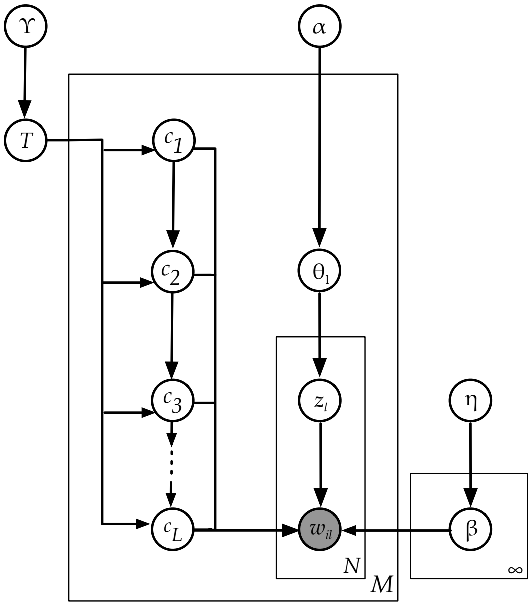


Figure 2.5: Graphical model depicting Hierarchical LDA (hLDA).

The original incarnation of hLDA[67] developed by Griffths et al. used a Dirichlet prior on the topic proportions θ thus allowing the use of Gibbs sampling to estimate model parameters unlike other LDA variants such as CTM and RTM. For evaluation of finite depth hLDA on real text, the researchers performed a qualitative and visual analysis of a three level hierarchy generated from 1,717 NIPS abstracts from 1987-1999 with a vocabulary of 1,600 terms. A later manuscript used hLDA abstracts from the *Journal of the ACM* from 1987-2004 as a corpus to compare LDA to hLDA[20]. The researchers found that hLDA yielded a near constant held-out log-likelihood around -20,000 for all K whereas LDA consistently exhibited a lower log-likelihood for all K. Their results showed that the log-likelihood increased similar to $\sqrt{K}$ but was always consistently lower than the log-likelihood from hLDA.

Several variations of hLDA exist for more specialized purposes. In [20], Blei, Griffiths and Jordan do not fix the cardinality of the set of topics K that they fit to the data. Instead, they used a stick-breaking process that imposes a reliance on the GEM distribution[137] to sample topic proportions θ. Chang and Blei combined hLDA with their previous work on RTM to develop a combined model that allows for fine-grained topics in a network setting; the application used in their work was a network of communication over the United States[32] . Researchers at Yahoo! and the University of Pennsylvania used a very similar formulation of LDA along with Pachinko allocation[106] in [90] to disambiguate named entities using weakly semi-supervised learning. Mao et al. describe in [115] semi-supervised hierarchical latent Dirichlet allocation (SSHLDA) which grows a taxonomy using a small training set of documents rather than using unlabeled data. Though researchers have used hLDA for a wide variety of text analysis involving hierarchies of topics, Pujara and Skomoroch commented in [141] that current techniques tend to focus on small corpora with thousands of documents and terms and not on large web scale sources.

2.8 Supervised Variations of LDA

The generalized linear model (GLM) is one of the most widely used statistical techniques. It allows one to model some dependent variable $\mathbf{y}$ as a linear combination of a set of independent variables $\mathbf{x}$ with real coefficients β. Analysts use different types of GLM models depending on the distribution and domain of the response variable $\mathbf{y}$ and the residuals. Typically, the independent variables $\mathbf{x}$ constitute real measurements but in the case of principal components regression, the independent variables are principal components, each containing a linear combination of correlated variables[116]. Supervised LDA (sLDA)[117] is an extension of classic LDA that fits topics according to the distribution of the dependent variable $\mathbf{y}$. sLDA is superficially similar to principal components regression in that clusters of words are used to explain the distribution of the dependent variable rather than just the independent features themselves. sLDA, like other variants of LDA, assumes a generative process that fits words into topics, but based on the distribution of the dependent variable. Unlike classic LDA which only models words into topics and is unsupervised, sLDA maps a response to each document and is thus supervised. Some examples of response variable types and use cases are

- ordinal discrete values such as scores on an essay, or the number of stars awarded to a restaurant in a Yelp review[109][10],
- counts such as the number of users that "Digg" a particular webpage on the popular news sharing site `digg.com`[104], a classic example developed by the father of sLDA, David Blei,
- binary indicators such as spam indication[14], or sentiment polarity[94],
- nominal topic classification.

The generative process for sLDA for every document d_j is in Algorithm 9 and the graphical model is depicted as Figure 2.6.

Algorithm 9: Generative Model for Supervised Latent Dirichlet Allocation (sLDA)

```
1. For each document $d_j$ sample a topic proportion vector $\theta_j|\alpha \sim \mathrm{Dir}(\alpha)$.
2. For each word occurrence in document $d_j$
    (a) Sample topic assignments $z_{ij}|\theta_j \sim \mathrm{Mult}(\theta_j)$.
    (b) Sample word $w_{ij}|z_{ij}, \beta_{1:K} \sim \mathrm{Mult}(\beta_{z_{ij}})$.
3. Sample response variable $y_j|z_{ij}, \eta, \delta \sim \mathrm{GLM}(\bar{z}, \eta, \delta)$.
```

where $\bar{z} = \frac{1}{N_{w_{d_j}}} \sum_{i=1}^{N_{w_{d_j}}} z_{ij}$, η is a vector of regression coefficients and δ is some scale parameter such as the variance σ^2 for the normal distribution. GLM$(\cdot)$ refers to the canonical link function for the GLM model in question.

For evaluation, the researchers studied two regression problems: predicting movie ratings (ordinal and discrete) and predicting the number of "Diggs" a webpage submission receives

[10] `https://www.yelp.com/academic_dataset`

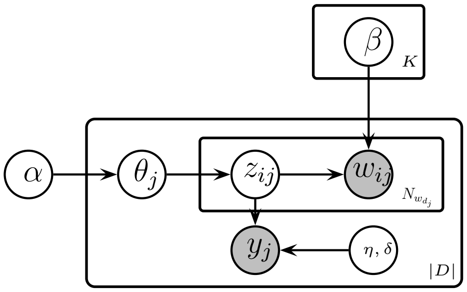


Figure 2.6: Graphical model depicting supervised LDA (sLDA).

on the Digg[11] service (counts). The researchers took the logarithm of the data to force normality so that they could use a standard linear model and avoid complicated computation with the GLM that was beyond the scope of their work. They used R^2 from a five-fold cross validation to compare the results on both datasets using sLDA, standard linear regression using LDA topics as the predictors, and LASSO with words as features instead of topics. For the movie prediction problem, sLDA consistently had a higher R^2 and held-out log-likelihood regardless of number of topics. For the "Diggs" prediction problem on the other hand, sLDA performed better than standard regression with LDA topics as predictors only when the researchers specified a small number of topics (approximately eight or less); both methods were comparable for larger numbers of topics. Finally, sLDA performed 8% better and 9.4% better on R^2 than LASSO for the Digg problem and the movie rating problem respectively. Despite these positive results, Hughes[82] states that the power of supervision decreases as the number of words per document increases. This is a very important and timely finding as it suggests that sLDA is a good candidate classifier to use for the research presented in the manuscript.

2.8.1 Other Supervised LDA Approaches

Researchers in natural language processing and computer vision have extended LDA in similar ways to form other supervised and semi-supervised variations of LDA aside from sLDA. Many of these methods are very similar to the technique presented as sLDA, but with

[11] `http://www.digg.com`

differences in particular nuances of the model or model assumptions. In [101], Lacoste-Julien et al. present DiscLDA, a discriminative (rather than generative) version of Latent Dirichlet Allocation that introduces a class-dependent linear transformation on the topic mixture proportions θ for dimension reduction. The topic mixture proportions are transformed to a mixture of Dirichlet distributions rather than relying on a single distribution. The other main difference between DiscLDA and sLDA is in parameter estimation; sLDA maximizes joint likelihood while DiscLDA maximizes conditional likelihood. Lacoste-Julien et al. compared the document classification performance of features generated from DiscLDA using a block matrix as the linear transformation, versus a classic LDA model with 110 topics using the classic *20 Newsgroups*[12] text corpus of Usenet messages. In an experiment, the researchers attempted to classify messages as belonging to `alt.atheism` or `talk.religion.misc`, a reportedly difficult task since the contents of both newsgroups is very similar in vocabulary. Using the topic proportions from DiscLDA with a binary SVM classifier, and using DiscLDA independently both yielded an improvement in binary misclassification rate of 3% over using the topic mixture proportions from classic (unsupervised) LDA with SVM.

According to [187], one disadvantage to DiscLDA and sLDA is that features extracted from them must pass to another classifier for them to be useful in document classification. In [187], Zhu et al. introduce minimum entropy discriminative LDA, or MedLDA for short. MedLDA is a maximum-margin approach similar to SVM which uses side information and the LDA model structure for document classification. Whereas sLDA learns a point estimate of the set of GLM parameters η, MedLDA uses a Bayesian approach to learn a distribution $q(\eta)$ that maximizes the margin. For a better perspective, the researchers described their method as a hybrid – a combination of a Bayesian sLDA where η is sampled from a prior $q(\eta)$, and ε-insensitive support vector regression[158]. For evaluation, the researchers trained a 110 topic MedLDA model and a 110 topic classic LDA model on the 20 Newsgroups dataset. They showed that qualitatively, MedLDA produced per-class distributions that were sharper and sparser for topics that had the most discriminative power whereas LDA overfit by discovering per-class distributions that modeled the fine details of the document

[12] `http://qwone.com/~jason/20Newsgroups/`

without regard to discriminative power. Quantitiatively, MedLDA was found to perform significantly better on the binary classification (`alt.atheism` vs. `talk.religion.misc`) task than DiscLDA, sLDA and a hybrid LDA/SVM solution. MedLDA with 80 topics or more was also found to perform significantly better than DiscLDA, sLDA and LDA/SVM with respect to accuracy for the multiclass problem using each of the 20 newsgroups as a class. Finally, the researchers compared MedLDA to sLDA and LDA for regression using the movie ratings data mentioned earlier and found that MedLDA and sLDA performed similarly with respect to per-word likelihood and predictive R^2, but consistently better than LDA. In summary, the goal of MedLDA is to find a latent topic representation of documents in a corpus that explain the data well, and also predict correctly with a large enough margin.

Ramage et al. presented Labeled LDA (L-LDA) in [144] which is a model for credit attribution, or tagging text with tokens. The authors state that Labeled LDA allows for information retrieval systems to attach tags to text, and extract snippets of text based on these tags for use in search engines. Unlike classic LDA and DiscLDA, Labeled LDA is appropriate for multi-label classification because it associates with each document a set of labels as training data rather than just one. For credit attribution it is simpler to assign labels to words in a document rather than a latent topic that may be less interpretable. For evaluation, the researchers took a sample of 29 web pages saved to the social bookmarking site `del.icio.us`, each containing two or more tags from a predefined set and compared the tags extracted by L-LDA to the results from a series of one-vs-rest SVMs. Human raters found that the tags generated by L-LDA were "superior" about 48% of the time and 14% of the time for the series of SVM classifiers. The authors found that L-LDA outperformed the SVM method for multi-label classification of web pages using macro-average F1 and micro-average F1 as evaluation metrics.

Several other variations of sLDA have been proposed, all with more specific use cases than the variations previously discussed. Multilingual Supervised LDA (MLSLDA)[25] uses sLDA for classification and regression across languages essentially simulating transfer learning among corpora. sLDA-bin[142] is another variation developed for classification in a multivariate binomial context for document and image labeling and annotation. Spatial

LDA (SLDA)[171] adapts classic LDA to a computer vision context by treating groups of pixels (such as a nose and an eye) as a "visual word" in an image (document) where the word-document pairs are an unobserved variable rather than observed as in classic LDA. Hierarchically Supervised LDA (HSLDA)[133] constructs a model using data partially categorized into a hierarchy as well as annotations provided by humans and can be used to classify documents into categories of a hierarchy together with tags. Feature LDA (feaLDA)[108] allows for specification of other types of supervision such as using both document labels and labeled features whereas Labeled LDA can only model the relationship between class labels and documents.

2.9 Chapter Summary

This chapter provided a history of the topic modeling literature from basic principles starting with Naïve Bayes through the early vector space model with Latent Semantic Analysis and PLSA, an early Bayesian model for text modeling. Then we saw a thorough coverage of Latent Dirichlet Allocation (LDA) including a derivation of the collapsed Gibbs sampler for parameter estimation, discussion of estimating hyperparameters and the number of topics while maintaining consistent notation. The rest of the chapter discussed variations of Latent Dirichlet Allocation to handle special, but common use cases: correlation in topic sets, linked documents, topic hierarchies and supervised learning. This discussion exposes the continued importance of topic modeling in the new frontier of statistical natural language processing and motivates the need for the individual contributions discussed later in this manuscript.

CHAPTER 3

Methods for Data Augmentation in Machine Learning and on Short Texts

In this chapter we review previous work in the literature related to machine learning and related tasks with short texts as well as the more general theory of data augmentation. While there has been a wealth of literature dedicated to machine learning tasks on text, short texts have not received nearly as much attention. The first widespread discussion of tasks with short texts appears in vintage literature dating back to the 1980s particularly for information retrieval and searching databases with short user generated queries. Much more recently, the discussion has reemerged in the literature related to Web content and other user-generated content. In much of the contemporary literature, the most popular machine learning tasks applied to short texts seems to be language identification, search and sentiment analysis. While the author of this manuscript refers to the process of making a short text longer as text augmentation, the literature, particularly the vintage literature, refers to this method as *query expansion* and both phrases are used interchangeably in this chapter.

Recently, much investigation has been performed on a wider problem: data augmentation, where more data is generated to reduce the effects of sparsity and improve generalization error[180]. The majority of the literature involving data augmentation discusses applications to image processing, computer vision, video and audio. These applications generally use neural networks and deep learning. The novel research discussed in this manuscript combines methods from query expansion and data augmentation, so in this chapter we review both.

3.1 Query Expansion

Query expansion is essentially a niche form of data augmentation where short texts are simply augmented with other semantic units such as words, phrases or concepts. Prior work on this problem can be categorized as follows: relevance feedback, domain specific, and lexical analysis.

3.1.1 Relevance Feedback

In a relevance feedback system, a user issues a short text as-is and the system returns the best ranked results to the user. Through various algorithms, the query system then adds words from these results to augment the original short text query. A major problem arises with this method when the returned results are not relevant to the original search query. Some of the vintage systems also used manual user feedback for expansion.

In [179], Xu and Croft proposed using both relationships among words as well as analyzing how well the returned documents match the original text in a concept space. The researchers refer to this method as global and local document analysis and it is the only work that uses a separate step to measure semantic cohesiveness in candidate query terms. Global document analysis referred to studying word relationships across the entire corpus whereas local document analysis only studied the set of results returned as part of the results for the initial query. They found that local analysis was more effective than global analysis in document retrieval applications. Local document analysis also included a check for semantic cohesiveness. Given these results, the author of this manuscript believes that semantic cohesiveness is an important factor to include in a method for augmenting short texts. Mitra et. al. study the problem of concept drift in [126]. Concept drift occurs when the intent of the original query is modified by irrelevant results from the initial issuing of the query. The researchers used several boolean logic methods as well as distance functions to measure the true cohesiveness of returned results to the original query and found their method superior to the baseline.

The vintage method has also appeared in contemporary work as in Chum et. al. [37] as a way to expand queries for image retrieval. In their work, a region of an image is submitted to a search system and it returns a set of image regions that match the query and all of the returned results are combined into one pseudo-image from which a latent concept can be discerned. The query is then reissued using the new learned concepts. The researchers accomplished a 23% improvement in retrieval performance, from 55% to 78% using their proposed method.

3.1.2 Domain Specific

In a domain specific approach, the researchers either manually constructed a thesaurus of synonyms, or used automated methods based on co-occurrence or semantic networks. This contemporary method was implemented as an automatic thesaurus using external data sources used as a corpus, or the full experimental corpus of text itself. The most commonly used external datasets used for augmenting short texts are Wikipedia and WordNet. Wikipedia[13] is a collaborative and free encyclopedia[14]. Since Wikipedia is updated very frequently by editors and other users, it has wide coverage and recency[181]. Additionally, the self-organizing nature of Wikipedia editors with domain expertise seems to have created a high-quality resource. The main incentive for using Wikipedia is in its structure and organization[181]. WordNet[15] is a lexical system manually developed by George Miller at Princeton University[122], and resembles a thesaurus. Words are grouped together into synsets based on their meanings and related to other words based on their senses, the context in which the word is used in language. WordNet also relates words based on `is-a` and `has-a` relationships.

In [53], the researchers augmented their short text data with concepts from Wikipedia articles to automatically annotate and tag text with topics. While classification was not of

[13] http://www.wikipedia.org

[14] https://en.wikipedia.org/wiki/Wikipedia:Introduction

[15] http://wordnet.princeton.edu

primary interest, it was implied that the concepts discovered by their TagMe system could be used to augment short texts. Using such external data resulted in a decent F-measure of 0.78 for topic identification and annotation. In similar work [9], Banerjee and colleagues used a hybrid of external data from Wikipedia and the legacy relevance feedback approach to cluster short texts. By issuing the short text as a query to Wikipedia, the researchers augmented the short text using titles from the returned articles. The hybrid approach accomplished an approximate 8% improvement in clustering accuracy over the baseline approaches on raw short texts and clustering algorithms available in CLUTO. A very similar method discussed in [81] used full Wikipedia articles and the entities contained within them, rather than just titles to augment short texts.

The method presented in [167] by Voorhees augmented search queries using a straightforward thesaurus approach induced by the WordNet semantic network as well as a similarity metric. In [128], Naigli et. al. made an attempt to expand on a similar method by using WordNet semantic networks to identify other words in the same sense rather than just using synonyms. They found that their method performed an average of 2% better when augmenting with synsets, hyperonyms and gloss hyperonyms[16] and about 27% better when augmenting with gloss words. It was unclear if this result was statistically significant and the researchers admit that they prefer to repeat their experiments on much larger corpora of short texts.

Mandala et. al. [112] presented an ensemble approach that combined the results of multiple thesauri. Each thesaurus was from a different paradigm: hand-crafted, co-occurrence based, and head-modifier based. In the hand-crafted approach, a domain expert constructs a thesaurus (i.e. WordNet) by hand. The co-occurrence based thesaurus used the number of times two terms appeared together in the corpus. The head-modifier based thesaurus was the most exotic and considered two terms similar if they appeared in the same linguistic construct, such as subject-verb, verb-object etc. Unsurprisingly, the researchers found that a combination of the co-occurrence based thesaurus and the head-modifier based thesaurus

[16] A *hyperonym* is a word that refers to a group of similar items – the `a` part of the `is-a` relationship. For example, `vegetable` is a hyperonym. A *gloss* is a very short definition of a term or group of terms.

performed the best. In [143], Qiu et. al. proposed a method for automatic query expansion by using statistical term co-ocurrence data and was the only research in this review that tested the effect of the number of terms to use to augment the text. The researchers found that the method performed best when a small number of terms was added relative to the maximum used in the experiment.

Cui et. al. [38] discussed a method that used query logs as the external dataset. Unlike the previously discussed methods in this section, this method uses "apples-to-apples" comparable data – online queries containing short text is augmented using logs of previously executed queries. The method discussed is self-sufficient with respect to data, similar to the approach that we will discuss in this manuscript in Chapter 4. A related way of resolving the sparsity induced by short texts is to combine multiple short texts into one large text, as presented in [175]. The task of interest to the researchers was to identify influential Twitter users given a particular topic. Similarly, [80] used the same method for training LDA models to identify topics in tweets. Rather than combine a user's tweets into one document, Hong and Davison instead combine all tweets containing the same word into a single large document. The researchers found that models trained from the larger text performed better than the models trained on the individual tweets. In both cases, the researchers made the assumption that all of a particular set of tweets was related to the same latent topic, but Twitter users tweet about many different topics. For this reason, the author of this manuscript considers combining short texts as a form of using external data. *This is a very commonly used "workaround" in dealing with short texts in industry.* Jin et. al. [87] proposed a solution to the potential mismatch in data by developing a variation of LDA called Dual LDA that uses transfer learning to learn a set of concepts on larger auxiliary texts and apply them back to the short texts. While Jin's method is much more dynamic than the others mentioned in this paragraph, it still relies on the existence of a large corpus of external data.

The author of this manuscript feels that although large corpora are easy to find on the World Wide Web, they may be too domain specific as the external data used to build a thesaurus must be applicable to the text being augmented. The use of Wikipedia and WordNet seems to imply that performance is best on scholarly or non-fiction texts rather than colloquial language typically used on the Web, on social media platforms and other services that produce short texts. Combining short documents into a larger one is also problematic because it assumes that the short texts are dimension (time, space, author, etc.) invariant, and in the types of systems that produce such short texts, this is rarely true.

3.1.3 Lexical Analysis

Other methods use feature selection and generation techniques from machine learning and text mining to extract meaning or other features to accomplish a task. Some of these methods include modifying topic models, using groups of words and characters as features instead of single words, and graph-based methods.

In [84], Islam and Inkpen note that most methods for short texts rely on domain knowledge. The researchers used Latent Semantic Analysis (LSA) and Pointwise Mutual Information (PMI) to identify similar texts. One unique aspect of their work is that they also used string similarity, particularly longest common subsequence, to correct for spelling errors in queries. The researchers compared their proposed similarity metric to human judgment and achieved a Pearson correlation of 0.853.

Gottron et. al. [66] researched language identification on short texts using classical machine learning techniques with little or no modification. The researchers used n-grams where n varied from 1 to 5. They found that Naive Bayes performed the best on this task for all n across the board. In Tromp and Pechenizkiy [164], the researchers constructed a network of character and word n-grams from short texts to construct a grammar for a language. The machine learning task was also language identification. They found that their method yielded a statistically significant improvement on classification accuracy and that it was particularly effective for jargon.

The fact that typical machine learning methods do not work well with short texts is unfortunately taken for granted in the literature as common sense. Yan et. al. [180] provided a discussion of machine learning on short texts and why it is problematic including term sparsity, and the lack of discriminative terms. The researchers proposed a modified version of LDA called the Biterm Topic Model (BTM). The major difference between LDA and BTM is that pairs of words at any location of a text, rather than a single word, are sampled from a multinomial distribution. Their results showed that purity increased logarithmically as the word distance between the terms in the pair increased, and was consistently better than LDA at non-negligible distances. Another method based on bigrams is presented in [51], instead using bigrams of syntactic labels such as parts of speech. When using these lexical features, the researchers accomplished an 86% accuracy score over 76% when using all feature types.

Dos Santos and Gatti proposed a method using deep convolutional neural networks for the sentiment analysis task (classification) on short texts in [46]. Their method uses word-level embeddings and character-level embeddings to capture syntactic and semantic information and morphological and shape information respectively. The character-level embeddings allowed the researchers to extract meaning from small tokens such as hashtags. Their work accomplished 85.7% accuracy in positive/negative sentiment classification compared with an average of 82% using standard methods presented in [159].

The author of this manuscript believes that the methods presented in this section serve as the most promising methods for general natural language processing tasks moving forward as they are scalable and represent language in a more holistic sense than the methods proposed in sections 3.1.1 and 3.1.2. One aspect of the research in this section with which the author disagrees is in the use of n-grams – the research in this section focused on using bigrams (or biterms) which seems arbitrary since n can essentially be any integer, and the author feels that researchers have not sufficiently defended why $n = 2$ was the best choice.

3.2 Data Augmentation

Recent data augmentation literature discusses making perturbations to the data points themselves to generate new data points, and thus an augmented dataset. The most common applications of data augmentation occur in vision and audio rather than in text, but introduces very interesting ideas that can also be applied to text. Another school of data augmentation methods can be found in the Bayesian modeling and MCMC literature[44][166][58][140][139], but will not be discussed in this manuscript as its purpose does not align with this research.

One of the most frequently cited manuscripts in the area of data augmentation is from Krizhevsky et. al. [99] on classifying images into one-thousand classes. One concern discussed in the paper is the possibility of overfitting. The researchers used a neural network architecture that required 60 million parameters. Data augmentation was used as one method to combat overfitting. From each 256×256 image, all patches of size 224×224 were extracted and used to augment the training data associated with the original image. When an image is classified, the 224×224 patches in the four corners of the image, the center of the image, as well as their horizontal transformations are all classified and their labels averaged. The researchers note that without this form of data augmentation, their neural network overfits and they would have had to use a smaller network architecture instead. The second form of data augmentation they used transformed the intensities of the RGB channels of the images in the training data. They used a transformation based on PCA to create more training instances from each existing training image. The researchers achieved significant reductions in testing error over existing methods on several datasets.

Simard, Steinkraus and Platt [157] included data augmentation as one of the their recommended best practices when working with convolutional neural networks applied to document analysis. They described a method of growing a training set so that it is very large. They proposed adding distorted versions of the original data into the training data to augment the training set. These distortions included both affine transformations such as translations, rotations and skewing as well as elastic transformations that they define as a convolution of Gaussian noise and random displacement fields. Using this extended dataset, the researchers trained a convolution neural network (CNN) and achieved improved classification error. Using no distortion with a two-layer Multi-Layer Perceptron achieved 1.6% classification error whereas using a simple convolutional neural network with affine and elastic transformations yielded lower classification errors of 0.6% and 0.4% respectively. This method would be approximately equivalent to adding synonyms, antonyms and other functions of words to a bag-of-words representation.

In [74], Haubert et. al. expand on the ideas presented in [157] by exploring the possibility of *learning* the transformations used for data augmentation rather than manually specifying them. The researchers proposed using label-invariant transformations – transformations performed on an image that do not change the label or class of the image. First, an unsupervised method was used to learn the different transformations that appear in each class. New data is then generated by sampling an image and then sampling a transformation from the distribution of transformations learned by the unsupervised method. By applying this transformation to the randomly selected image, we get a new data point. The process repeats until the training data is of a reasonable size. The researchers used the famous MNIST[17] dataset for their experiments and compared their results with the InfiMNIST[18] dataset which the researchers cited as the most extensive use of data augmentation in the literature. The proposed method achieved a test error of 0.44% on a convolutional neural network whereas the InfiMNIST dataset yielded a testing error of 0.49%.

Dosovitskiy et. al. used data augmentation to learn features in an unsupervised manner by building on the work of [157] and [74] in [47]. The researchers discuss the fact that feature learning usually involves labeled data and then learning proceeds in a supervised manner. The proposed method does not use labels or supervised learning on the surface. Instead, researchers take a single patch from each image and place it in its own class, y_i. Then a series of transformations are performed to obtain more patches based on the original one, and these transformed patches are also placed into the same class y_i. The researchers then trained a convolutional neural network to classify the surrogate classes. More specifically, all possible 32×32 patches of an image are extracted and one-hundred transformations were applied to each patch. Each transformation came from a base set: translation, scale, color and contrast. Researchers used several datasets and classifiers with their data augmentation method and found that on one dataset, accuracy using their proposed method was the best. The novel research described later in this manuscript also uses a number of iterations to ensure that each set of augmented documents produces each perspective of the original document.

[17] `http://yann.lecun.com/exdb/mnist/`

[18] `http://leon.bottou.org/projects/infimnist`

Bouthillier et. al. describe dropout as a method for data augmentation in [24]. The researchers argue that dropout permits data augmentation without any other domain expertise and that because it uses a bagging step to combine several networks with shared parameters, it yields better generalization error. They also developed an extension of dropout that increases variation in the sample by injecting random noise. The researchers state that in order to get better generalization error, a classifier must not only be able to classify the images in the dataset, but also all images that come from the same data distribution. Tests on MNIST and CIFAR-10[98] showed that the researchers' method using dropout on input noise yielded the best accuracy at 1.12% on MNIST and 40.9% on CIFAR-10. The next best method, performing dropout on both input noise and hidden layer noise yielded accuracies of 0.95% and 39% respectively. Baseline noise projection performed the worst at 0.99% and 37.9% respectively. While the researchers focused on data augmentation, the author of this manuscript feels that dropout is more of an ensemble method than a data augmentation method and that the ensemble is actually what contributed to better results.

Researchers have also used data augmentation for acoustic modeling. Jaitly and Hinton applied data augmentation to a speech recognition problem in [85]. In their work, the researchers transformed spectograms by applying a random and linear warping in the frequency domain of the audio. To generate variations of training data, the researchers added speaker-to-speaker variations to each input using Vocal Tract Normalization. The spectrogram's frequency axis is warped using a warp factor α. During the training phase, utterances from speakers are warped using random values of α, a process they refer to as Vocal Tract Length Perturbation (VLTP). The researchers made an improvement over the baseline of about 0.65% on the test set for TIMIT and a gain of about 1% using a baseline CNN. This paper was the only paper that *explicitly* discussed that data augmentation is useful for small datasets. Cui et. al. built on Jaitly and Hinton's work in [39] where they combined VLTP with stochastic feature mapping (SFM). The researchers explain that SFM basically is a form of voice conversion, though a statistical one. For a set of features H, and a speaker S who speaks an utterance u with label W, one would observe a sequence of features $O^{(S)} = \{o_1^{(S)}, \dots, o_N^{(S)}\}, o_t^{(S)} \in H$. Then, given we have speaker T speak the same utterance u

with the same label W, we infer the sequence of features as $O^{(T)} = \{o_1^{(T)}, \ldots, o_N^{(T)}\}, o_t^{(T)} \in H$. The goal of SFM is to infer this mapping between feature sets $O^{(S)}$ and $O^{(T)}$. Once this mapping is learned, it can be used to generate new data by mapping the existing data to other speakers. By combining VLTP and SFM, the researchers improved the maximum term-weighted value (MTWV) on an Assamese and Haitian Creole dataset. Combining VLTP and SFM on the Assamese data yielded an MTWV of 0.2862 and for Haitian Creole yielded an MTWV of 0.5178. Both of these values were higher than those for SFM and VTLP independently applied before training a deep neural network (DNN) and a convolutional neural network (CNN), and also higher than the baseline applied to a DNN and CNN.

Data augmentation has been applied in a wide variety of fields and for solving a wide variety of machine learning challenges. Ahmed et. al. [2] used data augmentation as a way to deal with class imbalance for a person re-identification problem. Li et. al. [107] also attempted to solve a person re-identification problem but used data augmentation alongside dropout and the bootstrap. Finally, Zeiler and Fergus [183] described the use of data augmentation with stochastic pooling for regularization of deep convolutional neural networks.

Data augmentation is a method parallel to query expansion, but relies more on statistical and machine learning theory. One can say that data augmentation is sometimes similar to ensemble method and this is especially true for dropout. Data augmentation is also similar to the parametric bootstrap in the case of adding noise to improve generalization error[72]. For short texts in particular, the author of this manuscript believes that combining vintage query expansion with data augmentation will yield better results than applying a classifier on unaltered short texts.

3.3 Chapter Summary

In this chapter we reviewed literature on a variety of machine learning tasks applied to short texts and their proposed solutions, as well as data augmentation methods used in other fields.

The researcher categorized the short text methods as relevance feedback, domain specific and lexical analysis. The vintage literature mostly considered the so-called query expansion problem by issuing an initial query and using human or automatic intervention to improve the short query. The solutions requiring a thesaurus such as WordNet assumed that writing style was consistent with typical English grammar norms and the method is based on a manually developed system. The methods involving external knowledge either used large datasets such as Wikipedia, or relied on combining documents under the assumption that all documents had the same semantic characteristics. The machine learning based methods typically use a single dataset along with feature generation, but the researchers seem to make arbitrary decisions regarding the generated features. The researcher believes that the machine learning based, and variations of the domain-specific approach are the most promising within the set of explicitly short text methods, and are also the most relevant to today's usage of language. The problem with using external datasets such as Wikipedia is that it makes the assumption that short texts are scholarly or can be classified into one of the topics discussed in Wikipedia. More broadly, it makes the assumption that the ontology and writing style is similar between the external dataset and the corpus. Mundane colloquial topics that carry little semantic discrimination power are unlikely to be found in Wikipedia and similar corpora. Additionally, combining multiple short texts into one larger text makes the assumption that all of the texts are produced from the same latent topics, and this is a very risky proposition when user-generated content with a dimensional component is the target.

We also discussed data augmentation where individual data points are perturbed or transformed to create new data points that can be added back to a training set. The result of data augmentation methods is not only a larger training set, but also more generalizable classifiers. In the image processing and computer vision literature, the most common transformations include rotation, reflection, translation and skewing though there are many more that may also transform color and other aspects of images. Researchers have even found ways to *learn* which transformations to use for data augmentation within a particular class. The common method of dropout in neural networks also forms a type of distortion by randomly turning off certain neurons in hidden layers and combining the results from such modifications into essentially an ensemble. Data augmentation has also been used with audio by perturbing spectrograms to produce new data for machine learning tasks such as speaker identification. The proposed data augmentation method shows similarities to these existing methods, but uses semantic relatedness to augment short texts. In particular, the idea of sampling an image, sampling a transformation, and repeating the process a large number of times, as presented in [74], is very relevant to the data augmentation method proposed in this manuscript.

CHAPTER 4

Data Augmentation for Short Texts

In this chapter we discuss a novel method to augment short texts for performing classification by modifying the concept of bootstrapping from classical statistics with a resampling step based on a semantic space. The proposed method attempts to derive additional meaning and classification signal based on a "population" created from a semantic space.

4.1 Review of the Bootstrap

In statistics, the classical application of the bootstrap is for computing the sampling distribution of an estimator[48]. The sampling distribution is constructed by sampling a large number of observations from some population independently, with replacement, and then computing some statistic on each such sample. This process is repeated until a sufficient number of samples of sufficient size has been selected. The resulting measurements form the sampling distribution of the estimator from which various metrics, such as standard error, can be computed. One use case of bootstrapping involves its use for small samples. By treating a small sample as a population and applying the bootstrap, one can better model the uncertainty present in the small sample[54][77]. Unfortunately, the word "small" in this context is vague and there is little guidance as to how small is too small for the bootstrap to perform effectively. One researcher mentions a sample size of eight as being sufficient[35]. A useful property of the bootstrap is that it takes advantage of the Central Limit Theorem – the sampling distribution of any estimator under these circumstances is Gaussian even if the population distribution may be anything else[52], or may even be so complex as to be intractable[121]. As an example, common measurements in natural language processing are word counts and TF-IDF scores, both of which typically follow power laws[95]. The use of the bootstrap on small samples raised interest with the researcher and inspired the question of whether or not resampling terms from a population could generate larger samples of text with properties that improve text classification.

4.2 Application to Short Texts: Three Modifications to the Bootstrap

In the bootstrap, we draw samples from a dataset D with replacement, independently. By sampling with replacement, we can construct a large number of new bootstrap samples all which contain an appropriate number of observations. We then compute some statistic on each of these samples and by the Central Limit Theorem, we see that the statistic has a normal distribution as long as some additional conditions are met. In this research, the computation of statistics and the use of the Central Limit Theorem is not of primary interest; rather, it is solely the sampling step that is of primary interest. This serves as the first modification of the original bootstrap framework. We note that while the observations in D may be independent, they all likely come from the same distribution in the most basic case of a simple dataset. One can think of a simple dataset as a nesting: observations in a dataset.

Things get much more complicated when the dataset D is a set of texts where each text is a bag-of-words[114] representation. As we have discussed earlier, such a dataset is called a corpus, and each entry in the corpus is an individual document[19] and each document consists of a set of terms. The nesting in this case is terms in documents in the dataset/corpus. One challenge applying the bootstrap to text is how we define an observation. When we speak of sampling observations, it becomes difficult to determine if an observation refers to a document, or a term. Even worse, the texts in a large corpus, even a simple one, typically do not come from the same distribution. Fortunately, this context allows us to make the

[19] In this manuscript, the word *document* is sometimes used interchangeably with the phrase *(short) text* and *bag-of-words.*

decision much easier since only one makes sense for text.

First, let us assume that a document is an observation, and we have the situation pictured in Figure 4.1, where we essentially sample documents with replacement from the corpus. All this does is make copies of existing texts and does not help classify each text into a category. A corpus typically contains a large number of documents, so it does not make sense to apply the bootstrap for this purpose using documents as observations anyway. Our second modification is that we instead consider each *term* an observation within a document, and a document serves as the original definition of a dataset. All documents are part of the same corpus. For this reason, the corpus itself is really of no consequence, it is simply a collection of texts. While we work with the corpus extensively in this research, the corpus itself does not serve any interesting purpose aside from serving as a denominator or summation index in many calculations. In the bootstrap framework, we treat the dataset as part of the population used for resampling. In this context, we would simply make duplicates of words already in the text at random and with replacement, a situation illustrated in Figure 4.2. But it is important to recognize that words induce context – certain words are more important than others[145] and the researcher hypothesizes that simply adding duplicate words to a text would not yield better results than with the original text. Instead, we make one final modification. We use a semantic space S as the population rather than the document itself. Using this semantic space, we add words to document d that are believed to be semantically similar to the words already in d. The new document is referred to as an *augmented bag-of-words* and is denoted d^*. For the rest of this work, since there are multiple documents and multiple possible augmented documents, we denote documents using the subscripts m and n, and we add another subscript l to the augmented documents to denote that each iteration of the process generates a different augmented bag-of-words. An example of this proposed resampling step is presented as Figure 4.3. This modification is the most significant one and serves as the bulk of this research. Throughout the rest of this chapter, we answer several questions. What is the semantic space S? How do we sample words from it? How do we account for the disproportional importance of certain words in the text over others? How many terms should be sampled? How do we ensure that words sampled from S make sense?

Finally, there are cases where the bootstrap is not appropriate because the sample size is simply too small. This could be a problem for short texts, but the researcher believes that if we resample in an intelligent manner, the size of the original text may not matter.

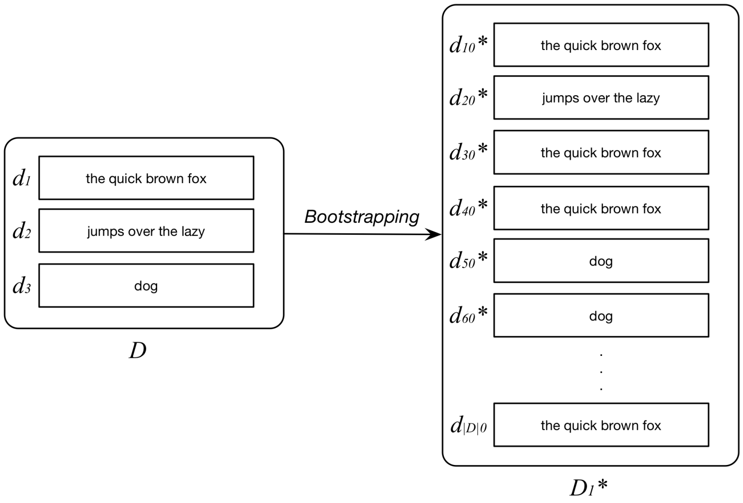


Figure 4.1: An example of the bootstrap sampling step treating the corpus as a population and the *documents* as the sampling units.

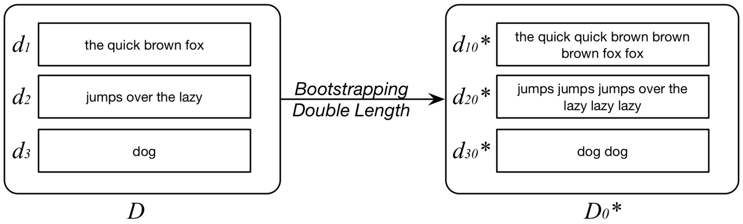


Figure 4.2: A hypothetical example of the bootstrap treating each document as a population and the *terms within the documents* as the sampling units.

4.3 Implementing the Proposed Data Augmentation Procedure

The proposed data augmentation method augments short texts using semantically similar words sampled from a semantic space S. Each iteration l of augmentation produces a full corpus D_l^* where each augmented document d_{ml}^* contains the same terms from corpus-to-corpus, but also contains *additional* terms that are semantically similar to those in the original document.

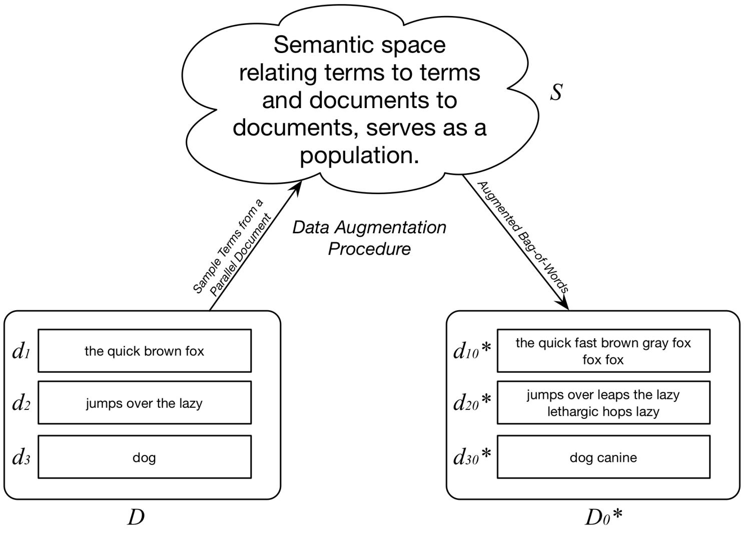

Semantic space relating terms to terms and documents to documents, serves as a population.
S
Sample Terms from a Parallel Document
Data Augmentation Procedure
Augmented Bag-of-Words
d_1
the quick brown fox
d_2
jumps over the lazy
d_3
dog
d_{10}*
the quick fast brown gray fox fox fox
d_{20}*
jumps over leaps the lazy lethargic hops lazy
d_{30}*
dog canine
D
D_0*

Figure 4.3: An illustration of the proposed data augmentation method, showing the semantic space S, the population from which terms are sampled as well as an example of a corpus D^* of augmented texts.

The data augmentation procedure is implemented as follows. We have some topic classification model we wish to apply to a sample of short text. We denote this short text as document d_m which is a bag-of-words representation. Instead, we augment the short text with related terms from a semantic space S formed using a matrix factorization that can model terms and documents and retain their relationship using some distance function. It is hypothesized by the researcher that a good class of such topic models involve matrix factorizations, such as Latent Semantic Analysis[43][102] which will be investigated in this manuscript. This augmented bag-of-words can then be be used to cluster or classify a test document.

More formally, we construct a term-document matrix $\mathbf{X}$ from training data where the entries X_{im} denote some quantity that represents the importance of term t_{im} to document d_m such as word presence/absence, word count, or in the case of this research, TF-IDF score. The matrix $\mathbf{X}$ is very sparse – only 0.04% of the values in the matrix are non-zero. This matrix $\mathbf{X}$ is decomposed using a matrix factorization that preserves distance between terms and between documents, in this case singular value decomposition (SVD) and Latent Semantic Analysis (LSA). Such a factorization constructs a set of spaces S for both terms

($\mathbf{S_t}$) and documents ($\mathbf{S_d}$) such that $S = \{\mathbf{S_t}, \mathbf{S_d}\}$ where distances can be computed and compared. For each document d_m in the corpus, we need to pick one or more terms to add to d_m. We first pick a *target* term that we will condition on in order to sample a new term to include in an augmented bag-of-words. This target term is simply some t_{im} in the document and is selected according to how discriminative the term is in the document according to TF-DF as in Equation 4.1:

$$P(t_{im}|d_m, X_{\cdot m}) = \frac{X_{im}}{\sum_q X_{qm}} \tag{4.1}$$

We then select a *parallel* document d'_m according to how similar it is to d_m. Using this pair of d_m and d'_m we pick a term t'_{im} from d'_m according to how similar it is to target t_{im}. Finally t'_{im} is accepted into the augmented bag-of-words representation $d^*_{ml} = d_m \cup t'_{im}$ with a probability proportional to how much the semantic meaning of d_m changes with the addition of t'_{im} to it. This process is repeated until the augmented bag-of-words d^*_{ml} reaches a sufficient length. The relative increase in document length is denoted ε. For example, $\varepsilon = 1$ means the augmented bag-of-words d^*_{ml} is twice the length of the original document d_m – a 100% increase in length. Figure 4.4 displays the data augmentation preprocessing workflow. A more concise and formal treatment of this data augmentation algorithm appears in the next chapter. In this chapter, the implementation details of the proposed method are established as well as the construction of the semantic spaces S, the selection of the classifiers and their hyperparameters.

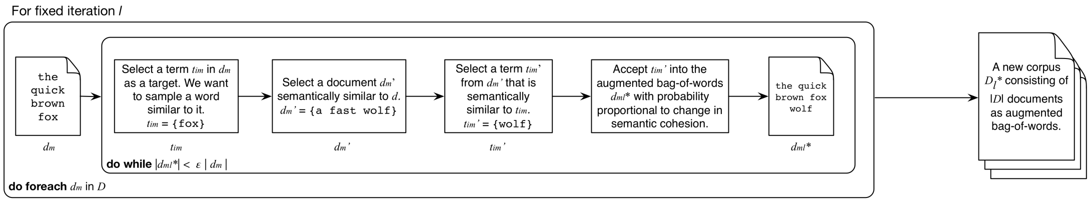

For fixed iteration l
the quick brown fox
d_m
Select a term t_{im} in d_m as a target. We want to sample a word similar to it. t_{im} = {fox}
t_{im}
Select a document d_m' semantically similar to d. d_m' = {a fast wolf}
d_m'
Select a term t_{im}' from d_m' that is semantically similar to t_{im}. t_{im}' = {wolf}
t_{im}'
Accept t_{im}' into the augmented bag-of-words d_{ml}* with probability proportional to change in semantic cohesion.
the quick brown fox wolf
d_{ml}*
do while $|d_{ml}^*| < \varepsilon\, |d_m|$
do foreach d_m in D
A new corpus D_l* consisting of |D| documents as augmented bag-of-words.

Figure 4.4: The generation of a new corpus D^*_l containing new documents represented as augmented bag-of-words representations.

Note that all of the conditional dependencies mentioned in the recipe above are a violation of the original independence assumption made by the statistical bootstrap[48] because text has complex dependencies and words are not sampled independently of each other; rather, terms and documents are sampled dependent on their similarity to other terms and documents. On the other hand, such dependencies also violate the independence assumption of the Naïve Bayes classifier, but in practice the classifier stills work very well for text[146]. It is the researcher's belief that despite these violations, this algorithm is the closest process to the statistical bootstrap that can be realized for text in practice.

4.4 Data

The World Wide Web is full of short texts. Tweets are a very popular source of short text, but their use requires traversing a deep "rabbit hole" of poor grammar, fake words, misspellings, SMS abbreviations etc. such that it is not possible to reasonably extract a coherent signal when using them for these experiments. Instead, titles of blog posts were used for the experiments in this research. Blog post titles tend to be in a more standardized format – concise, with a syntactic structure that entices the user to read more, but are also often too short for proper classification.

Technorati[20] is both a search engine for blogs and a directory of popular English blogs in a set of categories. Requiring the blogs to be in English removes a layer of preprocessing required to filter out foreign languages. While some foreign words still are present, they tend to be eliminated by preprocessing steps common in text mining. A small scale crawl was performed on the Technorati website in 2014. The crawl downloaded the first twenty pages of blog URLs in each of the nine Technorati categories, as well as an additional category which was essentially a "catch-all others" category called `overall`. Then, another crawler visited each of the URLs and attempted to automatically discover the URL of the RSS feed using the RSS specification[21]. For sites where the developer chose not to implement the RSS standard regarding auto-discovery, common URL patterns were added to the domain name to attempt to locate the feed URL. This process was not lossless; some blogs listed

[20] `http://www.technorati.org`

[21] `http://www.rssboard.org/rss-autodiscovery`

Category	% of Titles
autos	2.1
business	7.4
entertainment	20.8
green	2.1
living	21.6
politics	7.5
science	3.0
sports	14.3
technology	18.8
overall	2.4

Table 4.1: The ten categories used by Technorati and their relative frequencies in the corpus.

on Technorati no longer exist, some blogs had non-standard RSS feed URLs that could not be retrieved, and others could not be retrieved due to access restrictions or server issues. The data included 245,146 blog post titles each in one of Technorati's categories most of which are further subdivided into more refined categories. For a typical non-hierarchical topic model such as we have here, only the top level categories were used.

Several preprocessing steps were performed on the resulting text: numbers and punctuation marks were both removed from all text and text was converted to lowercase. Next, stopwords – words that appear frequently in the English language for syntax but do not convey any semantic meaning – were removed using the stopword database provided with the NLTK library[13] for Python. According to [114], words that appear too seldom and words that appear too often in a context should be removed as they introduce noise into any text mining activity. First, words that appear seldom are usually misspellings, or words that are too specialized to the particular document such that they are not representative of the corpus and would possibly cause overfitting. Second, words that appear in a large proportion of the documents in the corpus essentially serve as stopwords for a particular domain above and beyond the standard set of English stopwords[153]. For example, in this corpus the word "blog" appears across a large proportion of the corpus because the text consists of blog post titles. The word "blog" does not induce any meaning on the title text and thus should be removed. The choice of cutoff for words that appear too seldom or too frequently is often arbitrary but is performed empirically in this research.

Figure 4.5 shows the histogram of word counts and their frequencies. For example, $x = 1$, $y = 53650$ means that there are 53,650 *unique* words in the corpus that appear only once – approximately 57% of the entire vocabulary. From this histogram, it seems that a word count of 10 is a good cutoff and reduces the vocabulary from 94,071 words to 10,889 . The rationale behind this cutoff is that it preserves the long right tail of these word counts – words that appear less than 10 times are likely to be very specific to a particular blog or words that can be distractions such as misspellings and obscure slang. It also helps that 10 is a nice round number satisfying these conditions.

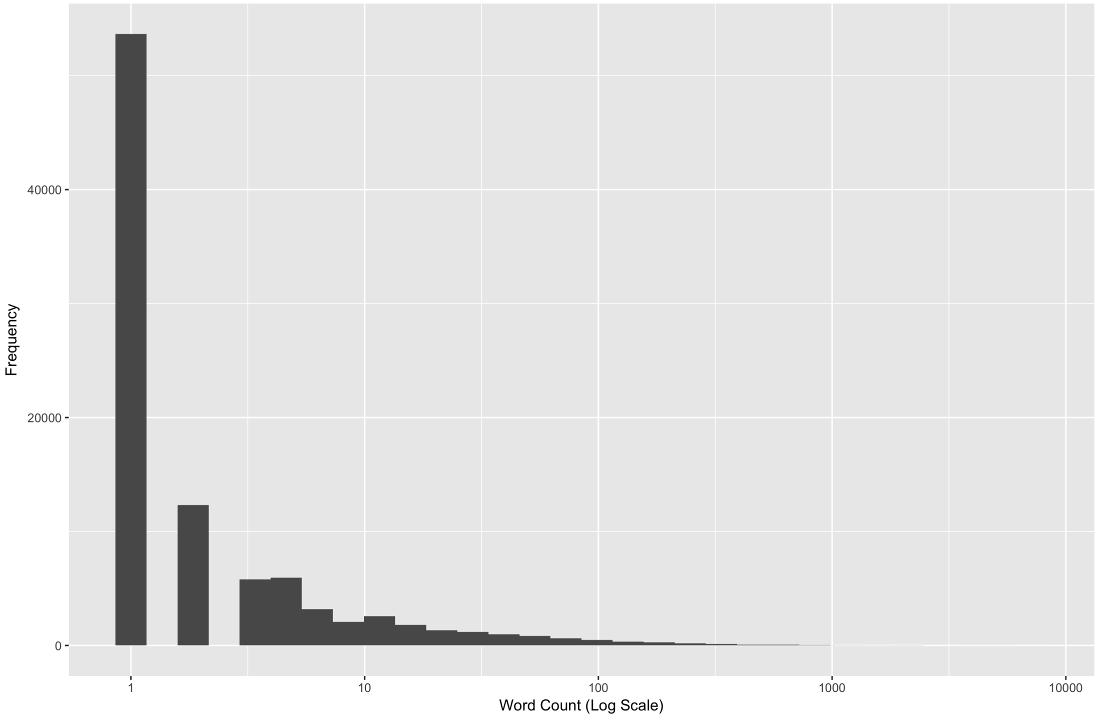


Figure 4.5: Distribution of word counts and their frequencies on a logarithmic scale.

Figure 4.6 is a similar histogram but this time displays the normalized inverse document frequency (the percent of documents in which the term is found), and the number of unique terms at each percentage. Any word that appears in more than 1% of the documents in the corpus was removed. The words “blog” and “official” occur in approximately 4% of all the documents in the corpus making them the most frequently used words in terms of IDF score. Table 4.2 displays the 18 words and tokens that appeared in more than 1% of the

web	best	blog	center	cloud	day
free	hosting	knowledge	new	official	one
rackspace	review	<<monthname>>	top	video	week

Table 4.2: Words and tokens that appear in more than 1% of the corpus and serve as contextual stopwords.

documents in the corpus lending support to the idea that certain words essentially serve as stopwords in this blog corpus.

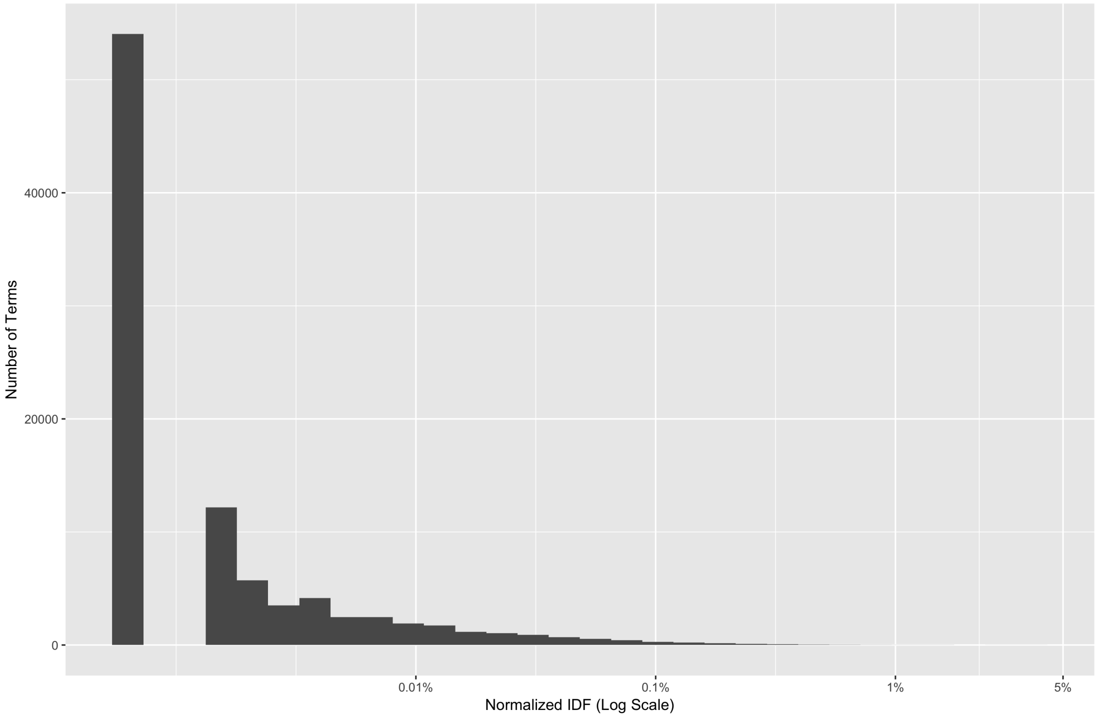


Figure 4.6: Distribution of normalized IDF scores vs. number of terms at each IDF score.

After removing stopwords and words that appear too infrequently or too frequently, there were many documents that either contained zero terms, or became duplicates and these documents were removed from the corpus. There were a few documents that seemed to result from text processing errors and had an unrealistic number of terms – one document had about 2,500 terms which is not realistic for a blog post title. These unrealistically long documents were removed from the corpus as well. The final corpus size used for this research contained 131,519 documents. Figure 4.7 shows the distribution of the lengths of the documents in the corpus on a log scale. One can see that the document lengths are a lognormal distribution with perhaps a slight right skew. The average document length is between 4 and 5 term occurrences with very few documents containing more than 10 words. This distribution is appropriate for this research because the vast majority of the documents are very short.

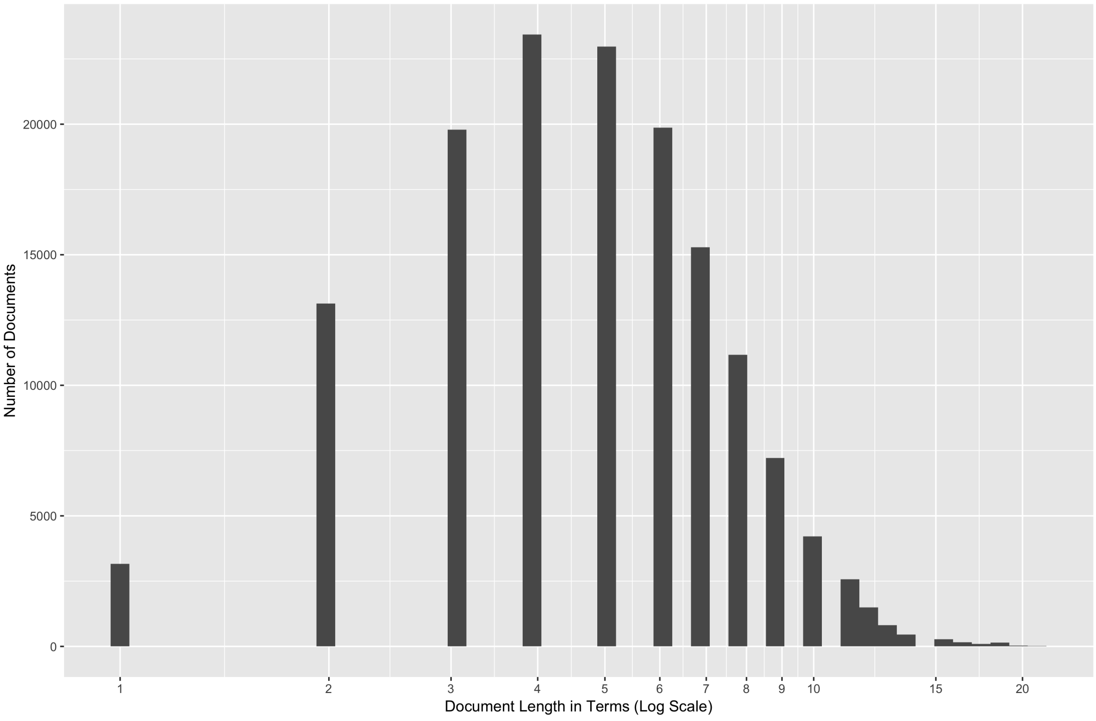


Figure 4.7: Distribution of document length in term occurrences, on a log scale.

4.5 Constructing the Semantic Spaces S

Constructing the semantic space S is perhaps the most computationally intensive part of the proposed algorithm. Constructing S requires performing a matrix factorization using singular value decomposition and then computing a similarity metric between each pair of words and between each pair of documents of the corpus in a lower dimensional space. For the reader's convenience, an explanation of the notation is provided in Appendix A.

4.5.1 Latent Semantic Analysis (LSA)

To compute the semantic spaces S, a singular value decomposition (SVD) based method called Latent Semantic Analysis[43][102] was performed on the training data to decompose the matrix into a term similarity space, document similarity space, and a set of singular values denoting the variance explained by each dimension. In the LSA literature, each dimension is referred to as a "topic." The TF-IDF score was calculated for each term in each document in the corpus and represented in a term-document matrix $\mathbf{X}$. The decomposition was realized as

$$\mathbf{X} \approx \mathbf{D_k \Sigma_k W_k}^{\mathbf{T}}$$

where $\mathbf{D_k}$ represents the first k singular vectors associated with documents and $\mathbf{\Sigma_k}$ is a diagonal matrix containing the singular values. $\mathbf{W_k^T}$ represents the first k singular vectors associated with the terms[22]. Then, from Chapter 2, we can construct a semantic space on terms using

$$\mathbf{S_t} = \mathbf{W_k \Sigma_k}^{\mathbf{2}}\mathbf{W_k}^{\mathbf{T}}$$

and on documents using

$$\mathbf{S_d} = \mathbf{D_k \Sigma_k}^{\mathbf{2}}\mathbf{D_k}^{\mathbf{T}}$$

There are several rules for choosing K, the number of dimensions to preserve, but most of them do not seem practical for this dataset. The Kaiser criterion[89] would require keeping almost all dimensions as most singular values are greater than 1, and the cumulative variance[129] criterion is similar in that an exorbitant number of dimensions would be retained. The typical criterion used in psychometrics for principal components analysis (PCA)

[22] While $\mathbf{T}$ may be a more obvious choice for notation, it would clash with the transpose operator, so $\mathbf{W}$ was chosen to represent the term "words," the set of unique terms in the corpus.

and factor analysis is the graphical Cattell scree test[29] and is also sometimes used for assessing the number of eigenvectors to retain from SVD[185]. The scree test looks for the “elbow” in the scree plot generated from plotting the magnitude of the singular values against the index of the singular values in non-increasing order; dimensions corresponding to singular values to the left of the elbow are retained. The author chose to pick $K = 500$ as the 494th singular value is the point farthest from the hypothetical straight line that joins the points associated with the minimum and maximum singular values (the elbow point)[64], and rounded up to 500 for ease of use. While application dictates the use of method for choosing the K dimensions, it has been determined that 300-600 dimensions are typically enough for LSA[102], and this result from the scree test matches this rule of thumb. The scree plot showing the magnitude of the singular values from SVD is shown in Figure 4.8. The singular value decomposition was computed using `scipy`’s SVD library[88] which is based on `ARPACK`.

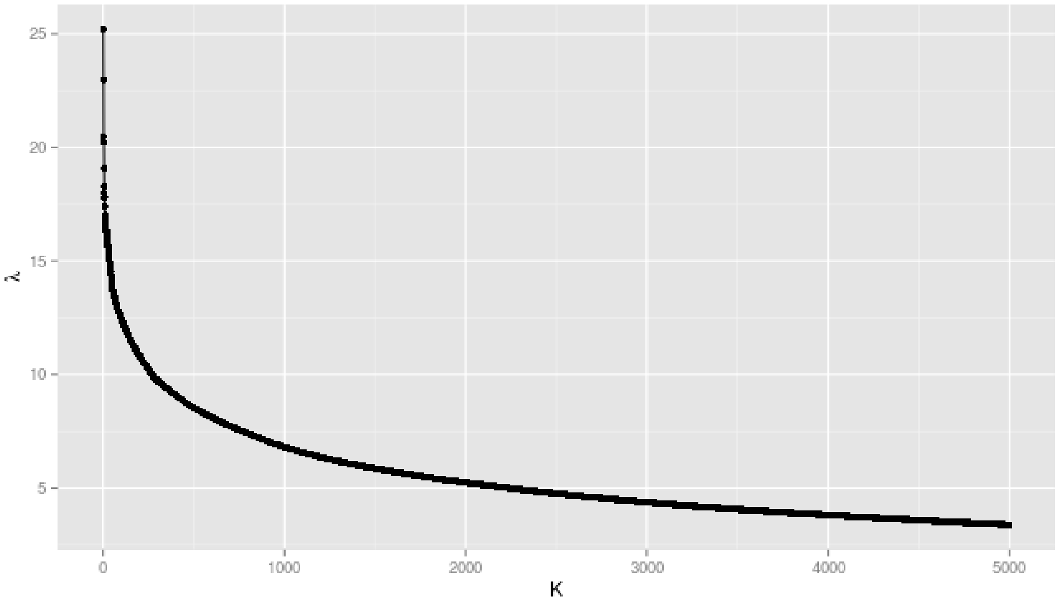


Figure 4.8: Scree plot showing singular value magnitude versus their index.

4.5.1.1 Document Similarity

Before a new term can be sampled to include in an augmented bag-of-words, we must know *where* to get the candidate term t'_{im}. The document space from LSA is used to select a document that is semantically similar to the current document d_m. A distance metric is computed between each pair of documents in $\mathbf{S_d}$. Any of the common distance metrics used in text mining should be sufficient, but cosine similarity was chosen since it is the most popular for semantic relatedness[113]. Equation 4.2 shows the definition of cosine similarity given two documents d_m and d_n in $\mathbf{S_d}$.

$$\delta_d(d_m, d_n) = \cos\left(S_{d_m}, S_{d_n}\right) = \frac{S_{d_m} \cdot S_{d_n}}{||S_{d_m}||||S_{d_n}||} \tag{4.2}$$

Using these computations, it is now possible to select a semantically related document d'_m from this semantic space $\mathbf{S_d}$ conditional on an existing document d_m as follows. Given a document d_m, we construct a probability distribution over all other documents in the corpus by taking the cosine distance between each candidate document d'_m denoted $\delta_d(d'_m, d_m)$ and dividing by the sum of all of cosine distance values given d_m as in Equation 4.3. Note that $|\min \delta_{\mathbf{d}}(d_m, \cdot)|$ is just the magnitude of the minimum cosine similarity given d_m. Adding the minimum is necessary because cosine similarities may be negative due to the logarithm calculation in the definition of TF-IDF and allows us to impose a lower bound of 0 on $P\left(d'_m|d_m, \delta_{\mathbf{d}}\right)$.

$$P(d'_m|d_m, \delta_{\mathbf{d}}) = \frac{\delta_d(d'_m, d_m) + |\min \delta_{\mathbf{d}}(d_m, \cdot)|}{\sum_q \left(\delta_d(d_m, d_q) + |\min \delta_{\mathbf{d}}(d_m, \cdot)|\right)} \tag{4.3}$$

Figure 4.9 shows the distribution of the document selection probabilities to have a distribution with a left skew suggesting the probabilities themselves may have something close to a lognormal distribution in the population. Note that the probabilities are very tiny in the left tail; most documents have a very small probability of being sampled from $\mathbf{S_d}$. We are mostly interested in the documents that are not in the left tail as they provide the highest relevance given d_m. For computational efficiency, only the top 100 candidates were considered for each document and their probabilities renormalized.

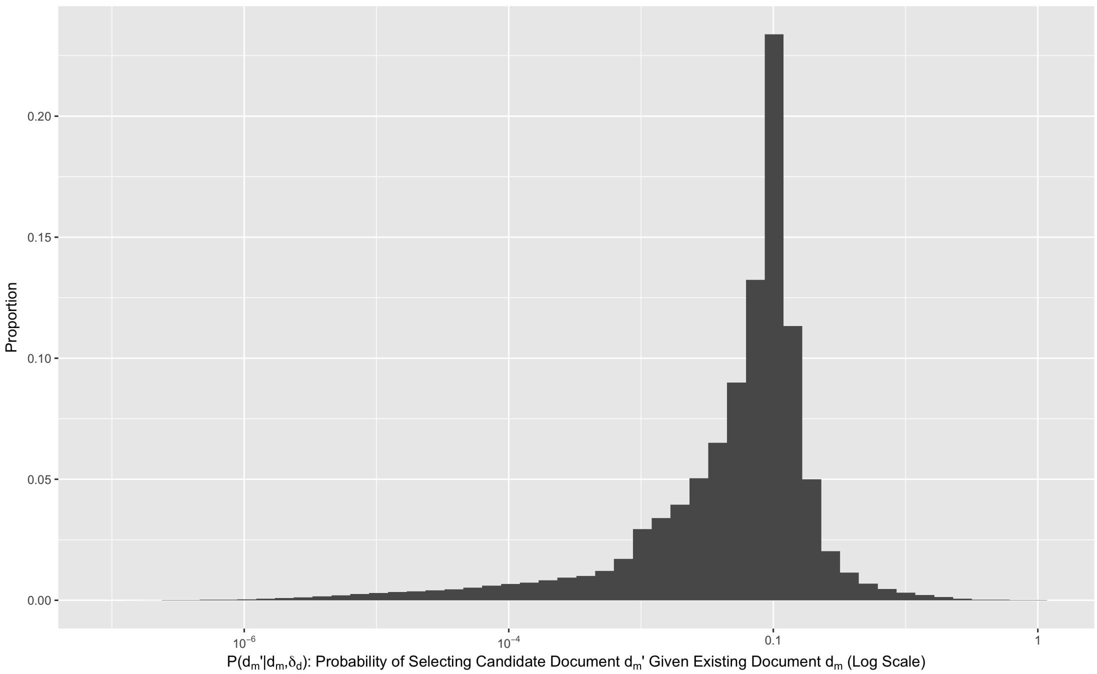


Figure 4.9: Histogram of document selection conditional probabilities on a log scale.

4.5.1.2 Term Similarity

Next, a distance metric is computed between each pair of terms in $\mathbf{S_t}$. The mathematics for term similarity are identical to that of document similarity, but computed on terms using $\mathbf{S_t}$. For completeness, Equation 4.4 shows the definition of cosine similarity given two terms $t_{i\cdot}$ and $t_{j\cdot}$ in $\mathbf{S_t}$ regardless of document, and Equation 4.5 shows the term selection probability.

$$\delta_t(t_{i\cdot}, t_{j\cdot}) = \cos\left(S_{t_i}, S_{t_j}\right) = \frac{S_{t_i} \cdot S_{t_j}}{||S_{t_i}||||S_{t_j}||} \tag{4.4}$$

Just as with document sampling from $\mathbf{S_d}$, $|\min \delta_{\mathbf{t}}(t_{i\cdot}, \cdot)|$ is just the minimum cosine similarity and adding it to each value allows us to impose a lower bound of 0 on $P\left(t'_{im}|t_{im}, d'_m, \delta_{\mathbf{t}}\right)$.

$$P\left(t'_{im}|t_{im}, d'_m, \delta_{\mathbf{t}}\right) = \frac{\delta_t(t'_{im}, t_{im}) + |\min \delta_{\mathbf{t}}(t_{im}, \cdot)|}{\sum_q \left(\delta_t(t_{im}, t_{qm}) + |\min \delta_{\mathbf{t}}(t_{im}, \cdot)|\right)} \tag{4.5}$$

Figure 4.10 shows the distribution of the selection probabilities to exhibit a near half-normal distribution[96], or an exponentially decaying tail except near probability zero, likely due to rounding. Note that the probabilities are very tiny; most words have a very small probability of being sampled from $\mathbf{S_t}$. We are mostly interested in the words in the right tail which are few and far between, but provide the most relevant words given $t_{i\cdot}$. Just as with the document sampling, only the top 100 candidates were considered for each word and their probabilities renormalized. Table 4.3 shows the top words for some seed $t_{i\cdot}$ terms. Note that the words closest to the word `hair` all involve beauty terms. The words closest to the seed term `egg` all involve food, and the words closest to the seed term `dog` are associated with aspects of owning a dog or associated with dogs themselves. The word `magnolia` is associated with many articles discussing furniture with the exception of `star` which is a part of the name of a plant. The seed term `cat` yields a more interesting result. The cat is a very common animal used throughout Internet culture and memes particularly with respect to "lolcats"[22][33]. While obvious terms such as `fur` and `litter` are closely associated with the common use of the word `cat`, the word `imgur`[23] is actually the name of a website for uploading small images for free for use on Internet forums and social media sites. Some of the most common images on `imgur` are memes including "lolcats"[105], some of which are `grumpy`.

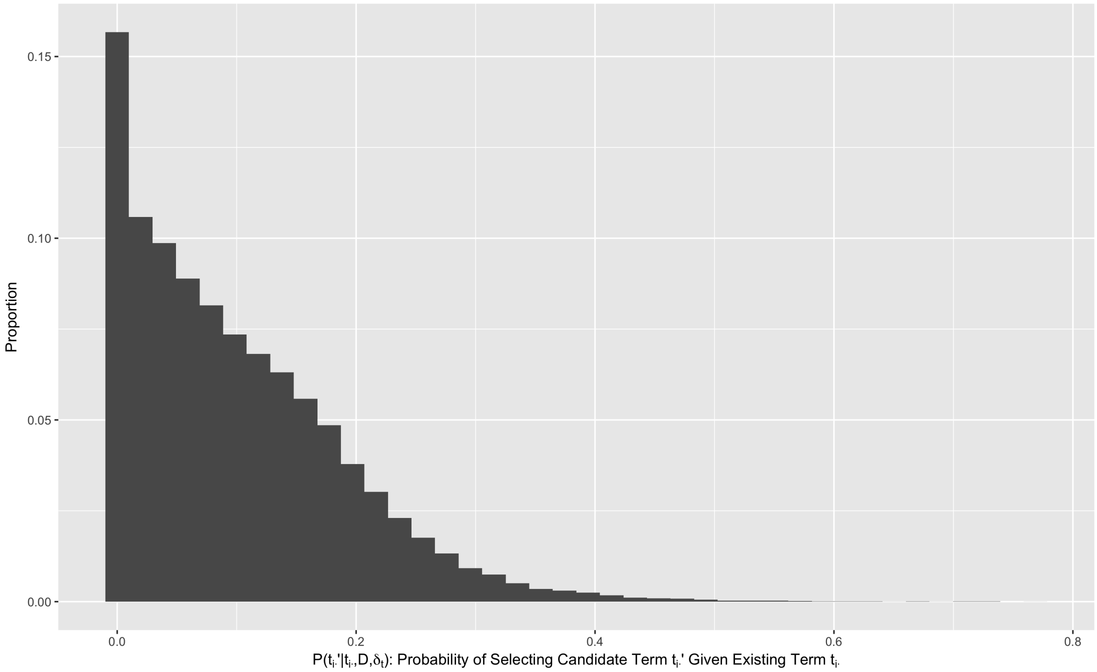


Figure 4.10: Histogram of term selection conditional probabilities.

[23] `http://www.imgur.com`

$t_{i\cdot}$	hair	egg	cat	magnolia	dog
$t'_{i\cdot}$	coloring	sandwich	imgur	room	breeds
	dye	sauce	grumpy	formal	collars
	natural	buttermilk	fur	teal	thirsty
	highlight	creamy	litter	star	pointer
	beauty	grilled	dining	pajamas	retriever

Table 4.3: Terms $t'_{i\cdot}$ maximizing $P(t'_{i\cdot}|t_{i\cdot}, D, \delta_{\mathbf{t}})$.

4.5.2 Overcoming the Violation of the Assumption of Independence between Terms and Documents

Although word and document selections are based on similarity up to this point, there is no guidance as to how adding a new term occurrence t' changes the semantic cohesion between some d and d^*. Rather than *always* sampling a word from $\mathbf{S}_{\mathbf{t}}$, we augment the bag-of-words with a new term according to the change in semantic cohesiveness (or lack thereof) between the original short text d_m and the augmented bag-of-words d^*_{ml}. We start with a bag-of-words d_m corresponding to a short text. Next, propose a change to this bag-of-words as discussed in Section 4.3. Then, we accept the new word into the augmented bag-of-words according to how well it improves the overall cohesiveness of the overall text. In natural language processing, pointwise mutual information (PMI) is a common metric used to measure entropy and cohesiveness between two words or among a set of words[59]. Pointwise mutual information is defined in Equation 4.6 for this use case as

$$I(d_m) = \sum_{i \neq j} P(t_{im}, t_{jm}) \log \left(\frac{P(t_{im}, t_{jm})}{P(t_{i\cdot}) \cdot P(t_{j\cdot})} \right) \tag{4.6}$$

and is dependent on the probabilities of observing terms $t_{i\cdot}$ and $t_{j\cdot}$ in the corpus D – it is computed over every pair of words in short document d_m. Next, the same computation is calculated on d^*_{ml} and denoted $I(d^*_{ml})$. Then, the decision rule illustrated in Equation 4.7 was used to determine whether or not to accept new word t'_{im} from d'_m into the augmented bag-of-words d^*_{ml}. This probability of state change is inspired by simulated annealing[91] and similar Markov Chain Monte Carlo methods[150], and a variant was proposed by Xu and Croft in [179].

$$P(d_m \to d^*_{ml} | t'_{im}) \propto \min \left(1, \exp \left\{ \frac{I(d^*_{ml}) - I(d_m)}{T} \right\} \right) \tag{4.7}$$

Just as in simulated annealing, T represents the temperature of the system and was chosen to be 1 so that the change in PMI is the only driving factor in the acceptance probability.

4.6 Topic Models and Classifiers

Once we have a corpus (or set of corpora as will be discussed in the next chapter) containing augmented bag-of-words representations, we can then use a classifier that should hypothetically perform better than the original classifiers used on the unaltered short texts. To illustrate and test the theory, a few different topic models and classification methods were constructed including Supervised Latent Dirichlet Allocation (sLDA) and a promising, but classical approach using linear support vector machines (SVM) implemented as `libshorttext`. We will also compare the experimental results to a linear SVM that uses semantic vectors from LSA as features instead of the simple bag-of-words approach. This method was studied in [111], [83] and [12] and is referred to as `SVM+LSA` for brevity. In the next chapter, we test the proposed data augmentation method using an ensemble of LSA and either sLDA or SVM to perform several experiments. In this chapter, we focus only on the analysis to pick the proper models and hyperparameters that yield the best model fit.

4.6.1 `libshorttext`: a Linear SVM Classifier Approach

`libshorttext`[182] is a package that uses SVM classification and prediction from a similar package called `LIBLINEAR` and uses some of the state-of-the-art preprocessing algorithms for short text classification. Feature scaling such as word count and TF-IDF is included as a user option and the user can remove stopwords, perform stemming and use bigrams[24] as features.

[24] A phrase containing two words treated as one token[1].

Since these features are already implemented as part of this research, the corresponding features in `libshorttext` were disabled. `libshorttext` supports Support Vector Classification using L1 and L2 penalties as well as logistic regression. The library also supports an automatic grid search for locating the optimal values for the hyperparameter C. For this research, part of the training data was sequestered as a validation set and the researcher used grid search on this validation set to pick the parameter C. Table 4.4 shows the micro- and macro-averaged performance metrics for each type of classifier generated by `libshorttext` and the LSA-based SVM model on the original testing data. The micro- averaged precision, recall, and F1-score are excluded from Table 4.4 because they are all equivalent to the micro-accuracy for the special case of a binary classifier. The micro-accuracy takes into account the distribution of the categories in the original corpus and sums the true positives and true negatives individually for each category and then normalizes by the total number of observations. The results show that there is a not much variation in classifier F1-score performance across the feature scaling, penalty types and modeling methods in `libshorttext` with micro-accuracy ranging from 0.57 to 0.72 and F1-score ranging from 0.638 to 0.645. Using word counts with a linear SVM using L2 penalty performs the best by a marginal amount compared to other binary classifiers according to F1-score. Table 4.5 shows the classical machine learning diagnostic metrics computed individually on each class for the best binary SVM model. Similar tables for the other classifiers are provided in Appendix B.1.

From Table 4.5 we see that the F1-score for the category classifiers ranges from 0.14 to 0.72. Three categories, `green`, `overall`, and `science` perform terribly under this classifier. Excluding these three categories, the F1-score ranges from 0.52 to 0.72 which calls for improvement. `overall` is a catch-all category for posts not labeled in any other category. While `overall` should probably be removed from modeling, and `green` and `science` should perhaps be merged with `technology`, the researcher felt that it would be of interest to see how the classifiers would perform leaving them as is. As with the rest of this manuscript, the hypothesis is that data augmentation can improve these metrics for `libshortttext` – most notably the linear L2 SVM with word counts as features. In conclusion, the models and feature selection methods used in `libshorttext` do not yield great results, likely because

Model / Metric	Accuracy Macro	Accuracy Micro	Precision Macro	Recall Macro	F1 Score Macro
Linear SVM L2, TF-IDF	0.911	0.696	0.720	0.624	0.668
Linear SVM L2, Binary	0.889	0.645	0.697	0.628	0.644
Linear SVM L2, Word Count	0.832	0.627	0.725	0.638	0.679
Linear SVM L2, Term Frequency	0.899	0.587	0.701	0.628	0.663
Linear SVM L1, TF-IDF	0.894	0.628	0.703	0.619	0.638
Linear SVM L1, Binary	0.933	0.614	0.691	0.625	0.640
Linear SVM L1, Word Count	0.885	0.622	0.691	0.625	0.639
Linear SVM L1, Term Frequency	0.900	0.601	0.691	0.626	0.641
Logistic Regression, TF-IDF	0.931	0.623	0.700	0.624	0.642
Logistic Regression, Binary	0.840	0.675	0.700	0.619	0.639
Logistic Regression, Word Count	0.898	0.622	0.700	0.620	0.639
Logistic Regression, Term Frequency	0.849	0.567	0.700	0.620	0.639
Linear SVM with LSA Vectors	0.920	0.732	0.855	0.481	0.576

Table 4.4: Evaluation metrics for each category using `libshorttext`.

the texts included in each document (title) are still too short.

Table 4.6 shows the macro- and micro- averaged performance metrics for `SVM+LSA`, the SVM model trained on LSA vectors and this model is used as a second baseline. We see that the F1-score for the category classifiers ranges from 0.06 to 0.78. We see that the same three categories: `overall`, `science` and `green` still perform poorly. Again, if we exclude these three categories, the F1-score ranges from 0.64 to 0.78 which is significantly better than the SVM model we selected using words as features and L2 penalty.

Metric / Class	`autos`	`business`	`entertainment`	`green`	`living`	`overall`	`politics`	`science`	`sports`	`technology`
Accuracy	0.887	0.852	0.763	0.883	0.755	0.878	0.849	0.875	0.771	0.810
True Positive Rate/Recall	0.400	0.528	0.558	0.143	0.597	0.088	0.526	0.219	0.544	0.592
True Negative Rate	0.994	0.972	0.908	0.993	0.891	0.998	0.969	0.991	0.902	0.953
False Positive Rate	0.006	0.028	0.092	0.007	0.109	0.002	0.031	0.009	0.098	0.047
False Negative Rate	0.487	0.359	0.329	0.744	0.290	0.975	0.361	0.668	0.343	0.295
Precision	0.761	0.775	0.778	0.564	0.773	0.360	0.751	0.652	0.652	0.907
F1 Score	0.524	0.628	0.650	0.228	0.674	0.141	0.619	0.328	0.593	0.716
Estimate of C	1.000	1.000	1.000	1.000	1.000	0.250	1.000	1.000	1.000	1.000

Table 4.5: Per-class metrics for linear SVM with L2 penalty and word count as features, via `libshorttext`.

Metric / Category	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.975	0.932	0.819	0.977	0.797	0.979	0.954	0.966	0.873	0.925
True Positive Rate/Recall	0.522	0.702	0.601	0.294	0.692	0.033	0.526	0.143	0.590	0.704
True Negative Rate	0.997	0.990	0.972	0.999	0.954	1.000	0.996	0.999	0.987	0.988
False Positive Rate	0.003	0.010	0.028	0.001	0.046	0.000	0.004	0.001	0.013	0.012
False Negative Rate	0.478	0.298	0.399	0.706	0.308	0.967	0.474	0.857	0.410	0.296
Precision	0.812	0.884	0.847	0.765	0.848	0.933	0.862	0.842	0.887	0.872
F1 Score	0.636	0.783	0.703	0.425	0.762	0.063	0.653	0.245	0.709	0.779
Estimate of C	1.000	1.000	1.000	1.000	1.000	0.250	1.000	1.000	1.000	1.000

Table 4.6: Per-class metrics for `SVM+LSA`.

4.6.2 Supervised Latent Dirichlet Allocation (sLDA)

Supervised Latent Dirichlet Allocation (sLDA) was described in Section 2.8. As opposed to classic Latent Dirichlet Allocation, sLDA fits words into topics such that the fitted topics maximize the likelihood of the target classification, whereas LDA fits topics based on word counts. sLDA provides a powerful advantage over LDA because there is an inherent coupling between the fitted topics and the response variable[117]. sLDA with logistic loss also fits nicely within a binary classifier framework because all of the usual binary evaluation metrics can be computed. sLDA also has the advantage over SVM in that it supports posterior class probabilities for each document whereas SVM under most loss functions does not[138]. This means that we can evaluate the model using the standard ROC curve and the holistic area under the ROC curve (AUC) measure that evaluates a classifier across all choices of threshold.

Currently there are only two major implementations of sLDA, both provided by the original researchers. The first uses variational EM to fit a discrete choice model[25] [36] and the second is included as part of a larger R package `lda` package [31] for topic modeling and uses Gibbs sampling. After experimenting with both, the R package by Chang was chosen for this work as it has a much more intuitive interface, supports logistic loss for classification, and allows the user to easily construct other loss functions, such as multinomial logit for

[25] `http://www.cs.cmu.edu/~chongw/slda/`

multiclass classification among others. Chang's Gibbs sampling implementation is also a few orders of magnitude faster than the variational EM implementation.

As with LDA, several parameters must be determined a priori or estimated: the number of topics K and the topic distribution over documents hyperparameter α. A series of 10 binary classifiers, one for each label in the corpus was constructed – thus, a series of `label/not-label` classifiers, also known as one-hot encoding[26] or a one-versus-all classifier. The same validation set from the SVM experiment was used to perform experiments varying the following parameters by performing cross-validation and taking the average of the performance:

1. the number of topics at values of $K \in \{5, 10, 50, 100\}$, where higher values of $K \geq 100$ were pre-determined to display very unstable and inconsistent results, thus not worth pursuing.

2. the hyperparameter α at typical default scalar values from the topic modeling literature including 1.0 and $\frac{1}{K}$.

3. the hyperparameter η was fixed at 0.1 as is common throughout the literature, and very unwieldy to estimate.

4. numerical results are presented for each classifier using the optimal cutoff threshold from the ROC curve – all of the ROC curve and precision-recall plots are provided in Appendix C.

It is worth noting that for this work, the Dirichlet distributions were assumed to be symmetric thus requiring a scalar for α rather than a vector. The rationale of this choice was that this is the most common treatment of the Dirichlet in general applied topic modeling literature and an investigation of non-symmetric Dirichlet priors would have required another dissertation.

The ROC curves and precision/recall curves compute the false positive rate, true positive rate and precision and recall respectively, at a large number of different cutoff thresholds. A cutoff threshold is defined as the posterior probability required to assign an observation as part of a class[64]. The choice of optimal threshold is context agnostic in this research – in a business setting, the choice of the cutoff threshold is often predicated by some cost whereas this work attempts to investigate a general case. The optimal cutoff threshold is the cutoff that yields performance closest to perfect in theory. In a ROC plot, the perfect theoretical classifier is one such that the true positive rate is 1.0 and the false positive rate is 0. Visually, the perfect theoretical classifier is the one that has a ROC curve extending closest to the top left corner of the plot. The worst theoretical classifier is one that is equivalent to chance, such that as the true positive rate increases, the false positive rate also increases. Visually, the worst theoretical classifier is the one closest to the line with slope 1 and intercept 0 in a ROC curve.

Table 4.7 displays some general global performance metrics derived from ROC and precision/recall curves for various K and α for each sLDA classifier. The AIC is the Akaike Information Criterion often used for model selection in regression and lower values purportedly represent better models[5][27]. It is important to note that only the AIC values for each classifier can be compared to AIC values from the same classifier as it is required that the same data and same response variable be used for AIC comparisons; naturally, each classifier uses a different response variable and thus AIC values cannot be compared across target classes. AIC attempts to prevent overfitting by penalizing based on the number of free parameters. In Table 4.7 we see that as K increases, AIC in general decreases and bottoms out at $K = 50$. This suggests that higher values of K yield better models until $K = 50$. Note that AIC does not tell the whole story. Since we are mainly interested in *predictive performance*, the area under the ROC curve (AUC) provides a globally descriptive metric for an sLDA model since it can output posterior class probabilities. In Table 4.7, we see that the AUC is maximized at $K = 50$. Since both AIC and AUC are optimal at $K = 50$ and a higher value of K does not, this suggests that the true optimal K is somewhere between 50 and 100 but for this research we will only consider a discrete set of values for K. The author of this manuscript feels that AIC is an unusual metric for the machine learning domain whereas ROC curves, precision/recall curves, F score and AUC are more widely accepted. Chang's `lda` package provides AIC for sLDA models, thus it was included in this analysis due to its widespread use in bread-and-butter statistical modeling. The AUC was computed using the `AUC`[8] package for R. AUC can be computed either on the ROC curve or the precision/recall curve[42], but since the author is interested in both positive and negative classification equally and generally, the ROC AUC is more useful and is already the more universal of the two metrics.

Table 4.7 and the graphs in Section C.1.2 show the ROC and precision/recall performance for each classifier at $\alpha \in \left\{1, \frac{1}{K}\right\}$ and various numbers of topics $K \in \{5, 10, 50, 100\}$. The ROC curves suggest that as the number of topics K increases, the performance of most of the classifiers improves, but that at some $K > 50$, performance decays, perhaps due to overfitting. For all stable K, the classifier for `technology` outperforms all other classifiers with an F1-score ranging from 0.537 to 0.624 and a ROC AUC ranging from 0.875 to 0.904. The classifier for `overall` consistently performs much worse than the other classifiers with an F1-score below 0.1 and an AUC ranging from 0.59 to 0.68. This is not surprising because the category `overall` in the corpus includes blogs that are not placed into any other category, as well as a spattering of blogs that are actually in other categories, but for some reason were not labeled as such by Technorati editors and/or users. This classifier's performance does not improve with increasing K – its F1-score, AUC and AIC remain relatively stable across K. The classifiers for `autos`, `business`, `green`, `politics` and `science` show significant improvement in F1-score with increasing K up to optimal K. `politics` and `science` also show significant improvements in ROC performance. The researcher hypothesizes that this may be because the vocabularies for the texts in these categories are more diverse and discriminative than those in the other categories and thus more topics leads to a better model. Table 4.7 also shows how classifier performance varies over the hyperparameter α. It appears that there is no clear effect of choosing one value of α over the other. For that reason, we will use the common default value $\alpha = 1$. Using a constant value makes more sense anyway because it prevents us from worrying about the correlation between K and α in this research.

		$\alpha = 1$			$\alpha = \frac{1}{K}$		
Category	K	AUC	AIC	F1	AUC	AIC	F1
autos	5	0.8410	17457	0.29022	0.8267	17783	0.27333
	10	0.8546	16057	0.36616	0.8261	18130	0.27929
	50	0.8738	12281	0.50462	0.8705	13916	0.42189
	100	0.8602	13124	0.54466	0.8473	14538	0.42391
business	5	0.8865	33256	0.54260	0.8744	34540	0.50367
	10	0.8699	34525	0.49937	0.8583	37845	0.45957
	50	0.8902	27647	0.58395	0.8909	30589	0.58487
	100	0.8741	30647	0.58819	0.8763	30732	0.54964
entertainment	5	0.8008	66183	0.57653	0.7667	76634	0.51021
	10	0.8087	65659	0.57032	0.8141	65947	0.58099
	50	0.8149	59617	0.56598	0.8285	58642	0.58601
	100	0.8131	63579	0.55965	0.8073	62577	0.55784
green	5	0.7278	19347	0.11974	0.8153	17746	0.17103
	10	0.7389	19519	0.13449	0.7805	18675	0.16358
	50	0.7528	17749	0.20099	0.7749	17963	0.18116
	100	0.7457	18302	0.24065	0.7671	17513	0.18352
living	5	0.8215	68893	0.61790	0.8496	60833	0.65834
	10	0.8225	68095	0.60908	0.8258	69429	0.61226
	50	0.8366	53608	0.63875	0.8429	58664	0.63478
	100	0.8269	59504	0.62571	0.8327	60993	0.61944
overall	5	0.6254	19036	0.06093	0.6834	18623	0.08057
	10	0.6305	18954	0.06376	0.6846	18355	0.09524
	50	0.6128	18742	0.06024	0.6605	18361	0.09228
	100	0.5856	19090	0.05533	0.6545	18389	0.09016
politics	5	0.7636	32433	0.22069	0.7750	32895	0.26921
	10	0.8159	28301	0.30459	0.7874	31757	0.26755
	50	0.8201	26674	0.39929	0.8500	24298	0.41927
	100	0.7989	27327	0.39374	0.8397	24349	0.41046
science	5	0.6930	25601	0.14701	0.6474	27166	0.10337
	10	0.7489	22182	0.24242	0.7240	24566	0.16279
	50	0.7642	19474	0.33949	0.7487	22272	0.22000
	100	0.7234	22732	0.25645	0.7207	22408	0.21996
sports	5	0.7926	60133	0.51160	0.8111	55315	0.58050
	10	0.8092	52456	0.52896	0.8063	52946	0.53240
	50	0.8107	46423	0.53887	0.8224	49590	0.52513
	100	0.7952	51657	0.52608	0.8123	50041	0.52200
technology	5	0.8753	38299	0.53739	0.9025	31796	0.61009
	10	0.8788	35736	0.54438	0.8984	33088	0.58476
	50	0.9005	27684	0.62394	0.9038	28319	0.61734
	100	0.8960	28674	0.61921	0.8961	28612	0.60722

Table 4.7: sLDA global evaluation metrics for $\alpha = 1$ and $\alpha = \frac{1}{K}$.

In conclusion, the F1-scores for sLDA on most of the classifiers are across all tested values of K and α were poor, which gives us a lot to work with for the proposed algorithm. Further research in this manuscript focuses on the sLDA model with $\alpha = 1$ and $K = 50$ topics based on the previous analysis. While optimal K varied slightly across categories, it is believed that the difference is not significant. Similar to picking one value for α, picking the most common optimal value for K is simpler for comparisons. Additionally, this research is not interested in obtaining optimal results for the baseline classifiers – rather, we are interested in the change in performance between the baseline classifiers and the classifiers using augmented data. Figure 4.11 shows the ROC curve for this particular sLDA model. Table 4.8 shows the classical machine learning metrics for the sLDA model at its optimal threshold.

Metric	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.849	0.830	0.743	0.678	0.774	0.575	0.742	0.733	0.770	0.846
True Positive Rate/Recall	0.763	0.808	0.735	0.687	0.745	0.592	0.752	0.663	0.702	0.821
True Negative Rate	0.851	0.833	0.746	0.677	0.784	0.574	0.741	0.735	0.783	0.849
False Positive Rate	0.149	0.167	0.254	0.323	0.216	0.426	0.259	0.265	0.217	0.151
False Negative Rate	0.237	0.192	0.265	0.313	0.255	0.408	0.248	0.337	0.298	0.179
Precision	0.129	0.331	0.448	0.049	0.554	0.029	0.143	0.084	0.378	0.383
F1 Score	0.505	0.584	0.566	0.201	0.639	0.060	0.399	0.339	0.539	0.624
AUC Score	0.874	0.890	0.815	0.753	0.837	0.613	0.820	0.764	0.811	0.901

Table 4.8: sLDA experimental results with optimal $K = 50$ topics and optimal $\alpha = 1$.

4.6.3 Comparison of SVM and sLDA for Short Texts

The linear SVM models implemented in `libshorttext` and the generative model implemented in sLDA provide two different approaches to a similar problem, classifying text. SVM uses sparse symbolic representations whereas sLDA uses clusters of words as predictors. The SVM method can use a variety of different methods to score each word/feature including count, binary occurrence and TF-IDF score. The SVM method also can use two different penalties (L1 and L2) and both a standard fit and a fit with regularlization. On the other hand, sLDA uses clusters of words grouped into topics such that the topics and other parameters identified maximize the likelihood of the data, and depend on the number of top-

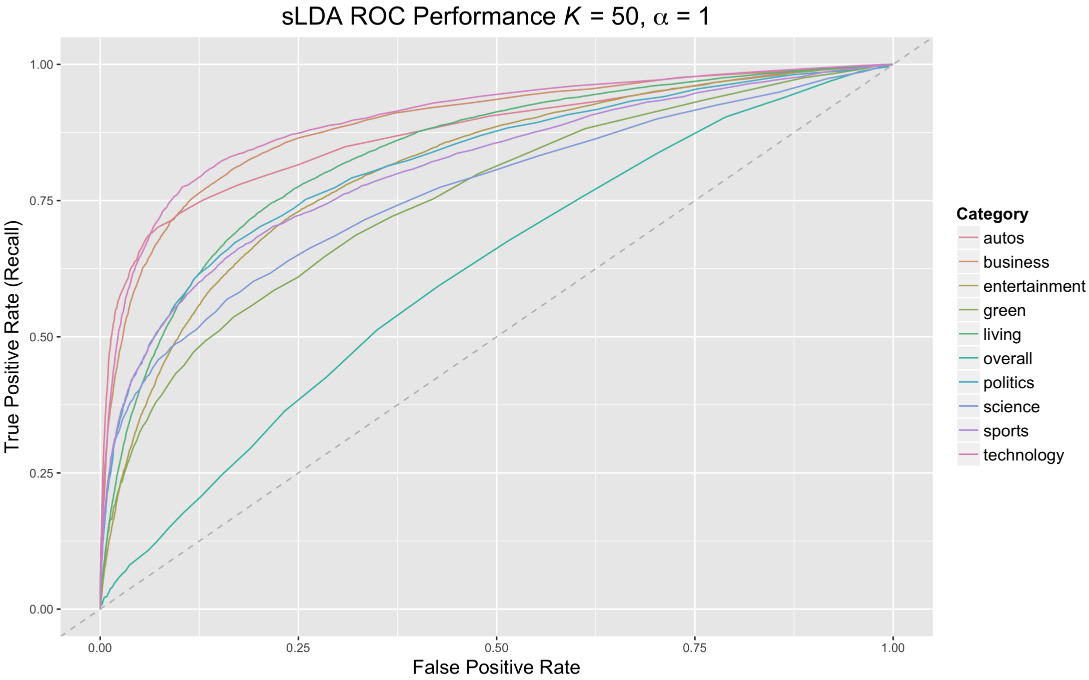


Figure 4.11: ROC curve for optimal $(K = 50, \alpha = 1)$ sLDA model trained and tested on original data.

ics/predictors K and a hyperparameter α on the Dirichlet prior guiding the document-topic assignment distribution. For sLDA to work as theorized, words are scored using their frequencies. Binary presence/absence and TF-IDF violate the assumptions made in the LDA model based on conjugacy since TF-IDF scores follow a different distribution than word counts[19]. sLDA in this work used a logit link function. Based on all of these parameters, the closest comparable SVM model to sLDA is a linear SVM using L2 penalty and word count feature scaling. Finally, based on the assumptions of the underlying LDA model, TF-IDF models are not comparable. It is worth noting that the linear SVM model is compared to sLDA in this section only to determine which model classifies short texts better a priori. In Chapter 6, the linear SVM method will be compared to sLDA using the proposed data augmentation method.

Based on the results in Tables 4.5 and 4.8, linear SVM performs superior to sLDA at baseline. For categories `overall`, `entertainment`, `politics` and `technology`, SVM yielded most impressive differences with sLDA with SVM performed better. The `science` category was the only category that performed worse under SVM than under sLDA at baseline.

4.6.4 A Comment on Classic Latent Dirichlet Allocation (LDA)

A thorough treatment of LDA was presented in Section 2.4. Initially, LDA was considered for clustering and classifying documents. Using LDA as a benchmark is difficult for this research because most standard treatments of LDA in the literature use *qualitative* measures to evaluate the clustered topics, or researchers simply compare models using likelihood or perplexity[113] without considering a response variable. Additionally, LDA is an unsupervised method thus topic labeling and ordering is not preserved across samples. That is, topic 2 may refer to topic `technology` whereas in another sample it may be topic 5 – all we know is that each topic contains similar documents, but we do not know what the topic label is. It is also possible that an individual topic estimated from LDA may not map to a single topic in the Technorati ontology. In an effort to standardize the experimental results, a manual method of computing metrics and labeling clusters was considered as follows. Given K topics,

1. classify an augmented document d^*_{ml} as part of topic k if the majority of words in that document are assigned to topic k,
2. given the true labels for each document, assign the most frequently appearing label in each topic as the label for topic k, to produce a mapping across iterations,
3. then measure classification error in the same way as supervised learning.

Unfortunately, this method did not yield useful analysis because the topics never mapped to topics in the original Tehcnorati ontology, even after tuning the hyperparameter α to an appropriate value for the data. Instead, a method that assigns words to topics to better model the original classification is preferred. For this reason, LDA was dropped as a proposed model and replaced with sLDA.

4.6.5 Other Variants of LDA

Aside from sLDA, there are several other variants of LDA; most of the base variants were described in Chapter 2. The corpus used for this research extends the document categories into a hierarchy that can be used with hLDA. For example, the `politics` category is further subdivided in the data for U.S. politics and international politics. The hierarchical nature of the data was ignored for this research and hLDA was not investigated. Another feature of the data is that some of the topics and words are correlated in the corpus. For example, the correlated topic model (CTM) may better model certain categories such as `green` and `science`. Naturally, blog posts are dynamic with topics changing over time. A dynamic topic model (DTM) may model data as more and more data is scraped. While this dataset provides lots of opportunity for future study with LDA variants, only sLDA was considered for this manuscript.

4.7 Chapter Summary

In this chapter, we introduced a preprocessing framework using data augmentation inspired by the bootstrap. The concepts of the statistical bootstrap were reintroduced to the reader and a description of the modifications used for this research was presented. We also introduced the reader to the experimental data used in this research – a categorized list of blog post titles extracted from the Technorati blog aggregator. Various exploratory statistics about the documents and the terms in each document were presented as well as statistics regarding document and term selection probabilities used in the proposed sampling technique. We also discussed the development of the semantic space forming a population for resampling from a matrix factorization technique called Latent Semantic Analysis. The `libshorttext` package and its suite of SVM models based on `LIBLINEAR` was introduced, and several types of SVM models, penalty functions and feature types were investigated. We saw that Linear SVM with L2 penalty and word counts as features performed the best out of all the binary classifiers. We also looked at another baseline using LSA vectors as features in a linear SVM and found that this model performed better than the models using bag-of-words in some cases and comparable in others. Supervised LDA (sLDA) was introduced for classification of augmented texts. We saw that SVM did markedly better than sLDA and found that the best sLDA results occurred when the hyperparamer $\alpha = 1$ and the number of topics $K = 50$.

CHAPTER 5

Experimental Setup

In this chapter, we introduce several research questions and describe the experiments developed to answer each question. Naturally, describing the experiments requires formalizing the proposed data augmentation method as an algorithm. As discussed in Chapter 4, we will use Latent Semantic Analysis (LSA) to induce a population for document and term sampling, and use linear SVM, and Supervised Latent Dirichlet Allocation (sLDA) as the topic classifiers. The SVM classifier uses word counts as features and the L2 penalty function. The hyperparameter C was chosen using grid search on a dedicated validation set each time an SVM classifier was trained. The Supervised Latent Dirichlet Allocation (sLDA) classifier uses the logit link function, $K = 50$ topics and Dirichlet hyperparameter $\alpha = 1$.

5.1 Research Questions

This manuscript investigates and answers several research questions. Naturally, the main question is what the effect, if any, the proposed data augmentation preprocessing step has on classifier performance, and this will be investigated through the following research questions:

1. How many words should be sampled to create the larger augmented bag-of-words? Or, more precisely, how much longer should the augmented document d^* be to improve classification?

2. Where should the data augmentation preprocessing occur: in the training phase only, the testing phase only, or both?

3. Is the final term acceptance step presented in Section 4.5.2 and proposed by [179]

necessary? Or, can we accept all candidate terms t'?

4. Which model, SVM or sLDA, yields best performance under data augmentation?

5. Does the optimal model under data augmentation perform better than a baseline SVM with LSA vectors as features (`SVM+LSA`)?

5.2 Experiments

Each research question posed in the previous section is associated with an experiment.

5.2.1 Effect of Augmentation Size ε

The proposed data augmentation method is inspired by the bootstrap sampling step and relies on augmenting a bag-of-words representation with semantically similar terms. It is important to consider *how many* terms to add. The parameter ε specifies the relative change in length between some original short text document bag-of-words d and the augmented bag-of-words d^*. For example, $\varepsilon = 1$ means that the augmented representation is 100% larger than the original document length – $|d^*_{ml}| = (1 + \varepsilon)|d_m| = 2|d_m|$. Naturally, this parameter could be unbounded, but that would be impractical for research. Additionally, it is the researcher's belief that as ε grows, there is potential for a great amount of bias to be introduced, particularly as the vocabulary of $\mathbf{S_t}$ is exhausted. This is an assumption that can be tested in further research, but is not within the scope of this manuscript. A value of $\varepsilon = 0$ denotes a completely unchanged document since substituting and removing words from a short document is not considered in this research. The special case (baseline) where $\varepsilon = 0$ was discussed in Chapter 4 and simply consisted of training and testing with the raw data. This research experiments with varying ε within the range $0.1 \leq \varepsilon \leq 2$ in increments of 0.1. The upper bound of 2 was chosen somewhat arbitrarily, and represents the situation where there are many more new terms in the augmented representation than there were in the original text. The upper bound could have been set higher, but such a value would be redundant.

5.2.2 Effect of When Terms are Sampled for Augmentation

Another point of interest in this research is when the augmentation step should be applied. Initially, this research only considered applying the augmentation step to both the training and testing data as it is customary to perform the same preprocessing on both the training and test data in machine learning[65]. The researcher determined that it would be interesting to also experiment with two non-standard approaches – using the data augmentation method on only the training data (denoted as variation `DA1`), and using it only on the testing data (denoted as variation `DA2`). The hypothesis was that performing the augmentation step on only the training data would add more signal to the resulting classifiers and make them more "sensitive" to the terms in the testing data. On the other hand, the reasoning behind only using the augmentation method on the testing data is that augmented data may "activate" more features in the classifier allowing more accurate prediction. To illustrate this concept, consider standing in a noisy room with a microphone at the other end that records only one particular voice and nobody else's. `DA1` represents the case where the microphone is better tuned to the frequency of the speaker's voice and is thus more sensitive to it. `DA2` would be the case where the microphone remains as is, but the speaker yells very loudly to overcome the noise in the room and thus be recorded. Variation `DA0` refers to the original raw data where augmentation was not used anywhere, and `DA12` refers to the case where augmentation is used for both the training and testing data. In [65], Gonzalez and colleagues suggest that `DA1` and `DA2` are unmatched training and testing sets, whereas `DA12` is the traditional matched training and testing set situation.

5.2.3 Effect of the Final Acceptance Probability Step

A third experiment involves the addition of the final acceptance probability step introduced in Section 4.5.2. This experiment compares performance of classifiers both with and without this final acceptance probability step. The experiments without the final term acceptance step consists of augmented corpora D_l^* that accept all sampled t' terms regardless of their change on semantic cohesiveness. The experiments with the final term acceptance step consist of augmented corpora containing only the terms selected probabilistically based on change in semantic cohesiveness. The researcher calls each one of these setups an augmentation strategy for brevity. In the experiments, the augmentation strategy is often combined with the variations discussed in Section 5.2.2 and together referred to as a experimental configuration, for brevity.

5.2.4 Choice of Model

As discussed in Chapter 4, linear SVM and supervised LDA were chosen as the two models for analysis. Separate experiments were evaluated for both models for research questions 1, 2, and 3. We used the AUC to evaluate model performance for the sLDA model since it allows the construction of posterior class probabilities, and we used F1-score to evaluate model performance for the SVM model. Although accuracy was provided in the tables in Chapter 4, it was not used to evaluate either classifier because the class distributions are imbalanced, so it is not a good measure for this reason. In Experiment 4, we compare SVM and sLDA directly using the F1-score metric.

5.2.5 Performance of Data Augmentation vs. Baseline

An SVM model using LSA vectors as features was used as a second baseline. This final experiment compares performance for the optimal classifier under data augmentation against this baseline.

5.3 Controlling for Variability in Experiments

As mentioned in Section 4.1, bootstrapping is a probabilistic process and generates a new sample at each iteration. In this work, each iteration of the data augmentation step generates a new corpus D_l^* containing augmented texts of length $|d_{ml}^*| = (1+\varepsilon)\,|d_m|$ each. Each corpus consists of a training, validation and testing set and the documents within each set remains consistent across all iterations. Depending on the particular experimental variation, only certain parts of the corpus may be augmented. In `DA1` and `DA12`, the training set is augmented. In `DA2` and `DA12`, the testing/unseen set is augmented. The validation set is processed the same as the training set for consistency. Since each iteration yields different data, the researcher decided to perform 100 iterations at each parameter set which generated 100 different versions of each short text under each experimental configuration, and was also proposed by [74]. Each corpus was used to train a separate classifier and each classifier has its own set of performance metrics. The researcher believes that these 100 sets of results infer the population distribution of the performance metrics for each experimental configuration. It is worth noting that each time an SVM classifier was trained, we used a grid search to select the hyperparameter C on a separate validation set since its value relies heavily on the distribution of the data which is different in each corpus[34][151]. The C parameter tended to vary quite a bit depending on the category and the value of ε. The validation set was not used for the sLDA classifier as the hyperparameters were held constant based on the analysis performed in Chapter 4.

Using the two (2) classifiers described in Chapter 4, 100 iterations were run for all combinations of ε (20), the three `DAx` variations (3), with and without the final acceptance step (2), for each target class (10), resulting in a total of approximately 240,000 trained classifiers. For the analyses in Chapter 6 the metrics were averaged over various experimental parameters.

5.4 Formalizing the Experimental Algorithms

We have described the data augmentation procedure and discussed several parameters and experiments to test the efficacy of short text classification using the proposed method. In this section, we formalize the experiments as algorithms. Due to the number of moving parts, we provide the framework of the experiments as pseudocode in this section. Table 5.1 shows which models and datasets are used for each of the data augmentation variations.

	Where augmentation occurs...	Training Data	Testing Data	Model Used
DA0	It does not; raw data, not augmented	Raw data not augmented	Raw data not augmented	Trained from raw data
DA1	Training data only	Augmented	Raw data not augmented	Trained from augmented training data
DA2	Testing phase only	Raw data not augmented	Augmented	Use classifier from DA0 to predict on augmented data.
DA12	Both phases; fully augmented.	Augmented	Augmented	Use classifier from DA1 to predict on augmented data.

Table 5.1: Models and data used for each data augmentation algorithm variation.

5.4.1 Data Generation

The proposed data augmentation method is a preprocessing framework, so most of its character comes to life in the data generation phase. Algorithm 10 shows the pseudocode for generating experimental training and test sets that answer each of the questions posed in the previous section. Data is generated for the following situations and parameters:

1. differing amounts of augmentation ($0.1 \leq \varepsilon \leq 2$),
2. phase at which augmentation occurs (the `DAx` variation),
3. each augmentation strategy: with and without the final probabilistic acceptance step based on PMI,
4. for each, the term sampling process is performed 100 times to account for variability in the sampling process.

The pseudocode in Algorithm 10 shows the general control flow of the data generation stage, but most of the work is performed in the `generate_data` routine and is illustrated in Algorithm 11 where we can more clearly see the changes in processing that occur based on variation.

Algorithm 10: Experimental data generation (augmentation)

```
Data: Corpus $D$ of short texts divided into $D_{\text{train}}$, $D_{\text{val}}$ and $D_{\text{test}}$, semantic space $S$
Result: Training, validation sets, testing sets, one set for each combination of experimental parameters
foreach $aug\_strategy \in \{\mathtt{without\_pmi}, \mathtt{with\_pmi}\}$ do
    foreach $variation \in \{\mathtt{DA1, DA2, DA12}\}$ do
        for $\varepsilon = 0.1$ to 2.0 by 0.1 do
            for $l = 1$ to 100 do
                $D^*_{l\,\text{train}}, D^*_{l\,\text{val}}, D^*_{l\,\text{test}} =$
                    $\text{generate\_data}(D_{\text{train}}, D_{\text{val}}, D_{\text{test}}, S, \text{aug\_strategy}, \text{variation}, \text{iteration}, \varepsilon)$
                $\text{write}(D^*_{l\,\text{train}}, D^*_{l\,\text{val}}, D^*_{l\,\text{test}})$
```

Algorithm 11: Procedure `generate_data`

```
Input: Train, validation and test sets $D_{\text{train}}, D_{\text{val}}, D_{\text{train}}$, semantic space $S$, aug_strategy
       $\in \{without\_pmi, with\_pmi\}$, variation, iteration, $\varepsilon$
Output: Augmented train and test sets, $D^*_{l\,\text{train}}, D^*_{l\,\text{test}}$
if $variation = \mathtt{DA1}$ or $variation = \mathtt{DA12}$ then
    // augment the training set
    $D^*_{l\,\text{train}} = \text{augment\_documents}\left(D_{\text{train}}, \varepsilon, aug\_strategy, S\right)$
    $D^*_{l\,\text{val}} = \text{augment\_documents}\left(D_{\text{val}}, \varepsilon, aug\_strategy, S\right)$
if $variation = \mathtt{DA2}$ or $variation = \mathtt{DA12}$ then
    // augment the testing set
    $D^*_{l\,\text{test}} = \text{augment\_documents}\left(D_{\text{test}}, \varepsilon, aug\_strategy, S\right)$
return $D^*_{l\,train}, D^*_{l\,val}, D^*_{l\,test}$
```

Algorithm 12 shows the pseudocode for the resampling stage, the process described in detail in Section 4.3. The procedure `weighted_sample` is taken to be a function that chooses an element proportional to a similarity function, in this case cosine similarity. If f is the mathematical equivalent of `weighted_sample`, we have

$$f(v, x_i) \propto \cos\left(\frac{v \cdot x_i}{||v||||x_i||}\right)$$

where v is a vector representing a lower dimension representation of a term or a document and x_i represents a sampling unit in the semantic space being queried. The procedure **`tfidf_sample`** samples a word from the current document d_m proportional to its TF-IDF score and that of the other terms in the document.

Algorithm 12: Procedure `augment_documents`

```
Input: Some corpus $D$, semantic space $S$, aug_strategy $\in \{without\_pmi, with\_pmi\}$, iteration $l$, $\varepsilon$
Output: Augmented corpus, $D^*_l$
for $d_m \in D$ do
    $d^*_{ml} = d_m$
    while $|d^*_{ml}| < (1+\varepsilon)\,|d_m|$ do
        // select a document $d'_m$ from $S_d$ according to semantic similarity to $d_m, \delta_d$
        $d'_m = \text{weighted\_sample}\,(d_m, S_d)$
        // select a term $t_{im}$ in document $d_m$ according to influence via TF-IDF,
        // as a target for resampling
        $t_{im} = \text{tfidf\_sample}\,(d_m)$
        // Sample a word $t'_{im}$ from $d'_m$ most related to $t_{im}$
        $t'_{im} = \text{weighted\_sample}\,(t_{im}, S_t, d'_m)$
        if $aug\_strategy = \texttt{with\_pmi}$ then
            // Compute PMI with and without the new term $t'_{im}$.
            $I\,(d_m) = \text{pmi}\,(d_m)$ ;                                  // See Equation 4.6
            $I\left(d^*_{ml}\right) = \text{pmi}\left(d^*_{ml}\right)$
            $p \propto \min\left(1, \frac{1}{Z} \times \exp\left\{I(d_m) - I(d^*_{ml})\right\}\right)$
            $r = \text{rand}()$
            if $r \leq p$ then
                $d^*_{ml} = d_m \cup t'_{im}$
        else
            $d^*_{im} = d_m \cup t'_{im}$
        $D^*_l = D \cup d^*_{ml}$
    return $D^*_l$
```

5.4.2 Classifier Training and Evaluation

Once all of the experimental data was generated using the data augmentation framework, we trained classifiers using the data and then evaluated their performance using the true labels. Algorithm 13 shows the process of training each of the experimental models over 100 iterations for each experimental configuration, category and choice of augmentation rate ε. The procedures

- `train_svm_model`,
- `train_slda_model`,
- `predict`,
- `evaluate_model`,
- `grid_search`

and the various `load_*` procedures are black box that need not be defined in the pseudocode.

5.5 Chapter Summary

In this chapter, several research questions related to the proposed data augmentation method were proposed and discussed. For each research question, the researcher developed an experiment to determine if the augmentation process improves classification performance on short texts, and which parameters, if any, affect the performance of the technique. Three variations of the data augmentation method were presented – each variation introduces the data augmentation step during different phases. Two modes of analysis were also introduced, a baseline mode that simply accepts all sampled terms into an augmented bag-of-words, and a second mode that accepts words according to how the term changes the semantic cohesiveness of the text using pointwise mutual information. We also look at the effect of the size of the augmentation, a parameter denoted as ε. Of course, we also consider whether or not performance is better for the linear SVM method or the supervised LDA method. Pseudocode was also presented as a way to formalize the experiments.

Algorithm 13: Experimental model training and evaluation

```
Data: Input corpora $D^*$ for training, cross-validation, and testing, true class labels $\mathbf{y}$.
Result: trained models $\mathbf{m}$ and evaluation data $\mathbf{e}$.
foreach $model \in \{svm, slda\}$ do
    foreach $aug\_strategy \in \{without\_pmi, with\_pmi\}$ do
        foreach $variation \in \{DA1, DA2, DA12\}$ do
            for $\varepsilon = 0.1$ to $2.0$ by $0.1$ do
                foreach $cat \in categories$ do
                    for $l = 1$ to $100$ do
                        if $variation = DA1$ then
                            $D^*_{l\,\text{train}}$ = load_augmented_training_data(aug_strategy, $\varepsilon$, cat)
                            $D^*_{l\,\text{val}}$ = load_augmented_validation_data(aug_strategy, $\varepsilon$, cat)
                            $D^*_{l\,\text{test}}$ = load_raw_testing_data()
                        else if $variation = DA2$ then
                            $D^*_{l\,\text{train}}$ = load_raw_training_data()
                            $D^*_{l\,\text{val}}$ = load_raw_validation_data()
                            $D^*_{l\,\text{test}}$ = load_augmented_testing_data(aug_strategy, $\varepsilon$, cat)
                        else if $variation = DA12$ then
                            $D^*_{l\,\text{train}}$ = load_augmented_training_data(aug_strategy, $\varepsilon$, cat)
                            $D^*_{l\,\text{val}}$ = load_augmented_validation_data(aug_strategy, $\varepsilon$, cat)
                            $D^*_{l\,\text{test}}$ = load_augmented_testing_data(aug_strategy, $\varepsilon$, cat)
                        if $model = svm$ then
                            $C$ = grid_search($D^*_{l\,\text{val}}, \mathbf{y}_{\mathbf{val}}$)
                            $\mathbf{m}$ = train_svm_model($D^*_{l\,\text{train}}, C$)
                        else if $model = sLDA$ then
                            $\mathbf{m}$ = train_slda_model($D^*_{l\,\text{train}}, K = 50, \alpha = 1$)
                        // predictions may be labels (SVM) or probabilities (sLDA)
                        $\hat{\mathbf{y}}$ = predict ($\mathbf{m}, D^*_{l\,\text{test}}$)
                        $\mathbf{e}$ = evaluate_model($\hat{\mathbf{y}}, \mathbf{y}$)
                        write($\mathbf{m}$)
                        write($\mathbf{e}$)
```

CHAPTER 6

Results

In this chapter we review the results of the experiments introduced in Chapter 5 and provide discussion of the results. The results are presented for both the Supervised Latent Dirichlet Allocation (sLDA) and `libshorttext` Support Vector Machine (SVM) models. Since sLDA is capable of reporting posterior probabilities, ROC curves are presented as a visual means of evaluating the sLDA classifier's performance. The AUC summarizes ROC performance by computing the area under the ROC curve. It is the researcher's opinion that the use of posterior probabilities is preferred, when possible, because it allows the use of a visual aide in determining classifier performance rather than simply a statistic.

Another metric used to evaluate a classifier's performance is the F1-score, the harmonic mean of precision and recall. According to [86], F1-score may be a more appropriate measure when the class distribution is imbalanced as it is in this dataset (see Table 4.1) *and* when the positive class is more crucial than the negative class. However, the F1-score is threshold dependent – that is, the F1-score is computed by first picking a probability threshold along the ROC curve (or precision-recall curve) and then computing the F1-score from it. This differs from the AUC as the AUC describes a classifier's performance across *all* thresholds. Researchers typically pick the threshold that maximizes a particular metric such as accuracy, true positive rate, or F1-score[132]. Despite this advantage, we lose a lot of information by using only the F1-score with the sLDA model. On the other hand, the SVM model does not natively allow the computation of posterior probabilities so the ROC curve concept does not apply. For this reason, the researcher instead chose to use F1-score to summarize the performance of the SVM models. Since there is no ROC curve, there are also no thresholds to consider, only one F1-score exists for each SVM classifier.

In Sections 6.1, 6.2, and 6.3, results are reported for each model *separately* and only compared *within* each model type. That is, there is a separate analysis for the sLDA model and the SVM model and comparisons are only made for like classifiers – across sLDA classifiers or across SVM classifiers, but not between sLDA and SVM classifiers, except in passing. In Section 6.4, sLDA and SVM models are compared directly. For that one experiment, both models must use the same metric so the researcher identified the optimal threshold for the sLDA model from the ROC curve and computed the F1-score. This F1-score can then be directly compared to the F1-score from the SVM model. Finally, in Section 6.5 we compare the best model under the optimal data augmentation strategy to a baseline SVM model using LSA vectors as features.

Before proceeding, it is important to describe some of the vocabulary used in this chapter. The parameter ε is referred to the augmentation rate and defines how many terms we sample into the bag-of-words representation. An experimental *variation* specifies when the data augmentation step takes place – in the training data only, in the testing data only, or in both datasets. These variations are denoted `DA1`, `DA2` and `DA12` with `DA0` denoting the raw data where augmentation occurs in neither the training set nor the testing set. There is also the final probabilistic term acceptance step using pointwise mutual information. Experiment 3 tests classifier performance both with and without this acceptance step, and is referred to as an *augmentation strategy*. For simplicity, each combination of variation and augmentation strategy is referred to as an experimental *configuration*.

6.1 Experiment 1 – Effect of Augmentation Size ε

The parameter ε denotes the size of some augmented bag-of-words d^* relative to the original short text document d as follows

$$|d^*| = (1 + \varepsilon)\,|d|$$

A particular value of ε is associated with a $\varepsilon\%$ increase in the length of d. For example, $\varepsilon = 1$ represents doubling the length of d, a 100% increase. The case where $\varepsilon = 0$ is synonymous with variant DA0, that is, the raw data – no augmentation performed at all.

6.1.1 sLDA

Figure 6.1 shows the effect of the choice of ε on the sLDA classifier performance as measured by AUC. A subplot exists for each experimental variation. Under each experimental configuration for the sLDA classifiers, it seems that $\varepsilon = 0.3$ yielded the highest AUC with surrounding values providing similar performance in AUC. At values $\varepsilon \geq 0.8$, the classifiers for each topic begin to perform worse than the DA0 raw data baseline. We see that overall, the augmentation process seems to give an immediate boost to performance over the classifiers trained on the raw data alone (DA0). The only category classifier that did not seem to be affected by the data augmentation is the elusive overall category. There is a significant bump in classifier performance as measured by AUC for smaller values of ε. In the raw data, AUC ranged from 0.61 to 0.9 but at $\varepsilon = 0.3$ it ranged from 0.61 to 0.93 (the lower bound is 0.8 if we exclude overall). While it may seem odd that this bump is followed by a steady decline in performance, the researcher has a hypothesis. It is believed that the bump in AUC for small values of ε is due to the addition of a *single* word to the bag-of-words representation, and the sampling procedure ensures that a highly discriminative or influential term is selected at the optimal ε. Adding further words to the text *may* add more noise in the form of bias deteriorating the performance of classifiers with higher values of ε. As we saw in Figure 4.7, the average text in this corpus contained between 4 and 5 words. For a document with 4 terms and $\varepsilon = 0.3$, we have $4\varepsilon = 1.2$ so we would add one additional term at each iteration, and a second term 20% of the time. Similarly, for a document with 5 terms and the same ε, we have $5\varepsilon = 1.5$ and would sample an additional word 100% of the time and a second additional word 50% of the time.

Since it is apparent that aside from overall, the category does not really influence the best choice for ε in this plot, Figure 6.2 collapses out category and shows the AUC across all experimental configurations. From this plot, it is easier to determine an optimal value of ε in this research. It is more apparent that AUC peaks at $\varepsilon = 0.3$ with a small window such

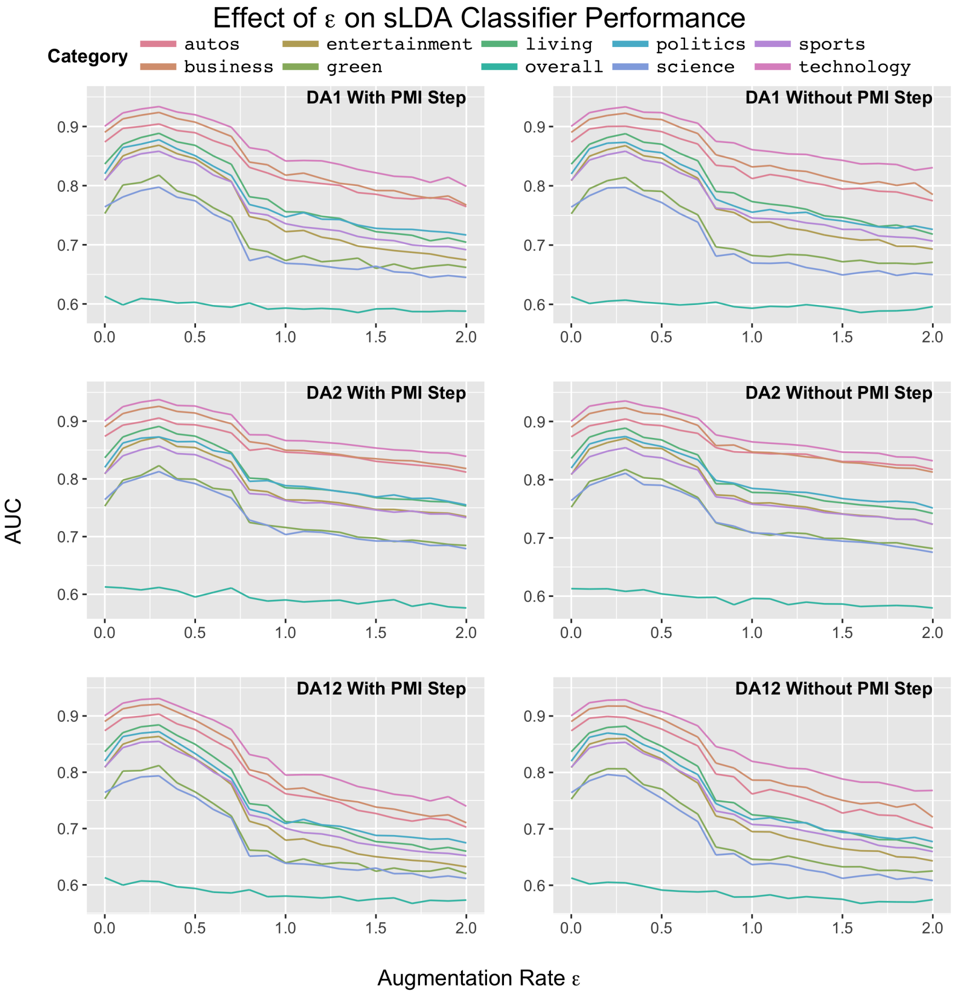


Figure 6.1: sLDA classifier performance by augmentation size ε, experimental configurations and category.

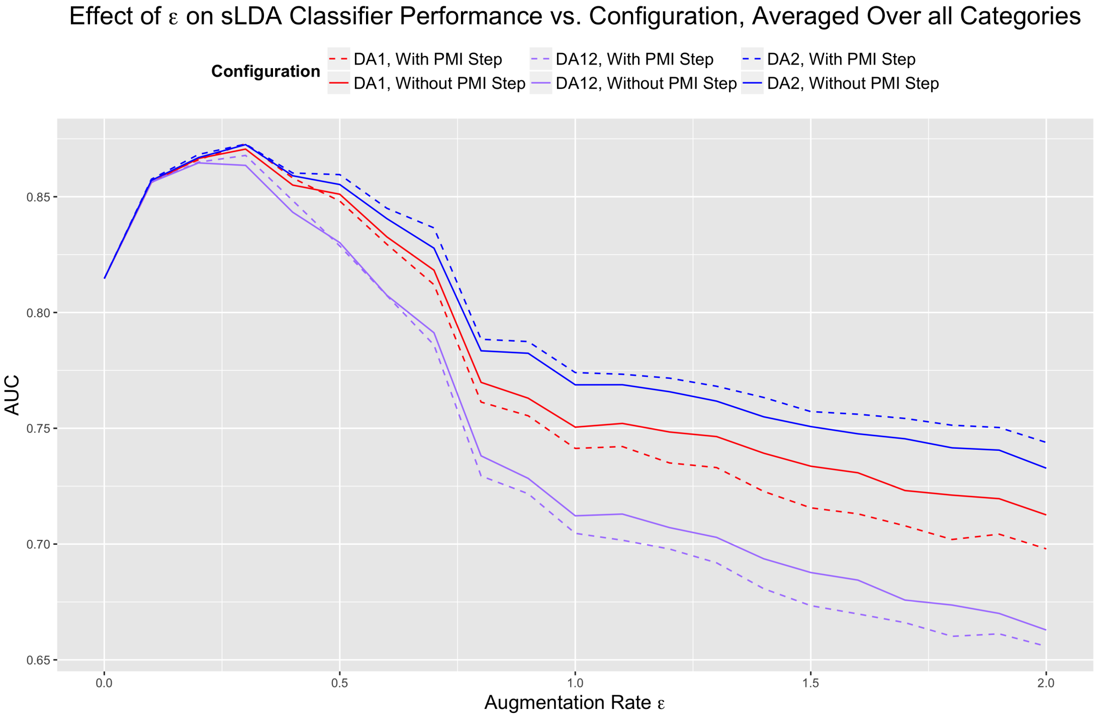


Figure 6.2: sLDA ROC AUC performance by experimental configuration averaged over category.

that values of ε in $0.2 \leq \varepsilon \leq 0.4$ performed optimally in this experiment. The results also suggest that classifier performance is good for $\varepsilon < 1$ if we take the researcher's AUC cutoff of 0.7 as a definition of "good" ([49] suggests 0.65 to 0.85 as a lower bound for a good AUC). When $\varepsilon \geq 1$, AUC performance drops below 0.7. This conclusion makes sense because once $\varepsilon = 1$, the number of terms added to the bag-of-words begins to exceed the number of terms in the original bag-of-words representation and if our sampling technique is not ideal, it is hypothesized that these new terms add more noise and/or bias. It appears that AUC performance was dependent on experimental configuration. DA2, where the augmentation method is only used on the test set seems to perform the best as ε increases, and performance decays more slowly than with the other variations; in fact, it never drops below 0.7. Surprisingly, DA12 decays the fastest and performs the worst as ε increases. This is surprising because in machine learning the same transformations are usually applied to the training and testing sets[65] and so it would seem that matched datasets would yield the best performance. The PMI word acceptance probability step does not seem to have a consistent effect. In DA1 and

DA12, the PMI acceptance probability step seems to improve AUC, whereas in the DA2 it is associated with worse performance, though the difference is likely not to be significant.

Finally, Figure 6.3 displays the diagnostic ROC curve for each classifier at optimal $\varepsilon = 0.3$ under each experimental configuration. The black dashed curve shows the performance of the classifier on the original data without augmentation (DA0). A model with higher predictive performance will have a curve that extends closer to the upper-left corner of the plot[64]. We see that under each configuration, the classifiers developed using the data augmentation method outperform the classifiers developed without. As usual, the only exception is the overall category which seems to not cooperate under any circumstances.

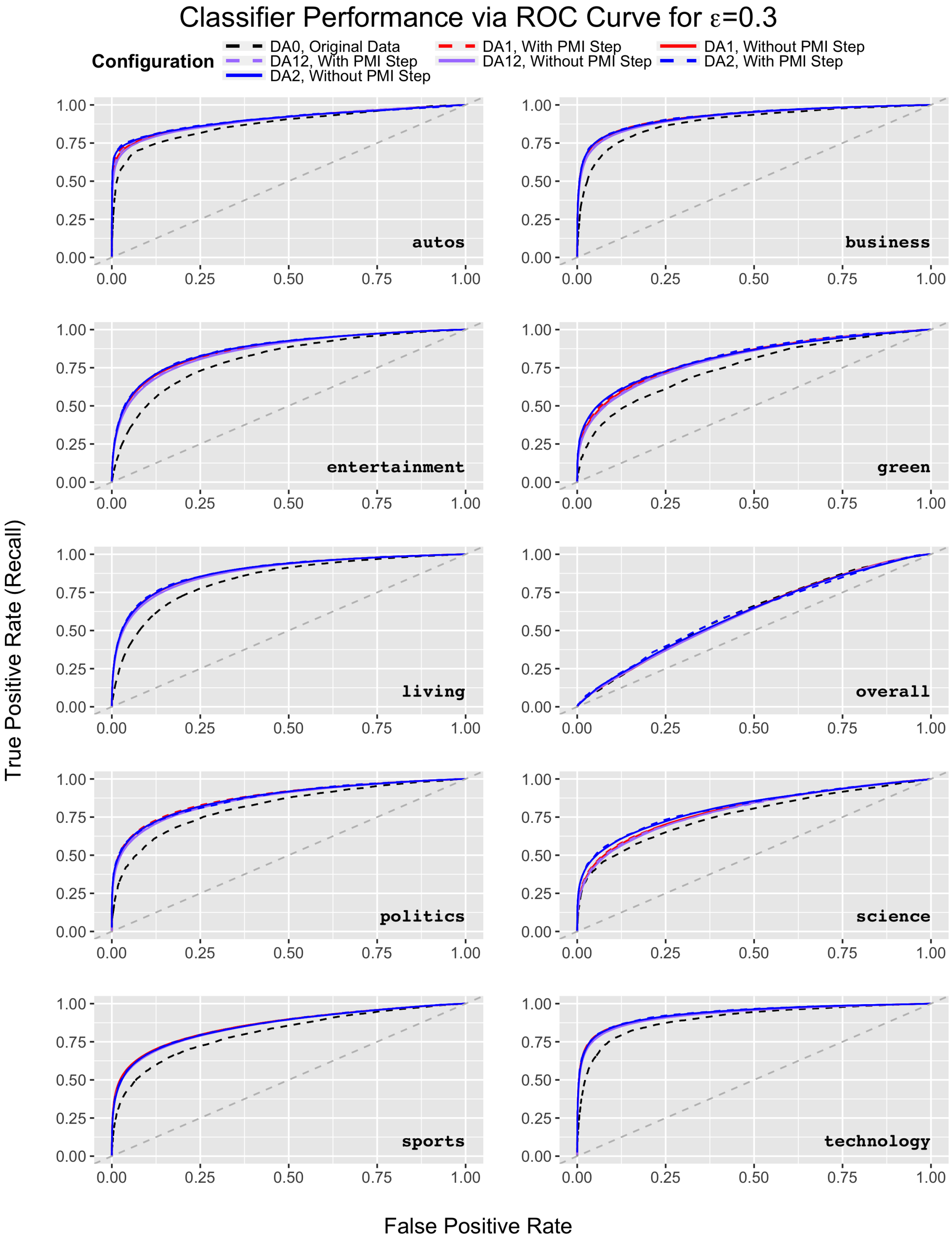


Figure 6.3: sLDA ROC performance by category and experimental configuration.

6.1.2 SVM

Figure 6.4 displays the effect of ε on the F1-score used to evaluate model performance for SVM. Similar to the sLDA models, performance is best at lower values of ε in the approximate range of $0.2 \leq \varepsilon \leq 0.4$ and then declining for higher values. In the raw data with no augmentation, F1-score ranges from 0.05 to 0.741 (the lower bound is 0.323 without `overall`). At $\varepsilon = 0.3$, F1-score ranges from 0.02 to 0.84 (the lower bound is 0.45 excluding `overall`) Interestingly, the improvement at lower values of ε is more stark than it was with with the sLDA models. In Figure 6.5 we average F1 performance across all categories since the majority of categories exhibited the same behavior. The `DA2` variation again seemed to perform the best over all values of ε. The two other variations `DA1` and the legacy `DA12` performed significantly worse as ε increased. The term acceptance step did not seem to have a significant effect on variations `DA1` and `DA2` but had a substantially positive effect on `DA12` as ε increased, a difference of about 8% in AUC at its most divergent point.

6.1.3 Experiment 1 Conclusions

In the previous chapter, it was hypothesized that a value of ε between 0.5 and 1.0 would yield the best performance because such a choice would yield a substantial number of resampled terms without overwhelming the bag-of-words representations with new, potentially poor, terms. The results from this experiment disproves this hypothesis and the optimal value of ε for both the sLDA and SVM models is approximately 0.3. It turns out that this optimal value is likely associated with sampling only a single term and that resampling one influential term is enough to improve classifier performance under data augmentation. Interestingly, this phenomenon of best performance occurring at lower augmentation rates was also noted in [143].

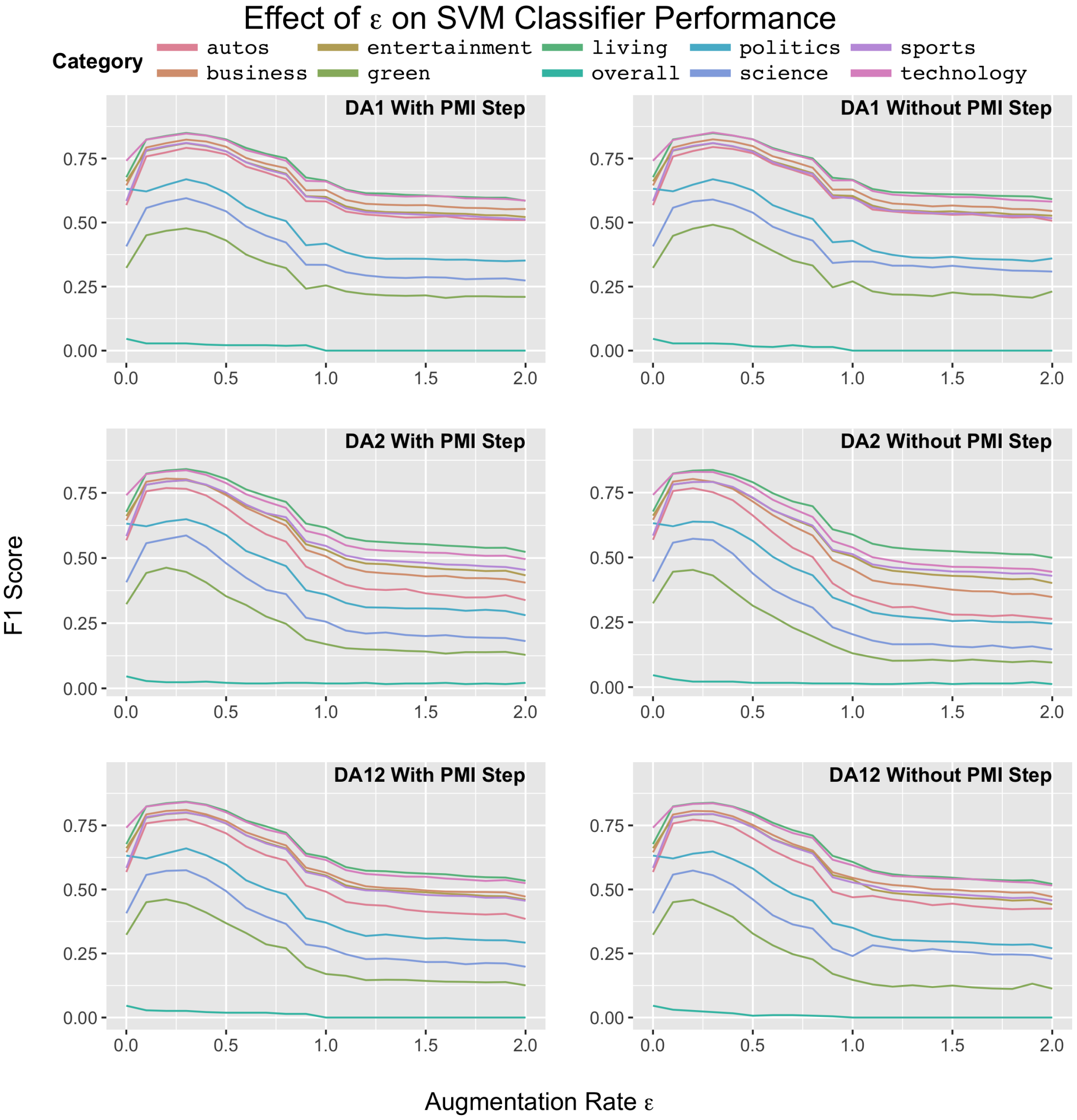


Figure 6.4: SVM classifier performance by experimental configuration, augmentation rate ε and category.

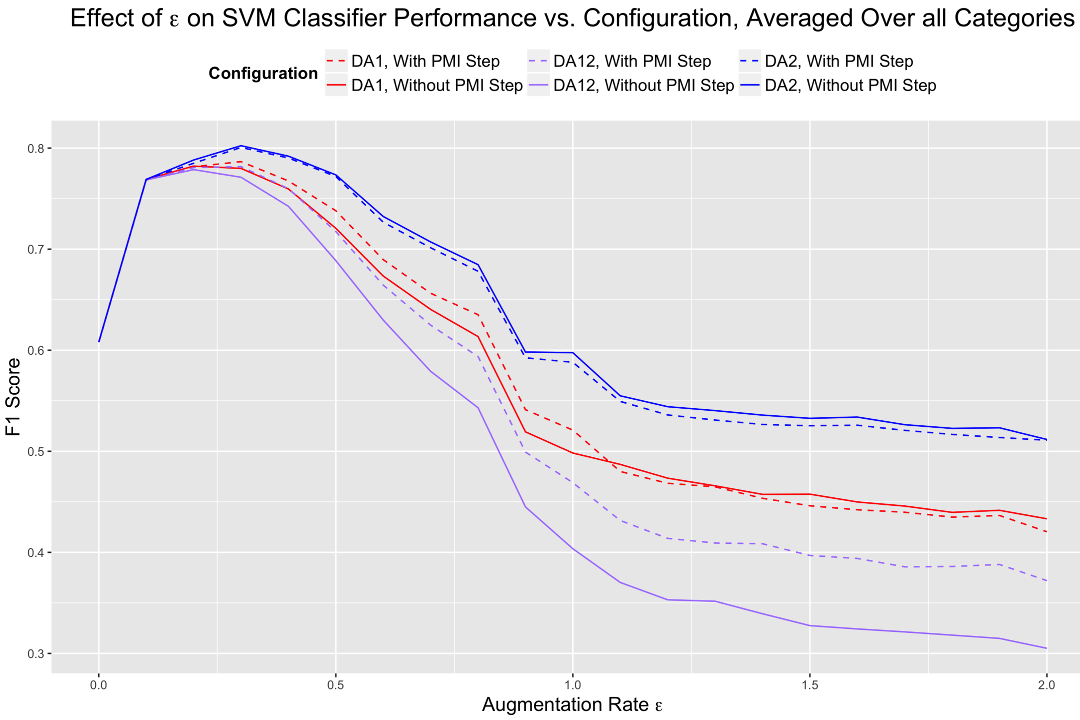


Figure 6.5: SVM F1-score performance by experimental configuration averaged over category.

6.2 Experiment 2 – Effect of When Terms are Resampled

Recall that the experimental variations `DA1`, `DA2` and `DA12` differ only in *when* the data augmentation algorithm is applied. `DA0` is the original data with no augmentation – the raw data. In most machine learning applications, the same transformations are applied to both the data used to train a classifier as well as the unseen data passed to the model for classification or for evaluation [65]. When both the training and testing data is augmented with the proposed method, it is referred to as variation `DA12`, matched training and testing sets, both augmented. In `DA1`, the proposed data augmentation algorithm is only applied on the data used to train the model (the training data) and the raw unseen data is classified by the model. `DA2` is the inverse, where a classifier is trained on the raw data without augmentation, but the unseen data is augmented by the proposed algorithm and then classified by the model trained on the augmented data. To test if there is a difference in performance for both the sLDA and SVM models based on when the data augmentation step is applied, the researcher compared performance for all three experimental variations for each category and model. Finally, category was “averaged out” and differences over ε were compared.

6.2.1 sLDA

Figure 6.6 shows the effect on ε and variation across each category. The main takeaway from this graph is that `DA12` with the matched training and test sets consistently yielded the worst performance under the proposed data augmentation method than the two variations using unmatched training and testing sets. The difference in performance is nearly static across all categories, though the decay in performance under `DA12` is most prominent in the `autos`, `business` and `technology` categories.

Figure 6.7 displays the ROC AUC performance of the sLDA classifier at optimal $\varepsilon = 0.3$ across experimental configurations. We can see that the AUC performance for each classifier improves by about 3% to 7% except for the usual `overall` category which does not see any improvement.

Figure 6.8 provides a summary of sLDA classifier performance at the optimal variation `DA2` and at the optimal augmentation rate $\varepsilon = 0.3$. We see that AUC ranges from 0.61 to 0.94 (the lower bound is 0.81 excluding `overall`) under this optimal configuration, whereas in the raw data in the white bars the AUC ranged from 0.61 to 0.9 (the lower bound was 0.75 excluding `overall`).

After averaging over all categories, we can see in Figure 6.9 that the matched training and testing sets of `DA12` yielded significantly worse performance over ε than the unmatched variations `DA1` and `DA2`. We can then take a closer look at classifier performance not only at the optimal augmentation rate ε but also the optimal experimental variation, `DA2`. Figure 6.9 is an abbreviated version of Figure 6.2, the latter displaying all augmentation strategies as well.

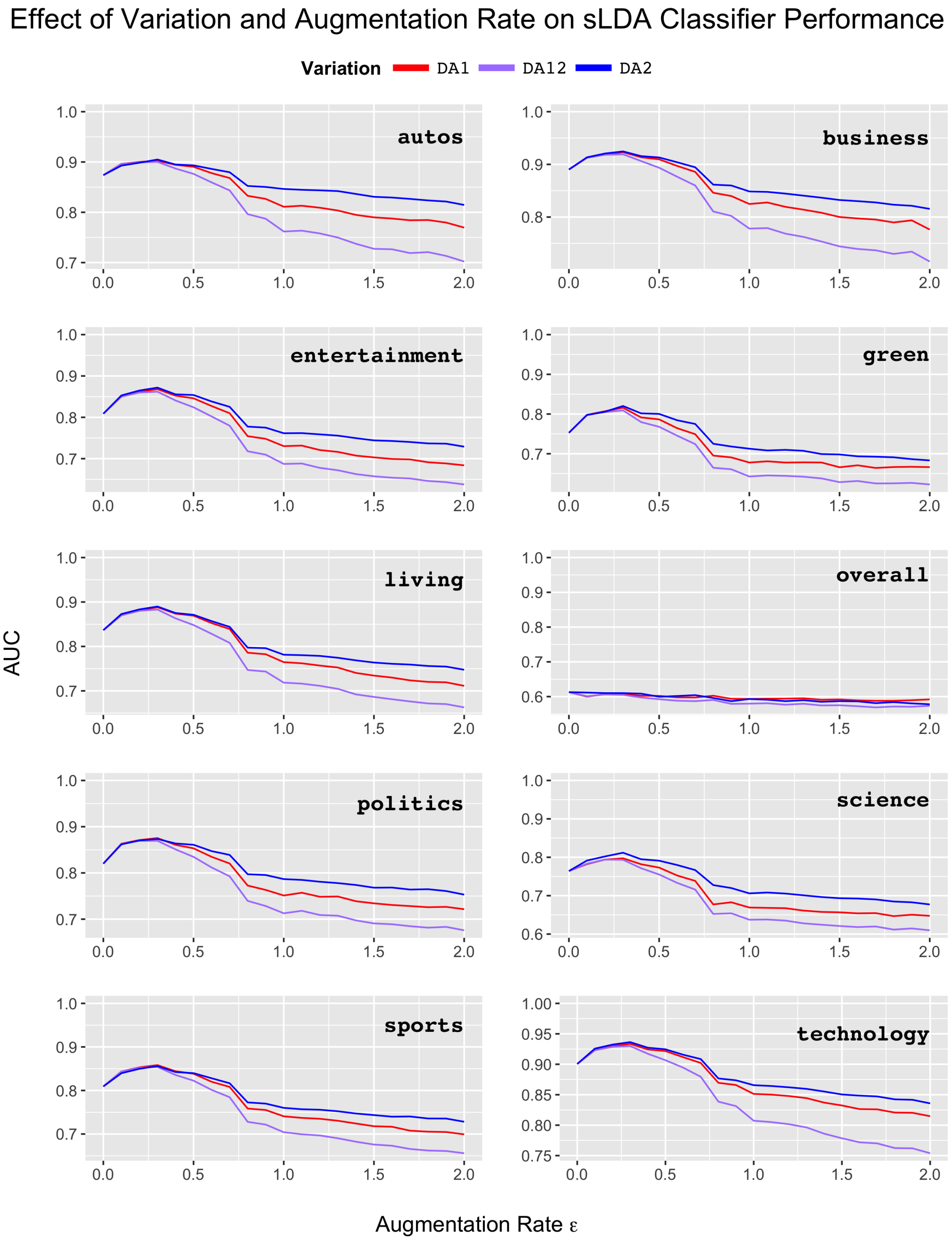


Figure 6.6: sLDA ROC AUC performance by experimental variation, ε and category.

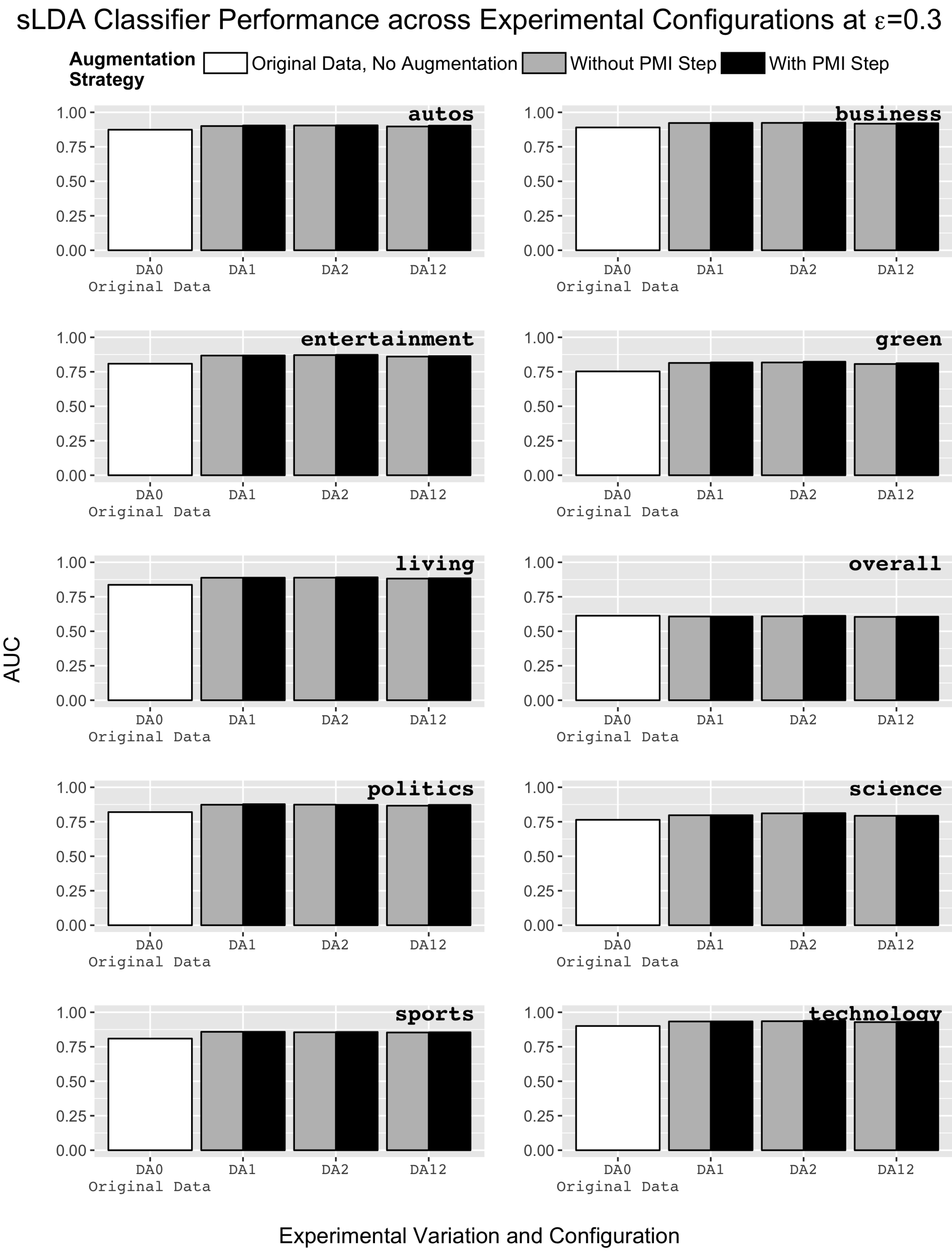


Figure 6.7: sLDA model ROC AUC performance by augmentation strategy, variation and category at $\varepsilon = 0.3$.

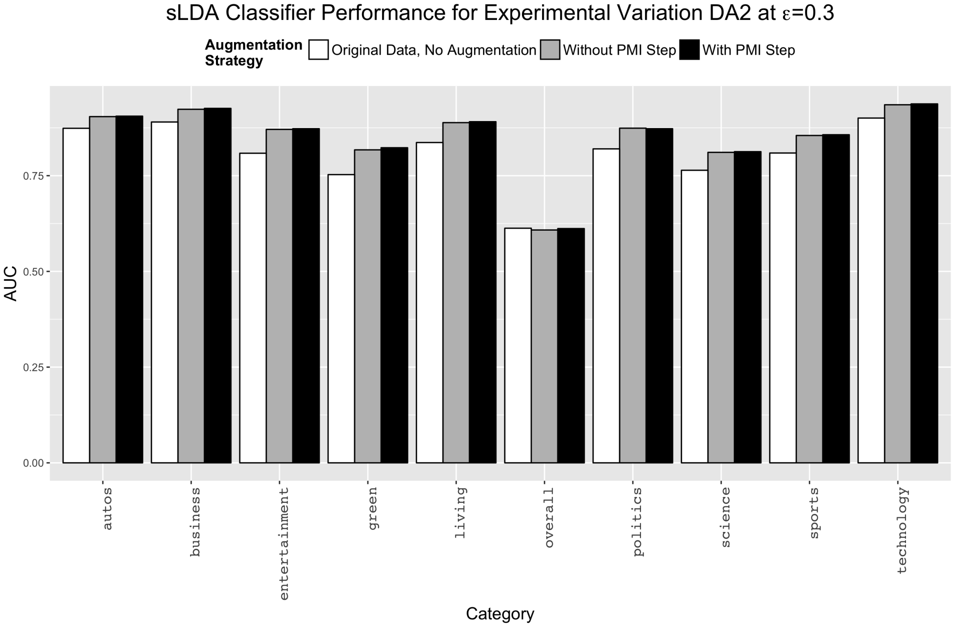


Figure 6.8: sLDA model ROC AUC performance at DA2 variation by category and augmentation strategy at $\varepsilon = 0.3$.

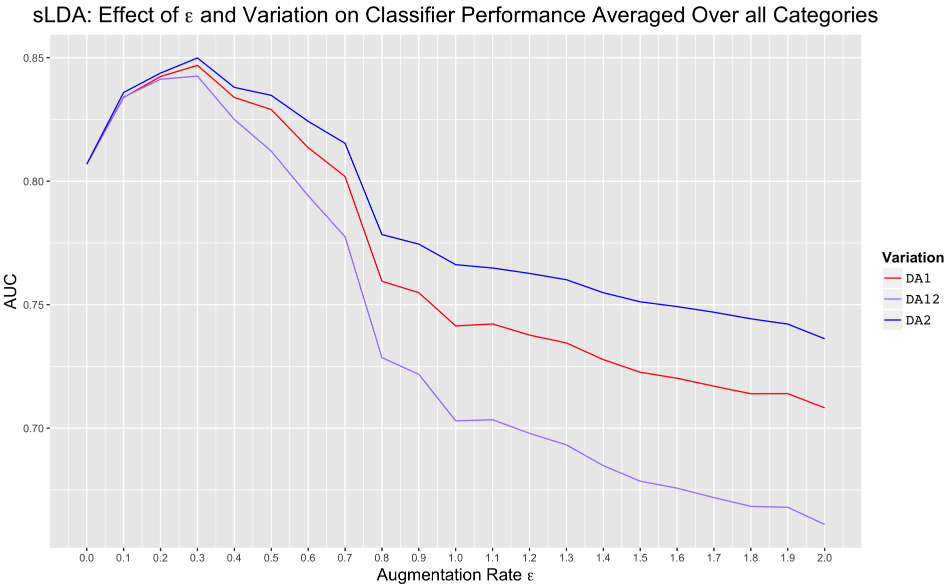


Figure 6.9: sLDA ROC AUC model performance by variation and ε averaged over category.

6.2.2 SVM

Figure 6.10 shows the effect of ε and variation on SVM F1-score performance across each category. We see that `DA2` performs the best across ε as usual, and the matched case variation `DA12` performing the poorest.

Figure 6.11 shows F1-performance for $\varepsilon = 0.3$ for each category across all experimental configurations. Performance improvement depends on category with gains of 14% for the `entertainment` to 22% for `sports`. There does not appear to be a significant difference among the experimental variations. Of course, the `overall` category classifier is the only one that performs worse than the `DA0` baseline, by about 2%. Improvement for the `politics` and `technology` categories was modest at 4% and 11% respectively.

Figure 6.12 shows SVM performance for each augmentation strategy at optimal ε and optimal variation `DA2`. We again see that in all categories except `overall`, the experimental variations outperform the `DA0` baseline. Improvement over baseline varied greatly over category with improvement as small as 1% for `politics` to 21% for `sports`. `overall` performed about 2% worse than the baseline. Finally, Figure 6.13 shows F1-score performance averaged over all augmentation strategies and categories. Again, we see that the matched training/testing set case performs the worst, about 5% worse than `DA1` and about 8% worse than `DA2`

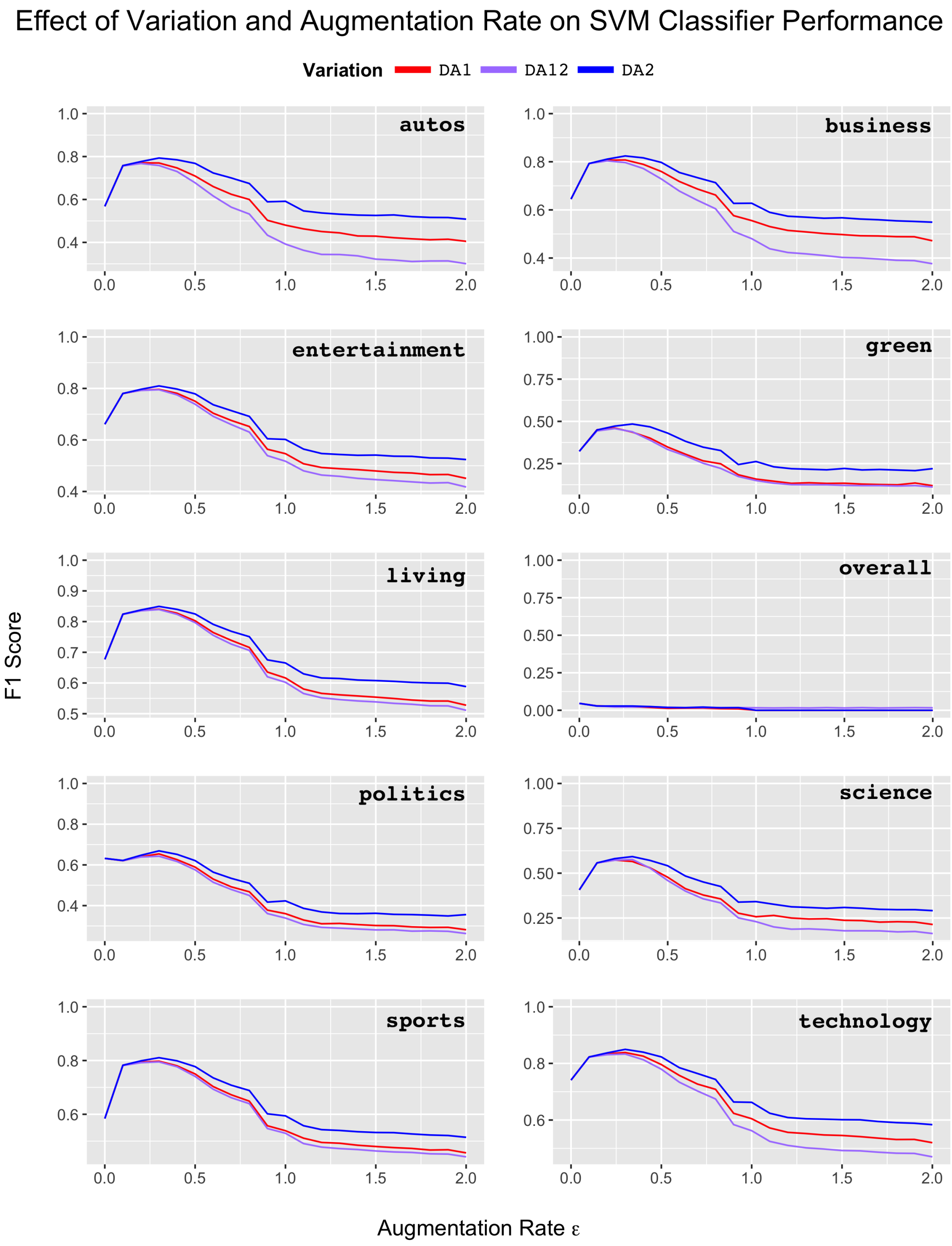


Figure 6.10: SVM F1 performance by experimental variation, ε and category.

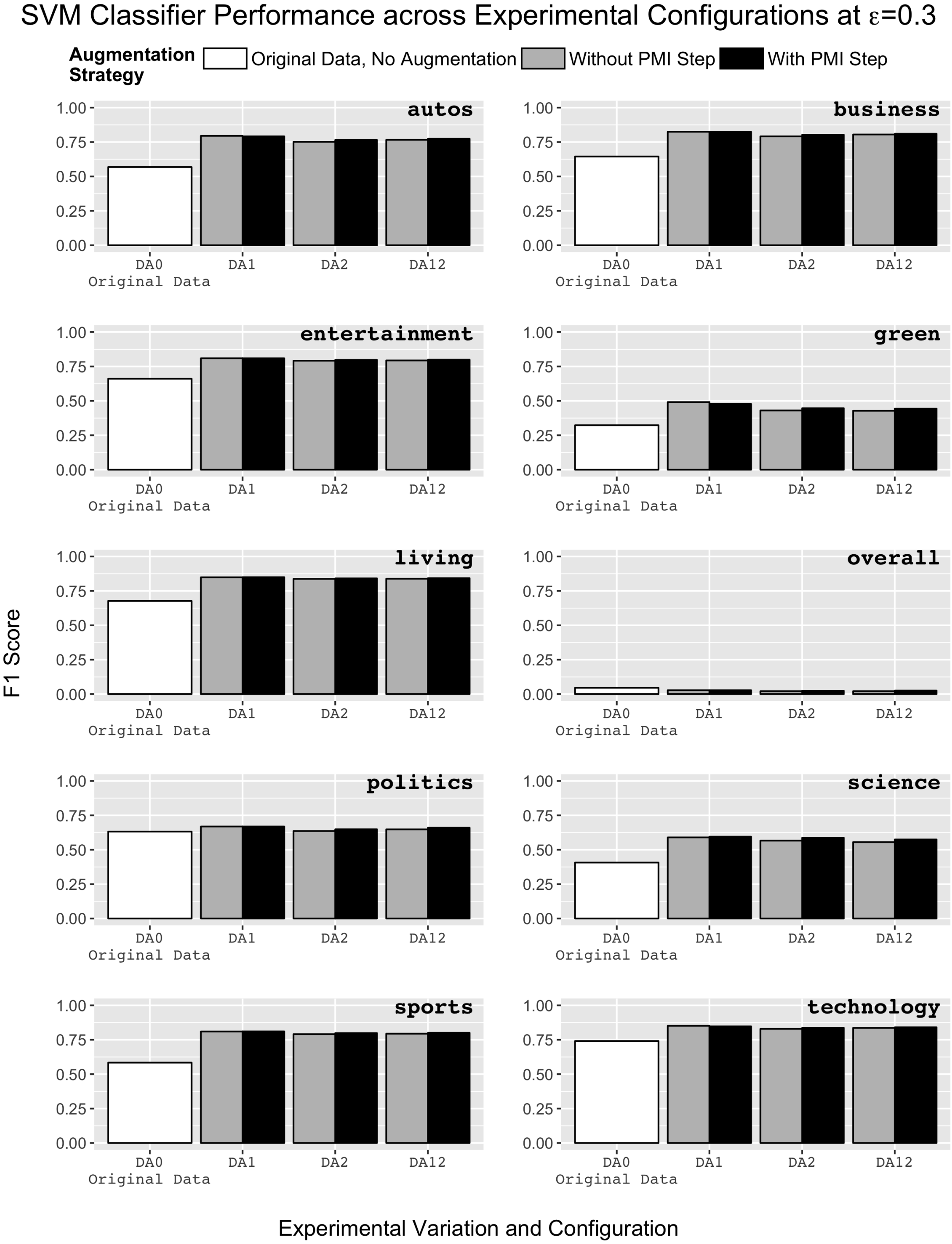


Figure 6.11: SVM model F1-score performance by augmentation strategy, variation and category at $\varepsilon = 0.3$.

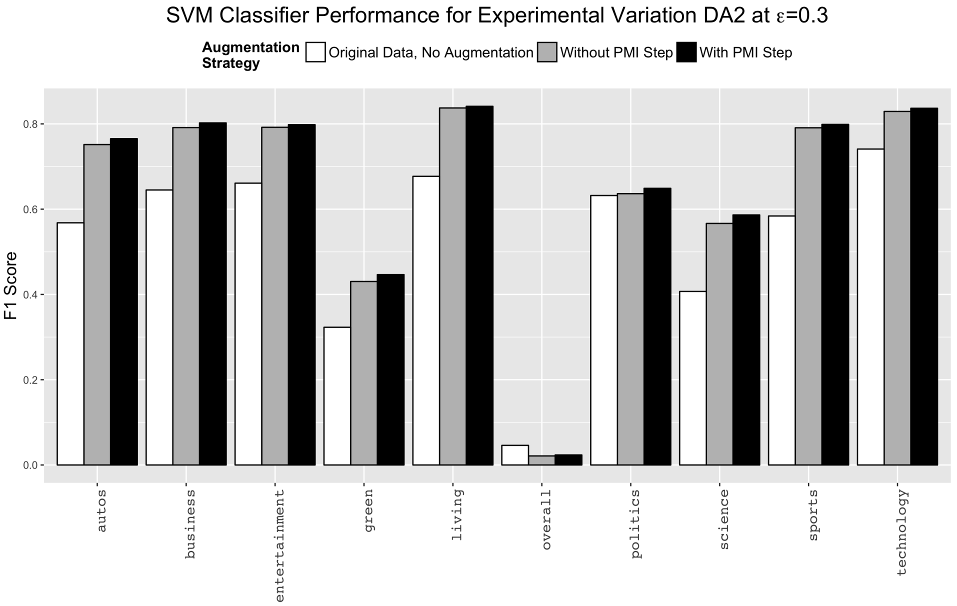


Figure 6.12: SVM model F1 performance for DA2 variation by category and augmentation strategy, averaged over all ε.

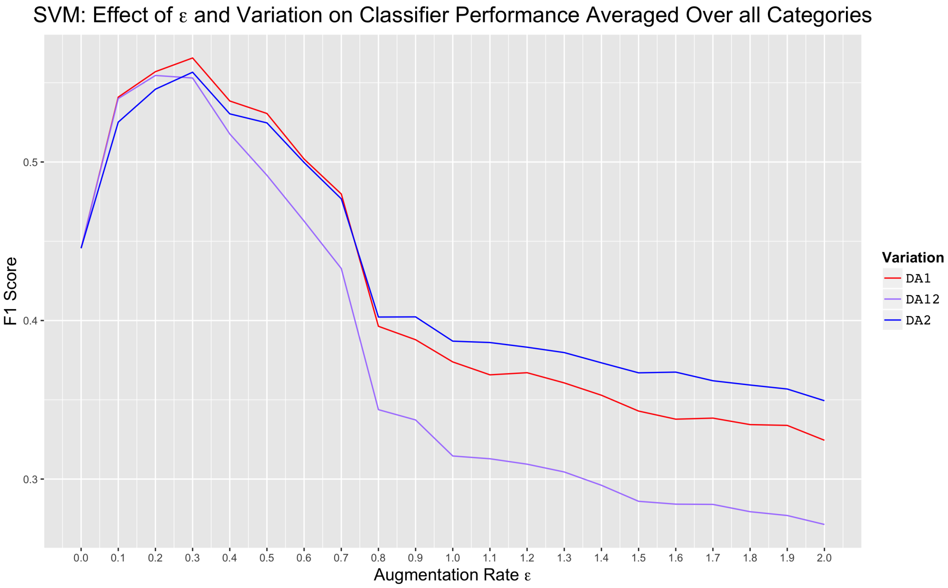


Figure 6.13: SVM model F1 performance by variation and ε averaged over all categories.

6.3 Experiment 3 – Effect of the Augmentation Strategy

One concern that the researcher had when developing this algorithm was the heavy reliance on conditional probabilities for selecting terms within each text. Recall that the bootstrap samples observations independently from a population. In the proposed data augmentation method, observations are sampled conditionally based on the other terms in the bag-of-words representation and on the semantic space S where S represents an approximate population. The concern is that the sampling dependency *within* each bag-of-words may be weighted too heavily when compared to the population. To counter this potential issue, the researcher proposed adding a final probabilistic term acceptance step (see Section 4.5.2) using pointwise mutual information (PMI). This final step accepts a term into the augmented bag-of-words according to how likely the term is to occur in the bag-of-words organically based on the semantic space S. Both models were evaluated across all variations both with and without this final step.

6.3.1 sLDA

Figure 6.14 shows AUC performance for the sLDA model across all values ε both with and without the term acceptance step. There is practically no difference for $\varepsilon < 1$ and for higher ε the difference is negligible, at most 2%. The bar chart in Figure 6.15 shows the average AUC performance for the optimal ε and variation `DA2` for all categories and confirms this finding. In this chart, it seems that both augmentation strategies improved performance slightly, by about 5% over the raw data with no augmentation. Thus, under the proposed data augmentation method, there was no significant difference in performance with and without the final probabilistic term acceptance step.

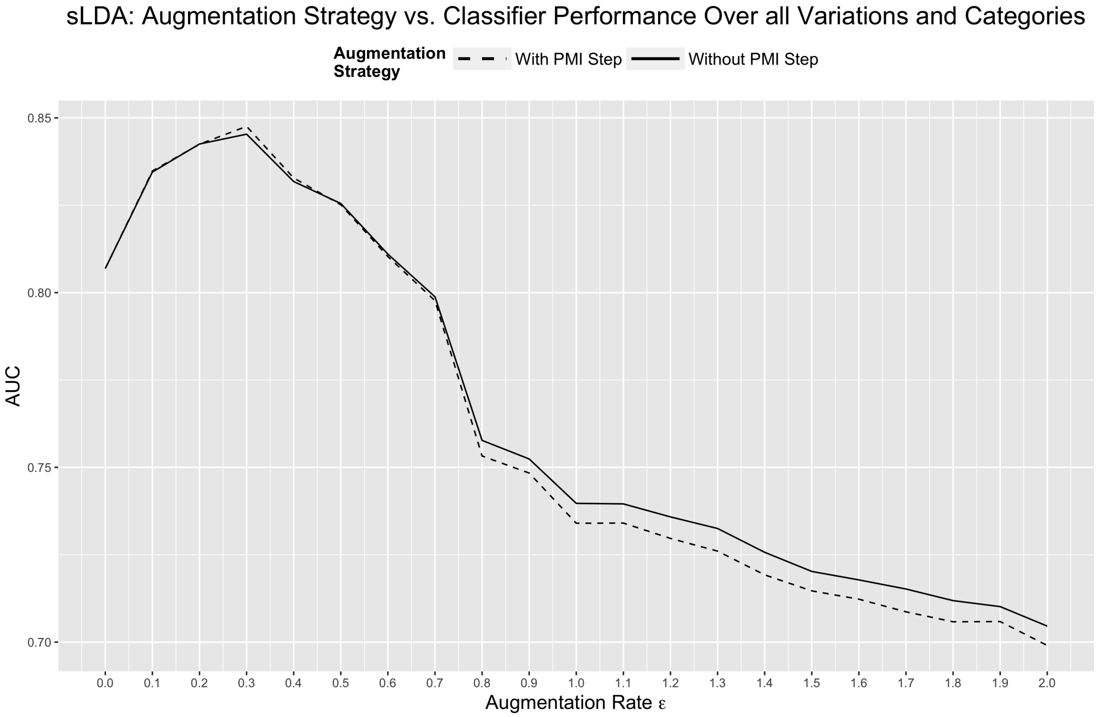


Figure 6.14: sLDA model ROC AUC performance by augmentation strategy averaged over category and ε

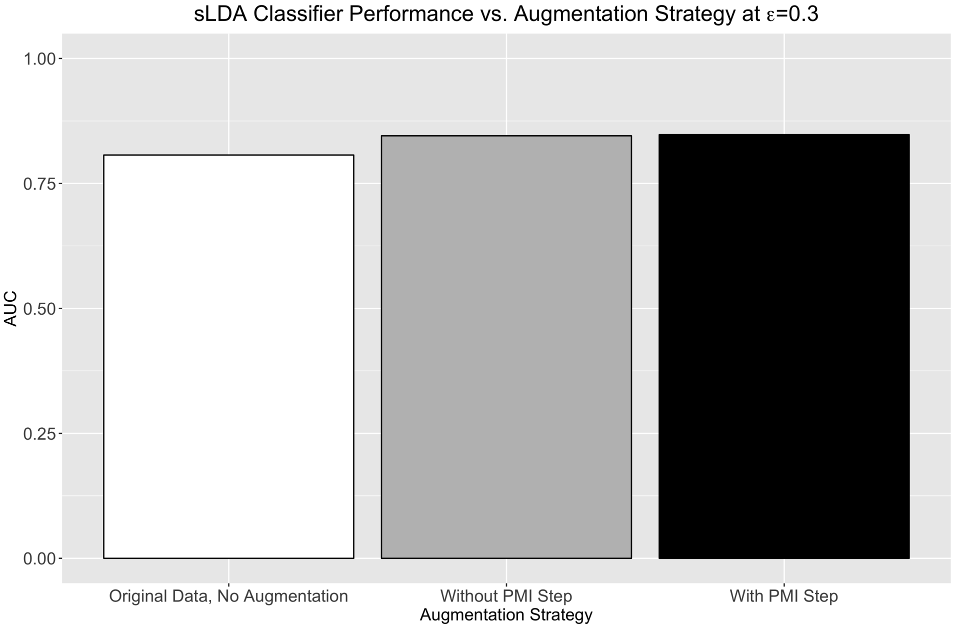


Figure 6.15: sLDA model ROC AUC performance by augmentation strategy averaged over category and variation.

6.3.2 SVM

Figures 6.16 and 6.17 imply a similar conclusion with respect to the effect of augmentation strategy and ε on classifier performance. The effect of augmentation strategy seems to be negligible across all ε though the probabilistic term acceptance step performed narrowly better by about 2%. When looking at only the augmentation strategies, it appears that SVM performed about 15% better over baseline while there was practically no difference between the experiments with and without the term acceptance step.

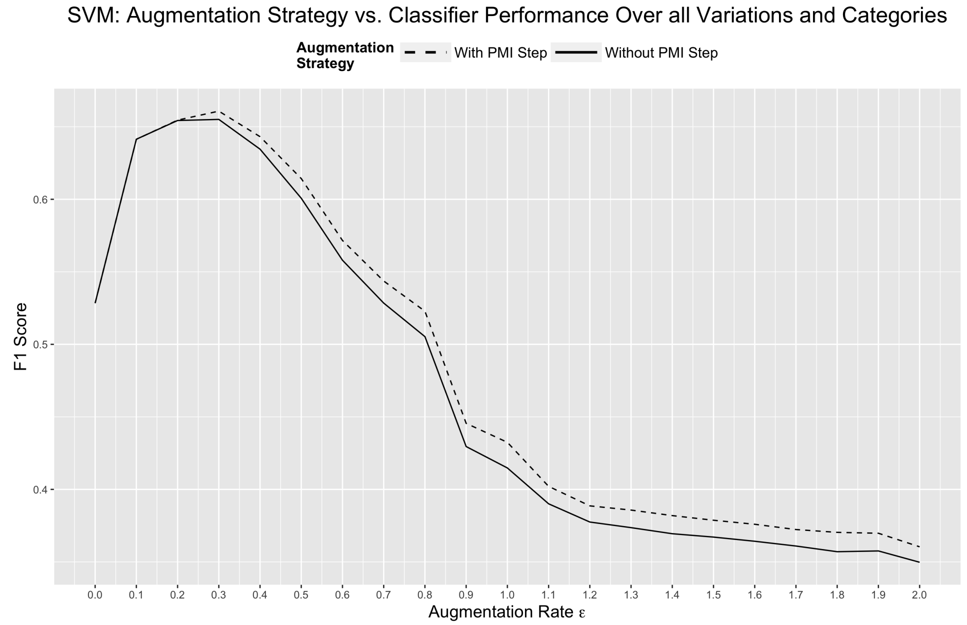


Figure 6.16: SVM model F1 performance by augmentation strategy averaged over category and ε.

6.3.3 Experiment 3 Conclusion

The purpose of adding the probabilistic term acceptance step was to correct for heavy dependencies in sampling within each bag-of-words, most notably due to the reliance of conditional probabilities in resampling terms. This experiment shows that this term acceptance step made no significant difference in classifier performance and thus we can conclude that it is not an important or necessary part of the proposed data augmentation algorithm at this time. While the implementation of the term acceptance step was in good faith, it is

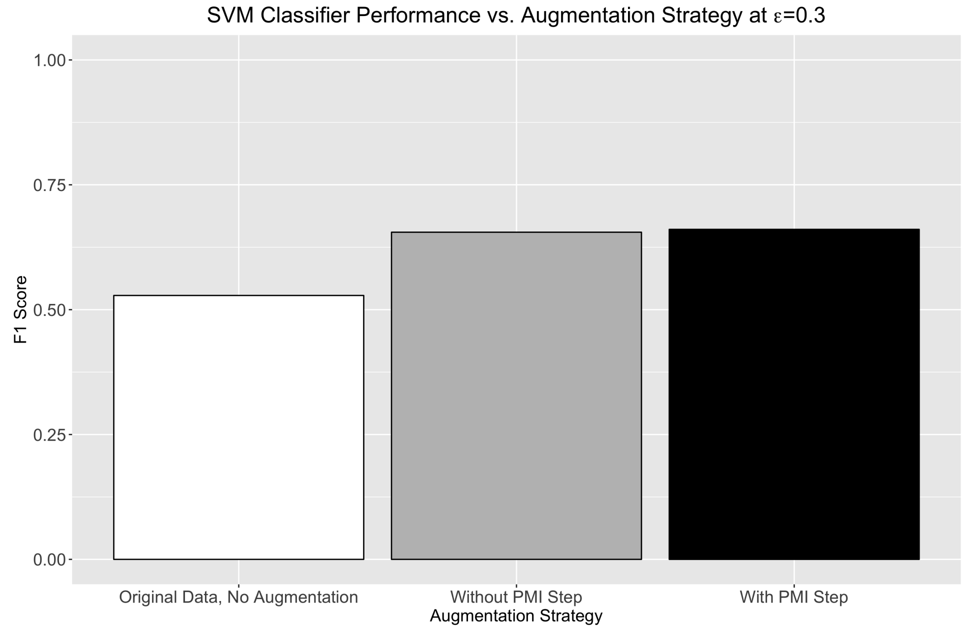


Figure 6.17: SVM model F1 performance by augmentation strategy averaged over category, ε and variation.

important to recognize that the pointwise mutual information (PMI) calculation relies on the same semantic space S as the original term resampling. Since both processes rely on the same underlying distribution, it could be reasonably assumed that they would both yield similar results.

6.4 Experiment 4 – Effect of Model Choice: SVM and sLDA

Next, we look at how the choice of model affects classifier performance. Two models were chosen for this research: Supervised Latent Dirichlet Allocation (sLDA) and a modification of the linear SVM implemented as `libshorttext` specifically for short texts. It is impossible to test the proposed data augmentation method on every text classification algorithm known to man, but topic models are timely and SVM is classic and both classes of models are used extensively in the literature. sLDA can use a variety of link functions, in this case the logit. This leaves Naïve Bayes and perhaps Random Forests as the only two "base" classifiers as a subject for future research.

Previously in this chapter we used AUC as a measure of performance for the sLDA model. In order to competently compare performance between sLDA and SVM, the F1-score is used instead for the analysis of this experiment since there is no concept of AUC in this base SVM model. First we look at the F1-score for both models across variation, augmentation strategy, augmentation rate ε and category. Figure 6.18 shows F1 performance for both classifier types across all ε for each category and is essentially a combination of Figures 6.6 and 6.10 that includes augmentation strategy and uses F1-score instead of AUC for the sLDA classifier. The major takeaway is that SVM consistently performed better than sLDA for lower, more optimally performing values of ε for all categories except `overall`. Somewhere in the range $0.7 \leq \varepsilon \leq 1.0$ the difference between the two classifiers becomes negligible for most categories as classifier performance for both also drops.

Figure 6.19 shows classifier performance for all variations for both the sLDA and SVM models. Similar to the results discussed previously in this chapter, we see that at small ε there is a negligible difference among the variations at low ε including optimal ε, but as ε increases, `DA2` performs the best, followed by `DA1` with `DA12` performing the worst. For the sLDA model, the unmatched training and testing set variations `DA1` and `DA2` perform significantly better than the matched case `DA12`. For the SVM on the other hand, the three variations perform significantly different from each other.

Since F1-score for both sLDA and SVM are on the same scale, we studied the effect of classifier type by computing the difference in F1-score by subtracting the sLDA F1-score from the SVM F1-score for the rest of this analysis. Figure 6.20 shows the difference in F1 performance between both classifiers for the three experimental variations and the two augmentation strategies.

SVM performed 17% to 22% better than sLDA across all experimental variations and augmentation strategies. For variation `DA12`, the improvement over sLDA is greater for the cases including the term acceptance step than without by about 5%. In the other two variations, SVM performed better than sLDA by about the same amount.

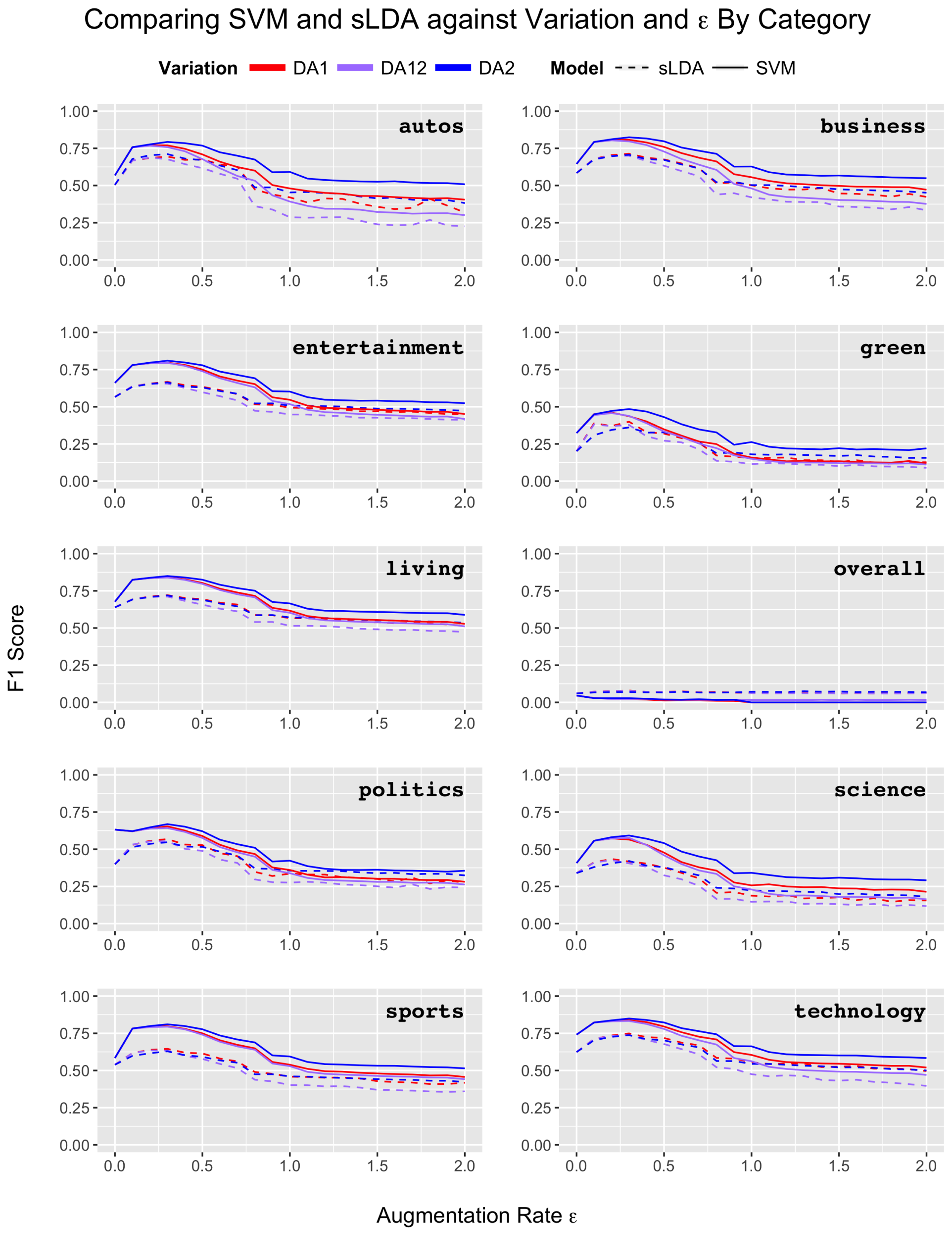


Figure 6.18: Comparison of sLDA and SVM performance via F1-score by category and variation, averaged over ε and augmentation strategy.

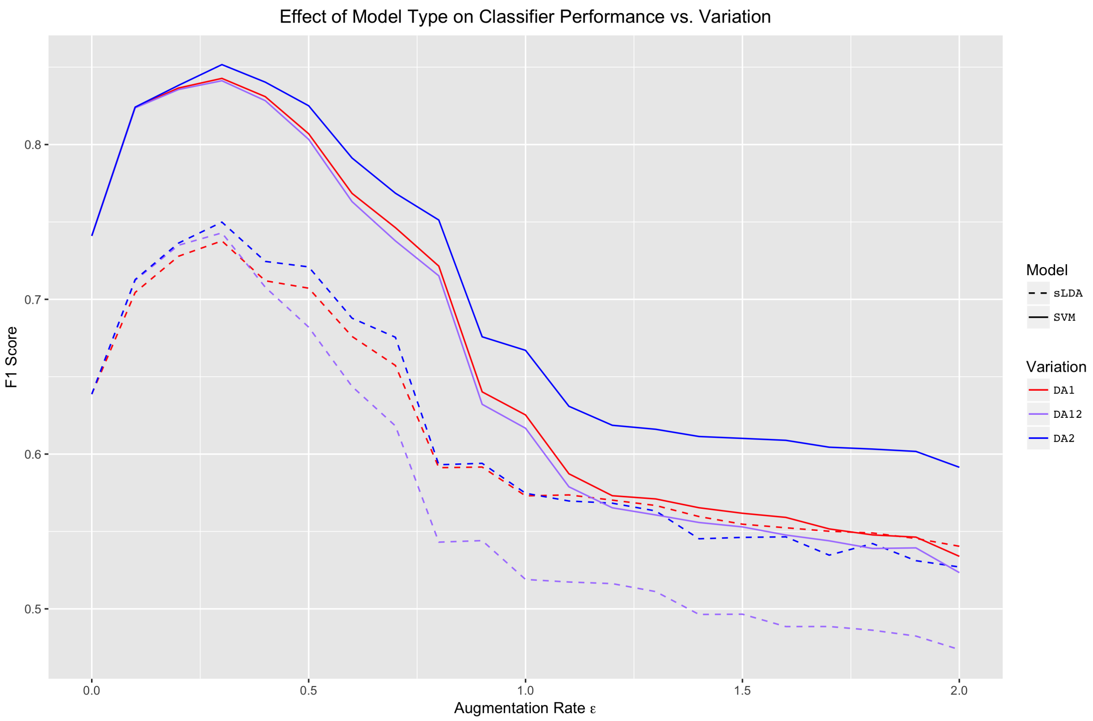


Figure 6.19: Comparison of sLDA and SVM performance via F1-score by variation and ε, averaged over category and augmentation strategy.

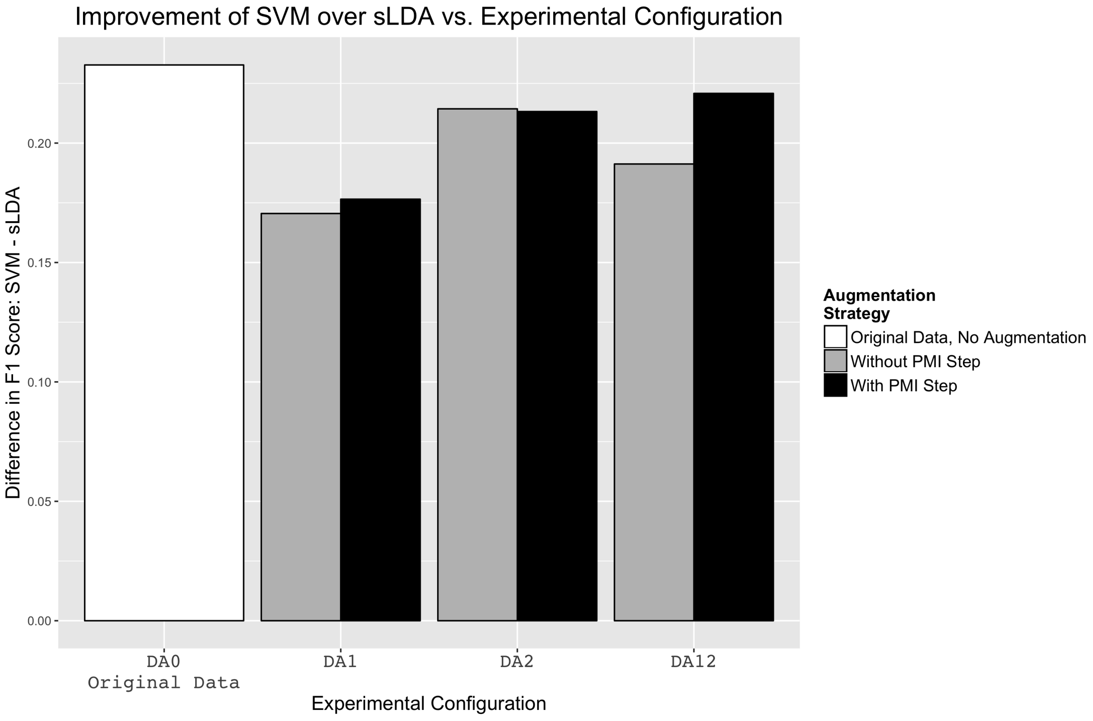


Figure 6.20: Difference in F1-score between sLDA and SVM by experimental configuration, averaged over category and ε.

Figure 6.21 quantifies how much better SVM performs than the sLDA classifier across all of the categories in the corpus. Since performance decreases significantly as ε increases and very few differences can be discerned at higher values of ε, we observe the performance only at the optimal $\varepsilon = 0.3$. In nine out of the ten categories, SVM performed better than sLDA across all variations and augmentation strategies. The elusive `overall` category seems to be better predicted by sLDA by about 4%. On the remaining categories, sLDA performed *at least* 8% better and as high as 18% better. SVM saw the largest improvement over sLDA on the `science`, `sports` and `entertainment` categories.

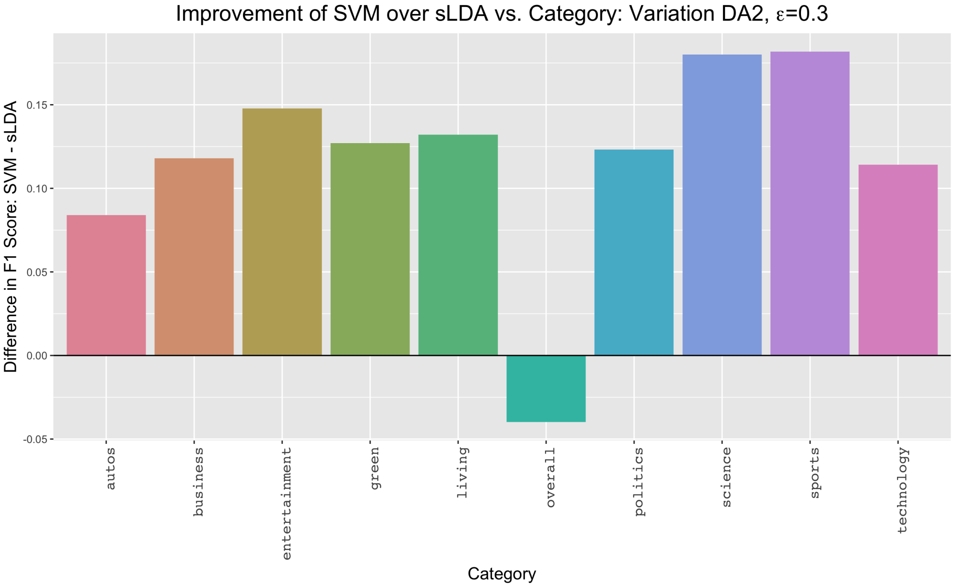


Figure 6.21: Difference in F1-score between sLDA and SVM by category, averaged over configuration and ε.

Finally, Figure 6.22 shows the improvement of sLDA over SVM for each category over all variations and augmentation strategies and is provided for completeness. We see that for small ε, the difference between the two classifiers is fairly small and in many cases increases significantly as ε increases. This suggests that SVM may be a more “stable” model with respect to ε as performance does not drop off as sharply as it does with sLDA. It does appear that sLDA performed better than SVM on the `DA1` variation at higher ε, but this is not of much significance since these values of ε are much higher than the optimal $\varepsilon = 0.3$.

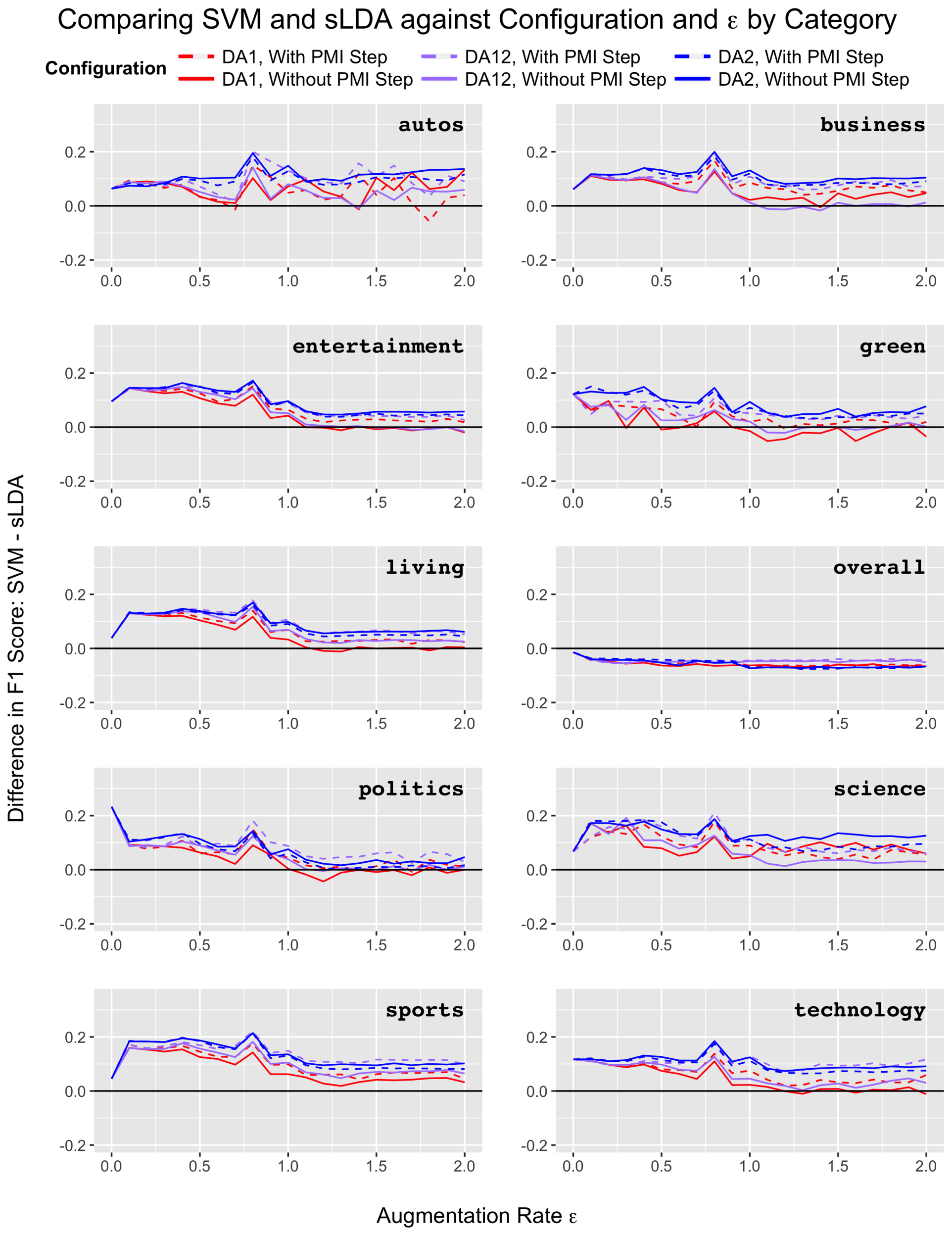


Figure 6.22: Difference in F1-score between sLDA and SVM by category and configuration, averaged over ε.

6.4.1 Experiment 4 Conclusion

The purpose of this experiment was to determine how well the proposed data augmentation method generalizes to different models types and study the differences between them under the proposed framwork. Supervised Latent Dirichlet Allocation (sLDA) with logistic loss and `libshorttext` implementation of SVM were the models studied. As one would expect, results in the literature vary regarding which model provides "better" results, with [172] finding that SVM performed better than sLDA and [7] finding that sLDA performed better than SVM. In this research, `libshorttext` SVM, a variant of SVM designed specifically for short texts, performed better than sLDA. The only case in which sLDA performed better than SVM was in variation `DA1` without the term acceptance step for large or suboptimal ε, and also when predicting the peculiar `overall` category.

6.5 Experiment 5 – Comparison of Data Augmentation with SVM using LSA Vectors

Finally, we compare the results of the best classifier under the proposed data augmentation method to the SVM that uses LSA vectors as features. We denote this special SVM model as `SVM+LSA` in this section. The optimal model under data augmentation was the `DA2` test-set-only augmentation with $\varepsilon = 0.3$ and no final term acceptance step applied to SVM, and in this section we denote this model $\mathtt{DA}^*_{\mathtt{SVM}}$. The original data with no augmentation applied to SVM is denoted $\mathtt{DA0}_{\mathtt{SVM}}$.

Table 6.1 displays the F1-score for each category classifier across all three models as well as the mean and median F1-score across all categories. The $\mathtt{SVM} + \mathtt{LSA}$ baseline produced F1-scores between 0.063 and 0.783, but without the outliers resulting from `green`, `overall` and `science`, the range was 0.576 to 0.783. The optimally augmented SVM model $\mathtt{DA}^*_{\mathtt{SVM}}$ yielded F1-scores between 0.02 and 0.84, or between 0.636 and 0.84 once the three low performers were removed. In two categories, `overall` and `politics`, $\mathtt{DA}^*_{\mathtt{SVM}}$ performed worse than the `SVM+LDA` baseline by about 0.04 and 0.02 respectively. This may suggest that the problematic overall category and the politics should be fit with some sort of topic model by itself rather than using bag-of-words with augmentation. It should be noted that the politics category also produced one of the lowest gains from data augmentation. The improvement yielded by $\mathtt{DA}^*_{\mathtt{SVM}}$ over SVM+LSA had a large range between 0.1 and 0.11 with a gain of 0.32 being a high outlier. The science category, a particularly low performer across the board, experienced a 0.32 improvement in F1-score on the experimental model compared to the SVM+LSA baseline and was this high outlier. The categories business and green only saw negligible improvements in performance over SVM+LSA at 0.005 and 0.008 respectively, but both categories performed better than the $\mathtt{DAO}_{\mathtt{SVM}}$ baseline. All other categories except those previously mentioned saw a boost in F1-score performance between 0.1 and 0.11. These findings are illustrated in Figure 6.23 and Figure 6.24.

Category	$\mathtt{DAO}_{\mathtt{SVM}}$	SVM+LSA	$\mathtt{DA}^*_{\mathtt{SVM}}$
autos	0.5680	0.6355	0.7515
business	0.6450	0.7829	0.7914
entertainment	0.6610	0.7029	0.7920
green	0.3230	0.4251	0.4304
living	0.6770	0.7621	0.8374
overall	0.0460	0.0629	0.0212
politics	0.6320	0.6533	0.6363
science	0.4070	0.2449	0.5665
sports	0.5840	0.7085	0.7909
technology	0.7410	0.7792	0.8292
Mean	0.5284	0.5757	0.6447
Median	0.5840	0.6533	0.7515

Table 6.1: Per-class comparison of F1-scores between the baseline with no augmentation ($\mathtt{DAO}_{\mathtt{SVM}}$), a baseline using topics rather than bag-of-words (SVM+LSA), and the optimal data augmentation configuration applied to SVM $\mathtt{DA}^*_{\mathtt{SVM}}$.

6.6 Experiment 5 Conclusion

The purpose of this experiment was to determine if the optimal augmented model performed better than a baseline model that used a lower dimensional representation of text rather than bag-of-words. We fit a model using LSA vectors as features to an SVM and denoted this model SVM+LSA. When we compared the models using F1-score, we saw that the optimal

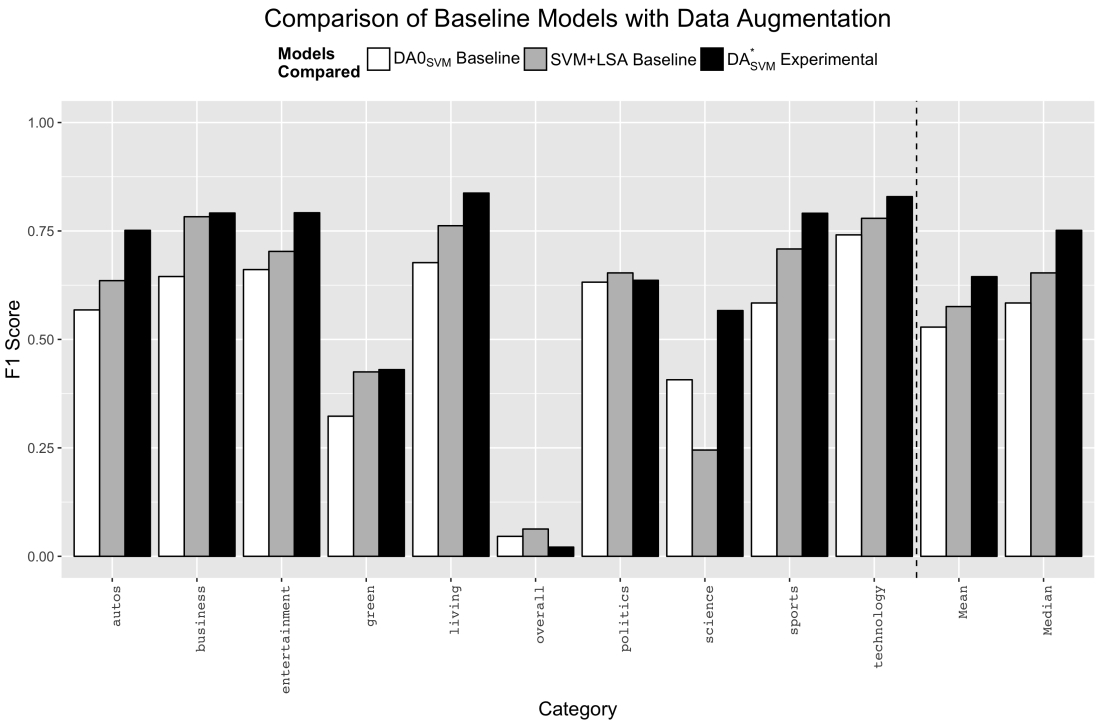


Figure 6.23: Per-class comparison of F1-scores between `DA0`$_{\texttt{SVM}}$, `SVM+LSA`, and `DA`$^*_{\texttt{SVM}}$.

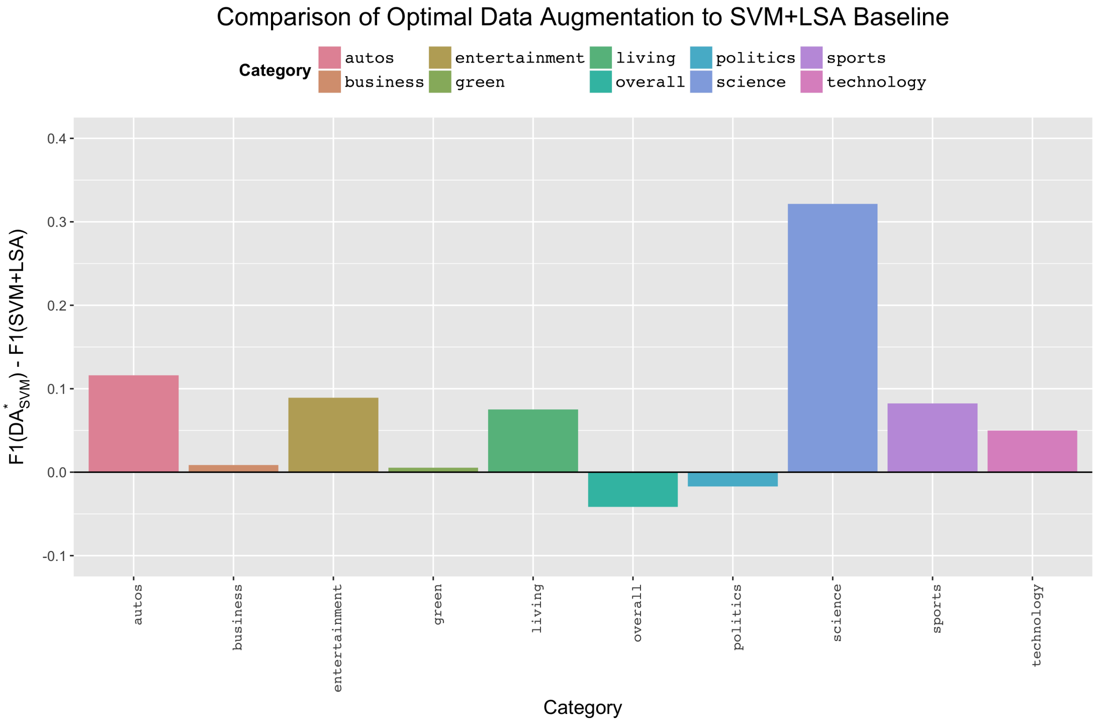


Figure 6.24: Per-class difference in F1-score between a `SVM+LSA` baseline, and the optimal data augmentation configuration applied to SVM `DA`*.

model, the SVM model evaluated on augmented testing data with $\varepsilon = 0.3$ performed better than `SVM+LSA` in every category except `overall` and `politics`.

6.7 Chapter Summary

In this chapter, we answered five questions about the performance of the proposed data augmentation algorithm:

1. How many terms should be used to augment a bag-of-words?
2. When should terms be selected for addition? In both the training and testing data, only in the training data, or only in the testing or unseen data?
3. Do we need to use a probabilistic term acceptance step to correct strong within-text dependencies?
4. Which classifier works better under this framework?
5. How does the best performing model under data augmentation perform better than an SVM model using LSA vectors as features?

We concluded that $\varepsilon = 0.3 \pm 0.1$ was an optimal choice for ε and that this coincides with sampling a single highly influential term. This is a significant development because that single term is selected probabilistically according to influence across the entire corpus using the semantic space S and shows that the resampling strategy served its purpose. Once $\varepsilon \geq 0.5$, classifier performance decreased substantially and was often worse than the classifiers trained on the original non-augmented data (`DA0`). The point at which term sampling occurred also had a significant effect on classifier performance. While variation `DA2` enjoyed being the optimal variation in most cases, `DA12` consistently performed the worst particularly as ε increased. This is a very interesting finding because `DA12` represents the *status quo* of machine learning where both the training and testing sets undergo the same transformations and both datasets follow the same distribution. It was the researcher's hypothesis that using the data augmentation method on only one of the sets may have introduced some additional signal either into the classifier (under `DA1`), or into the unseen testing data (under `DA2`). In [65], the researchers actually found that mismatched training and testing sets can *outperform* the performance of matched training and testing sets. The proposed data augmentation method appears to exhibit this behavior as `DA1` and `DA2` always or nearly always outperformed the matched case `DA12`. The researcher also hypothesized that a probabilistic term acceptance step may serve as a final sanity check when augmenting a bag-of-words to account for the high level of dependency within each text and the resampling step by only including words if they were a good “fit” for the text according to S. We saw that this augmentation strategy had no significant effect on performance and that it sometimes decreased performance though further research is required. We also looked at how sLDA and SVM performance compared to each other. We observed that in most cases SVM performed better than sLDA. Finally, we considered whether or not the optimal data augmentation configuration performed better than an SVM trained on lower-dimensional features generated from LSA, and we found that the data augmentation method performed better for all categories except two categories that underperformed throughout this work, `overall` and `politics`.

CHAPTER 7

Concluding Remarks

In this manuscript, the researcher addressed the challenge of classifying short texts into classes, an idea that is important, yet is not frequently studied at a deep level as engineers typically find workarounds or have their own way of augmenting text. This research proposed a preprocessing step based on the bootstrap, a concept used frequently in statistics. The researcher describes modifications to the original implementation of the bootstrap that allow it to be applicable to text. We saw that this new method had success in improving classifier performance on short texts, particularly when the original text is augmented to be about 30% larger and the preprocessing only occurs on unseen data and categorized by a classifier that is trained on the original short texts. The researcher tested the new approach on two models for text classification and found that while classification improved for both, the size of the effect depended heavily on the type of classifier. The researcher considers this work to be a "moonshot" – it addresses a large problem in natural language processing using a method that has not been attempted in quite this way. For this reason, there is a lot of room for further research and improvement.

7.1 Unexpected Results and Failed Experiments

The experiments in this manuscript all had a substantial contribution to the work at hand; however, some of them yielded unexpected results that either negated the researcher's hypothesis or ended up pointing out something interesting that was not known a priori. The researcher hypothesized that either doubling the length of the text, or increasing its length by about 50% would yield the best results. It turns out that the best augmentation size was

30%, but that this figure corresponded to sampling one highly relevant term from a semantic space. This is still a significant finding because it suggests that using a semantic space to augment short texts actually works. But this also calls into the question whether or not the rest of the proposed data augmentation framework is in fact required, or if we can simply stop once an arbitrary number of terms has been added to each text. Another unexpected result was that augmenting only the unseen/test data improved performance considerably. This finding suggests that practitioners can train classifiers on short texts, and then simply augment unseen observations (such as in an online system) to generate more signal for better performance. The result was unexpected because datasets are typically matched; that is, the training set and the testing or unseen datasets are processed in the same way and this fact is drilled over and over in the machine learning community though research suggests that the contrary can be true. One particularly surprising finding to the researcher was that sLDA performed rather poorly compared with SVM. While comparing these two classifiers was not the purpose of this research, the researcher was quite surprised that using groups of words in topics did not improve base performance. Instead, using plain old word counts with plain old SVM performed much better. A pleasant surprise was that data augmentation consistently performed significantly better than another oldie but goodie: an SVM model trained with LSA vectors as features rather than words.

The researcher also performed some other experiments that are not discussed in this manuscript because they did not contribute to this research in any meaningful way. Improving Latent Dirichlet Allocation (LDA) was the original goal of this research, but experimentation determined that there was no way to force LDA into a coherent and consistent supervised context. Instead, Supervised Latent Dirichlet Allocation (sLDA) was used in place of LDA.

Since the bootstrap is a simulation method, several iterations of augmentation were performed on each text to construct a distribution of performance metrics to summarize performance since every run would be different. With the number of iterations being large, the researcher noticed that bagging or boosting may yield better results. The researcher determined that adding this step would be counterproductive because the purpose of this research was to improve classification of short texts and not develop an ensemble method that would require significant runtime and significant effort for the end user. The bootstrap was already heavily modified for this research, and bagging would require further modifications to work with text. It was unclear if these modifications would make sense and could in fact be realized. The other option was boosting where we use incorrectly labeled observations to train further classifiers, but the same concern over runtime and practicality made this impractical for the time being.

7.2 Future Research

Due to the breadth of this research topic, there are a lot of areas where more can be done to improve classification, or to expand the research to cover additional topic models, classifiers and data types.

7.2.1 Nuances of the Proposed Data Augmentation Method

There were several areas where the researcher had to make an "engineering decision"[26] about minute details in the proposed algorithm. This manuscript proposes a new algorithm at a macro level and evaluates it. Further research is required for micro level details. The choice of the distance metrics used to build and query semantic space S is one such micro level detail to investigate. Another detail that requires further research is the resampling scheme. In this research, we used a systematic method of first identifying "target" terms that were used to assist in drawing a new candidate term to add to the text. Could there be a simpler more effective way of accomplishing the same goal?

It would also be useful to investigate whether or not the optimal ε discovered is global and unique to this implementation of data augmentation, just unique to this particular formulation of it, or if ε is actually a function dependent on other variables that were not

[26] A phrase commonly used by Lixia Zhang, Professor of Computer Science, UCLA, to refer to a careful and educated decision made to further a goal and not expected to have a measurable effect in a particular context.

studied. While the experimental variations studied in this research cover the full case of training and testing set transformations, the final term acceptance step could potentially yield performance gains using a metric other than pointwise mutual information (PMI) or by experimenting with the other parameters used in the probability computation.

One disappointing finding to the researcher was how poor sLDA performed at baseline. Using cross-validation, values were chosen for the number of topics K and the topic distribution hyperparameter α. The values tested were based on what is common in the literature and we saw that these parameters affected the baseline classifier significantly. The researcher assumed one optimal K and one optimal α and it may be useful to perform this analysis again with a more thorough experiment to choose these parameters and to vary these parameters for each category rather than choosing them globally.

Finally, to accommodate for the high level of variation in the term sampling, we essentially created 100 copies of each short text and augmented each of them with sampled terms. It is unclear if multiple iterations are actually necessary, and if so, how many of them to perform. For this research, the author decided to err on the side of controlling for variation though it would be interesting to investigate the use of bagging or boosting to combine all of these augmented bags-of-words, classifiers and predictions into one potentially superior estimate. Additionally, it would be interesting to investigate if a boosting step, where each classifier's response is weighted according to some error measure, would improve performance over a bagging step.

7.2.2 Additional Classifiers and Techniques

In this research, two machine learning techniques were tested: SVM and sLDA. The researcher hypothesizes that other classical methods such as the random forest would be interesting to study given that it already uses bagging. It would be interesting to see what the effect on performance of having two bagging operations, one with the random forest and the other from this data augmentation method would be. Another promising classical model is Naïve Bayes. The researcher found that Naive Bayes can work very well on large corpora despite the violation of feature independence[27].

In Chapter 2 we saw a plethora of topic models and many more are developed each year. Most of these topic models fall into the Bayesian statistics or Bayesian non-parametrics paradigm of methods and has a dedicated following of researchers. This research discussed classical Latent Dirichlet Allocation (LDA) thoroughly as a basis for Bayesian topic models and studied Supervised Latent Dirichlet Allocation (sLDA) as a target classifier used after augmentation using the proposed method. Language is not independent and often appears in hierarchies, in networks, and with correlated themes. It would be enlightening to apply the proposed data augmentation method to these topic models. The challenge is that these topic models are not necessarily supervised; they would need to be combined with some supervised model to be able to serve as a classifier.

A paradigm that has seen a major rebirth in the past few years is the neural network. Deep learning has seen success particularly in natural language processing for speech recognition, machine translation, and parsing structures[103], as opposed to the bag-of-words approach. The author of this manuscript believes that the neural network and Bayesian paradigms accomplish very similar goals, but in different ways and that while one should make an educated choice as to which one to use, the author chose to focus on topic models out of research preference. It is unclear if this data augmentation method could be applied directly to techniques based on deep learning but dropout[24] provides a promising first step.

One final aspect of the proposed data augmentation method to investigate is the method used to construct the semantic space S. In this research we used Latent Semantic Analysis (LSA). It would be interesting to investigate other distance-preserving methods such as Non-negative Matrix Factorization (NMF)[162].

[27] *Sentiment Analysis with `scikit-learn`* presented by Ryan Rosario at *PyData Silicon Valley 2014* on behalf of Facebook, Inc. `https://www.youtube.com/watch?v=y3ZTKFZ-1QQ&t=1s`

7.2.3 Highly Unstructured Text

As mentioned in Chapter 1, the researcher originally proposed to use tweets for this research. Tweets extensively complicate research and are an example of a wider class of unstructured text. For topic classification, tweets may not have any signal related to the set of topics to be classified because they are about a theme of no consequence to the research. Unlike blog post titles, tweets often do not carry with them any sort of context on their own.

One can append some sort of context with the text by processing the tweets of the user in question, or considering a neighborhood of tweets that surround a particular tweet in time, but even this has its own challenges that the researcher believes to depend on the particular user's frequency of authoring tweets and the evolution of the topics he or she discusses. Another way to append context would be to process the tweets of all other users at the same time for time-based insights, or all other users in the same location at the same time for space-time insights. As an example, consider an earthquake that occurs at a particular time in a small town in California. Several tweets from users in the same location and in the same timeframe may tweet interjections such as `yikes!` or `that was scary!`. Topic classification would not be able to do much with such tweets except place them in a topic associated with fear. Other users in the region may author tweets containing conjugations of the word `shake` that can be used to add context the tweets in the region. It would be interesting to study how well this method works in conjunction with data augmentation and to determine if applying both yields any interesting improvements.

APPENDIX A

Guide to Notation Used in Chapters 4 and 5

Terms	
i	An index over terms.
j	An index over a second term, $i \neq j$.
t_i, t_j	Casual reference to two different terms.
$t_{i\cdot}, t_{j\cdot}$	Refers to two terms in the lexicon globally in the corpus without regard to document.
t_{im}	Refers to term i within document d_m.
t'_{im}	Some term selected from parallel document d'_m based on t_{im}.
$t'_{i\cdot}$	Refers to a term selected from the lexicon without regard to document, based on $t_{i\cdot}$.

Table A.1: Notation used to reference terms.

Documents	
l	An index representing an iteration of data augmentation, used with d^*.
m	An index over documents.
n	An index over a second document, $m \neq n$.
d	Casual reference to some short text document.
d^*	Casual reference to some augmented document.
d_m	Refers to the mth short text document in corpus D.
d^*_{ml}	Refers to the augmented version of document d_m after iteration l.
d'_m	Some document selected based on similarity to d_m.

Table A.2: Notation used to reference documents.

Corpus	
l	An index representing an iteration of data augmentation, used with D^*
D	The original corpus of short texts. There no subscript as it remains fixed.
D^*	The collection of all augmented corpora from all iterations.
D^*_l	The corpus of augmented documents generated from iteration l.

Table A.3: Notation used to reference the corpus.

Semantic Spaces	
S	General; refers to the set of semantic spaces $\{\mathbf{S_t}, \mathbf{S_d}\}$.
$\mathbf{S_t}$	The semantic space associated with terms (a matrix).
$\mathbf{S_d}$	The semantic space associated with documents (a matrix).
S_{t_i}, S_{t_j}	Singular vectors for terms t_i and t_j respectively.
S_{d_m}, S_{d_n}	Singular vectors for documents d_m and d_n respectively.
$\delta_{\mathbf{d}}$	A cosine similarity matrix comparing all documents in D.
$\delta_{\mathbf{t}}$	A cosine similarity matrix comparing all terms in V.
$\delta_d(d_m, d_n)$	Cosine similarity between two documents d_m and d_n.
$\delta_d(d'_m, d_m)$	Cosine similarity between d_m and the parallel document d'_m.
$\delta_{\mathbf{d}}(d_m, \cdot)$	Vector of cosine similarities between d_m and all other documents in D.
$\delta_t(t_{i\cdot}, t_{j\cdot})$	Global cosine similarity between terms t_i and t_j without regard for document.

Table A.4: Notation related to the semantic spaces.

Sampling Distributions	
$P(t_{im}\|d_m, X_{\cdot,m})$	Probability of selecting a particular term in a document, t_{im} according to TF-IDF.
$P(d'_m\|d_m, \delta_d)$	Probability of selecting a parallel document d'_m conditioned on similarity to d_m.
$P(t'_{im}\|t_{im}, d'_m, \delta_{\mathbf{t}})$	Probability of selecting an arbitrary term t'_{im} from parallel document d'_m given its similarity to t_{im}.
$P(t'_{i\cdot}\|t_{i\cdot}, D, \delta_{\mathbf{t}})$	Probability of selecting some term t' given its similarity to t based on similarity. In this case, no parallel document is involved, only cosine similarity between terms.
$P(d_m \to d^*_{ml}\|t'_{im})$	Probability of transitioning from short document d to augmented document d^* via adding term t'_{im}.

Table A.5: Notation related to sampling probabilities.

Latent Semantic Analysis (LSA)	
K	A variable holding number of dimensions retained from truncated SVD.
k	A particular instance of K.
V	The vocabulary/lexicon of D.
$\mathbf{X}$	The term-document matrix induced by D.
$\mathbf{X_k}$	The lower k-dimensional approximation of $\mathbf{X}$.
X_{im}	A TF-IDF score, element of the term-document matrix for term t_i in document d_m.
$\mathbf{D_k}$	The singular vectors associated with documents.
$\mathbf{\Sigma_k}$	A square diagonal matrix of singular values.
$\mathbf{W_k^T}$	The singular vectors associated with terms/words.

Table A.6: Notation related to the the construction of the semantic spaces with Latent Semantic Analysis (LSA).

Latent Dirichlet Allocation (LDA / sLDA)	
k	An index over topics.
K	Number of latent topics used in the model.
α	A hyperparameter of the LDA model.

Table A.7: Notation related to Latent Dirichlet Allocation (LDA).

Support Vector Machines	
C	A hyperparameter of the SVM model selected by grid search.

Table A.8: Notation related to the SVM model.

Experiments	
ε	The augmentation rate. The relative increase in document size.
$I(d_m), I(d^*_{ml})$	The mutual information of documents d_m and d'_{ml} respectively.

Table A.9: Notation related to experiments.

APPENDIX B

Per-class Performance for Benchmark Models

B.1 Per-class Performance for Baseline SVM

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.984	0.948	0.859	0.980	0.850	0.975	0.944	0.972	0.867	0.905
True Positive Rate/Recall	0.507	0.635	0.675	0.254	0.717	0.021	0.630	0.321	0.654	0.704
True Negative Rate	0.994	0.973	0.906	0.994	0.888	0.999	0.970	0.992	0.903	0.952
False Positive Rate	0.006	0.027	0.094	0.006	0.112	0.001	0.030	0.008	0.097	0.048
False Negative Rate	0.493	0.365	0.325	0.746	0.283	0.979	0.370	0.679	0.346	0.296
Precision	0.649	0.654	0.649	0.484	0.643	0.309	0.625	0.561	0.526	0.774
F1 Score	0.569	0.644	0.662	0.333	0.678	0.040	0.627	0.408	0.583	0.737

Table B.1: Per-class metrics for SVM with L2 penalty and TF-IDF features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.948	0.859	0.979	0.852	0.974	0.944	0.971	0.867	0.906
True Positive Rate/Recall	0.513	0.641	0.671	0.255	0.710	0.025	0.639	0.332	0.657	0.705
True Negative Rate	0.994	0.972	0.908	0.993	0.891	0.998	0.969	0.991	0.902	0.954
False Positive Rate	0.006	0.028	0.092	0.007	0.109	0.002	0.031	0.009	0.098	0.046
False Negative Rate	0.487	0.359	0.329	0.745	0.290	0.975	0.361	0.668	0.343	0.295
Precision	0.634	0.649	0.651	0.439	0.647	0.234	0.624	0.526	0.526	0.781
F1 Score	0.567	0.645	0.661	0.323	0.677	0.046	0.631	0.407	0.584	0.741

Table B.2: Per-class metrics for SVM with L2 penalty and word presence features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.948	0.859	0.979	0.851	0.974	0.945	0.971	0.867	0.906
True Positive Rate/Recall	0.513	0.641	0.671	0.256	0.710	0.025	0.639	0.332	0.657	0.705
True Negative Rate	0.994	0.972	0.908	0.993	0.891	0.998	0.969	0.991	0.902	0.953
False Positive Rate	0.006	0.028	0.092	0.007	0.109	0.002	0.031	0.009	0.098	0.047
False Negative Rate	0.487	0.359	0.329	0.744	0.290	0.975	0.361	0.668	0.343	0.295
Precision	0.635	0.649	0.652	0.438	0.647	0.234	0.625	0.526	0.526	0.781
F1 Score	0.568	0.645	0.661	0.323	0.677	0.046	0.632	0.407	0.584	0.741

Table B.3: Per-class metrics for SVM with L2 penalty and word count features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.948	0.859	0.979	0.851	0.974	0.945	0.971	0.867	0.906
True Positive Rate/Recall	0.513	0.641	0.671	0.256	0.710	0.025	0.639	0.332	0.657	0.705
True Negative Rate	0.994	0.972	0.908	0.993	0.891	0.998	0.969	0.991	0.902	0.953
False Positive Rate	0.006	0.028	0.092	0.007	0.109	0.002	0.031	0.009	0.098	0.047
False Negative Rate	0.487	0.359	0.329	0.744	0.290	0.975	0.361	0.668	0.343	0.295
Precision	0.635	0.649	0.652	0.438	0.647	0.234	0.625	0.526	0.526	0.781
F1 Score	0.568	0.645	0.661	0.323	0.677	0.046	0.632	0.407	0.584	0.741

Table B.4: Per-class metrics for SVM with L2 penalty and term frequency features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.984	0.946	0.858	0.979	0.850	0.975	0.945	0.972	0.866	0.903
True Positive Rate/Recall	0.521	0.635	0.668	0.231	0.716	0.019	0.619	0.308	0.654	0.702
True Negative Rate	0.994	0.971	0.907	0.995	0.888	0.999	0.971	0.992	0.901	0.950
False Positive Rate	0.006	0.029	0.093	0.005	0.112	0.001	0.029	0.008	0.099	0.050
False Negative Rate	0.479	0.365	0.332	0.769	0.284	0.981	0.381	0.692	0.346	0.298
Precision	0.647	0.635	0.648	0.473	0.642	0.243	0.629	0.558	0.522	0.768
F1 Score	0.577	0.635	0.658	0.310	0.677	0.035	0.624	0.397	0.581	0.734

Table B.5: Per-class metrics for SVM with L1 penalty and TF-IDF features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.946	0.860	0.979	0.852	0.974	0.944	0.971	0.865	0.905
True Positive Rate/Recall	0.526	0.644	0.664	0.244	0.709	0.024	0.626	0.326	0.660	0.702
True Negative Rate	0.993	0.970	0.911	0.994	0.892	0.998	0.969	0.990	0.899	0.952
False Positive Rate	0.007	0.030	0.089	0.006	0.108	0.002	0.031	0.010	0.101	0.048
False Negative Rate	0.474	0.356	0.336	0.756	0.291	0.976	0.374	0.674	0.340	0.298
Precision	0.629	0.630	0.657	0.442	0.649	0.227	0.621	0.511	0.520	0.776
F1 Score	0.573	0.637	0.660	0.315	0.677	0.043	0.623	0.398	0.581	0.737

Table B.6: Per-class metrics for SVM with L1 penalty and word presence features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.946	0.859	0.979	0.852	0.975	0.943	0.970	0.865	0.905
True Positive Rate/Recall	0.525	0.643	0.664	0.246	0.709	0.022	0.627	0.325	0.659	0.701
True Negative Rate	0.993	0.970	0.910	0.994	0.892	0.998	0.969	0.990	0.900	0.952
False Positive Rate	0.007	0.030	0.090	0.006	0.108	0.002	0.031	0.010	0.100	0.048
False Negative Rate	0.475	0.357	0.336	0.754	0.291	0.978	0.373	0.675	0.341	0.299
Precision	0.626	0.630	0.654	0.449	0.649	0.234	0.619	0.509	0.521	0.776
F1 Score	0.571	0.637	0.659	0.318	0.677	0.040	0.623	0.397	0.582	0.737

Table B.7: Per-class metrics for SVM with L1 penalty and word count features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.946	0.860	0.979	0.852	0.974	0.943	0.971	0.865	0.905
True Positive Rate/Recall	0.525	0.644	0.663	0.253	0.709	0.025	0.627	0.326	0.659	0.702
True Negative Rate	0.993	0.970	0.911	0.993	0.893	0.998	0.969	0.990	0.900	0.953
False Positive Rate	0.007	0.030	0.089	0.007	0.107	0.002	0.031	0.010	0.100	0.047
False Negative Rate	0.475	0.356	0.337	0.747	0.291	0.975	0.373	0.674	0.341	0.298
Precision	0.626	0.632	0.656	0.439	0.649	0.232	0.616	0.512	0.521	0.777
F1 Score	0.571	0.638	0.660	0.321	0.678	0.045	0.622	0.398	0.582	0.738

Table B.8: Per-class metrics for SVM with L1 penalty and term frequency features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.984	0.948	0.858	0.978	0.850	0.975	0.944	0.972	0.867	0.906
True Positive Rate/Recall	0.498	0.637	0.676	0.251	0.708	0.025	0.631	0.327	0.658	0.704
True Negative Rate	0.994	0.973	0.905	0.993	0.890	0.998	0.969	0.991	0.902	0.953
False Positive Rate	0.006	0.027	0.095	0.007	0.110	0.002	0.031	0.009	0.098	0.047
False Negative Rate	0.502	0.363	0.324	0.749	0.292	0.975	0.369	0.673	0.342	0.296
Precision	0.643	0.655	0.648	0.433	0.644	0.242	0.625	0.538	0.526	0.780
F1 Score	0.561	0.646	0.662	0.318	0.675	0.045	0.628	0.407	0.585	0.740

Table B.9: Per-class metrics for logistic regression with TF-IDF features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.948	0.857	0.979	0.849	0.975	0.944	0.971	0.867	0.904
True Positive Rate/Recall	0.491	0.632	0.678	0.249	0.710	0.023	0.623	0.319	0.650	0.701
True Negative Rate	0.994	0.973	0.904	0.994	0.888	0.998	0.969	0.991	0.902	0.952
False Positive Rate	0.006	0.027	0.096	0.006	0.112	0.002	0.031	0.009	0.098	0.048
False Negative Rate	0.509	0.368	0.322	0.751	0.290	0.977	0.377	0.681	0.350	0.299
Precision	0.643	0.655	0.644	0.449	0.641	0.248	0.621	0.534	0.525	0.775
F1 Score	0.557	0.643	0.661	0.320	0.673	0.041	0.622	0.400	0.581	0.736

Table B.10: Per-class metrics for logistic regression with word presence features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.948	0.857	0.979	0.849	0.975	0.944	0.971	0.867	0.904
True Positive Rate/Recall	0.491	0.632	0.678	0.250	0.710	0.023	0.623	0.320	0.651	0.701
True Negative Rate	0.994	0.973	0.904	0.994	0.888	0.998	0.970	0.991	0.902	0.952
False Positive Rate	0.006	0.027	0.096	0.006	0.112	0.002	0.030	0.009	0.098	0.048
False Negative Rate	0.509	0.368	0.322	0.750	0.290	0.977	0.377	0.680	0.349	0.299
Precision	0.643	0.655	0.644	0.450	0.640	0.248	0.622	0.535	0.525	0.775
F1 Score	0.557	0.643	0.661	0.321	0.673	0.041	0.622	0.400	0.581	0.736

Table B.11: Per-class metrics for logistic regression with word count features.

Metric / Class	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.983	0.948	0.857	0.979	0.849	0.975	0.944	0.971	0.867	0.904
True Positive Rate/Recall	0.491	0.632	0.678	0.250	0.710	0.023	0.623	0.320	0.651	0.701
True Negative Rate	0.994	0.973	0.904	0.994	0.888	0.998	0.970	0.991	0.902	0.952
False Positive Rate	0.006	0.027	0.096	0.006	0.112	0.002	0.030	0.009	0.098	0.048
False Negative Rate	0.509	0.368	0.322	0.750	0.290	0.977	0.377	0.680	0.349	0.299
Precision	0.643	0.655	0.644	0.450	0.640	0.248	0.622	0.535	0.525	0.775
F1 Score	0.557	0.643	0.661	0.321	0.673	0.041	0.622	0.400	0.581	0.736

Table B.12: Per-class metrics for logistic regression with term frequency features.

B.2 Per-class Performance Metrics for Baseline sLDA

Metric	α	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.2	0.775	0.817	0.682	0.758	0.782	0.640	0.667	0.610	0.797	0.855
False Positive Rate	0.2	0.225	0.183	0.339	0.242	0.220	0.360	0.339	0.390	0.188	0.143
False Negative Rate	0.2	0.225	0.177	0.245	0.246	0.213	0.361	0.226	0.375	0.284	0.167
True Negative Rate	0.2	0.775	0.817	0.662	0.758	0.780	0.640	0.661	0.610	0.813	0.858
Precision	0.2	0.091	0.316	0.385	0.070	0.563	0.037	0.116	0.055	0.418	0.400
Recall (True Positive Rate)	0.2	0.775	0.823	0.755	0.754	0.787	0.639	0.774	0.625	0.716	0.833
F1 Score	0.2	0.273	0.504	0.510	0.171	0.658	0.081	0.269	0.103	0.581	0.610
AUC Score	0.2	0.827	0.874	0.767	0.815	0.850	0.683	0.775	0.647	0.811	0.903
Accuracy	1.0	0.765	0.832	0.755	0.651	0.748	0.592	0.660	0.649	0.751	0.799
False Positive Rate	1.0	0.235	0.166	0.243	0.350	0.261	0.408	0.346	0.351	0.239	0.203
False Negative Rate	1.0	0.232	0.191	0.254	0.293	0.228	0.402	0.245	0.368	0.308	0.187
True Negative Rate	1.0	0.765	0.834	0.757	0.650	0.739	0.592	0.655	0.649	0.762	0.798
Precision	1.0	0.087	0.334	0.463	0.047	0.515	0.031	0.112	0.062	0.353	0.314
Recall (True Positive Rate)	1.0	0.768	0.809	0.746	0.707	0.772	0.598	0.755	0.632	0.692	0.814
F1 Score	1.0	0.290	0.543	0.577	0.120	0.618	0.061	0.221	0.147	0.512	0.537
AUC Score	1.0	0.841	0.887	0.801	0.728	0.822	0.625	0.764	0.693	0.793	0.875

Table B.13: sLDA experimental results with $K = 5$ topics.

Metric	α	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	0.1	0.782	0.804	0.753	0.747	0.744	0.643	0.726	0.679	0.770	0.832
False Positive Rate	0.1	0.218	0.194	0.250	0.252	0.263	0.356	0.272	0.320	0.218	0.168
False Negative Rate	0.1	0.232	0.222	0.236	0.311	0.236	0.387	0.307	0.349	0.298	0.175
True Negative Rate	0.1	0.782	0.806	0.750	0.748	0.737	0.644	0.728	0.680	0.782	0.833
Precision	0.1	0.093	0.292	0.461	0.062	0.511	0.036	0.128	0.069	0.377	0.360
Recall (True Positive Rate)	0.1	0.768	0.778	0.764	0.689	0.764	0.613	0.693	0.651	0.703	0.825
F1 Score	0.1	0.279	0.460	0.581	0.164	0.612	0.095	0.268	0.163	0.532	0.585
AUC Score	0.1	0.826	0.858	0.814	0.781	0.826	0.685	0.787	0.724	0.806	0.898
Accuracy	1.0	0.794	0.800	0.748	0.690	0.744	0.600	0.741	0.691	0.761	0.822
False Positive Rate	1.0	0.206	0.199	0.251	0.310	0.256	0.399	0.259	0.309	0.230	0.175
False Negative Rate	1.0	0.226	0.214	0.254	0.341	0.255	0.424	0.257	0.320	0.290	0.211
True Negative Rate	1.0	0.794	0.801	0.749	0.690	0.744	0.601	0.741	0.691	0.770	0.826
Precision	1.0	0.098	0.289	0.455	0.049	0.511	0.030	0.142	0.075	0.367	0.340
Recall (True Positive Rate)	1.0	0.774	0.787	0.746	0.659	0.745	0.576	0.743	0.680	0.711	0.789
F1 Score	1.0	0.366	0.499	0.570	0.134	0.609	0.064	0.305	0.242	0.529	0.544
AUC Score	1.0	0.855	0.870	0.809	0.739	0.823	0.631	0.816	0.749	0.809	0.879

Table B.14: sLDA experimental results with $K = 10$ topics.

Metric	α	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	.02	0.808	0.833	0.756	0.682	0.767	0.616	0.796	0.646	0.747	0.840
False Positive Rate	.02	0.191	0.163	0.244	0.319	0.229	0.384	0.201	0.357	0.245	0.159
False Negative Rate	.02	0.219	0.198	0.243	0.275	0.242	0.386	0.260	0.282	0.297	0.169
True Negative Rate	.02	0.809	0.837	0.756	0.681	0.771	0.616	0.799	0.643	0.755	0.841
Precision	.02	0.106	0.335	0.465	0.052	0.543	0.034	0.175	0.069	0.350	0.373
Recall (True Positive Rate)	.02	0.781	0.802	0.757	0.725	0.758	0.614	0.740	0.718	0.703	0.832
F1 Score	.02	0.422	0.585	0.586	0.181	0.635	0.092	0.419	0.220	0.525	0.617
AUC Score	.02	0.871	0.891	0.829	0.775	0.843	0.661	0.850	0.749	0.822	0.904
Accuracy	1.0	0.849	0.830	0.743	0.678	0.774	0.575	0.742	0.733	0.770	0.846
False Positive Rate	1.0	0.149	0.167	0.254	0.323	0.216	0.426	0.259	0.265	0.217	0.151
False Negative Rate	1.0	0.237	0.192	0.265	0.313	0.255	0.408	0.248	0.337	0.298	0.179
True Negative Rate	1.0	0.851	0.833	0.746	0.677	0.784	0.574	0.741	0.735	0.783	0.849
Precision	1.0	0.129	0.331	0.448	0.049	0.554	0.029	0.143	0.084	0.378	0.383
Recall (True Positive Rate)	1.0	0.763	0.808	0.735	0.687	0.745	0.592	0.752	0.663	0.702	0.821
F1 Score	1.0	0.505	0.584	0.566	0.201	0.639	0.060	0.399	0.339	0.539	0.624
AUC Score	1.0	0.874	0.890	0.815	0.753	0.837	0.613	0.820	0.764	0.811	0.901

Table B.15: sLDA experimental results with $K = 50$ topics.

Metric	α	autos	business	entertainment	green	living	overall	politics	science	sports	technology
Accuracy	.01	0.799	0.813	0.731	0.690	0.752	0.585	0.755	0.627	0.759	0.841
False Positive Rate	.01	0.200	0.185	0.272	0.311	0.249	0.416	0.246	0.375	0.231	0.155
False Negative Rate	.01	0.228	0.209	0.257	0.298	0.246	0.368	0.229	0.326	0.294	0.189
True Negative Rate	.01	0.800	0.815	0.728	0.689	0.751	0.584	0.754	0.625	0.769	0.845
Precision	.01	0.100	0.305	0.433	0.052	0.522	0.032	0.153	0.062	0.365	0.374
Recall (True Positive Rate)	.01	0.772	0.791	0.743	0.702	0.754	0.632	0.771	0.674	0.707	0.811
F1 Score	.01	0.424	0.550	0.558	0.184	0.619	0.090	0.411	0.220	0.522	0.607
AUC Score	.01	0.847	0.876	0.807	0.767	0.833	0.655	0.840	0.721	0.812	0.896
Accuracy	1.0	0.852	0.817	0.743	0.705	0.756	0.566	0.737	0.730	0.751	0.840
False Positive Rate	1.0	0.145	0.180	0.251	0.294	0.245	0.434	0.262	0.266	0.238	0.158
False Negative Rate	1.0	0.253	0.212	0.278	0.352	0.242	0.448	0.292	0.391	0.303	0.179
True Negative Rate	1.0	0.855	0.820	0.749	0.706	0.755	0.566	0.738	0.734	0.762	0.842
Precision	1.0	0.129	0.310	0.446	0.051	0.527	0.027	0.135	0.077	0.355	0.373
Recall (True Positive Rate)	1.0	0.747	0.788	0.722	0.648	0.758	0.552	0.708	0.609	0.697	0.821
F1 Score	1.0	0.545	0.588	0.560	0.241	0.626	0.055	0.394	0.256	0.526	0.619
AUC Score	1.0	0.860	0.874	0.813	0.755	0.827	0.586	0.799	0.723	0.795	0.896

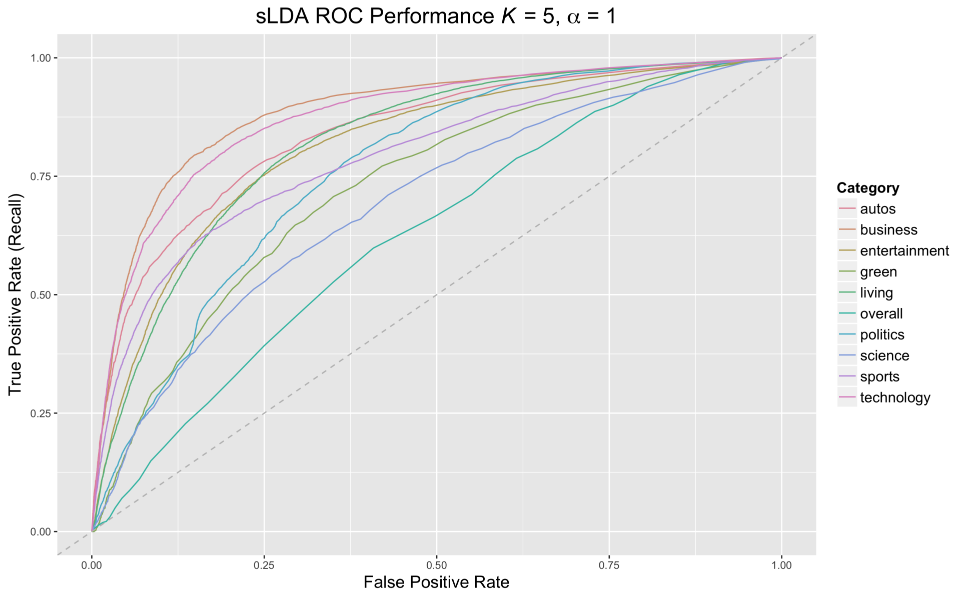
Table B.16: sLDA experimental results with $K = 100$ topics.

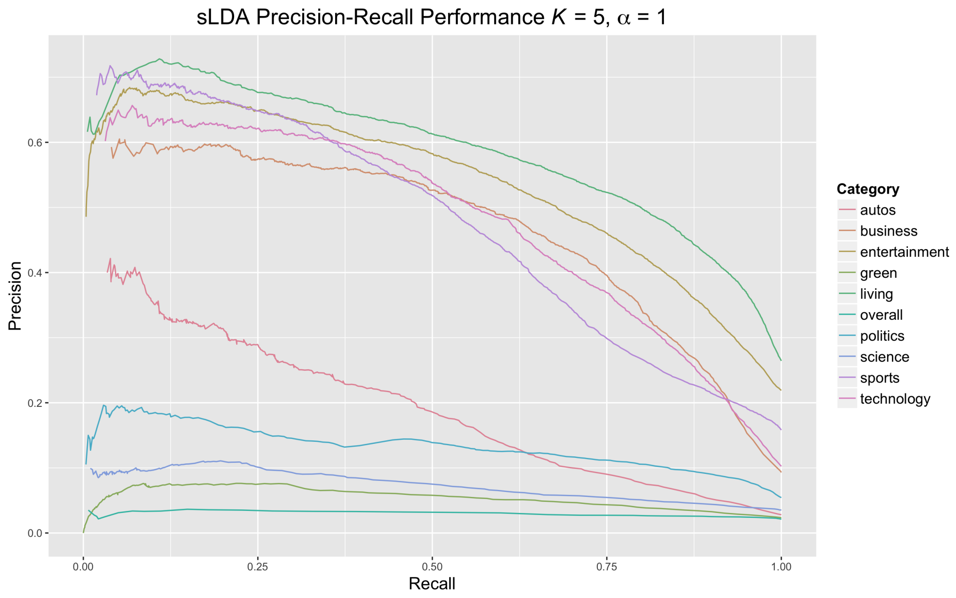
APPENDIX C

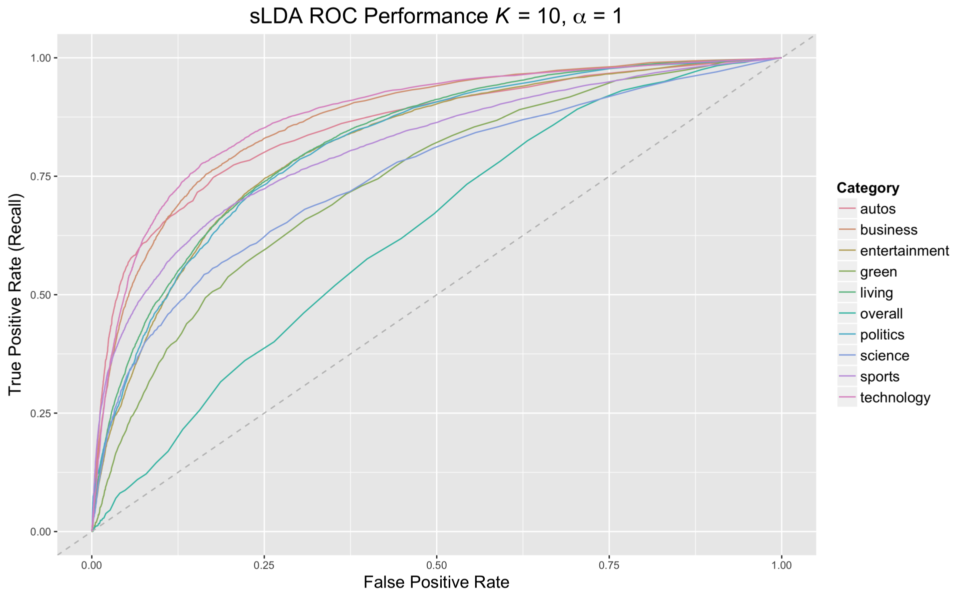
Ancillary Plots

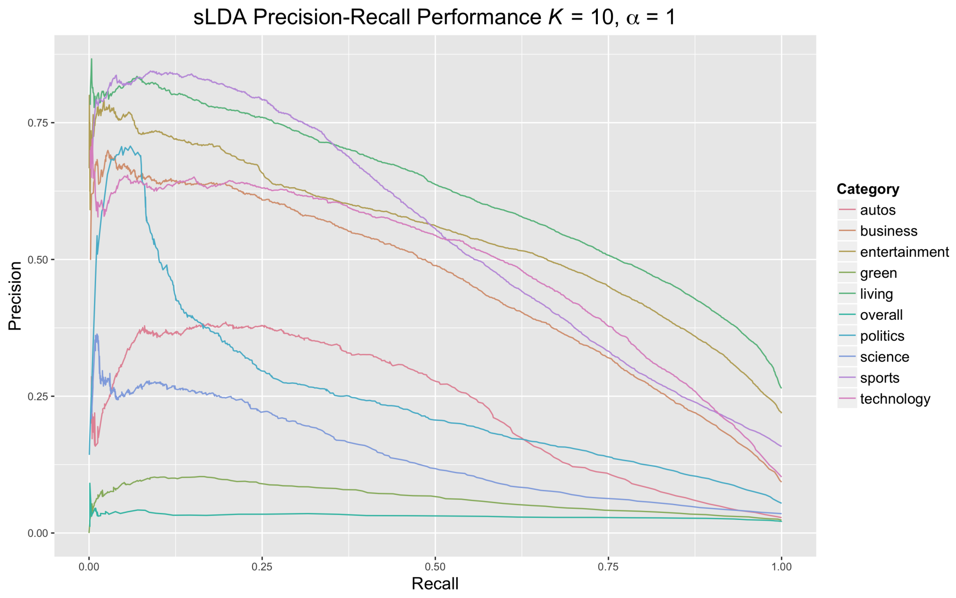
C.1 sLDA Benchmark Experiments

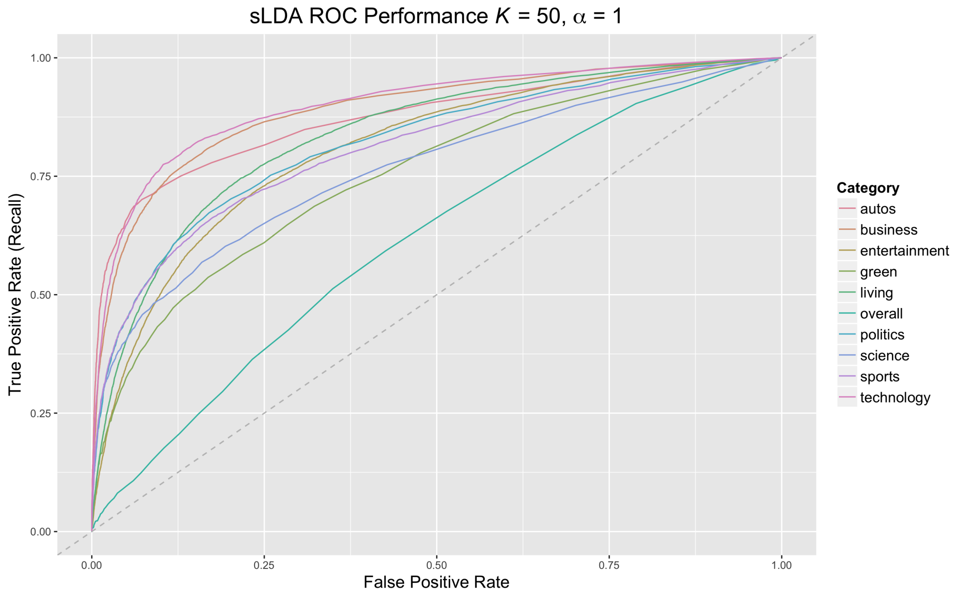
C.1.1 Performance for All Categories at Various K and α

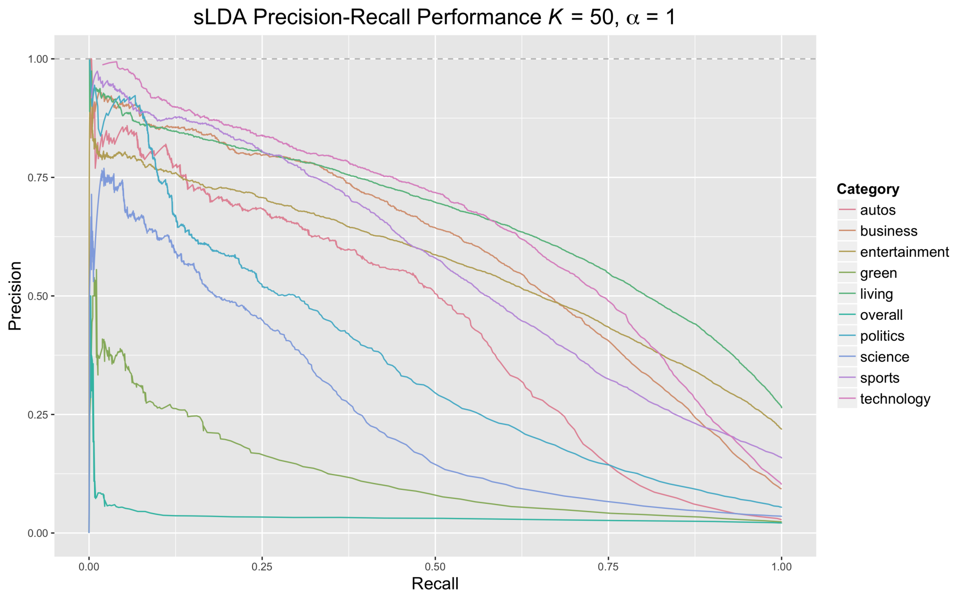

sLDA ROC Performance K = 5, α = 1

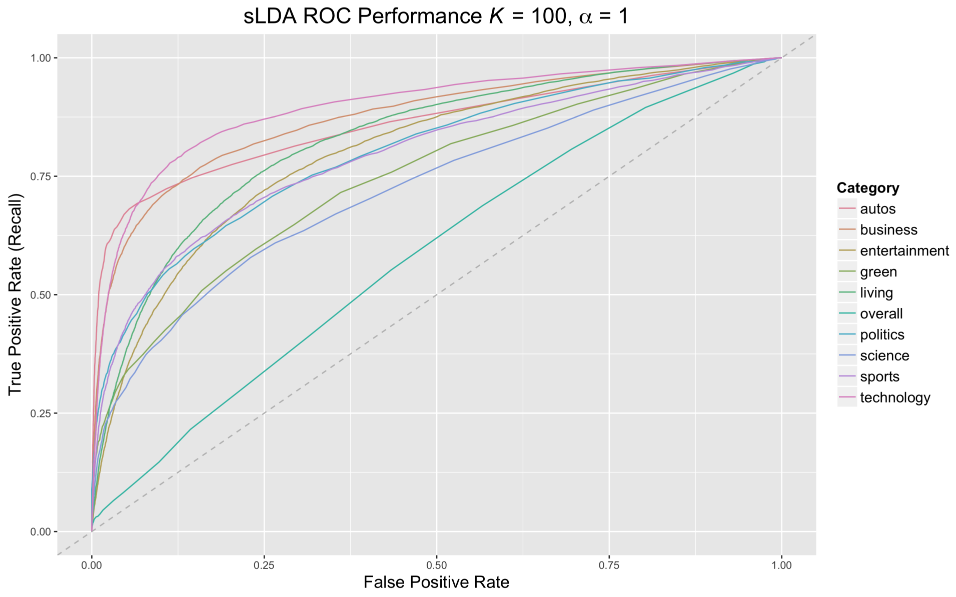

sLDA Precision-Recall Performance K = 5, α = 1

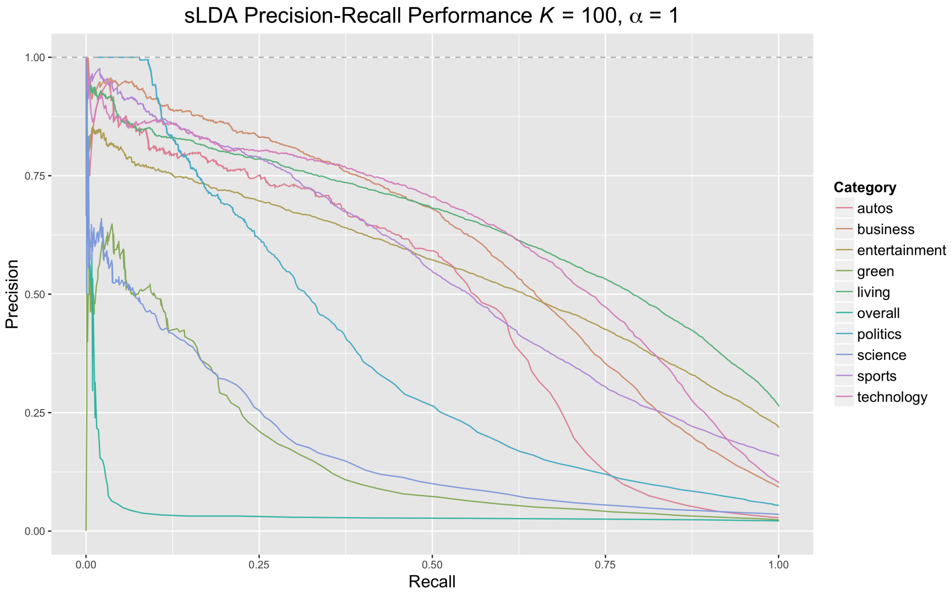

sLDA ROC Performance K = 10, α = 1


sLDA Precision-Recall Performance K = 10, α = 1
Precision
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
Recall
Category
autos
business
entertainment
green
living
overall
politics
science
sports
technology

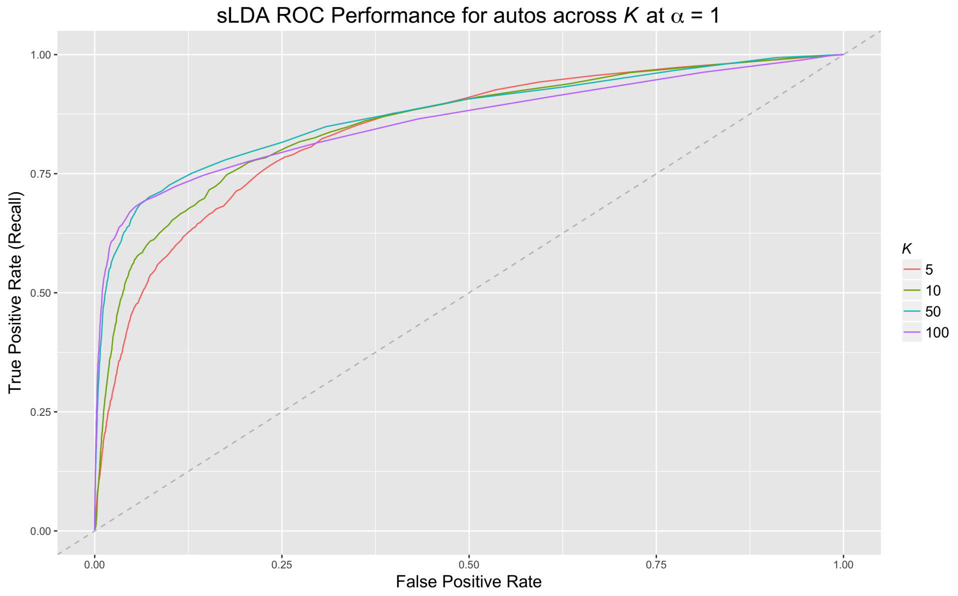

sLDA ROC Performance K = 50, α = 1
True Positive Rate (Recall)
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
False Positive Rate
Category
autos
business
entertainment
green
living
overall
politics
science
sports
technology

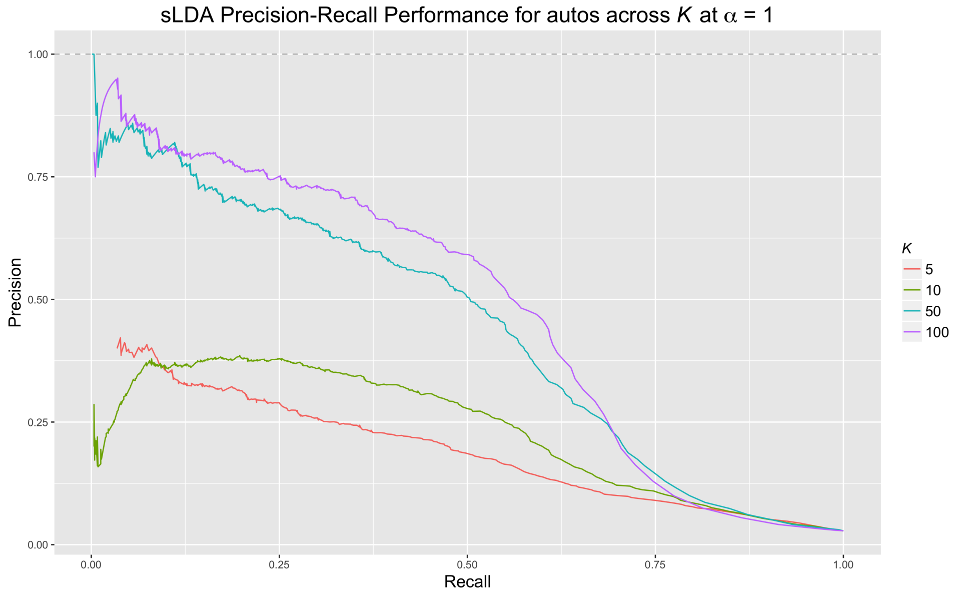

sLDA Precision-Recall Performance K = 50, α = 1
Precision
Recall
0.00
0.25
0.50
0.75
1.00
Category
autos
business
entertainment
green
living
overall
politics
science
sports
technology

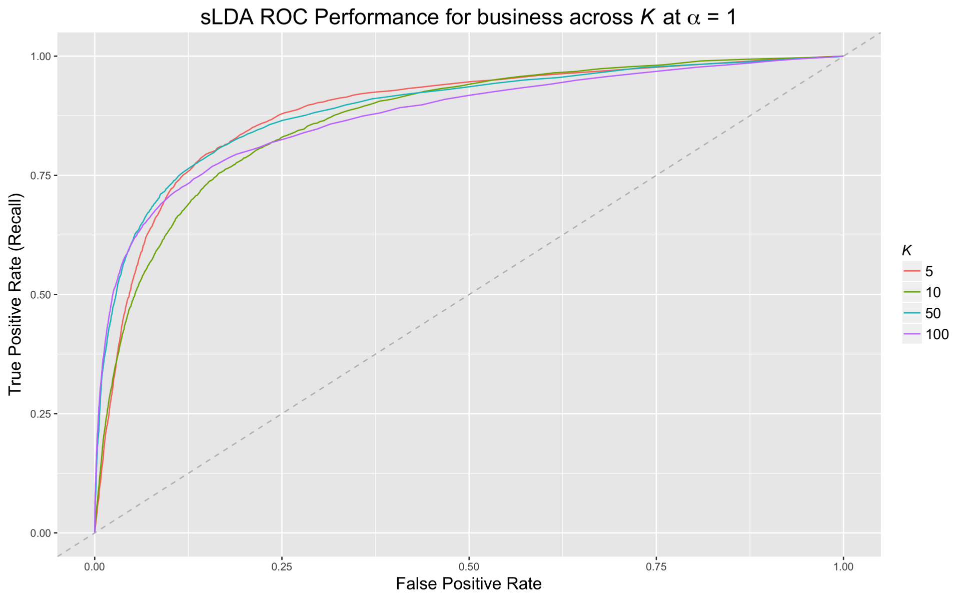

sLDA ROC Performance K = 100, α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
Category
autos
business
entertainment
green
living
overall
politics
science
sports
technology

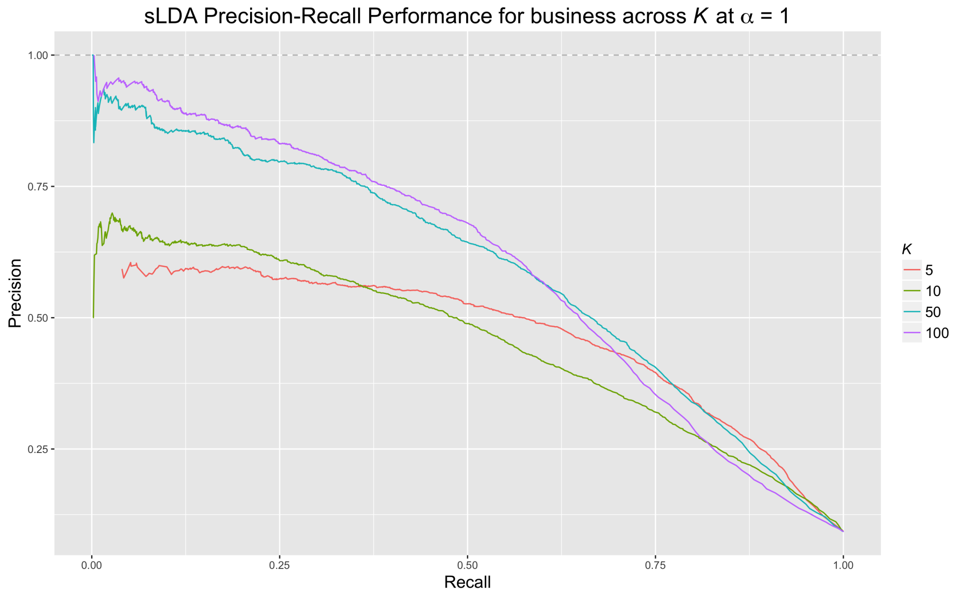

sLDA Precision-Recall Performance K = 100, α = 1
Precision
1.00
0.75
0.50
0.25
0.00
Recall
0.00
0.25
0.50
0.75
1.00
Category
autos
business
entertainment
green
living
overall
politics
science
sports
technology

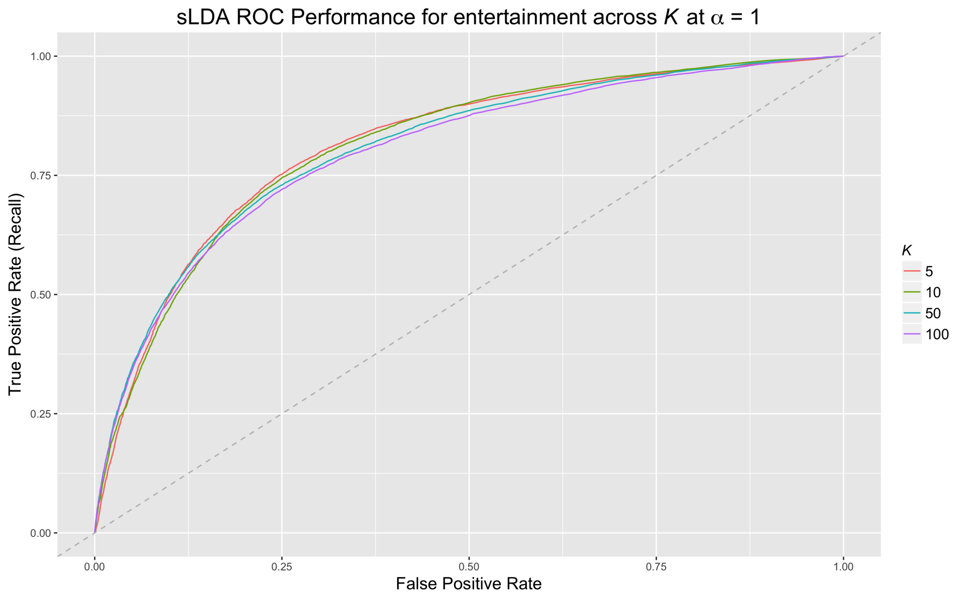
C.1.2 Performance for each Category at Various K and $\alpha = 1$

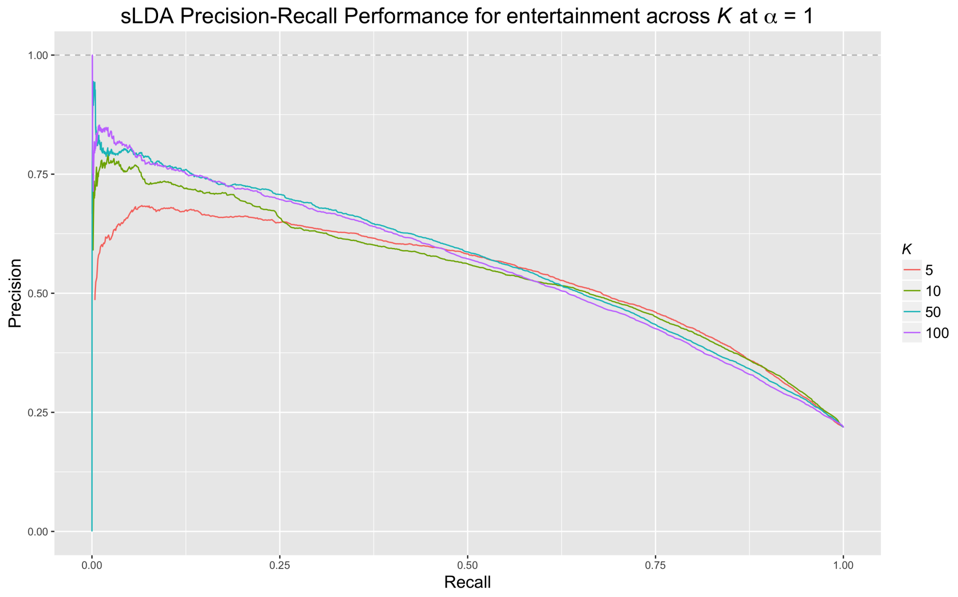


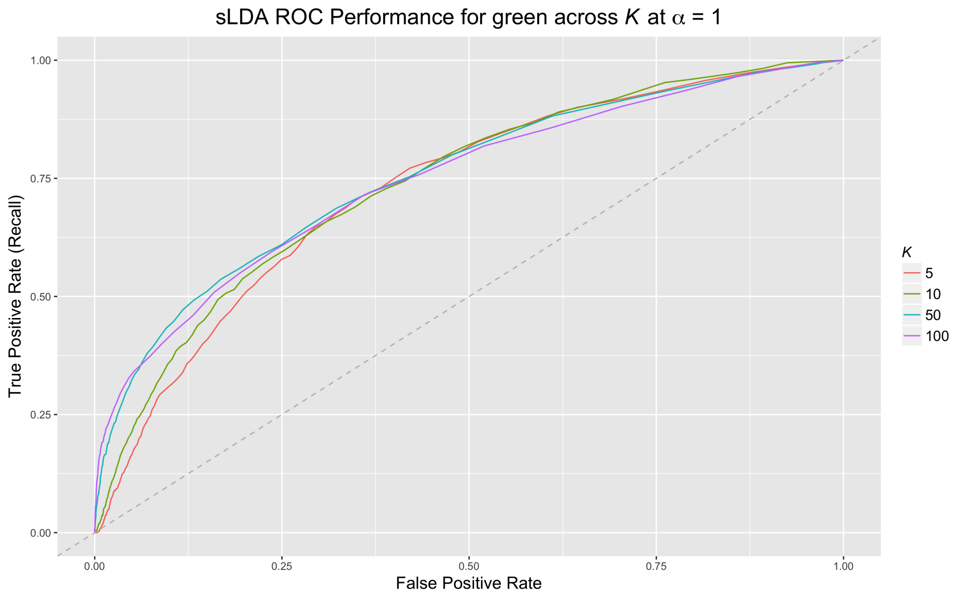


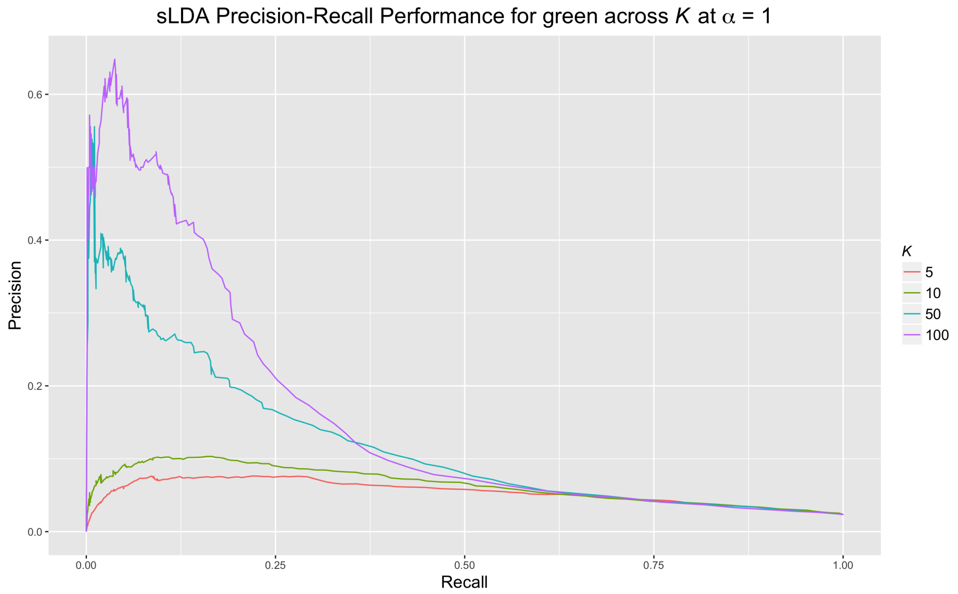

sLDA ROC Performance for business across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

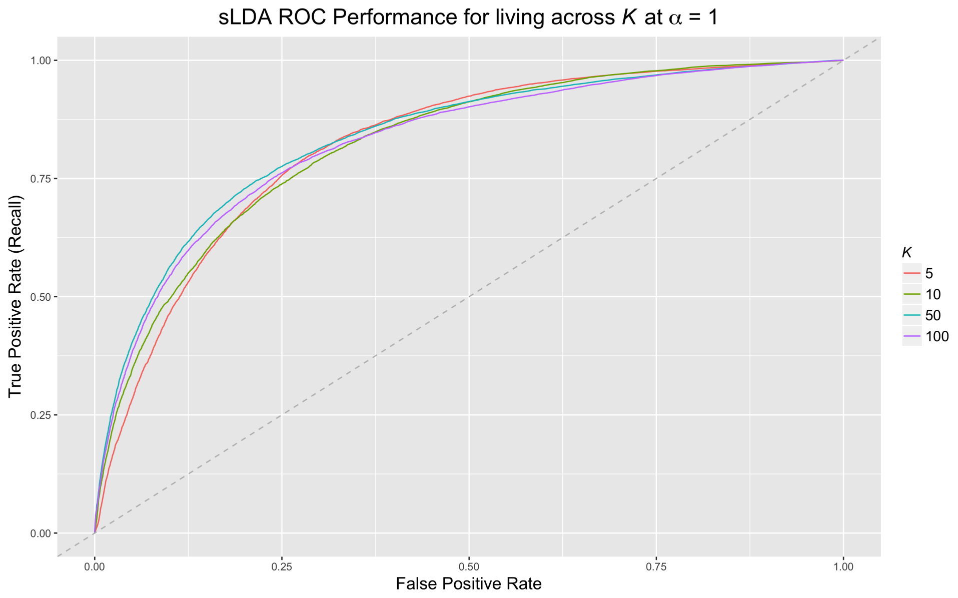

sLDA Precision-Recall Performance for business across K at α = 1
Precision
Recall
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

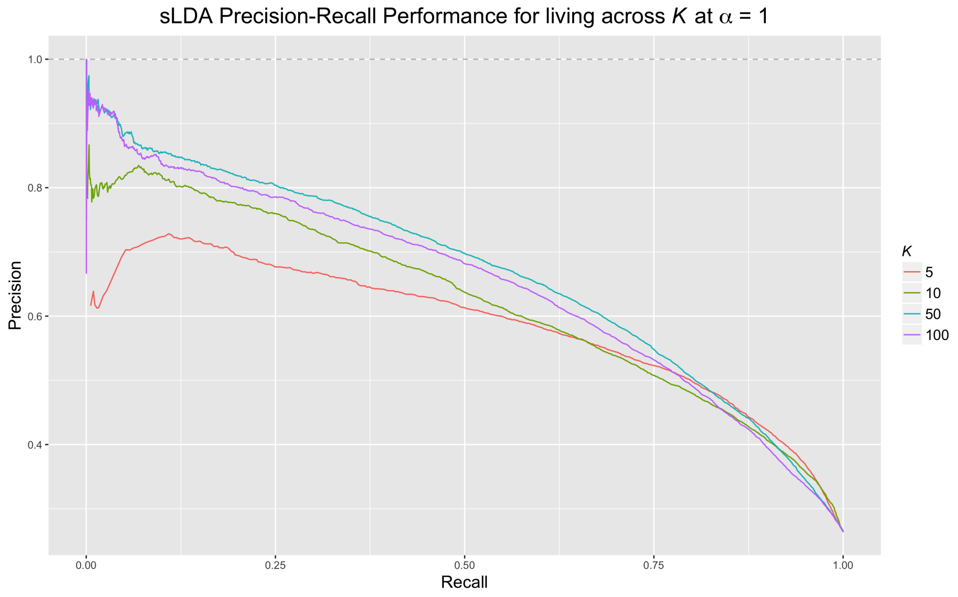

sLDA ROC Performance for entertainment across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

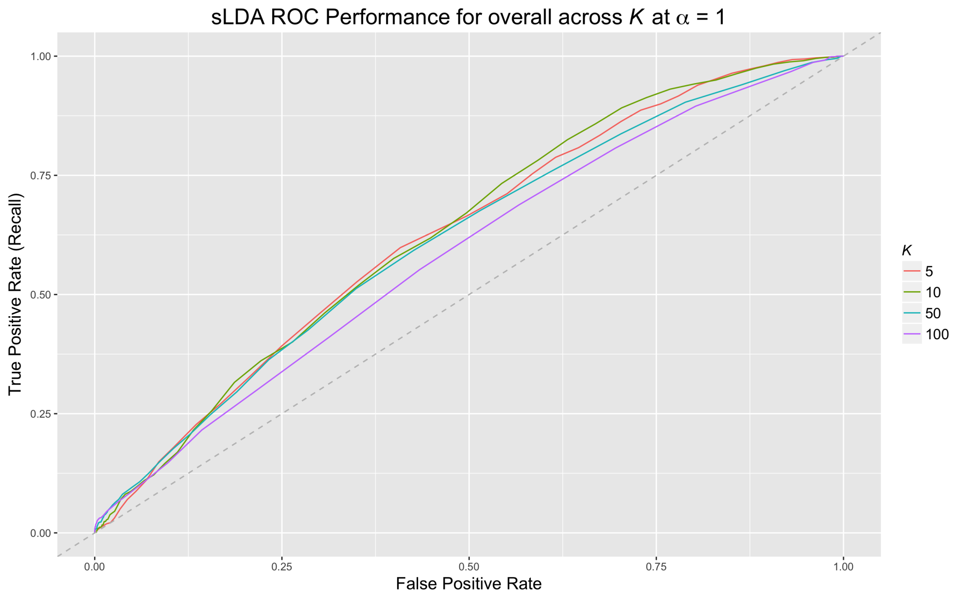

sLDA Precision-Recall Performance for entertainment across K at α = 1
Precision
Recall
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

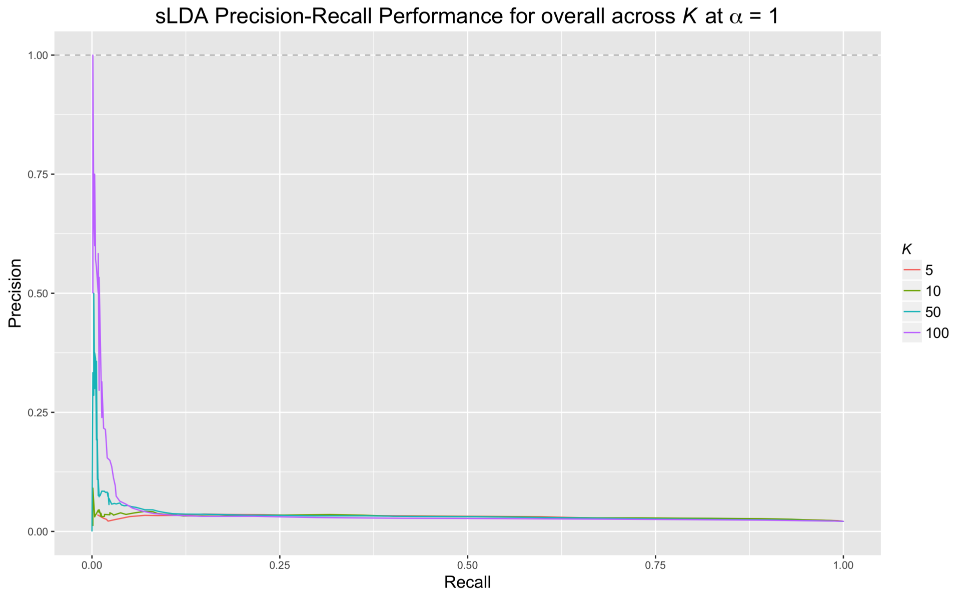

sLDA ROC Performance for green across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

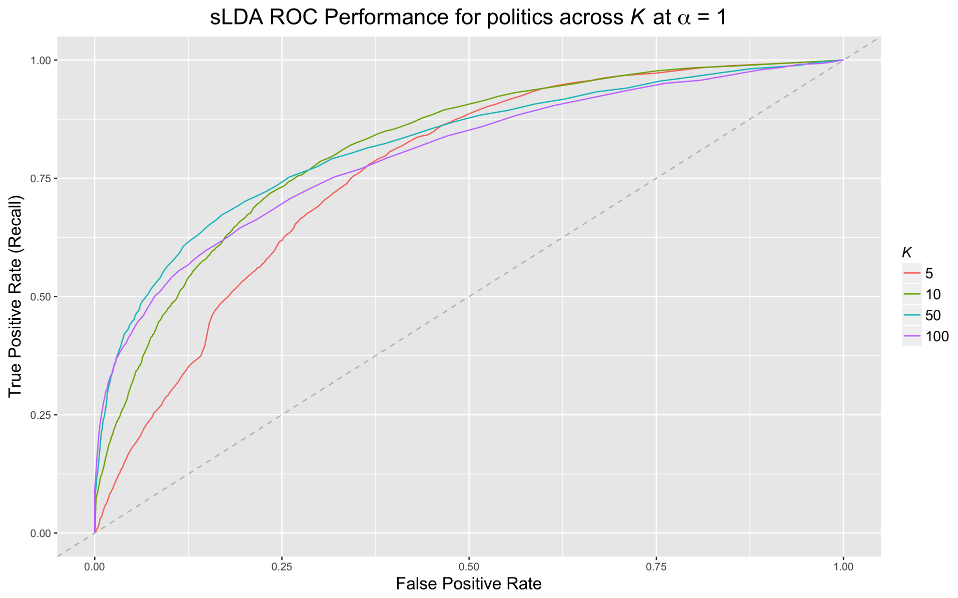

sLDA Precision-Recall Performance for green across K at α = 1
Precision
Recall
0.0
0.2
0.4
0.6
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

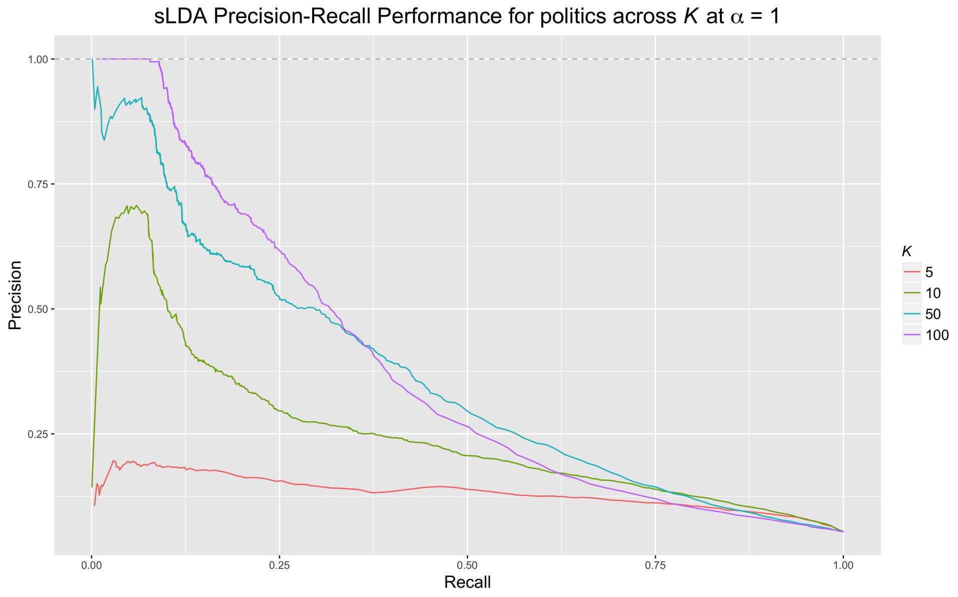

sLDA ROC Performance for living across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

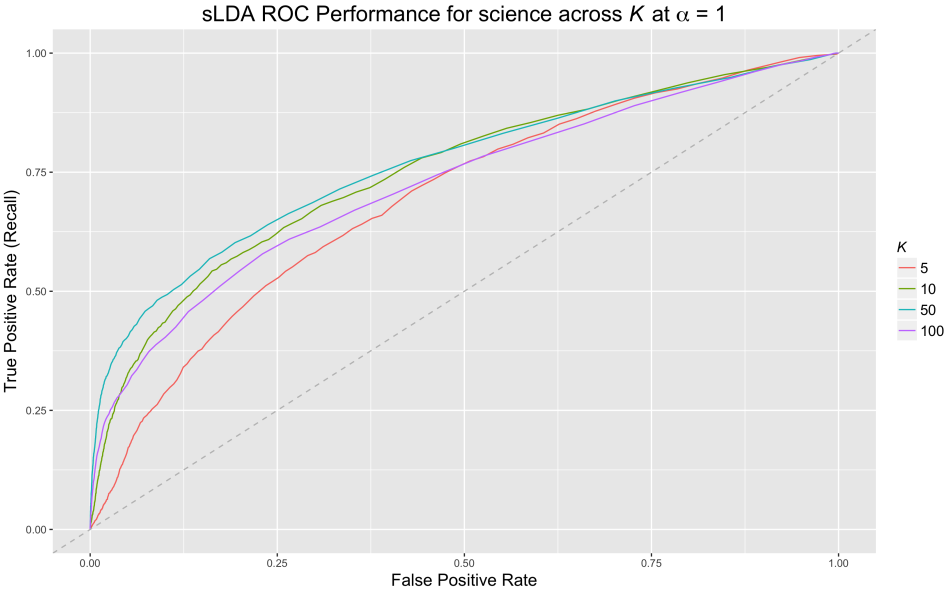

sLDA Precision-Recall Performance for living across K at α = 1
Precision
Recall
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

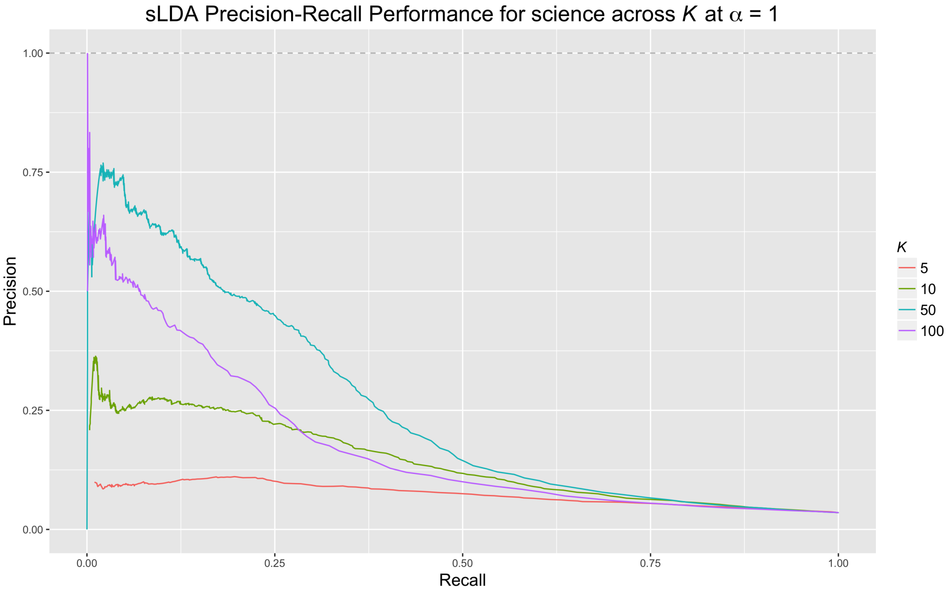

sLDA ROC Performance for overall across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

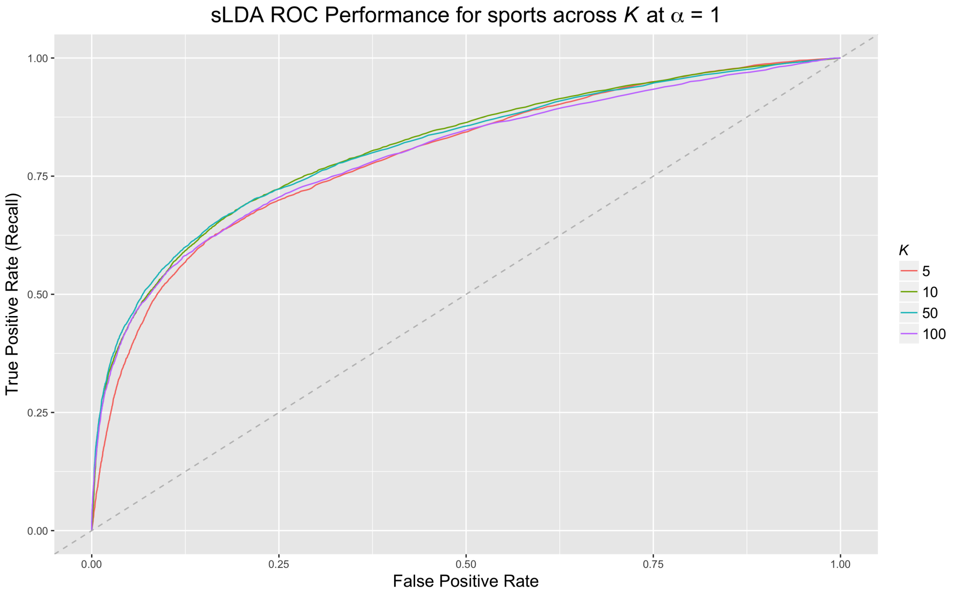

sLDA Precision-Recall Performance for overall across K at α = 1
Precision
Recall
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

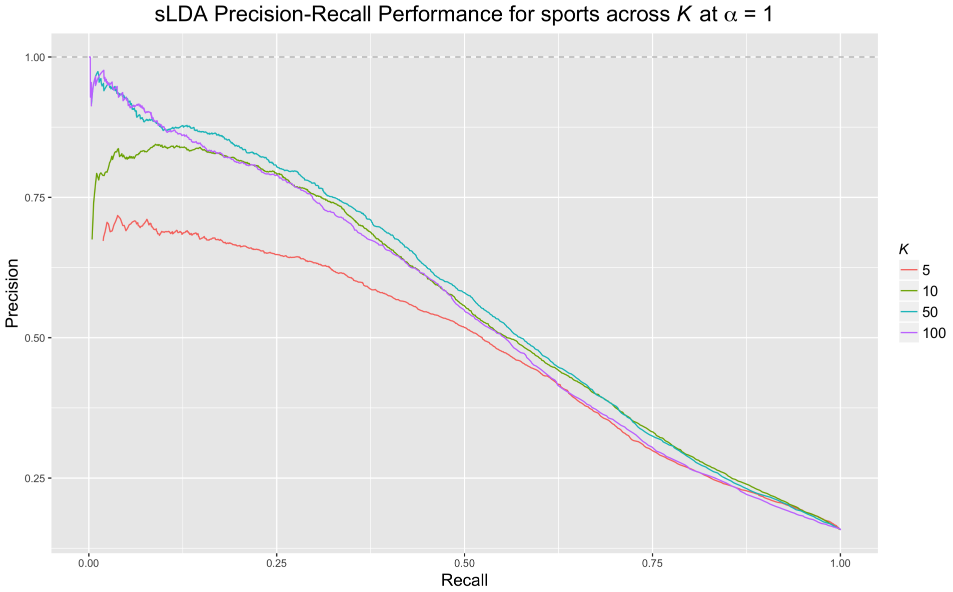

sLDA ROC Performance for politics across K at α = 1
True Positive Rate (Recall)
False Positive Rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

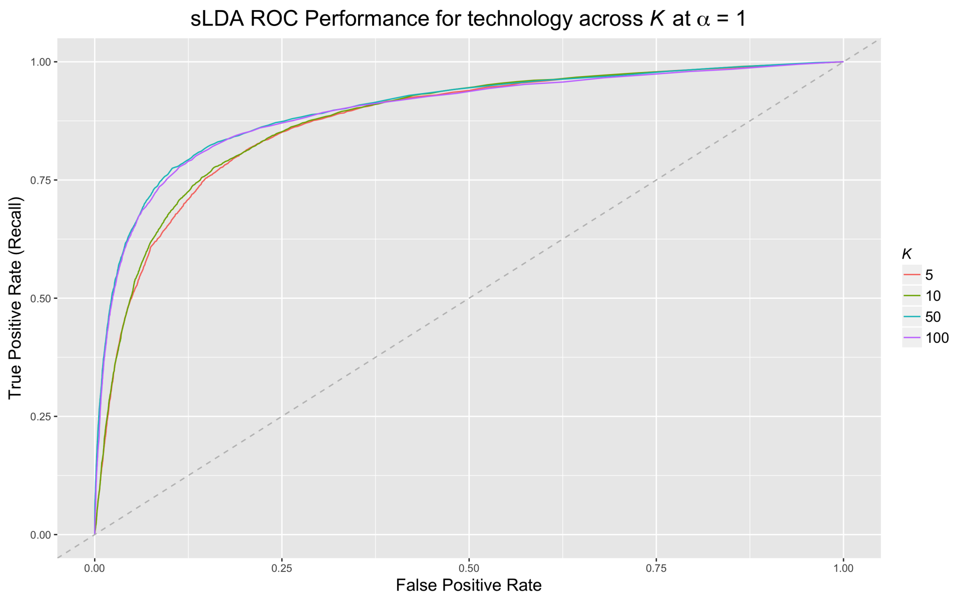

sLDA Precision-Recall Performance for politics across K at α = 1
Precision
Recall
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

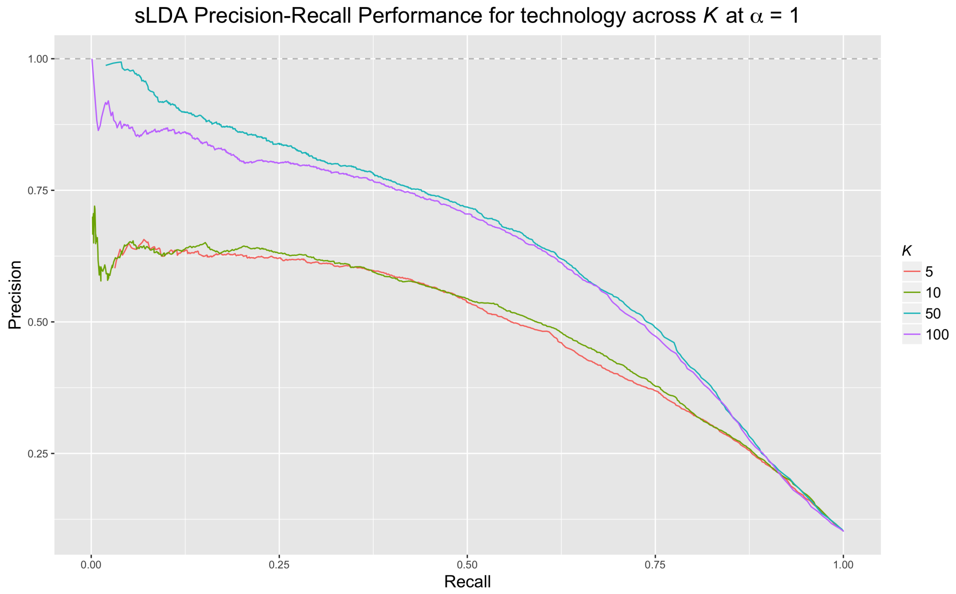

sLDA ROC Performance for science across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100


sLDA Precision-Recall Performance for science across K at α = 1
Precision
Recall
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

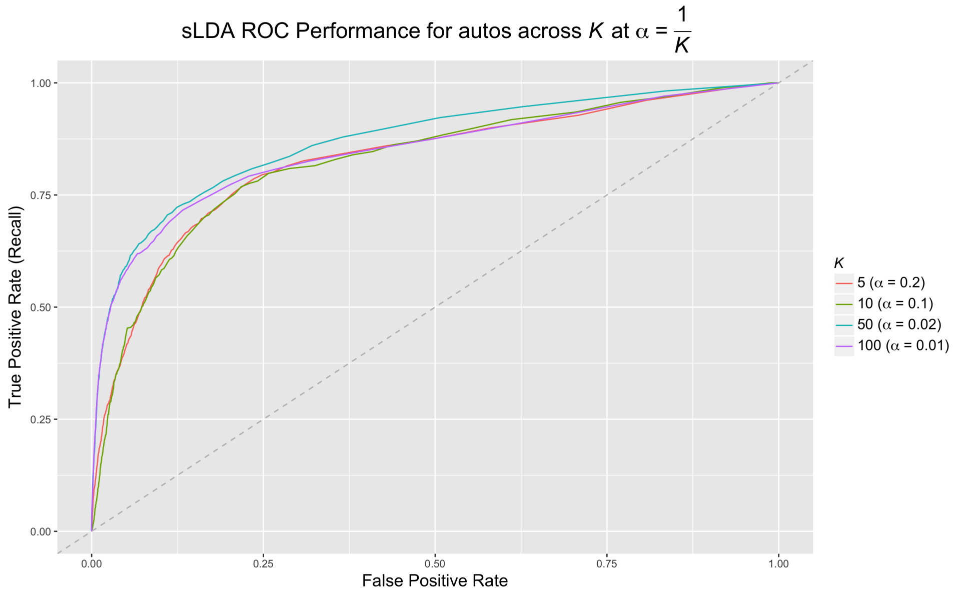

sLDA ROC Performance for sports across K at α = 1
True Positive Rate (Recall)
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
False Positive Rate
K
5
10
50
100

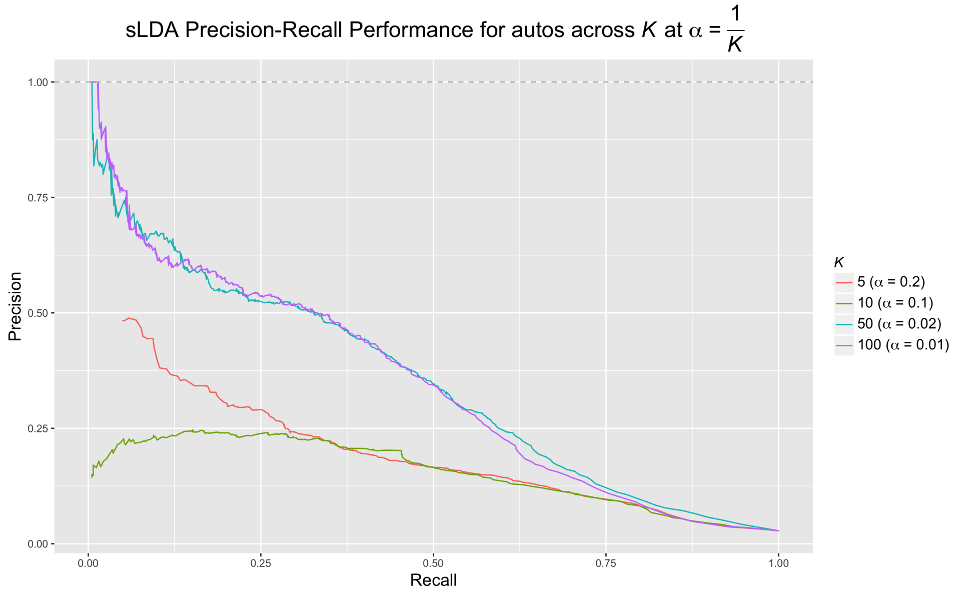

sLDA Precision-Recall Performance for sports across K at α = 1
Precision
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
Recall
K
5
10
50
100

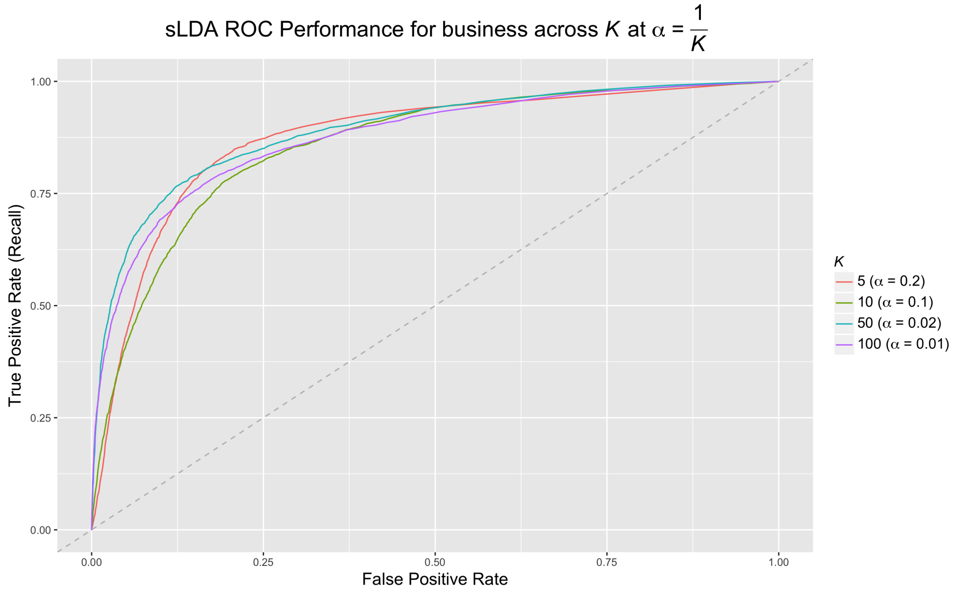

sLDA ROC Performance for technology across K at α = 1
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

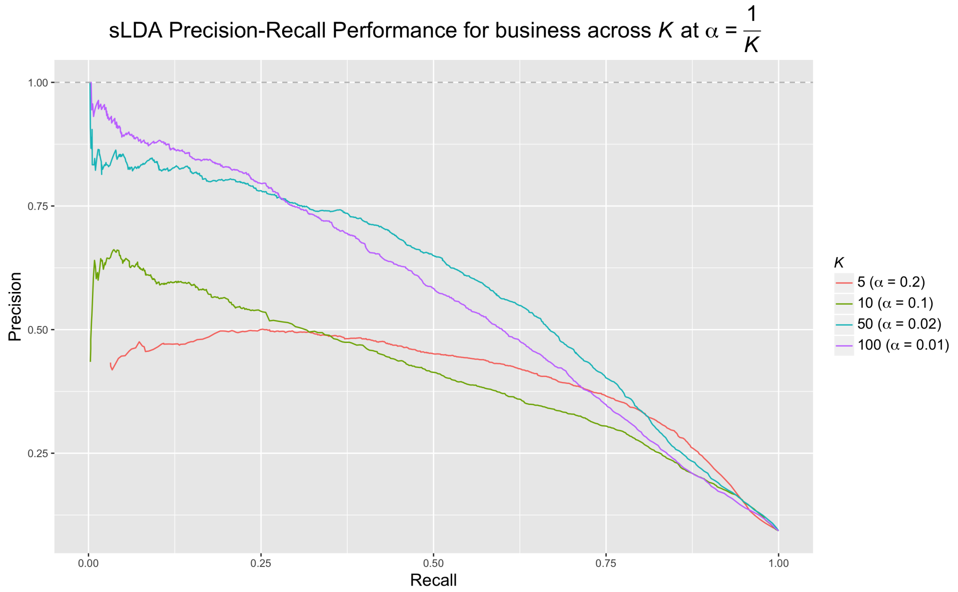

sLDA Precision-Recall Performance for technology across K at α = 1
Precision
Recall
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5
10
50
100

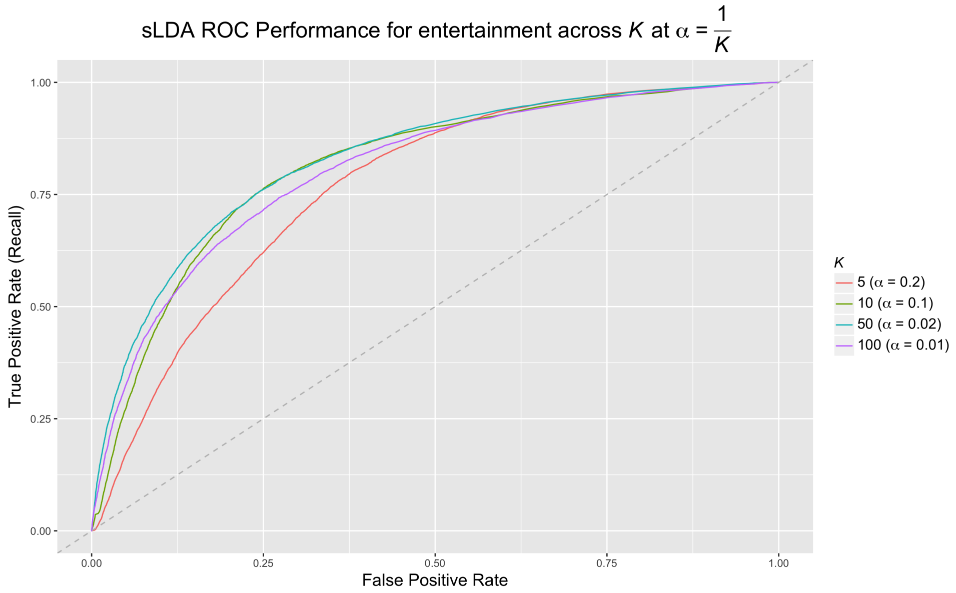
C.1.3 Performance for each Category at Various K and $\alpha = \frac{1}{K}$

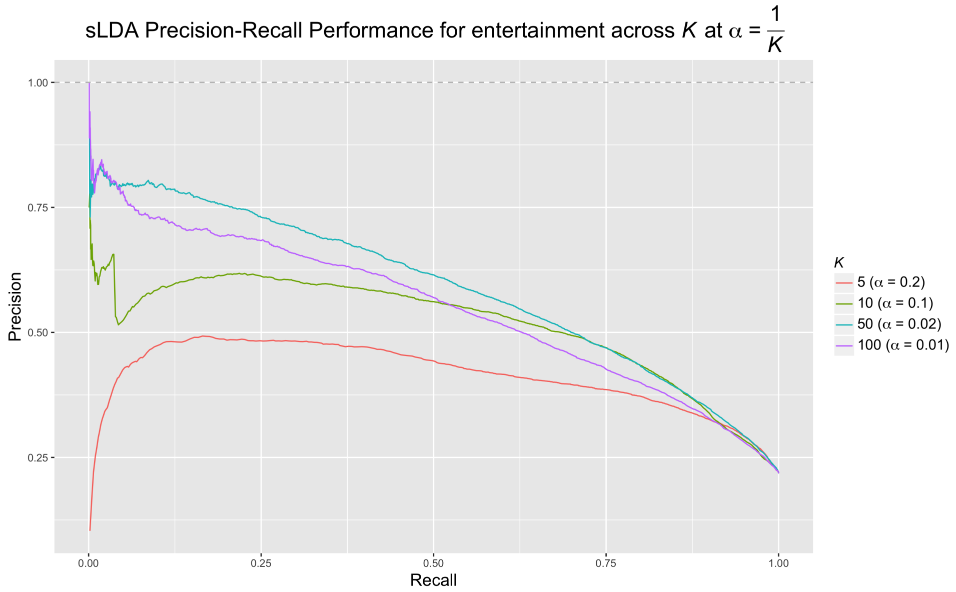


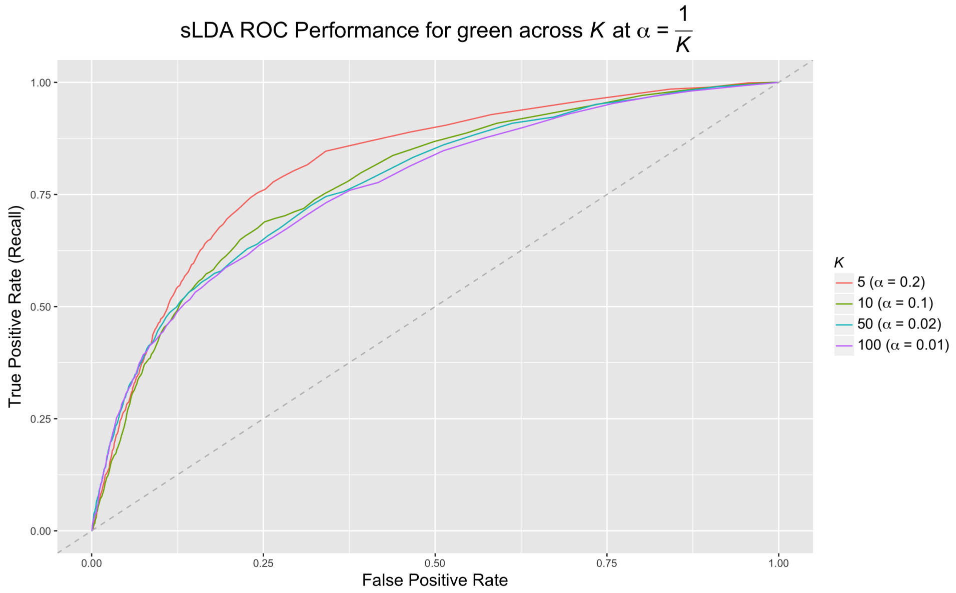


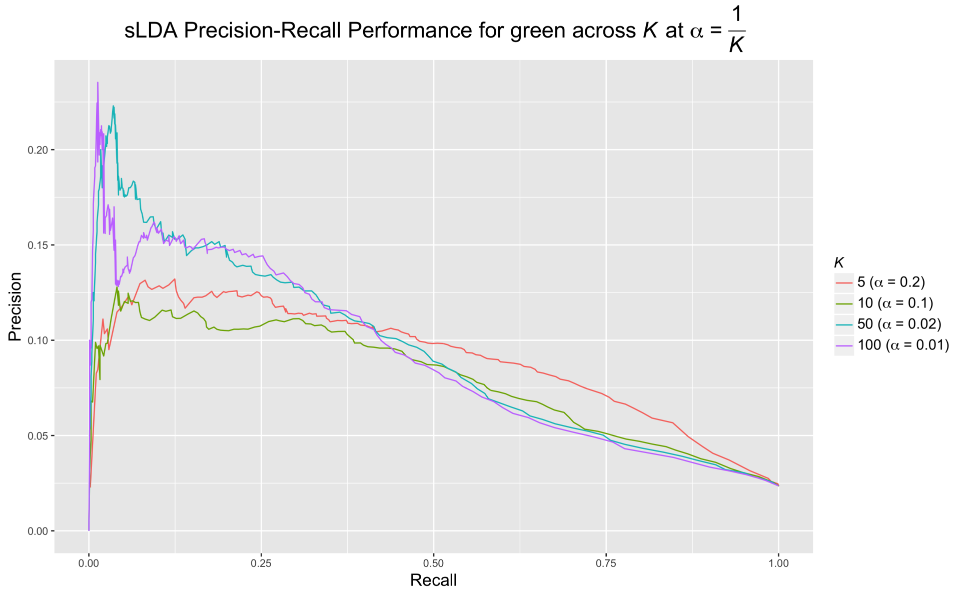

sLDA ROC Performance for business across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

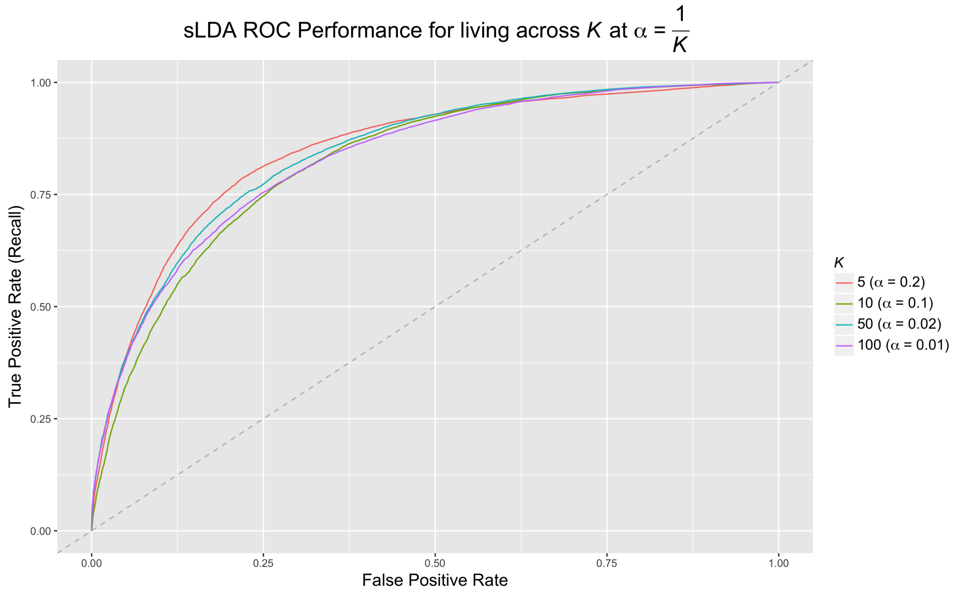

sLDA Precision-Recall Performance for business across K at $\alpha = \frac{1}{K}$
Precision
Recall
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

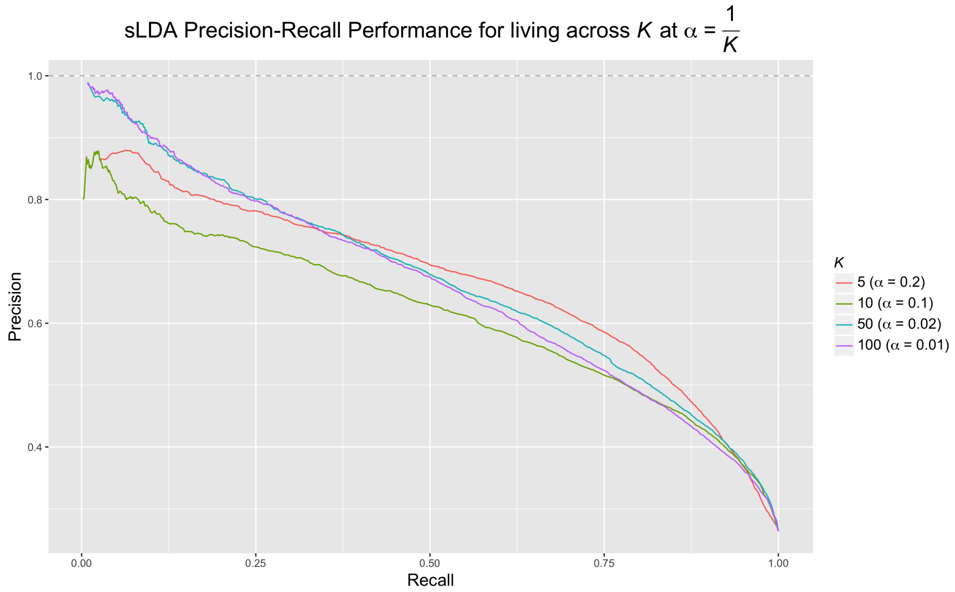

sLDA ROC Performance for entertainment across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

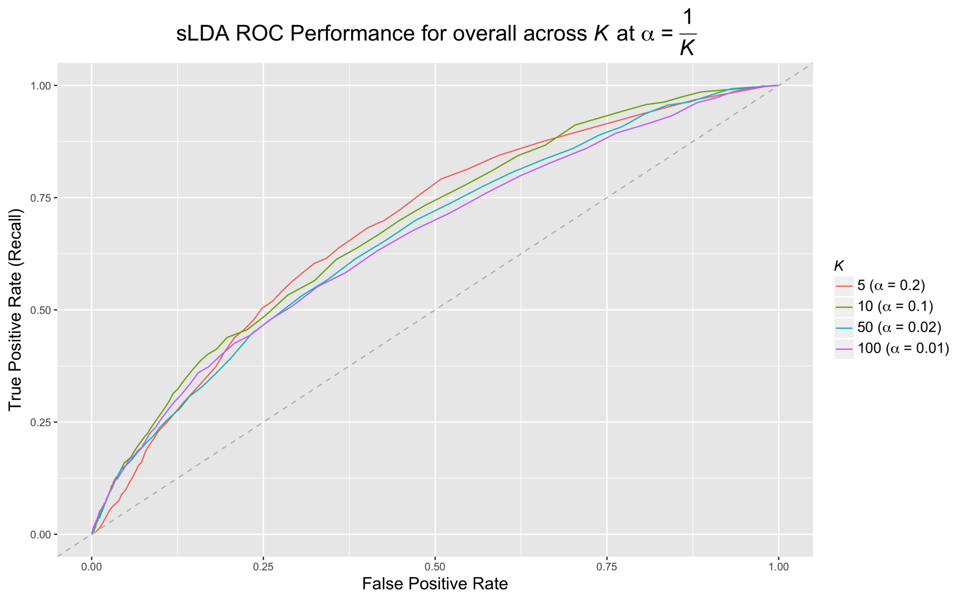

sLDA Precision-Recall Performance for entertainment across K at $\alpha = \frac{1}{K}$
Precision
Recall
0.25
0.50
0.75
1.00
0.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

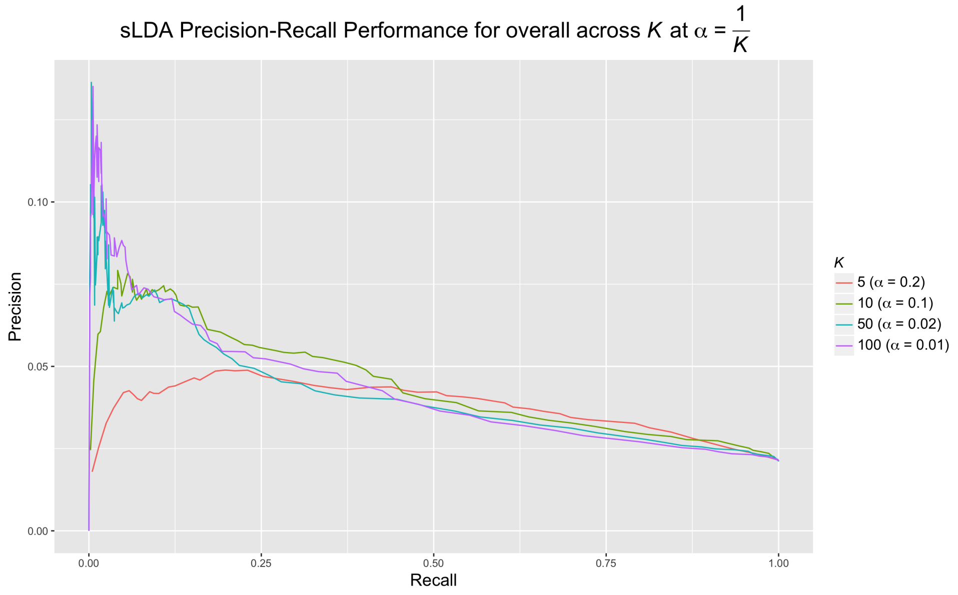

sLDA ROC Performance for green across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

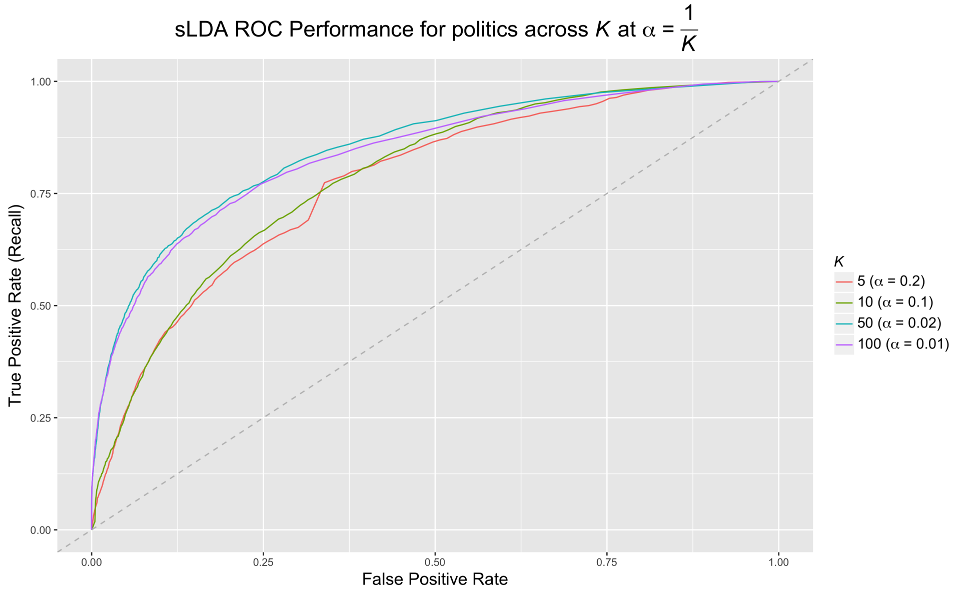

sLDA Precision-Recall Performance for green across K at $\alpha = \frac{1}{K}$
Precision
Recall
0.00
0.05
0.10
0.15
0.20
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

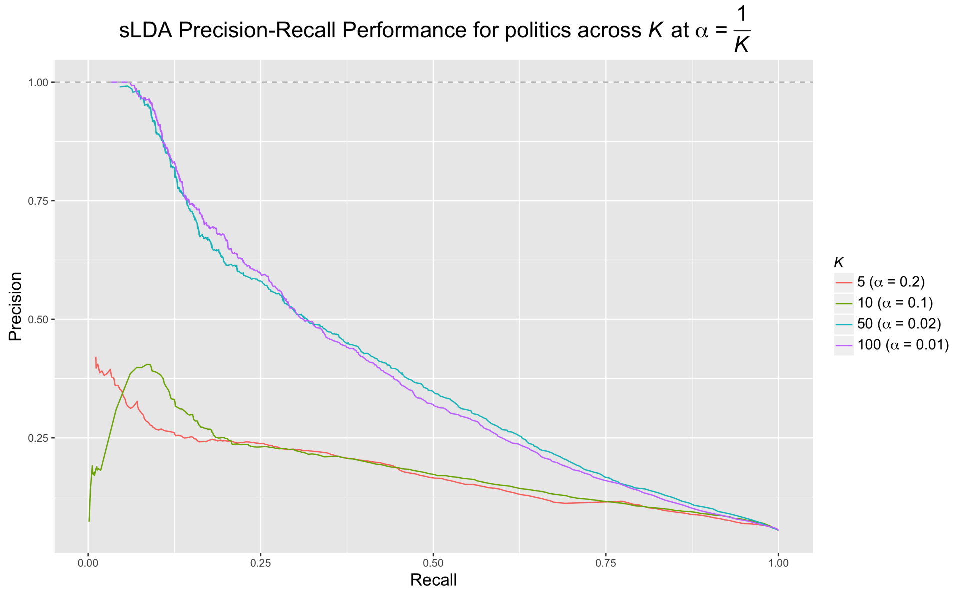

sLDA ROC Performance for living across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
1.00
0.75
0.50
0.25
0.00
False Positive Rate
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

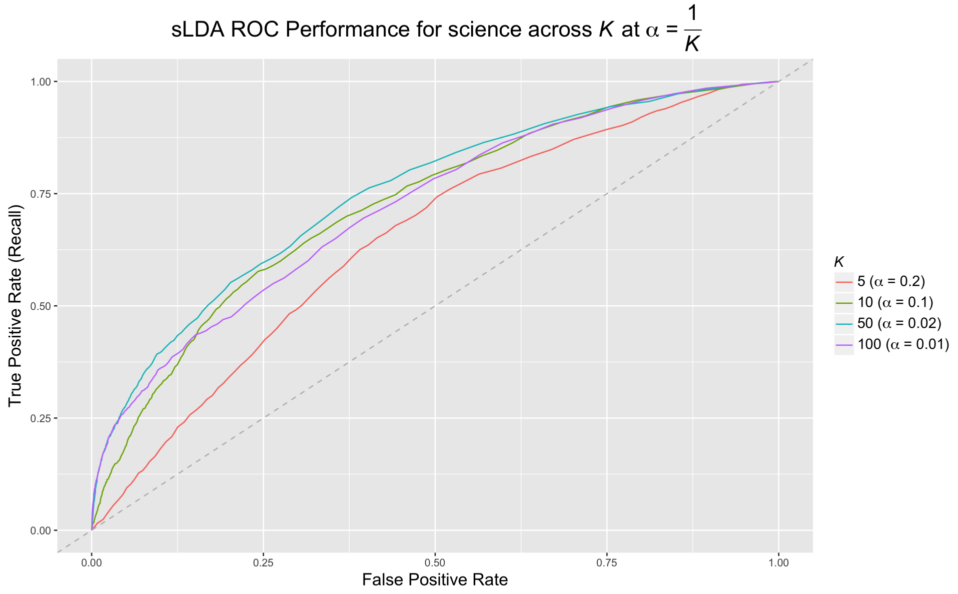

sLDA Precision-Recall Performance for living across K at $\alpha = \frac{1}{K}$
Precision
1.0
0.8
0.6
0.4
Recall
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

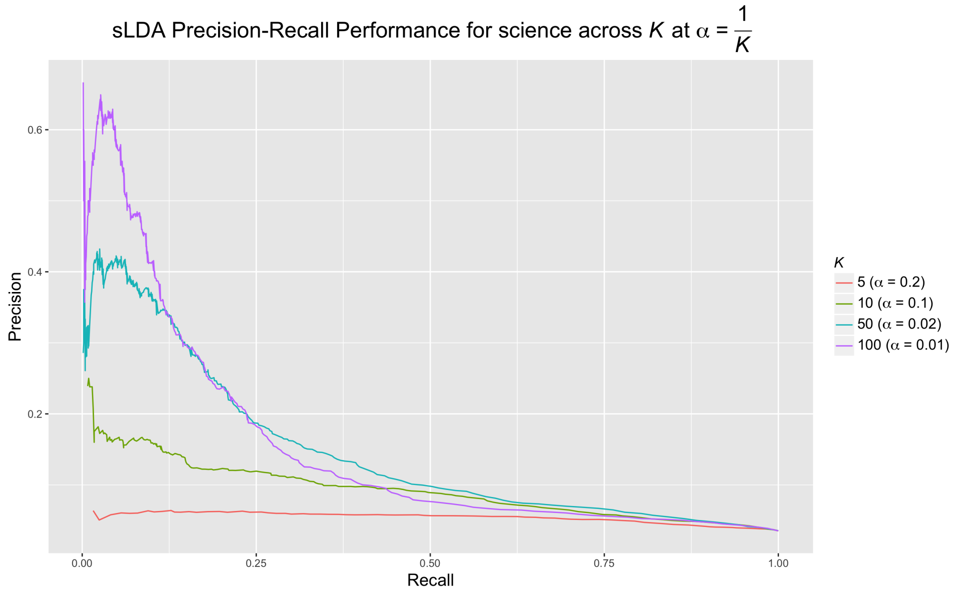

sLDA ROC Performance for overall across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

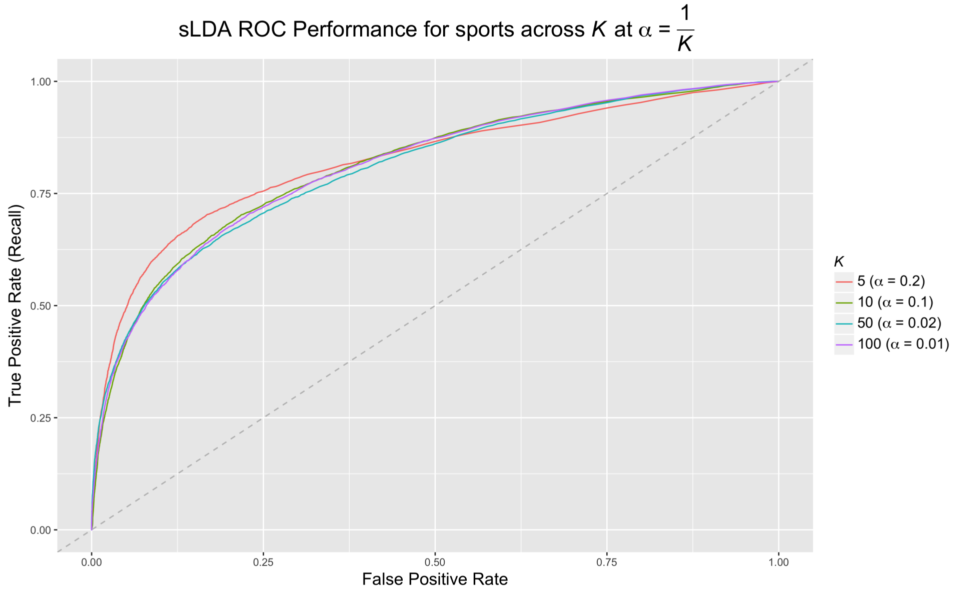

sLDA Precision-Recall Performance for overall across K at $\alpha = \frac{1}{K}$
Precision
Recall
0.00
0.05
0.10
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

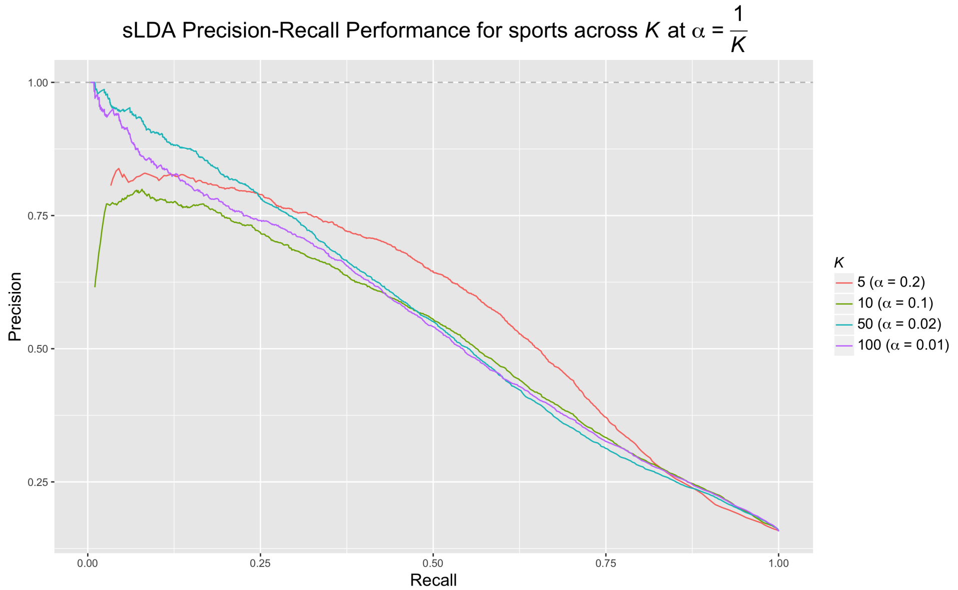

sLDA ROC Performance for politics across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

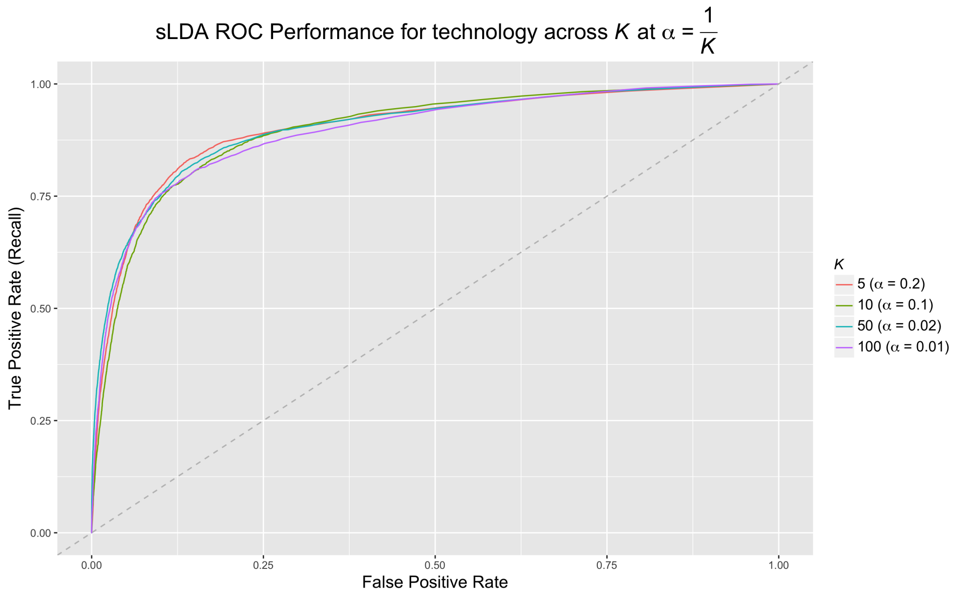

sLDA Precision-Recall Performance for politics across K at $\alpha = \frac{1}{K}$
Precision
Recall
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)


sLDA ROC Performance for science across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)

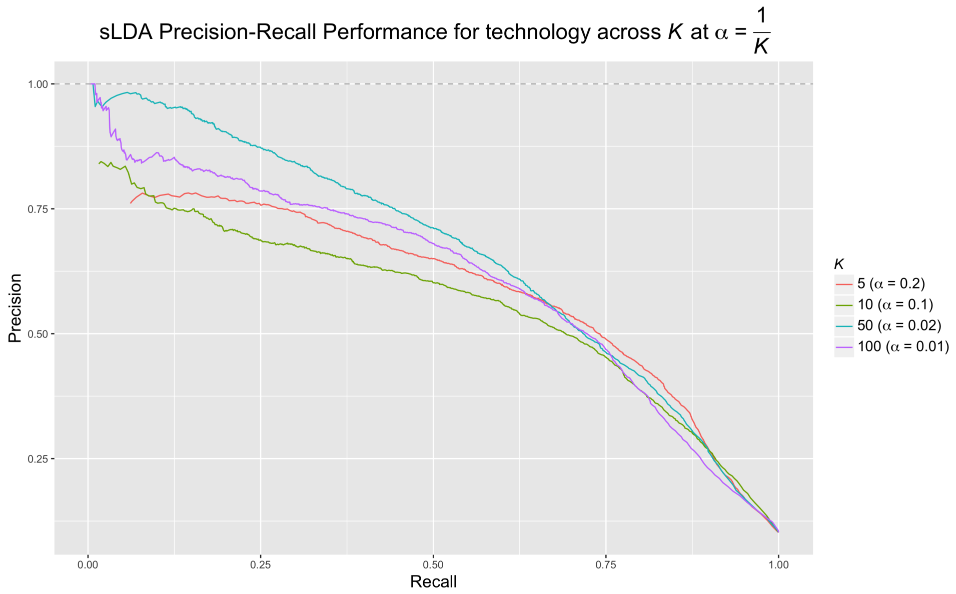

sLDA Precision-Recall Performance for science across K at $\alpha = \frac{1}{K}$
Precision
Recall
0.2
0.4
0.6
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)


sLDA ROC Performance for sports across K at $\alpha = \frac{1}{K}$
True Positive Rate (Recall)
False Positive Rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)


sLDA Precision-Recall Performance for sports across K at $\alpha = \frac{1}{K}$
Precision
Recall
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
K
5 (α = 0.2)
10 (α = 0.1)
50 (α = 0.02)
100 (α = 0.01)



sLDA ROC Performance for technology across K at $\alpha = \frac{1}{K}$



sLDA Precision-Recall Performance for technology across K at $\alpha = \frac{1}{K}$

APPENDIX D

Relevant Code

Relevant code is available at

`https://github.com/RyanRosario/dissertationcode`.

BIBLIOGRAPHY

[1] Charu C Aggarwal and ChengXiang Zhai. *Mining text data.* Springer Science & Business Media, 2012.

[2] Ejaz Ahmed, Michael Jones, and Tim K Marks. An improved deep learning architecture for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3908–3916, 2015.

[3] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9: 1981–2014, 2008.

[4] J. Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 139–177, 1982.

[5] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[6] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 27–34. AUAI Press, 2009.

[7] Ramnath Balasubramanyan, William W Cohen, Doug Pierce, and David P Redlawsk. What pushes their buttons?: predicting comment polarity from the content of political blog posts. In *Proceedings of the workshop on languages in social media*, pages 12–19. Association for Computational Linguistics, 2011.

[8] Michel Ballings and Dirk Van den Poel. *AUC: Threshold independent performance measures for probabilistic classifiers.*, 2013. URL `http://CRAN.R-project.org/package=AUC`. R package version 0.3.0.

[9] Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. Clustering short texts using wikipedia. In *Proceedings of the 30th annual international ACM SIGIR con-*

ference on Research and development in information retrieval, pages 787–788. ACM, 2007.

[10] Adam Bermingham and Alan F Smeaton. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1833–1836. ACM, 2010.

[11] Tim Berners-Lee, Mark Fischetti, and Michael L Foreword By-Dertouzos. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor.* Harper Information, 2000.

[12] Marin Bikić. *Text Classification using Support Vector Machine.* PhD thesis, Fakultet elektrotehnike i računartsva, Svučilište u Zagrebu, 2010.

[13] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python.* O'Reilly, 2009.

[14] István Bíró, Jácint Szabó, and András A Benczúr. Latent dirichlet allocation in web spam filtering. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 29–32. ACM, 2008.

[15] D. Blei and J. Lafferty. Correlated topic models. *Advances in neural information processing systems*, 18:147, 2006.

[16] David Blei. Correlated Topic Model in C. `http://www.cs.princeton.edu/~blei/ctm-c/index.html`, 2007.

[17] David Blei. Latent Dirichlet Allocation in C. `http://www.cs.princeton.edu/~blei/lda-c/`, 2007.

[18] David Blei. Topic models, 2009. URL `http://videolectures.net/mlss09uk_blei_tm/`.

[19] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[20] David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[21] D.M. Blei and J.D. Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

[22] Linda K Börzsei. Makes a meme instead: A concise history of internet memes. *New Media Studies Magazine*, 7, 2013.

[23] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Scene classification via plsa. In *Computer Vision–ECCV 2006*, pages 517–530. Springer, 2006.

[24] Xavier Bouthillier, Kishore Konda, Pascal Vincent, and Roland Memisevic. Dropout as data augmentation. *arXiv preprint arXiv:1506.08700*, 2015.

[25] Jordan Boyd-Graber and Philip Resnik. Holistic sentiment analysis across languages: Multilingual supervised latent dirichlet allocation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 45–55. Association for Computational Linguistics, 2010.

[26] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[27] Kenneth P Burnham and David Anderson. *Model selection and multi-model inference*. Springer-Verlag, 2003.

[28] B. Carpenter. Integrating out multinomial parameters in latent dirichlet allocation and naive bayes for collapsed gibbs sampling. Accessed from `http://lingpipe.files.wordpress.com/2010/07/lda1.pdf` on 2011-11-11, 2010.

[29] Raymond B Cattell. The scree test for the number of factors. *Multivariate behavioral research*, 1(2):245–276, 1966.

[30] J. Chang and D. Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009.

[31] Jonathan Chang. *lda: Collapsed Gibbs sampling methods for topic models.*, 2012. URL `http://CRAN.R-project.org/package=lda`. R package version 1.3.2.

[32] Jonathan Chang and David M Blei. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, 4(1):124–150, 2010.

[33] Carl Chen. The creation and meaning of internet memes in 4chan: Popular internet culture in the age of online digital reproduction. *Habitus Vol. III, Spring*, pages 6–19, 2012.

[34] Vladimir Cherkassky and Yunqian Ma. Selection of meta-parameters for support vector regression. In *International Conference on Artificial Neural Networks*, pages 687–693. Springer, 2002.

[35] Michael R Chernick. *Bootstrap methods: A guide for practitioners and researchers*, volume 619. John Wiley & Sons, 2011.

[36] Wang Chong, David Blei, and Fei-Fei Li. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1903–1910. IEEE, 2009.

[37] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[38] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on World Wide Web*, pages 325–332. ACM, 2002.

[39] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(9):1469–1477, 2015.

[40] Marie Davidian and Thomas A Louis. Why statistics? *Science*, 336(6077):12, 2012.

[41] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics, 2010.

[42] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[43] Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[44] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[45] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[46] Cícero Nogueira Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78, 2014.

[47] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014.

[48] Bradley Efron and Robert Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1993.

[49] D. Egger. Mastering data analysis in excel – lecture 15: How to calculate the area under the roc curve. `http://www.coursera.org/learn/analytics-excel/lecture/RkU17/how-to-calculate-the-area-under-the-roc-curve`, 2014.

[50] Francesca Farina and Fiona Lyddy. The language of text messaging: "linguistic ruin" or resource? *Irish Psychologist*, 37(6):145–149, 2011.

[51] O Feiguina and G Hirst. Authorship attribution for small texts: Literary and forensic experiments. paper presented to the international workshop on plagiarism analysis, authorship identification and near-duplicate detection. In *30th Annual International ACM SIGIR Conference (SIGIR'07)*, 2007.

[52] Thomas Shelburne Ferguson. *A course in large sample theory*, volume 49. Chapman & Hall London, 1996.

[53] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE software*, 29(1):70–75, 2012.

[54] Nicholas I Fisher and Peter Hall. Bootstrap algorithms for small samples. *Journal of statistical planning and inference*, 27(2):157–169, 1991.

[55] Apache Software Foundation. Mahout. `http://mahout.apache.org/`, 2011–2016.

[56] Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, 2012.

[57] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[58] Zhe Gan, Ricardo Henao, David Carlson, and Lawrence Carin. Learning deep sigmoid belief networks with data augmentation. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 268–276, 2015.

[59] Dmitriy Genzel and Eugene Charniak. Entropy rate constancy in text. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 199–206. Association for Computational Linguistics, 2002.

[60] Zoubin Ghahramani, Michael I Jordan, and Ryan P Adams. Tree-structured stick breaking for hierarchical data. In *Advances in Neural Information Processing Systems*, pages 19–27, 2010.

[61] Kevin Gimpel. Modeling topics. *Information Retrieval*, 5:1–23, 2006.

[62] Mark Girolami and Ata Kabán. On an equivalence between plsi and lda. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 433–434. ACM, 2003.

[63] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[64] M. Gonen. *Analyzing Receiver Operating Characteristic Curves with SAS*. SAS Institute, 2015.

[65] Carlos R González and Yaser S Abu-Mostafa. Mismatched training and test distributions can outperform matched ones. *Neural computation*, 2015.

[66] Thomas Gottron and Nedim Lipka. A comparison of language identification approaches on short, query-style texts. In *European Conference on Information Retrieval*, pages 611–614. Springer, 2010.

[67] David M Blei Thomas L Griffiths and Michael I Jordan Joshua B Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, volume 16, page 17. MIT Press, 2004.

[68] Thomas L Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. In *NIPS*, volume 18, pages 475–482, 2005.

[69] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[70] Bettina Grün and Kurt Hornik. topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30, 2011. URL `http://www.jstatsoft.org/v40/i13/`.

[71] Bettina Grün and Kurt Hornik. topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30, 2011. URL `http://www.jstatsoft.org/v40/i13/`.

[72] Peter Hall and Tapabrata Maiti. On parametric bootstrap methods for small area prediction. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(2):221–238, 2006.

[73] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[74] Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John Fisher, and Lars Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 342–350, 2016.

[75] Gregor Heinrich. Parameter estimation for text analysis. Technical report, University of Leipzig, 2008.

[76] Gregor Heinrich. Infinite lda implementing the hdp with minimum code complexity. , Technical Report, arbylon.net, 2011.

[77] Alexander Hinneburg, Heikki Mannila, Samuli Kaislaniemi, Terttu Nevalainen, and Helena Raumolin-Brunberg. How to handle small samples: bootstrap and bayesian methods in the analysis of linguistic change. *Literary and linguistic computing*, 22(2): 137–150, 2007.

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

[79] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1-2):177–196, 2001.

[80] Liangjie Hong and Brian D Davison. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM, 2010.

[81] Xia Hu, Nan Sun, Chao Zhang, and Tat-Seng Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 919–928. ACM, 2009.

[82] Michael C Hughes. Supervised topic models for video activity recognition. Unpublished manuscript, 2010.

[83] Naohiro Ishii, Takeshi Murai, Takahiro Yamada, Yongguang Bao, and Susumu Suzuki. Text classification: combining grouping, lsa and knn vs support vector machine. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 393–400. Springer, 2006.

[84] Aminul Islam and Diana Inkpen. Semantic similarity of short texts. *Recent Advances in Natural Language Processing V*, 309:227–236, 2009.

[85] Navdeep Jaitly and Geoffrey E Hinton. Vocal tract length perturbation (vtlp) improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, 2013.

[86] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 245–251. IEEE, 2013.

[87] Ou Jin, Nathan N Liu, Kai Zhao, Yong Yu, and Qiang Yang. Transferring topical knowledge from auxiliary long texts for short text clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 775–784. ACM, 2011.

[88] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–2015. URL `http://www.scipy.org/`. [Online; accessed 2015-08-15].

[89] Henry F Kaiser. The application of electronic computers to factor analysis. *Educational and psychological measurement*, 1960.

[90] Saurabh S Kataria, Krishnan S Kumar, Rajeev R Rastogi, Prithviraj Sen, and Srinivasan H Sengamedu. Entity disambiguation with hierarchical topic models. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1037–1045. ACM, 2011.

[91] AG Khachaturyan, SV Semenovskaya, and B Vainstein. A statistical-thermodynamic approach to determination of structure amplitude phases. *Soviet physics, crystallography*, 24:519–524, 1979.

[92] Dongwoo Kim, Yohan Jo, Il-Chul Moon, and Alice Oh. Analysis of twitter lists as a potential source for discovering latent characteristics of users. In *ACM CHI workshop on microblogging*, page 4. ACM, 2010.

[93] Sang-Bum Kim, Hee-Cheol Seo, and Hae-Chang Rim. Poisson naive bayes for text classification with feature weighting. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11*, pages 33–40. Association for Computational Linguistics, 2003.

[94] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367. Association for Computational Linguistics, 2004.

[95] Zipf George Kingsley. Selective studies and the principle of relative frequency in language, 1932.

[96] Samuel Kotz, Narayanaswamy Balakrishnan, and Norman L Johnson. *Continuous multivariate distributions, models and applications*. John Wiley & Sons, 2004.

[97] A.D.I. Kramer and K. Chung. Dimensions of self-expression in facebook status updates. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 169–176, 2011.

[98] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[99] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pages 1097–1105. Curran Associates Inc., 2012.

[100] Cherukuri Aswani Kumar and Suripeddi Srinivas. Latent semantic indexing using eigenvalue analysis for efficient information retrieval. *International Journal of Applied Mathematics and Computer Science*, 16(4):551–558, 2006.

[101] Simon Lacoste-Julien, Fei Sha, and Michael I Jordan. Disclda: Discriminative learning for dimensionality reduction and classification. *Advances in Neural Information Processing Systems (NIPS)*, 21, 2008.

[102] T.K. Landauer. *Handbook of Latent Semantic Analysis*. University of Colorado Institute of Cognitive Science Series. Lawrence Erlbaum Associates, 2007.

[103] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[104] Kristina Lerman and Aram Galstyan. Analysis of social voting patterns on digg. In *Proceedings of the first workshop on Online social networks*, pages 7–12. ACM, 2008.

[105] Noah David Levinson. *LOLs, Lulz, and ROFL: The Culture, Fun, and Serious Business of Internet Memes*. PhD thesis, University of Pittsburgh, 2012.

[106] Wei Li and Andrew McCallum. Pachinko allocation: Scalable mixture models of topic correlations. *Journal of Machine Learning Research. Submitted*, 2008.

[107] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159, 2014.

[108] Chenghua Lin, Yulan He, Carlos Pedrinaci, and John Domingue. Feature lda: a supervised topic model for automatic detection of web api documentations from the web. In *International Semantic Web Conference*, pages 328–343. Springer, 2012.

[109] Jack Linshi. Personalizing yelp star ratings: a semantic topic modeling approach. *Yale University*, 2014.

[110] Ramon Lobato, Julian Thomas, and Dan Hunter. Histories of user-generated content: Between formal and informal media economies. *International Journal of Communication*, 5, 2011.

[111] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[112] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. Combining multiple evidence from different types of thesaurus for query expansion. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–197. ACM, 1999.

[113] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[114] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[115] Xian-Ling Mao, Zhao-Yan Ming, Tat-Seng Chua, Si Li, Hongfei Yan, and Xiaoming Li. Sshlda: a semi-supervised hierarchical topic model. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 800–809. Association for Computational Linguistics, 2012.

[116] William F Massy. Principal components regression in exploratory statistical research. *Journal of the American Statistical Association*, 60(309):234–256, 1965.

[117] Jon D Mcauliffe and David M Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.

[118] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Association for the Advancement of Artificial Intelligence (AAAI), 1998.

[119] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. 2002. URL `http://mallet.cs.umass.edu`.

[120] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367. International Machine Learning Society, 1998.

[121] BD McCullough and Hrishikesh Vinod. Implementing the single bootstrap: some computational considerations. *Computational Economics*, 6(1):1–15, 1993.

[122] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[123] D. Mimno, H. Wallach, and A. McCallum. Gibbs sampling for logistic normal topic models with graph-based priors. In *NIPS Workshop on Analyzing Graphs*, 2008.

[124] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[125] Thomas Minka. Estimating a dirichlet distribution. Technical report, MIT, 2000.

[126] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214. ACM, 1998.

[127] Ramesh M Nallapati, Amr Ahmed, Eric P Xing, and William W Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550. ACM, 2008.

[128] Roberto Navigli and Paola Velardi. An analysis of ontology-based query expansion strategies. In *Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*, pages 42–49, 2003.

[129] P.D.R.P. Norm O'Rourke and L. Hatcher. *A Step-by-Step Approach to Using SAS for Factor Analysis and Structural Equation Modeling, Second Edition*. SAS Institute, 2013.

[130] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[131] Ajay S Patil and BV Pawar. Automated classification of web sites using naive bayesian

algorithm. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2012.

[132] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[133] Adler Perotte, Nicholas Bartlett, Noémie Elhadad, and Frank Wood. Hierarchically supervised latent dirichlet allocation. In *Twenty-Fifth Annual Conference on Neural Information Processing Systems*, pages 12–15, 2011.

[134] Saša Petrovic. *Real-time event detection in massive streams.* PhD thesis, University of Edinburgh, 2012.

[135] Phan and Nguyen. Gibbslda++: A c/c++ implementation of latent dirichlet allocation. 2008. URL `http://gibbslda.sourceforge.net/`.

[136] Phan and Nguyen. Jgibblda: A java implementation of latent dirichlet allocation using gibbs sampling for parameter estimation and inference. 2008. URL `http://jgibblda.sourceforge.net/`.

[137] Jim Pitman et al. Combinatorial stochastic processes. Technical Report 621, Technical Report, Department of Statistics, University of California at Berkeley, 2002. Lecture notes for St. Flour course.

[138] John Platt. Probabilities for sv machines. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[139] Nicholas G Polson, Steven L Scott, et al. Data augmentation for support vector machines. *Bayesian Analysis*, 6(1):1–23, 2011.

[140] Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349, 2013.

[141] Jay Pujara and Peter Skomoroch. Large-scale hierarchical topic models. In *NIPS Workshop on Big Learning*, volume 128, 2012.

[142] Duangmanee Putthividhya, Hagai Thomas Attias, and Srikantan S Nagarajan. Supervised topic model for automatic image annotation. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 1894–1897. IEEE, 2010.

[143] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–169. ACM, 1993.

[144] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[145] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.

[146] Sebastian Raschka. Naive bayes and text classification: Introduction and theory. *arXiv preprint arXiv:1410.5329*, 2014.

[147] Manjeet Rege, Cecilia Ovesdotter Alm, and Reynold Bailey. Using word and phrase abbreviation patterns to extract age from twitter microtexts. Master's thesis, Rochester Institute of Technology, 2013.

[148] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. European Language Resources Association, 2010.

[149] J. D. M. Rennie. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning, 2003*, pages 616–623. International Machine Learning Society, 2003.

[150] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[151] Matthias Rychetsky. *Algorithms and architectures for machine learning based on regularized neural networks and support vector approaches*. Shaker, 2001.

[152] Youcef Saad. *Numerical methods for large eigenvalue problems*. Manchester University Press, 1992.

[153] Hassan Saif, Miriam Fernández, and Harith Alani. Automatic stopword generation using contextual semantics for sentiment analysis of twitter. In *CEUR Workshop Proceedings*, volume 1272, 2014.

[154] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[155] Lawrence Saul and Fernando Pereira. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the second conference on empirical methods in natural language processing*, pages 81–89. Association for Computational Linguistics, 1997.

[156] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[157] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition-Volume 2*, page 958. IEEE Computer Society, 2003.

[158] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[159] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642, 2013.

[160] Jeong-Woo Son, Yun-Seok Noh, Hyun-Je Song, and Seong-Bae Park. Location comparison through geographical topics. In *International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 1, pages 311–318. IEEE, 2012.

[161] Yang Song, Jian Huang, Isaac G Councill, Jia Li, and C Lee Giles. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 342–351. ACM, 2007.

[162] Rashish Tandon and Suvrit Sra. Sparse nonnegative matrix approximation: new formulations and algorithms. Technical Report 193, Max Planck Institute for Biological Cybernetics, 2010.

[163] Yee W Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Advances in neural information processing systems*, pages 1353–1360, 2006.

[164] Erik Tromp and Mykola Pechenizkiy. Graph-based n-gram language identification on short texts. In *Proc. 20th Machine Learning conference of Belgium and The Netherlands*, pages 27–34, 2011.

[165] Indiana University. Proc. of the National Academy of Sciences (PNAS) Dataset, 2012. URL `http://iv.slis.indiana.edu/db/pnas.html`.

[166] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[167] Ellen M Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69. Springer-Verlag, 1994.

[168] Hanna M Wallach, David M Mimno, and Andrew McCallum. Rethinking lda: Why priors matter. In *NIPS*, volume 22, pages 1973–1981, 2009.

[169] Hanna Megan Wallach. *Structured topic models for language.* PhD thesis, University of Cambridge, 2008.

[170] Jinqiao Wang, Lingyu Duan, Lei Xu, Hanqing Lu, and Jesse S Jin. Tv ad video categorization with probabilistic latent concept learning. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 217–226. ACM, 2007.

[171] Xiaogang Wang and Eric Grimson. Spatial latent dirichlet allocation. *Advances in neural information processing systems*, 20:1577–1584, 2007.

[172] Yang Wang and Greg Mori. Max-margin latent dirichlet allocation for image classification and annotation. In *British Machine Vision Conference (BMVC)*, 2011.

[173] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge University Press, 1994.

[174] Max Welling, Chaitanya Chemudugunta, and Nathan Sutter. Deterministic latent variable models and their pitfalls. In *SIAM International Conference on Data Mining*, 2008.

[175] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[176] Tim Weninger, Yonatan Bisk, and Jiawei Han. Document-topic hierarchies from document graphs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 635–644. ACM, 2012.

[177] Mike West and Michael D Escobar. *Hierarchical priors and mixture models, with application in regression and density estimation.* Institute of Statistics and Decision Sciences, Duke University, 1993.

[178] Xiaohui Wu, Jun Yan, Ning Liu, Shuicheng Yan, Ying Chen, and Zheng Chen. Probabilistic latent semantic user segmentation for behavioral targeted advertising. In *Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising*, pages 10–17. ACM, 2009.

[179] Jinxi Xu and W Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11. ACM, 1996.

[180] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456. ACM, 2013.

[181] Tae Yano and Moonyoung Kang. Taking advantage of wikipedia in natural language processing. Technical report, Carnegie Mellon University Language Technologies Institute, 2016.

[182] H Yu, C Ho, Y Juan, and C Lin. Libshorttext: a library for short-text classification and analysis. Technical report, National Taiwan University, 2013. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/libshorttext.pdf`.

[183] Matthew Zeiler and Robert Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2013.

[184] Haizheng Zhang, Baojun Qiu, C Lee Giles, Henry C Foley, and John Yen. An lda-based community structure discovery approach for large-scale social networks. In *Intelligence and Security Informatics, 2007 IEEE*, pages 200–207. IEEE, 2007.

[185] Lingsong Zhang, JS Marron, Haipeng Shen, and Zhengyuan Zhu. Singular value decomposition and its visualization. *Journal of Computational and Graphical Statistics*, 16(4):833–854, 2007.

[186] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, 38(3):2758–2765, 2011.

[187] Jun Zhu, Amr Ahmed, and Eric P Xing. Medlda: maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1257–1264. ACM, 2009.