TOPIC MODEL DEPLOYMENT FRAMEWORK AND GIBBS SAMPLING-BASED QUADRATIC TOPICS APPROXIMATION METHOD FOR LARGE TEXT ANALYTICS

GEOFFREY MARIGA WAMBUGU

DOCTOR OF PHILOSOPHY

(Information Technology)

JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

2019

Topic Model Deployment Framework and Gibbs Sampling-Based Quadratic Topics Approximation Method for Large Text Analytics

Geoffrey Mariga Wambugu

A thesis submitted in partial fulfilment for the Degree of Doctor of Philosophy in Information Technology in the Jomo Kenyatta University of Agriculture and Technology

2019

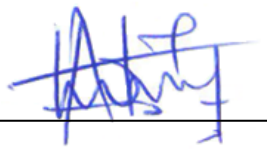
DECLARATION

This thesis is my original work and has not been presented for a degree in any other university.

Signature: _____ Date: _____

Geoffrey Mariga Wambugu

This thesis has been submitted for examination with our approval as University Supervisors.

Signature:  Date: 17/7/2019

Dr. George Okeyo, PhD

JKUAT, Kenya

Signature: _____ Date: _____

Prof. Stephen Kimani, PhD

JKUAT, Kenya

DEDICATION

Dedicated to:

My sweetheart Stella and Children: Joyce, Precious, Christine, Naomi and Esther,

You have made my life a very pleasant and exciting experience.

My mother Joyce,

you are such a wonderful mentor.

My siblings, Mbari ya Mbogo and Paul Gachoya Waweru,

'Ngai amuhingurire mirango ya irathimo'

and

In memory of my father Joseph,

I have always missed your wise counsel.

ACKNOWLEDGEMENT

Thanks to the Almighty God for granting me sufficient grace, peace of mind, strength and good health to accomplish this thesis.

My deepest gratitude goes to Dr. George Okeyo and Prof. Stephen Kimani for inspiring, encouraging and travelling with me in this academic journey. I have been fortunate to have supervisors who taught me to explore my own ideas, were patient and gave guidance and numerous feedbacks on my work. I am grateful for the lengthy discussions that helped to shape and complete this work.

To Dr. James Ciera, Dr. Calvins Otieno and Dr. Anthony Wanjoya, I appreciate the role you played in the formative stages of this study and continued consultation. Prof. Waweru Mwangi, Dr. Michael Kimwele, Dr. Richard Rimiru and the entire SCIT fraternity at JKUAT, thank you for the encouragement and your wise words.

My close friend and colleague Prof. Geoffrey Muchiri, thank you for your time, support, feedback on my work and encouragement. Stephen Obonyo, my research assistant, thank you for the effort. My classmates: Mohamed, Clive, Stephen, Charity and Cornelius, thank you for your friendship, encouragement and support.

I am extremely grateful to all my family members for their love, prayers and support, especially my dearest wife, Stella, for bearing with me during this journey.

MAY GOD BLESS YOU ALL

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS AND ACRONYMS

AD-LDA	Approximate Distributed LDA
ANN	Artificial Neural Network
API	Application Programming Interface
BI&A	Business intelligence and analytics
BNP	Bayesian Nonparametric
BPA	Back Propagation Algorithm
CRP	Chinese Restaurant Process
CTM	Correlated Topic Model
CVB	Collapsed Variational Bayes
DTM	Dynamic Topic Models
GMM	Gaussian Mixture Model
GNU	General Public License.
GPU	Graphics Processing Unit
TMDF	Topic Model Deployment Framework
HD-LDA	Hierarchical Distributed LDA
IBM	International Business Machines
IBP	Indian Buffet Process
iid	independent and identically distributed
IR	Information Retrieval
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSVB	Latent Space Variational Bayes
MCMC	Markov Chain Monte-Carlo
ML	Machine Learning

MPI	Message Passing Interface
NLP	Natural Language Processing
PLSI	Probabilistic Latent Semantic Indexing
QTAM	Quadratic Topics Approximation Method
SMS	Short Messaging Service
SVM	Support Vector Machine
TOT	Topic Over Time
VB	Variational Bayes

ABSTRACT

Probabilistic topic modelling provides computational methods for text analytics. Latent Dirichlet Allocation (LDA) is a popular Bayesian model that aids in discovering hidden thematic structure in large text datasets. Application of LDA requires that Dirichlet prior parameters alpha and beta, and number of topics be specified. The performance of many machine learning methods depends critically on parameter settings. Currently, parameter estimation is based on Markov Chain Monte-Carlo algorithms that must be run for many iterations before convergence. Further, deployment of topic models require definition of constituent constructs and their relationships. This thesis consists of two parts. In the first part, a topic model deployment framework (TMDF) is developed. TMDF identifies all components that are required for deployment of topic models. The second part of the thesis deals with the development of Quadratic Topics Approximation Method (QTAM) for fast estimation of optimal number of topics input parameter. QTAM models minimum perplexity against number of topics for any given dataset. Both the framework and the method are validated for application in large unstructured text data analytics. A python experimental environment was set up in the Google cloud compute engine's custom machine and then used to study parameter behaviour by manipulating selected existing datasets. Asymptotic time analyses as well as descriptive statistics were used to validate QTAM. A critical review of literature and expert opinion survey were used to identify, develop and validate TMDF. Results indicate improvement in inferential speed for the number of topics (K) and hyper parameter alpha thereby enhancing LDA application. This implies that QTAM is more efficient in estimating number of topics parameter than other existing methods currently being used for developing text applications, especially those that have large data processing requirements. Validation results also indicate that TMDF can be used in the deployment of topic models. Further study may be carried out to determine the best combination of pre-processing steps and topic model parameters that were outside the scope of this thesis but otherwise related. Another direction for further study is to conduct replica experiments to further enhance the results of the new method with the aim of setting it as an industry standard.

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

This chapter provides the foundation for this thesis by outlining our research problem, objectives, justification and contributions of the study. The main aim of the study was to develop a topic model deployment framework and parameter inference methodology for application in large unstructured text data analytics. The chapter concludes by providing an outline for the thesis.

Technologies of processing information have progressed over the last couple of years from large mainframes to personal computers to mobile devices to cloud computing. This has brought an information overflow, with transformative societal implications that affect all aspects of human life. A significant portion of this information is in the form of text data, such as books, news articles, microblogs and instant messages (Puurula, 2016). These vast quantities of text data can only be accessed and utilized using information technology devices, but the automated processing of text is only possible using technology specialized for human language. Text analytics in a broad sense refers to technology that allows the utilization of large quantities of text data, among them topic models. Tated and Ghonge (2015) note that although extracting useful information from texts is not an easy task, it is a need of this modern life to have business intelligence tools which are able to extract useful information as quickly as possible and at a low cost.

Increased prevalence of text analysis has given people an ability to access, analyse and process vast amounts of textual information in a fraction of a second and have yielded innovations that help people better understand and make use of the information in document repositories. The development of new technologies to tackle problems such as topic detection, tracking, and trending, where a machine automatically identifies emergent topics in a text corpus, is bound to have wide application in the future. Such applications can be found in universal consumer-

based applications as well as systems focused on banking and finance, health care, aerospace, manufacturing, and the natural sciences (Srivastava & Sahami, 2009).

Turning user generated large text datasets content into actionable information requires using computational techniques to show trends and patterns within and between these extremely large datasets (Letouze, 2012). Probabilistic models provide automatic text data analysis methods for classifying, filtering, organising, understanding, searching, summarizing and explaining large data sets. A latent topic model is a probabilistic model that encodes hidden patterns in data. We uncover these patterns from their conditional distribution and use them to summarize data and form predictions. Latent variable models are important in many fields, including computational biology, natural language processing, and social network analysis (Blei, 2014).

The basic idea behind a topic model can be stated as follows: (1) Take a big corpus of text data and uncover hidden topical patterns; (2) Annotate the documents according to the topics; (3) Use the annotations to organise and summarize. Several tools have been developed in the past to address the identified existing challenges for large text data. In text analytics, if the researcher knows the categories beforehand, automated methods can minimize the amount of labour needed to analyse documents. Dictionary methods use the frequency of key words to determine a document's class. One way to improve upon dictionaries are supervised methods. These methods begin with human hand coding of documents into a predetermined set of categories. The human hand coding is then used to train, or supervise statistical models to classify the remaining text documents (Grimmer & Stewart, 2013). But the performance of any one classifier can vary substantially across contexts so validation of a classifier's accuracy is essential to establish the reliability of supervised learning methods.

Classification is an important data mining technique used in many fields to classify data of various kinds. In this technique, a training data is used in order to construct a classification model, which relates the features in the underlying records to one of

the class labels (Sravani & Srinivasu, 2014). The trained model is then used to predict a label for any unknown class.

The problem of classification has been widely studied in the database, data mining, and information retrieval communities. As problems and applications are numerous in this area, there is no single algorithm that is better than the others on all the problems (Sravani & Srinivasu, 2014). Some of the applications of text classification are in News filtering and Organization (Benkhelifa & Laallam, 2016), Document Organization and Retrieval (Abdullah & Zamil, 2018), Opinion Mining (Mahendran et al., 2013, Kang et al., 2018), Email Classification and Spam Filtering (Kobayashi et al., 2017).

A wide variety of techniques have been designed for text analytics. This study highlights the following broad classes of techniques: Decision Trees; Pattern Rule-based; Support Vector Machines (SVM); Neural Networks; Generative Bayesian approaches and other techniques like nearest neighbours, and genetic algorithm-based techniques.

Decision tree is a classifier that is expressed as a recursive partition of the instance space. It creates a predictive model, which maps observations about a node to conclusions about the nodes' target value. In a tree structure leaves represent the class labels and branches represent conjunctions of feature leading to the class labels. Sumbaly et al., (2014) applied decision trees mining technique in diagnosis of Breast Cancer. According to Luo et al., (2014) decision tree algorithms have the advantages of high speed, high accuracy, are easily understood and make no assumption on distribution. However, classical decision tree classification algorithms can only deal with completely structured data and hence when they are used for unstructured data it is important to convert the data into structured data. The converted structured data tends to be sparse with high dimensionality leading to low algorithm efficiency.

Bayesian approaches fit a probability model to a set of data and summarize the results by a probability distribution on the parameters of the model and on unobserved quantities such as predictions for new observations (Gelman et al.,

2014). The practical disadvantage of this approach is that it requires computation of analytically intractable integrations over variables. As a result, much contemporary research in Bayesian approaches to machine learning relies on, or is directly concerned with, approximation techniques (Tipping & Bishop 2006).

The most widely used posterior inference methods in Bayesian nonparametric models are Markov Chain Monte Carlo (MCMC) methods. MCMC methods are guaranteed to converge to the posterior with enough samples but have two drawbacks: (1) the samplers must be run for many iterations before convergence making them slow and challenged in large dataset applications. (2) It is difficult to assess convergence. Many researchers fix the number of iterations heuristically (Gershman & Blei, 2012).

Gershman and Blei (2012) identifies the following software packages for approximating the posterior when implementing various Bayesian nonparametric models: Chinese Restaurant Process (CRP) mixture model implementing MCMC inference in Matlab and R and CRP mixture model implementing variational inference in Matlab; Indian Buffet Process (IBP) latent factor model implementing both MCMC and variational inference in Matlab.

Topic models are a class of generative Bayesian models that can be used to summarize text corpus into topics and see how the topics change with time. A lot of research in the area of Topic Modeling has been carried out, pioneered by Hofmann, (1999). Blei et al., (2003) developed Latent Dirichlet Allocation (LDA), a generative topic modeling technique referred to by Roberts et al., (2014) as a prominent example in the field of text data analysis. According to Wang & Blei, (2012) generative probabilistic modeling treats data as arising from a generative process that includes hidden variables. This generative process defines a joint probability distribution over both the observed and hidden random variables. Data analysis is performed by using that joint distribution to compute the conditional distribution of the hidden random variables given the observed variables. This conditional distribution is also called the posterior distribution. Probabilistic topic model

algorithms aims at discovering the hidden thematic structure in large archives of documents (Wang & Blei, 2012).

Bayesian nonparametric (BNP) models have emerged as an important tool for building probability models with flexible latent structure and complexity. Posterior inference in BNP models is intractable and posterior must be approximated (Wang & Blei, 2012).

1.2 Problem Statement

Popular probabilistic methods such as Bayesian statistics are used to build models that analyse vast amounts of data. Inference and learning mechanisms in such models usually rely on Markov Chain Monte-Carlo (MCMC) based algorithms that spend a lot of time when computing parameters, thereby rendering the process of text analytics slow. The said models are therefore challenged for application in analysing large datasets. Moreover, there is no framework that comprehensively defines deployment of topic models in terms of constituent constructs and their relationships. These two gaps, namely, slow parameter computation time and lack of a deployment framework, have not been addressed so far.

1.3 Research Objectives

1.3.1 Main objective

The main objective of this research was to develop a topic model deployment framework and parameter inference method for application in large unstructured text data analytics.

1.3.2 Specific objectives

To achieve the main objective, the research was guided by the following specific objectives:

i) To investigate topic model's deployment components and inference algorithms with the aim of extending them for application in large text data analytics.

ii) To develop a topic model deployment framework that serves as a blue print for large text analytics.

iii) To implement topic model and evaluate parameter trends on different text datasets with the aim of developing a faster method of selecting optimal values for use in applications.

iv) To validate the developed topic model deployment framework and fast parameter approximation method.

1.4 Research Questions

To achieve the stated objectives, the study sought to answer the following research questions:

i) What are the generic components and implementation algorithms of a topic model?

ii) How do you develop a topic model deployment framework that serves as a blue print for large text analytics?

iii) How do you implement a topic model, evaluate parameter trends on different text datasets and develop a fast parameter estimation method?

iv) Is the developed topic model deployment framework and topic model parameter approximation method be valid?

1.5 Justification

Analysis of large datasets requires huge memory and fast processing speed. Despite the fact that faster hardware and larger memory chips are continually appearing in the market, increases in hardware speed and computer memory are outstripped by increases in our ambitions. For example, even allowing for good advances in hardware speed, supremely efficient algorithms will be needed if advancement of state-of-the-art in realistic systems development is to be attained (Zhang et al., 2014). A research therefore aimed at improving the efficiency of an algorithm will always find its place, application and appreciation within the research community and practitioners.

Computing research scholars have been developing topic models to analyse large text datasets. As noted in section 1.3, most of the models rely on Makov Chain Monte Carlo (MCMC) which is slow and therefore challenged when applied in the context of large datasets analytics. This raises a practical motivation for fast approximate Bayesian approaches that bypass MCMC while maintaining the benefits of Bayesian analysis. This thesis developed a fast approximate Bayesian approach to enhance parameter inference for Latent Dirichlet Allocation (LDA) topic model.

There are many factors that come into play in deployment of popularly used algorithms which includes readability, extensibility, portability, reusability, ease of use and efficiency. The efficiency of an algorithm depends on its use of resources, such as: the time it takes the algorithm to execute; the memory it uses for its variables; the network traffic it generates; the number of disk accesses it makes. This study mainly focused on time complexity of existing algorithms. However, it is should be noted that trade-offs are often necessary. An algorithm may save time by using more space; or it may reduce disk accesses by using more main memory. Efficiency often also conflicts with other criteria such as readability and extensibility.

1.6 Contributions of the Thesis

This thesis has made the following contributions:

a) It developed a generic topic model deployment framework.

b) It developed the Quadratic Topics Approximation Method to be used in estimating number of topics (K) LDA model input parameter.

c) It implemented a tool based on QTAM for automating the process of finding number of topics (K) LDA model input parameter.

d) It provided theoretical and empirical evidence that QTAM is valid. Comparative run time analysis was used to compare QTAM against the Collapsed Gibbs Sampling (CGS) which was chosen as the baseline model since it converges to the true posterior when given sufficiently many iterations.

1.7 Thesis Structure

This thesis is divided into seven chapters. Chapter one presents the background of research, outlines the research objectives and research questions and provides a justification and contributions of this study.

Chapter two first reviews probabilistic topic models, introduces the basic concepts of topic modelling and the modelling process of Latent semantic indexing (LSI), Probabilistic Latent Semantic Indexing (PLSI) and Latent Dirichlet Allocation (LDA) in details, analyzes and compares their advantages and disadvantages. The chapter then reviews three main inference algorithms of LDA model parameter, such as variational inference, belief propagation and Gibbs sampling. Next, it introduces the basic concepts, main steps and parameter settings of genetic algorithms. Finally, it reviews some python frameworks like GraphLabs and SCIKIT learn which contains latent topic modelling libraries used in this study.

Chapter three is the methodology. This chapter presents research methods and materials used to define and validate generic topic model deployment framework, implement LDA model, and develop and validate the QTAM approach.

Chapter four identifies the major components of topic model deployment and gives detailed description of each component. A framework consisting of LDA implementation constructs is presented inform of a connected logical network diagram. The chapter also presents validation methodology, results and interpretation for the developed Topic Model Deployment Framework (TMDF).

Chapter five presents the experimental dataset, environment and results of model parameter trending and modelling. The chapter then presents a design of Quadratic Topics Approximation Method (QTAM) for inferring the best LDA input parameter number of topics (K). The chapter further presents validation of the developed Quadratic Topics Approximation Method (QTAM). For this method, a comparative theoretical worst case time complexity of CGS and QTAM inference for number of topics (K) is presented followed by comparative experimental real run-time analysis of the proposed QTAM algorithm against existing CGS.

Chapter six presents the conclusion of the thesis. It revisits the thesis research questions and recommends possible future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter starts by an overview of Topic Modelling (TM) where the following sections are covered: definition and history of TM, fields related to TM –Information Retrieval. The chapter further describes basic machine learning concepts that provide the tools required for defining and modelling analysis of large text data. The model focussed in this study, Latent Dirichlet Allocation (LDA), was presented as an information retrieval model (Blei et al., 2003), solvable by machine learning techniques. Topic models can also be seen as classical text mining or natural language processing tools, and are therefore these concepts are described as well.

2.2 Overview of Topic Modelling

2.2.1 Defining Topic Model

It is important to describe the definition of topic modelling since it is the main subject of this study. David Blei, an accomplished scholar in Statistics and Computer Science and the author of the Latent Dirichlet Allocation (LDA) defined a topic model in Blei (2012) as follows:

Definition (Topic Model):

> *A topic model is a suite of algorithms that uncover the hidden thematic structure in document collections.*

Blei (2012) further notes that the said algorithms help in the development of new ways to search, browse and summarize large archives of texts.

In topic modelling, a topic is a list of statistically significantly occurring words. A text can be a collection of news articles, an email or a collection of emails, a blog post, a book or book chapter, a journal article or any other kind of unstructured text. Being mathematical, topic models cannot understand the meaning of words in text

documents, word concepts and context. Instead, they suppose that any part of the text is obtained by selecting words from probable collections of words which we can call '*baskets of words*' where each basket corresponds to a topic. The model iterates over this process many times until the most probable distribution of words into baskets is obtained. This most probable distribution is called topics. Topic modelling can provide a useful summary of a large collection of documents in terms of the collection as a whole, the individual documents, and the relationships between the documents (Jelodar et al., 2017).

2.2.2 History of Topic Model

The latent semantic indexing (LSI) (Deerwester et al., 1990) formed the origin of topic modelling and have served as the basis for the development of a topic models. However, LSI is not a probabilistic model and therefore, it is not an authentic topic model. Based on LSI, probabilistic latent semantic analysis (PLSA) a genuine topic model, was developed by Hofmann (1999). Published after PLSA, latent Dirichlet. allocation (LDA) (Blei et al., 2003) is an even more complete probabilistic generative model and is the extension of PLSA. Nowadays, there is a growing number of probabilistic models that are based on LDA via combination with particular tasks.

In the history of Topic Modelling, Alghamdi & Alfalqi, (2015) identifies most popular models as follows: latent semantic analysis (LSA), probabilistic latent semantic analysis (PLSA), Latent Dirichlet Allocation (LDA), and Correlated topic model (CTM). Other topic models are topic over time (TOT), dynamic topic models (DTM), multi scale topic tomography, dynamic topic correlation detection and detecting topic evolution in scientific literature.

2.2.3 Fields Related to Topic Modelling

This section of thesis presents short introductions to information retrieval, natural language processing and text analytics, since they are closely related and sometimes overlap in their goals and tools.

a) Information Retrieval

Consider a collection of hundred thousand or more digital text documents, for example books in soft copy. Suppose you want to make these data accessible to a normal person through a computer interface. Ideally, when a user enters a few words that describe what s/he is looking for, the relevant results should be displayed in a ranked order immediately after submitting a querye. This serves as an entry point to the full texts. Also, the user should be notified when new documents that are similar to his or her previous results are added to the collection, and furthermore they should also be able to browse the data guided by a meaningful structure instead of just filtering it.

The following problems would be faced in designing and implementing a system according to specifications listed above: (1) How to match the meaning of a query to that of a relevant text document, (2) Consider the case of synonyms (different words with the same meaning) and homographs (same spelling, different meaning), (3) How to rank search results, and (4) How to find similar documents or how to process terabytes of data when the user expects instant results.

These and related questions are addressed by the computer science of information retrieval (IR), which deals with "searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web". The history of the research and development of information retrieval (IR) systems started with the creation of electro-mechanical searching devices, through to the early adoption of computers to search for items that are relevant to a user's query. An IR system typically searches in collections of unstructured or semi-structured data (e.g. web pages, documents, images, video, etc.). The need for an IR system occurs when a collection reaches a size where traditional cataloguing techniques can no longer cope. Similar to Moore's law of continual processor speed increase, there has been a consistent doubling in digital storage capacity every two years (Sanderson & Croft, 2012). Since then the tasks computers are expected to carry out for us has become an important aspect, most prominently shown by our everyday use of web search engines. The main goals of IR

may thus be reduced to enabling adhoc retrieval, filtering of new documents in a collection, based on a user profile and browsing.

b) Natural Language Processing

Natural Language Processing (NLP) is the practical field of Computational Linguistics (Moreno & Redondo, 2016). Sometimes NLP has been considered a sub-discipline of Artificial Intelligence, and more recently it sits at the core of Cognitive Computing, since most cognitive processes are either understood or generated as natural language utterances. NLP is a very broad topic, and includes a huge amount of subdivisions: Natural Language Understanding, Natural Language Generation, Knowledge Base building, Dialogue Management Systems (and Intelligent Tutor Systems in academic learning systems), Speech Processing, Data Mining – Text Mining (Text Analytics), and so on. Natural language processing (NLP) methods can be used to analyze texts in the pre-processing step of text analytics process, in order to better represent the text and extract more from its content. This thesis focuses specifically in Text Analytics (TA) and uses NLP methods for data pre-processing (Khurana et al., 2017).

c) Text Analytics

Text Analytics (TA) originates from several earlier research fields, such as data mining, machine learning, information retrieval and natural language processing. Like these fields, TA has a foundation in computer science, with considerable influence from applied artificial intelligence (Fayyad et al., 1996; Witten, 2004).

Text Analytics has gained a lot of popularity probably because unstructured free text accounts for 80% of data that aid decision in a business context, including tweets, blogs, wikis and surveys (Moreno & Redondo, 2016). Text Analytics is the discovery of new, previously unknown information, by automatically extracting information from different written resources (Gaikwad, 2014). Powerful trends in social media, e-discovery, customer services (call center transcriptions of voice calls, customer complaint emails and instant messaging) and customer centric business strategies are

driving IT leaders to consider text analytics as a powerful business tool (Grimes, 2011).

Text Analytics is an extension of data mining, which tries to find textual patterns from large non-structured sources, as opposed to data stored in relational databases. Text Analytics, also known as Intelligent Text Analysis, Text Data Mining, Information Extraction, Opinion Mining or Knowledge-Discovery in Text (KDT), refers generally to the process of extracting non-trivial information and knowledge from unstructured text (Alwidian et al., 2015; Moreno & Redondo, 2016). Text Analytics can cover unstructured or semi-structured data sets such as emails, full-text documents and HTML files, blogs, newspaper articles, academic papers, etc. Text Analytics is an interdisciplinary field which draws on information extraction, data mining, machine learning, statistics and computational linguistics.

The terms “text analytics” and “text mining” are largely interchangeable. They name the same set of methods, software tools, and applications. Their distinction stems primarily from the background of the person using either of the term with “text mining” mostly being used by data miners, and “text analytics” by individuals and organizations in domains where the road to insight was paved by business intelligence tools and methods.

The three typical steps in text analytics include: (1) Application of statistical, linguistic and structural techniques to identify, tag, and extract entities, concepts, relationships, and events within text document sets, (2) Creation of a categorization/taxonomy from the extracts, and (3) Application of statistical techniques to classify the documents (Grimes, 2005).

2.2.4 Topic Model Components

Despite topic modeling’s undisputed popularity, it is for many a difficult area to break into due to its relative complexity and the common practice of leaving out implementation details in papers describing new models (Darling, 2011). Further to this, existing literature indicates that a lot of studies have been directed towards the

architecture of topic models at the expense of other components that are required for a complete topic modelling process.

Darling (2011) further notes that while key update equations and other details on topic model inference are often included in the formulation, the intermediate steps used to arrive at these conclusions are often left out, and what details are given are rarely enough to enable most researchers to test the given results for themselves by implementing their own version of the described model. There is therefore a need to develop a framework that consolidates scattered literature involving model architecture theory and definitions provided in research publications and other practical steps believed to affect the quality of model output in the deployment of topic models. Ultimately, it is hoped that this framework will provide a holistic view of topic models.

2.2.5 Topic Model Application Domains

Text Modelling is used in a variety of application domains, such as biomedical sciences to extract novel knowledge from scientific text (Zhu et al., 2013) and business intelligence (Gupta & Narang, 2012; Ishikiriyama et al., 2015). The related fields have influenced Text Modelling in terminology and methodology, whereas the application domains have been influenced by Text Modelling. Figures 2.1 and 2.2 illustrates the relationship of Text Modelling to the major related fields and application domains.

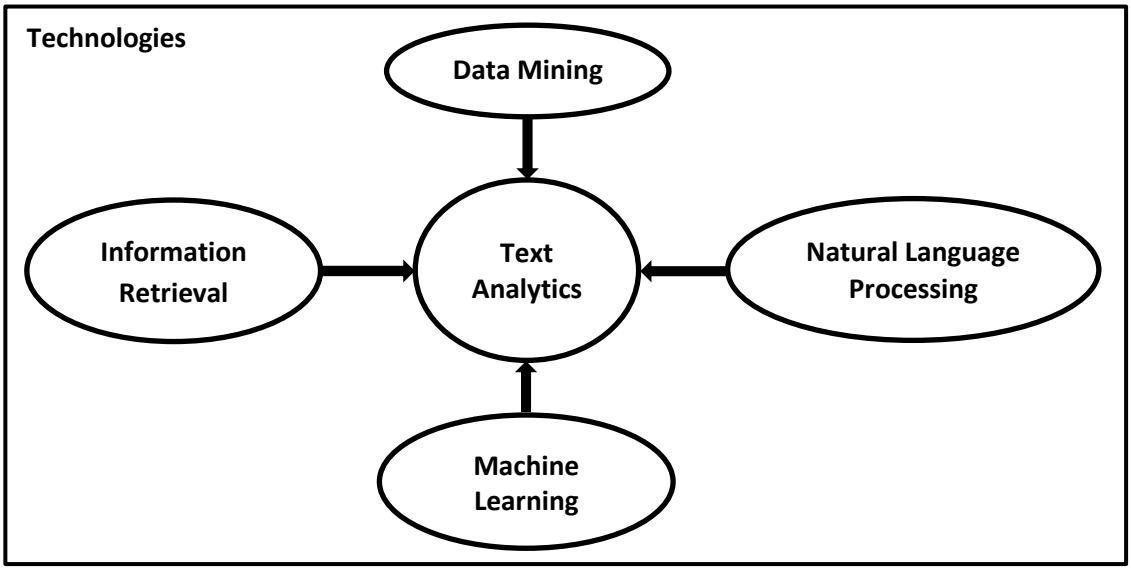


Figure 2.1: Relationship of Text Analytics to related technologies

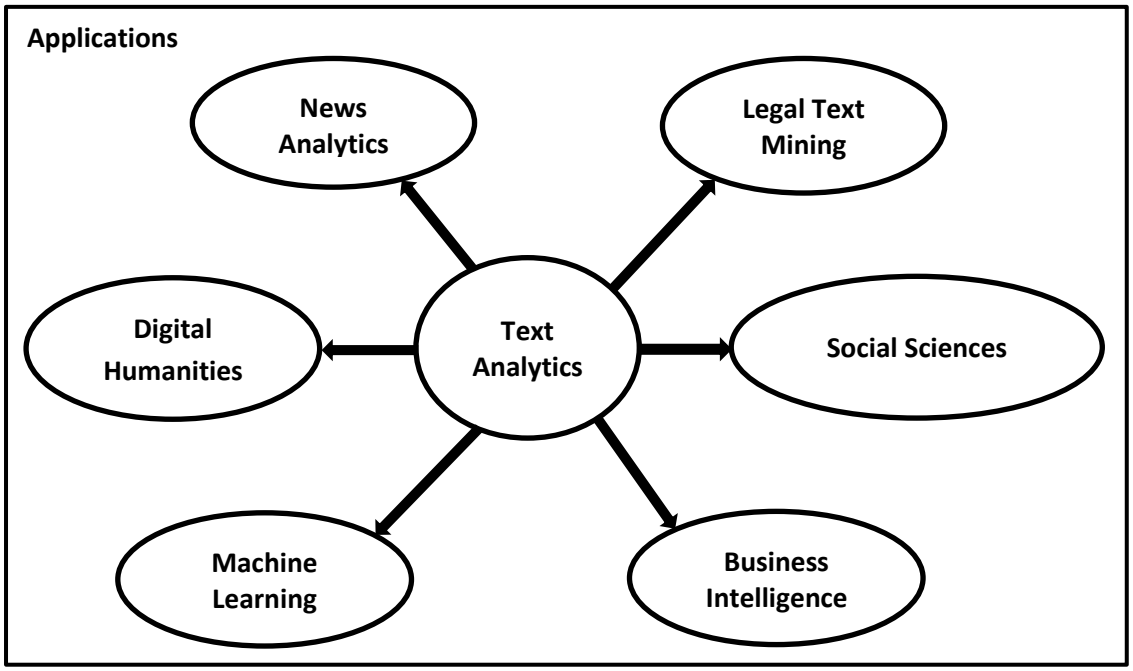


Figure 2.2: Relationship of TA to the major application domains

Nowadays, information technology opens the door through which humans step into a smart society and leads to the development of modern services such as: Internet e-commerce, modern logistics, and e-finance. It also promotes the development of

emerging industries, such as Telematics, Smart Grid, New Energy, Intelligent Transportation, Smart City, and High-End Equipment Manufacturing. Modern information technology is becoming the engine of the operation and development of all walks of life. But this engine is facing the huge challenge of big data (Zikopoulos et al., 2011). Various types of business data are growing by exponential orders of magnitude (Bell et al., 2009). Problems such as data collection, storage, retrieval, analysis, and the application of data can no longer be solved by traditional information processing technologies. These issues have become great obstacles to the realization of a digital society, network society, and intelligent society. The New York Stock Exchange produces 1 terabyte (TB) of trading data every day; Twitter generates more than 7 TB of data every day; Facebook produces more than 10 TB of data every day; the Large Hadron Collider located at CERN produces about 15 PB of data every year. According to a study conducted by the well-known consulting firm International Data Corporation (IDC), the total global information volume of 2007 was about 165 exabytes (EB) of data. Even in 2009 when the global financial crisis happened, the global information volume reached 800 EB, which was an increase of 62% over the previous year. In the future, the data volume of the whole world will be doubled every 18 months. The number will reach 35 (zettabytes) ZB in 2020, about 230 times the number in 2007, yet the written record of 5000 years of human history amounts to only 5 EB data. These statistics indicate the eras of TB, PB, and EB are all in the past; global data storage is formally entering the "Zetta era." (Tian & Zhao, 2014).

Most applications of big data in the beginning were in the Internet industry: the data on the Internet is increasing by 50% per year, doubling every 2 years (McLellan, 2015). Most global Internet companies are aware of the advent of the "big data" era and the great significance of data. In May 2011, McKinsey Global Institute published a report titled "Big data: The next frontier for innovation, competition, and productivity" (Manyika et al., 2011), and since the report was released, "big data" has become a hot topic within the computer industry.

2.3 Machine Learning (ML) Concepts

2.3.1 Defining ML

Machine Learning (ML) is the subfield of Artificial Intelligence which focuses on techniques for constructing computer programs that learn from experience to discover patterns and regularities in semi-structured or unstructured data. The many sub branches of ML include classification, clustering, probabilistic reasoning or decision theory (Rehurek, 2011).

ML involves studying the principles that govern learning processes, be it in artificial systems (as used in modern technology) or natural systems (e.g. in humans or animals). Theories and algorithms from machine learning are relevant to understanding aspects of human learning (Jordan & Mitchell, 2015). The field of ML lies at the intersection of computer science and statistics, and at the core of artificial intelligence and data science. Recent progress in machine learning has been driven both by the development of new learning algorithms and theory and by the ongoing explosion in the availability of online data and low-cost computation. The adoption of data-intensive machine-learning methods can be found throughout science, technology and commerce, leading to more evidence-based decision-making across many walks of life, including health care, manufacturing, education, financial modelling, policing, and marketing (Jordan & Mitchell, 2015). The role of Statistics is inference from a sample, while the role of Computer science is to develop efficient algorithms to solve the optimization problem and to represent and evaluate the model for inference (Alpaydin, 2014).

2.3.2 Goal of ML

The goal of machine learning is to take some training data and use it to induce a function *f*, which will map a new example to a corresponding prediction. This function *f* is evaluated on the test data. The machine learning algorithm has succeeded if its performance on the test data is high. Most research in the area of machine learning concentrates on designing systems that mimic human behaviour. However, as artificial systems are becoming more and more advanced and

autonomous, new challenges for machine learning arise. For example in Natural Language Processing (NLP), which is a use case of ML, natural language includes many ambiguities such as lexical, syntactic-level and anaphora ambiguity. Thus, it is not sufficient to build a system that is pre-programmed to mimic human behaviour or to achieve a certain goal. Present-day artificial systems must adjust their behaviour in accordance with the needs of the users and the surrounding environment. Consequently, machine learning researchers must design algorithms that will allow the artificial systems to be constantly learning through interaction with its environment and its users (Glowacka, 2012).

Despite its successful applications, current state-of-the-art ML techniques contain a lot of mathematical models that drive them behind the scene. These models are chosen for their computational tractability, and typically contain simplified assumptions; they are often challenging to interpret for human experts and their output rarely directly translates to new insights or knowledge about the problem but as Hamming (1962) famously noted that the purpose of computing is insight, not numbers. The success of these models is measured by their utility.

2.3.3 Classification of ML Algorithms

Machine learning algorithms are classified into two main categories: Supervised and Unsupervised learning. Supervised learning is where datasets have labels or true values and can further be grouped as either classification or regression problems depending on whether the labels are categorical or quantitative. Some examples of supervised learning classification algorithms are: Logistic regression, Support Vector Machines (SVM), Decision trees and k-Nearest Neighbours (k-NN). An example of supervised learning regression algorithm is linear regression.

In unsupervised learning, there is no label or true value. The data is not organised into a database or in any format. For our setting, the data conceptually exists in the form of simple text files if not, it needs to be converted to this format. The system tries to cluster things together or answers the question of which latent variables or data structure exists. It therefore takes an explorative dimension. Unsupervised

learning clustering algorithms mainly focus on K-means, Hierarchical clustering and Topic modeling while unsupervised learning latent variables or structure algorithms focuses on dimensionality reduction and topic modeling. Topic modeling therefore spans unsupervised area of machine learning. Figure 2.3 shows a classification of ML algorithms.

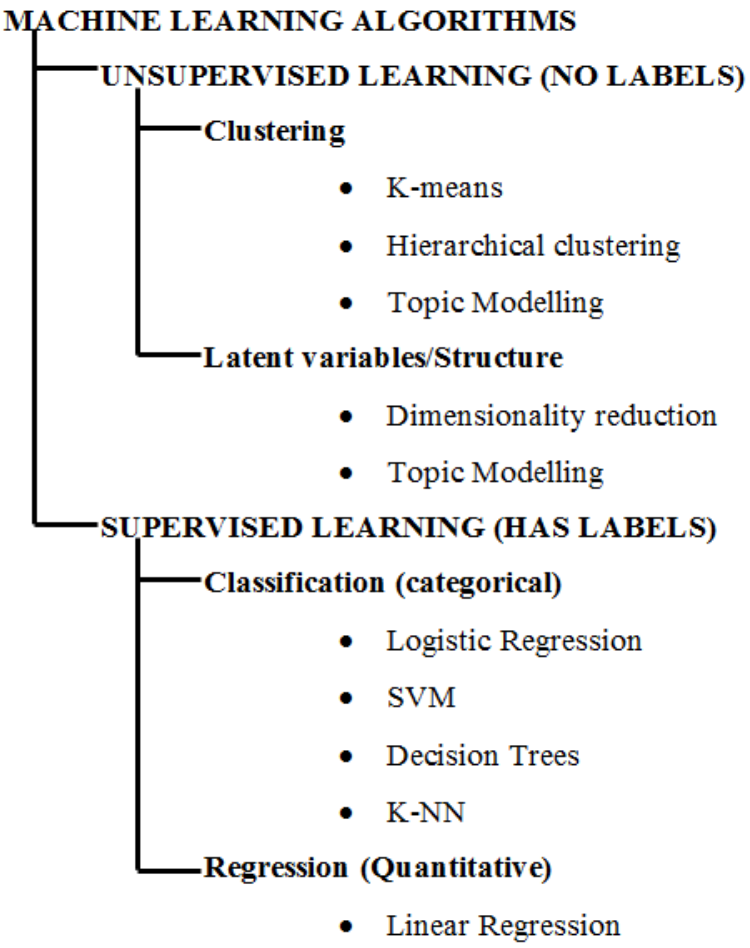


Figure 2.3: Classification of Machine Learning Algorithms

This work, focuses on topic modelling algorithms which fall under unsupervised category of machine learning algorithms, where data comes in such poor quality or in such quantities that human "supervision", as well as any sort of human inspection during its processing, is infeasible. Modelling systems in machine learning involves describing data that one could observe from a system by use of probability theory to express all forms of uncertainty and associated noise. In the expression of the model,

inverse probability (i.e. Bayes rule) allows us to infer unknown quantities, adapt our models, make predictions and learn from data.

2.3.4 Applications of ML

Machine learning is the preferred approach in many applications such as automatic speech recognition (ASR), Computer vision Medical outcomes analysis; Robot control and Computational biology. In ASR, ML provides analysis of spoken language in audio and video files, natural language processing tasks which includes speech tagging, chunking, named entity recognition, and semantic role labeling (Graves & Jaitly, 2014; Collobert et al., 2011). In computer vision, sparse representations predict well the visual features found in the early visual cortex. Generally, applications of Machine Learning can be grouped into the following categories: Association Analysis; Supervised Learning; Classification; Regression/Prediction ; Unsupervised Learning; Reinforcement Learning (Sathya & Abraham, 2013).

2.4 Probabilistic Modeling

Probabilistic modeling provides essential tools for modeling, searching, visualising and understanding the vast data sets that have become available in science, government, industry, and everyday life (Ghahramani, 2013).

A probabilistic model describes data that one could observe from a system. In machine learning, probabilistic models are described as belonging to one of two categories: generative or discriminative. Generative models are built to understand how samples from a particular category were generated. The category chosen for a new data-point is the category whose model fits the point best. Discriminative models are concerned with defining the boundaries between the categories. The category chosen for a new data-point then depends on which side of the boundary it belongs to (Lasserre, 2008)

2.4.1 Generative Models

Generative probabilistic modeling tells the story of how data came to be using the language of probability. The model randomly generates observable-data values from

a generative process given some hidden parameters by defining a joint probability distribution over both the observed and hidden parameters. Data analysis is performed by using that joint distribution to compute the conditional distribution of the hidden variables given the observed variables. This conditional distribution is also called the posterior distribution (Blei, 2012).

Generative models are used in machine learning for either modelling data directly (i.e., modelling observations drawn from a probability density function), or as an intermediate step to forming a conditional probability density function. A conditional distribution can be formed from a generative model through Bayes' rule. In the case of text analysis the generative model involves defining a data generating process for each document and then using the data to find the most likely values for the parameters within the model (Roberts et al., 2014).

More formally, if the set of all the variables (hidden and observed) of a system is denoted z, then a generative model will describe the system with a joint probability distribution over all the variables in z, i.e. writing $p(z)$, thereby providing a framework to model all of the interactions between the variables. This is called generative because it provides a way of sampling different possible states of the system. Since there is a distribution over all the variables of the system, inference can be performed for any variable, by marginalising out the others. Note that inference in this setting is able to give not only a solution, but also a confidence measure.

Generative probabilistic models provide a rigorous platform to define the prior knowledge experts have about the problem at hand, and to combine it with observed data. Their ability to model uncertainty while still absorbing prior knowledge is what provoked the transfer from traditional techniques mostly to generative probabilistic modelling. This is true for a number of application fields. In general computer vision, geometry has been put aside for probabilistic models: naive Bayes and numerous variants of the constellation model are common practice in object recognition, while techniques like Markov random fields have revolutionised segmentation, and there are numerous generative models for pose estimation. In natural language processing,

traditional rule-based systems have been overtaken by Markov models. In bio-informatics, to represent regulatory networks, the original dynamic models using differential equations have been challenged by Bayesian networks. In artificial intelligence, the list of applications where traditional methods have been augmented with probabilistic generative models is long: path planning, control systems, navigation systems, etc. However, recently a different shift has appeared, from generative to discriminative models, which have known a popular success in many scientific fields.

Probabilistic generative models are built to capture the interactions between all the variables of a system, in order to be able to synthesise possible states of this system. This is achieved by designing a probability distribution p modelling inputs, hidden variables and outputs z jointly. This is written $p(z/\theta)$, where θ represents the parameters of the model. Note that the different variables involved in z can be heterogeneous. To make the joint distribution simpler to estimate, conditional independencies can be added so as to factorise p, or a prior distribution can be defined over θ to prevent it from taking undesirable values. Because of their modelling power, generative models are usually chosen to inject prior knowledge in the system. Experts use their experience of the problem to choose reasonable distributions and reasonable conditional independencies.

In the context of classification, the system's input is the descriptor x of a data-point, and the output is the label c of this data-point. Probabilistically speaking, it means that $p(x, c/\theta)$ is defined, as shown in *Figure 2.4*. If the categories are cats and dogs, the question generative models will then answer is: 'what makes a cat a cat?' or 'why is a dog a dog?'.

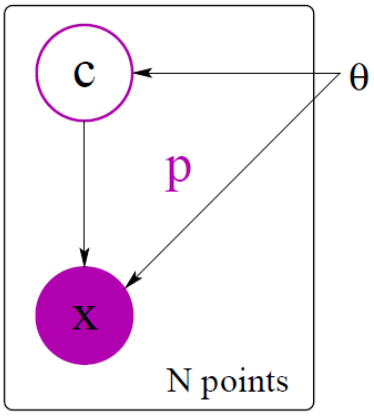


Figure 2.4: Graphical representation of a basic generative model.

Because the label is modelled by the joint distribution, generative models can easily perform classification by computing *p(c/x, θ)* using the common probability rules.

Popular generative models include naive Bayes, hidden Markov models, Markov random fields and so on. However, the simplest one is probably the Gaussian mixture model (GMM) (Bishop, 2006).

2.4.2 Discriminative Models

Discriminative models are also called conditional models and are widely used in Natural Language Processing (NLP), Speech Processing, Information Retrieval (IR) and generally machine learning (Bishop, 2013).

More formally, if the set of input variables in the system is denoted *x*, and the set of output variables is denoted *c*, then a discriminative model is a conditional probability distribution over the output variables in c, i.e. *p(c/x)*, therefore provides a framework to model the boundaries between the possible output states. This is called discriminative because it provides a way of discriminating directly between the different output states. Again, note that the prediction is given with a confidence measure.

The process in discriminative models is data driven unlike generative models which sketches a solution using prior knowledge and refines it with the data available. As a consequence, more effort is usually put into pre-processing the data. Discriminative models have shown enormous potential in many scientific areas. Object recognition, economics, bio-informatics and text recognition have known a huge improvement with the use of support vector machines over generative approaches, while conditional random fields are the state-of-the-art in segmentation, and discriminative variants of hidden Markov models have proved to be superior to their generative counterparts for speech recognition (Lasserre, 2008).

Discriminative probabilistic models are very efficient classifiers, since that is what they are defined for, however they have no modelling power. It is almost impossible to inject prior knowledge in a discriminative model (the best example of a model presenting this difficulty is probably the Neural Network). This makes them act like black boxes: training data → computation time → boundaries. Why / how? is not something a discriminative model will answer. Ironically though, in the recent years discriminative models have performed so well that they have been preferred to their generative cousins for their ability to classify, at the expense of clarity and modelling (Lasserre, 2008).

2.4.3 Bayesian Framework

Bayesian modelling describes data that one could observe from a system using mathematics of inverse probability to express all forms of uncertainty and associated noise. The models allow users to infer unknown quantities, adapt the models, make predictions and learn from data. Bayesian modelling and inference is based on Bayes' theorem which can be stated simply as follows:

$$P(hypothesis|data) = \frac{P(data|hypothesis\ )\ P(hypothesis)}{P(data)}$$

P(hypothesis|data) is called *posterior*, *P(data|hypothesis)* referred to as the *Likelihood, P(hypothesis) prior* and *P(data) Evidence*.

Bayes rule tells us how to do inference about the posterior from the evidence. Learning and prediction can be seen as a form of inference. Machine Learning seeks to learn models of data by defining a space of possible models; learning the parameters and structure of the models from data and making predictions and decisions. Our goal in Bayesian inference is to get an accurate representation of the posterior distribution, the *P(hypothesis|data)*.

In topic modelling, Bayesian inference is used to fit a probability model to a set of data and summarize the results using a probability distribution on the parameters of the model and on unobserved quantities such as predictions for new observations. Bayesian method's central characteristic is the explicit use of probability for quantifying uncertainty in inferences based on statistical data analysis.

The following three steps describe Bayesian data analysis: (1) Setting up a joint probability distribution for all observable and unobservable quantities in a problem. (2) Conditioning on observed data: calculating and interpreting the appropriate posterior distribution, the conditional probability distribution of the unobserved quantities of ultimate interest, given the observed data. (3) Evaluating the fit of the model and the implications of the resulting posterior distribution: how well does the model fit the data, are the substantive conclusions reasonable, and how sensitive are the results to the modelling assumptions in step 1? In response, one can alter or expand the model and repeat the three steps.

Bayesian modelling and inference is a popular and growing tool in statistics, machine learning and data mining. It is one of the two dominating perspectives used in probabilistic modelling and has certain interesting features for handling over-fitting, prior information and uncertainty, which can be useful in applications.

2.4.4 Basic Inference Process

The basic inferential process illustrated on Figure 2.5 is a three step process. The first step in any inference procedure is to collect data from the system of interest. In the second step, a statistical model which usually depends on a number of unknown

parameters is build. The last step involves making use of the data to infer the parameters of interest.

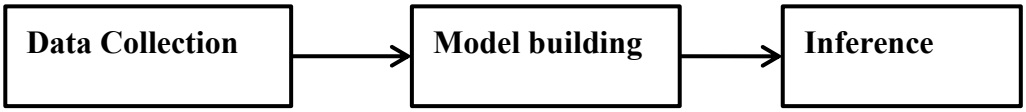


Figure 2.5: The Basic Inference Process

After the model has been inferred, it can be used to make forecasts or for making decisions by taking uncertainty into account.

2.5 Semantics Used in this Study

2.5.1 Corpus

In this work, a corpus is used to denote a body of (digital) text documents. A document is simply a chunk of text and does not necessarily correspond to the entirety of a manuscript as it was originally published. In Information Retrieval, documents are often individual paragraphs, passages, sentences, phrases or even just sequences of characters. The ideal granularity of what constitutes a "document" is dependent on the intended application of the corpus. In these generalized settings, documents are also sometimes called contexts or chunks. It can be advantageous to view and store documents as "semantically coherent blocks of text", where each chunk deals with a single idea or topic.

Firth (1957) postulated that "You shall know a word by the company it keeps". It has long been observed that words that occur in similar contexts are semantically related (Turney & Pantel, 2010). This is often phrased more generally as the statistical semantics hypothesis:

"Statistical patterns of human word usage can be used to figure out what people mean."

2.5.2 Bag-of-Words

One of the most widely used feature representation methods is bag-of-words (BoW). Under the bag-of-words hypothesis, each vector dimension corresponds to the

frequency of a particular token; these dimensions form descriptive facets of the data and are therefore also called features. Each document is then interpreted as a measurement in a multidimensional feature space. Bag-of-word models use features with quantitative domain (integer or real numbers). Features used in other fields of Machine Learning, such as unordered nominals (human gender, nationality, colours) or ordered nominals ("good-neutral-bad"), are either avoided or must be first carefully converted. Another peculiarity of the bag-of words approach is the very high dimensionality of the feature space (Rehurek, 2011; Tsai, 2012).

In mathematics, a bag, also called a multiset, is a set with duplicates allowed. A document can be represented by the bag of its constituent tokens, so that for example the utterance "to be or not to be" is represented by the multiset {to, be, or, not, to, be}. Order of the elements in (multi)sets is arbitrary, so this document may equivalently be represented by the vector [2, 1, 1, 2], given the understanding that the first element corresponds to the frequency of the word "be", second to that of "not", third of "or" and fourth of "to" (Rehurek, 2011).

In Information Retrieval, the bag-of-words hypothesis stipulates that such word frequency vectors can be used to assess semantic relevance of documents. In other words, it states that the frequencies of individual words are sufficiently indicative of semantic association between two documents. However the bag-of-words hypothesis is painfully naive from the linguistic perspective since it ignores word order, as well as any syntactic structure, which incurs a serious loss of information. Each observation is represented by a vector, i.e., by a single point in space. Similarity then translates to standard vector similarity measures.

On the positive side, the transformation from the surface text into a bag-of-words vector is computationally straightforward and efficient. Switching to the world of vectors and matrices also allows us to utilize powerful techniques and algorithms from the field of linear algebra. Perhaps the most convincing argument in this respect is the vast body of literature and successful applications based on this approach (Turney & Pantel, 2010).

2.5.3 Term-Document Matrix

A wide range of computational kernels in data mining and information retrieval from text collections involve techniques from linear algebra. These kernels typically operate on data that are presented in the form of large sparse term-document matrices (tdm). With a collection of many documents, corresponding vectors can be stacked into a single matrix. By convention, document vectors form the columns, while the vector elements (called features) form the matrix rows. With n documents and m features, an $m \times n$ matrix $\boldsymbol{A}$ can be formed. In more complex schemes, the integer event frequencies from bag-of-words are commonly reweighted according to their importance (reflecting the intuition that some words are more semantically important than others), and the resulting vector can be normalized to abstract from document length, so that $\boldsymbol{A} \in \boldsymbol{R}^{mxn}$ (Zeimpekis & Gallopoulos, 2006; Rehurek, 2011)

In many domains, only a handful of features are active in any given context. For example, in the "to be or not to be" example, only three words appear with non-zero frequency. All other words, such as "shopping" or "hippopotamus" have a frequency of zero. The vector that corresponds to this document would therefore have three non-zero elements, and typically tens or hundreds of thousands of zero elements. Naturally, algorithms that build on top of the bag-of-words paradigm must take sparsity into account if they are to be efficient (Rehurek, 2011).

The vanilla bag-of-words model described above is seldom the representation of choice. More advanced techniques used in practice exploit more complex vocabulary patterns, just as additional vector transformations and similarity metrics capture more of (or a different type of) document semantics.

2.6 TM and Inference Techniques

The main computational challenge in topic modelling is inference, namely estimating the posterior distribution over both parameters and hidden variables and ultimately estimating predictive probabilities and the evidence (Asuncion et al., 2009, Phan et al., 2008).

Probabilistic topic models describe documents in a two-step generation process. In the first step, documents are observed as mixtures of latent topics, while in the second step, topics are probability distributions over vocabulary words (Hu, 2009; Vulic & Moens, 2015).

The central goal of a topic model is to provide a thematic summary of a collection of documents. In other words, it answers the question: what themes are these documents discussing? A collection of news articles could discuss for example political, sports, and business related themes. Figure 2.6 shows a conceptual topic modelling algorithm indicating the input as document corpus and number of topics and output as a list of topics and a topical categorization.

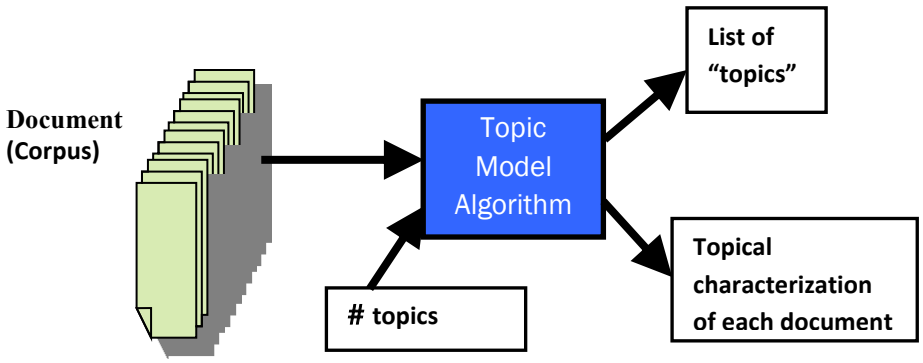


Figure 2.6: Conceptual Topic Model Algorithm

Topic models uses probability distributions to describe data emanating from a system. These probability distributions requires the definition of parameters. In a Bayesian context, these parameters are also distributed according to some distribution as opposed to the frequentist approach of point estimates where the parameters are estimated as single values. The parameters that describe the distribution of other parameters are called hyper-parameters i.e. parameters of parameters.

The algorithmic component of a topic model generally has three main objectives: (1) learning distributions on words called "topics" shared by documents; (2) Learning a distribution on topics for each document and (3) Assigning every word in a

document to a topic. In topic modelling, none of these objectives are known in advance and must be learned. Learning involves defining a model, and a learning algorithm. Each document is treated as a "bag of words".

While originally, the models were designed to analyse and classify large text document collections, they have nowadays been extended to model data from other fields, including computer vision for example (Li et al., 2009; Wang et al., 2009). Some of the examples where researchers have used topic models in computer vision problems include sorting multiple images into scene-level classes, annotating images with words, and segmenting and labelling objects within images. Statistical topic models have also been extended to analysis of video (Wang, 2009). Other fields where topic models have been applied are bio-informatics, finance and even social sciences (Vulic & Moens, 2015). In the original definition, a topic is defined to contain a cluster of words that frequently occur together. A topic model can associate words with comparable meaning and distinguish between uses of words with numerous meanings.

2.6.1 Popular Topic Model Algorithms

The main importance of topic modeling is to discover patterns of word-use and how to connect documents that shared similar patterns. So, the idea of topic models is that term which can be working with documents and these documents are mixtures of topics, where a topic is a probability distribution over words. In other word, topic model is a generative model for documents. It specifies a simple probabilistic procedure by which documents can be generated (Alghamdi & Alfalqi, 2015). Create a new document by choosing a distribution over topics. After that, each word in that document could choose a topic at random depends on the distribution. Then, draw a word from that topic. (Steyvers & Griffiths, 2007).

Topic models rely on the bag-of-words assumption which is ignoring the information from the ordering of words. Each document in a given corpus is thus represented by a histogram containing the occurrence of words. The histogram is modeled by a distribution over a certain number of topics, each of which is a distribution over

words in the vocabulary. By learning the distributions, a corresponding low-rank representation of the high-dimensional histogram can be obtained for each document. The various kind of topic models, such as Latent semantic analysis (LSA), Probabilistic latent semantic analysis (PLSA), Latent Dirichlet allocation (LDA), Correlated topic model (CTM) have successfully improved classification accuracy in the area of discovering topic modeling (Alghamdi & Alfalqi, 2015).

Probabilistic Latent Semantic Indexing was introduced by Hofmann (1999) as a novel approach to automated document indexing based on a statistical latent class model for factor analysis of count data. The model was fitted from a training corpus of text documents by a generalization of the Expectation Maximization algorithm. The basic idea of PLSI is that documents are generated by mixtures of topics, where a topic is a multinomial distribution over words. Blei et al., (2003) notes that while Hofmann's work is a useful step toward probabilistic modeling of text, it is incomplete in that it provided no probabilistic model at the level of documents. In Probabilistic Latent Semantic Indexing (PLSI), each document is represented as a list of numbers (the mixing proportions for topics), and there is no generative probabilistic model for these numbers. Blei et al., (2003) identified the following problems with PLSI: (1) the number of parameters in the model grows linearly with the size of the corpus, which lead to serious problems with over fitting, and (2) it was not clear how to assign probability to a document outside of the training set. With Hofmann's model therefore, it was not clear how the mixing proportions for topics in a document are generated. To overcome this limitation, Blei et al., (2003) proposed Latent Dirichlet Allocation (LDA).

The Latent Dirichlet Allocation (LDA) model was introduced by Blei et al., (2003) to address the shortcomings of Hoffmann (1999). Since then, LDA has been applied to numerous tasks, such as text mining (AlSumait et al., 2008), computer vision (Cao & Fei-Fei, 2007), social-network analysis (Fu et al., 2007) and bio-informatics (Pinoli et al., 2014).

LDA (Blei, Ng & Jordan, 2003; Blei, 2012) is a prominent generative probabilistic topic model in the field of text data analysis that uses a pattern of word co-

occurrences to discover latent themes across text documents by assuming that documents are a mixture distributions of language models associated with individual topics (Roberts et al., 2014).

LDA precisely falls into the generative model framework, previously described. The observed variables are the words of the documents; the hidden variables are the topic structure; and the generative process is defined. The computational problem of inferring the hidden topic structure from the documents is the problem of computing the posterior distribution, the conditional distribution of the hidden variables given the documents.

2.6.2 Topic Model Inference Techniques

A fully defined topic model, need to have the following: (1) well identified probability distributions of prior, likelihood and posterior over the problem in question; (2) A set of parameters associated with the defined model, to be estimated from data and fit into the model; (3) a notation to aid the mathematical analysis; (4) Conversion of data to digits that can be analysed using the defined probabilistic model. (5) Interpretation of the data obtained back to a human understandable form. (6) Presentation of the analytical results for utilisation by the expected audience. The last point includes visualisation which is the use of typography, animation and cinematography with modern human-computer interaction system to facilitate the comprehension of computer systems for easier and faster utility (Alice, 2014).

In probabilistic topic modelling the distributions are described as mathematical functions. Generally inference is concerned with drawing conclusions, from numerical data, about quantities that are not observed. Inferring posterior and parameter estimates involves integrations and because of the multi-parameter nature and correlation between the parameters of interest in the defined models, the complex integration leads to a challenge of solving intractable functions. Many researchers have used various mathematical approximations to deal with these intractable integrals that result to inference algorithms. These approximations are

subject to scientific research in establishing optimal solutions in contextual setups (LaRosa, Fiannaca, Rizzo & Urso, 2015).

Modern approximate posterior inference algorithms fall into two main categories: sampling approaches and optimization approaches. In sampling based approaches like the popular Markov Chain Monte Carlo (MCMC), the topic model parameters of interest are approximated through an iterative process where transitional matrices are improved to convergence. The conceptual idea of these methods is to generate independent samples that are provably distributed according to the posterior and then reason about the data of interest. This is achieved by drawing parameter values from approximate distributions and then correcting those draws to better approximate the target posterior distribution, say $p(\theta|y)$ where θ is the set of parameters to be estimated and y is the given data. The sampling is done sequentially, with the distribution of the sampled draws depending on the last value drawn; hence, the draws form a Markov chain. This is illustrated in the generic Gibbs sampler which is a specific case of MCMC (Hoffman, Bach & Blei, 2010).

2.6.3 Generic Gibbs Sampler

Gibbs Sampling is one member of a family of algorithms from the Markov Chain Monte Carlo (MCMC) framework (Gilks et al., 1999). The MCMC algorithms aim to construct a Markov chain that has the target posterior distribution as its stationary distribution. In other words, after a number of iterations of stepping through the chain, sampling from the distribution should converge to be close to sampling from the desired posterior. Gibbs Sampling is based on sampling from conditional distributions of the variables of the posterior.

In a general perspective, given a joint sample x^s of all the variables, a new sample x^{s+1} can be generated by sampling each component in turn, based on the most recent values of the other variables. For example, the following set of equations can be used if there are three variables:

- $x_1^{s+1} \sim p(x_1|x_2^s, x_3^s)$

- $x_2^{s+1} \sim p(x_2 | x_1^{s+1}, x_3^s)$
- $x_3^{s+1} \sim p(x_3 | x_1^{s+1}, x_2^{s+1})$

x_i is not sampled since it is a visible variable and therefore is already known.

The key to the method's success, however, is not the Markov property but rather that the approximate distributions are improved at each step in the simulation, in the sense of converging to the target distribution. The Markov property is helpful in proving this convergence (Gelman et al., 2014). Unlike variational inference, MCMC has asymptotic guarantees.

The second category referred to as optimization approaches are based on variational inference and also called the Variational Bayesian methods. They involve finding an approximation of the posterior within an analytically tractable family of distributions. These methods optimize the closeness of the posterior, based on the Kullback-Leibler divergence, to a simplified parametric distribution (Speh, Muhic & Rupnik, 2013). The idea is to choose a distribution within the simplified family that is closest to the true posterior.

2.6.4 Drawbacks of Markov Chain Monte Carlo

The most widely used posterior inference methods in Bayesian nonparametric models are the Markov Chain Monte Carlo (MCMC) (Gershman & Blei, 2012). Unlike Variational Bayesian methods MCMC are guaranteed to converge to the posterior with enough samples. However the methods have two major drawbacks: (1) the samplers must be run for many iterations before convergence and (2) it is difficult to assess convergence. These drawbacks slows the models in application hence raising a practical motivation for speed enhancement in MCMC approaches in order to enhance their usage in the era of Big Data analytics where mining of numerous data streams is performed in real-time.

2.6.5 Strategies for Accelerating Inference

Monte Carlo methods are useful for enabling Bayesian inference in many models. The main drawback with the methods is that a large number of samples from the posterior are usually required to obtain reasonable accuracy. This is often not a problem for simpler Monte Carlo methods, where samples can be generated efficiently. However, for more complicated models it can take considerable time to generate a sample (Dahlin, 2016).

There are two main approaches to mitigate this problem: (i) decreasing the computational cost of generating a sample or (ii) decreasing the required number of samples by making better use of them. Another aspect for the user is the time and complexity of the implementation of an inference algorithm for a new model. It can therefore be beneficial to apply simpler methods that are quicker to implement and tune than more elaborate algorithms that may be more efficient but takes longer time to get up and running. The total time for the entire inference can therefore be shorter in the former case than in the latter, even if the advanced algorithm is more efficient.

2.7 Latent Dirichlet Allocation (LDA) Model

2.7.1 Overview of the LDA Model

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is one of the most classical state of the art unsupervised probabilistic topic model which can be applied to a corpus of text documents in a bag-of-words form to discover semantic structure of documents, by examining statistical word co-occurrence patterns within a corpus of training documents.

LDA is acknowledged as the most popular approach for building topic models (Wei & Croft 2006; Boyd-Graber, 2010; Rosen-Zvi et al., 2004; Mei et al., 2007; Chemudugunta et al., 2007).

LDA assumes a hypothetical generative process is responsible for creating observed set of documents. The natural statistical assumption behind LDA is to model documents as arising from multiple topics, where a topic is defined to be a

distribution over a fixed vocabulary of terms. LDA assumes that, since documents in a corpus tend to be heterogeneous, K topics are associated with a collection, and that each document exhibits these topics with different proportions.

The original generative process for a document collection D under the LDA model is defined by Blei (2003) as follows:

1. For $k = 1, \ldots, K$

   (a) $\phi^{(k)} \sim Dirichlet\ (\beta)$

2. For each document $d \in \boldsymbol{D}$

   (a) Choose $\theta_d \sim Dirichlet\ (\alpha)$

   (b) For each word $w_i \in d$:

   i) $z_i \in Discrete(\theta_d)$

   ii) $w_i \in Discrete(\phi^{(zi)}$

where K is the number of latent topics in the collection, $\phi^{(k)}$ is a discrete probability distribution over a fixed vocabulary that represents the k^{th} topic distribution, θ_d is a document-specific distribution over the available topics, z_i is the topic index for word w_i, and α and β are hyperparameters for the symmetric Dirichlet distributions that the discrete distributions are drawn from.

The above described generative process results in the following joint distribution:

$$p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \phi|\alpha, \beta) = p(\alpha|\beta)\ p(\theta|\alpha)\ p(z|\theta)\ p(w|\ \phi_z) \qquad (2.1)$$

What is of interest to us is the hidden variables $\boldsymbol{z}$, $\boldsymbol{\theta}$, and ϕ. Each θ_d is a low-dimensional representation of a document in "topic"-space, each z_i represents which topic generated the word instance w_i, and each $\phi^{(k)}$ represents a $K\ X\ V$ matrix where $\phi_{i,j} = p(w_i|z_j)$. Therefore, one of the most interesting aspects of LDA is that it can learn, in an unsupervised manner, words that would be associated with certain topics, and this is expressed through the topic distributions ϕ. Table 2.1 is an example of top

10 words for 3 topics that can be learned using LDA on a sample dataset. On the table 2.1, topic labels and word selection have been added manually for illustration purposes.

Table 2.1 Three topics learned using LDA on Sample Dataset

"Environs"	**"Travel"**	**"War"**
Discharge	travel	combat
Conservational	tourism	fight
Air	Hotel	hatred
License	hostel	match
Herb	Fares	animosity
Facility	City	ethnic
Part	town	battle
Ape	Visit	war
Water	Miles	against
Location	Deal	team

The main idea of the LDA model is based on the assumption that each document may be viewed as a mixture of various topics, where a topic is represented as a multinomial probability distribution over words. Learning the mixing coefficients for the various document-topic and topic-word distributions is a problem of Bayesian inference. Blei et al., (2003) developed a variational Bayesian algorithm to approximate the posterior distribution in the original paper; and an alternative inference method using collapsed Gibbs sampling to learn the model from data was subsequently proposed by Griffiths et al., (2004). Both of the approaches have their advantages and disadvantages: the variational approach is arguably faster computationally but can possibly lead to inaccurate inferences and biased learning; though the collapsed Gibbs sampling approach is in principal more accurate, it has high computational complexity, which makes it inefficient on large datasets.

With the advent of the era of big data, the amount of data in our world has been exploding, and analyzing large datasets has become a key in many areas. LDA model, with the collapsed Gibbs sampling algorithm in common use, has now been

broadly applied in machine learning and data mining, particularly in classification, recommendation and web search, in which both high accuracy and speed are required. Some improvements have been explored for adapting to big data based on variational Bayes (Teh et al., 2007; Nallapati et al., 2007). The Collapsed Variational Bayes algorithm converges faster than collapsed Gibbs sampling, but the latter attains a better solution in the end with enough samples (Teh et al., 2007). So there is a significant motivation to speedup collapsed Gibbs sampling for LDA model.

Bayesian methods are especially appealing due to their ability to represent uncertainty in parameter estimates and latent variables. Real-world problems are rarely amenable to exact inference, and so require approximate inference in the form of Monte Carlo estimates or variational approximations. Unfortunately, approximate Bayesian inference can be challenging when modeling large data sets, as the target posterior density may become expensive to evaluate (Angelino et al., 2014). This challenge has motivated new methods for inferential computation that can take advantage of approximations to the target density, most often by examining only a subset of the data, or by exploiting closed form approximations such as Taylor series or by putting linear or Gaussian Process regressions (Christen & Fox, 2005; Angelino, 2014). Stochastic variational inference techniques achieved this by randomized approximations of gradients, while recent developments in Markov chain Monte Carlo (MCMC) have implemented efficient transition operators that lead to approximate stationary distributions (Hoffman et al., 2013; Welling & Teh, 2011).

Recent other work uses a lower bound on the local likelihood factor to simulate from the exact posterior distribution while evaluating only a subset of the data at each iteration (Maclaurin & Adams, 2014).

The key design principle is to formulate the MCMC method so that it is, in principle, applicable for functions; this may be achieved by use of proposals based on carefully chosen time-discretizations of stochastic dynamical systems which exactly preserve the Gaussian reference measure. Taking this approach leads to many new algorithms

which can be implemented via minor modification of existing algorithms, yet which show enormous speed-up on a wide range of applied problems.

In topic modelling the main problem is posterior inference of hidden variables given some parameters from observed data which amounts to reversing the original generative process for a document collection D under the LDA model defined above by Blei (2003), and learning the posterior distributions of the latent variables in the model. In Latent Dirichlet Allocation, this involves solving the following equation:

$$p(\theta, \phi, \boldsymbol{z}|\boldsymbol{w}, \alpha, \beta) = \frac{p(\theta, \phi, \boldsymbol{z}, \boldsymbol{w}|\alpha, \beta)}{p(\boldsymbol{w}|\alpha, \beta)} \qquad (2.2)$$

Many researchers such as (Xu, Hospedales, & Gong, 2017; Huang, Wang, Yang, & Xu, 2018; Puschmann, Barnaghi, & Tafazolli, 2018) have noted that this distribution is intractable to compute. However, a number of approximate posterior inference techniques that can be applied to the problem have been developed. Three of the most common are: one, variational inference which was used in the original LDA paper and further improved to Collapsed Variational Bayes (Teh & Welling, 2007). The second is Markov Chain Monte Carlo approach called Gibbs Sampling which samples from the posterior over topic assignments by repeatedly sampling the topic assignment conditioned on the data and all other topic assignments. Many researchers have improved this technique for topic modeling for example Qiu et al., (2014). This paper will further explore Gibbs Sampling. Lastly online variational Bayes algorithm is based on online stochastic optimization with a natural gradient step (Hoffman et al., 2010).

2.7.2 Gibbs Sampling in LDA

Gibbs sampling was developed in 1876 by Josiah Willard Gibb as a means of determining the energy states of gasses at equilibrium. The method he used was modelled as a Bayesian posterior distribution (Bolstad, 2010). The method defines a Markov Chain whose stationary distribution is the posterior of interest. Independent samples are collected from the stationary distribution to approximate the posterior.

This produces a Monte Carlo estimate of an expectation using independent samples from the posterior (Gyori & Paulin, 2016).

For example, to sample x from the joint distribution $p(x) = p(x_1, \ldots, x_m)$, where there is no closed form solution for $p(x)$, but a representation for the conditional distributions is available, one would perform the following using Gibbs Sampling:

1. Randomly initialize each x_i
2. For $t = 1, \ldots T$:

2.1 $x_1^{t+1} \sim p(x_1 | x_2^{(t)}, x_3^{(t)}, \ldots, x_m^{(t)}$

2.2 $x_2^{t+1} \sim p(x_2 | x_1^{(t+1)}, x_3^{(t)}, \ldots, x_m^{(t)}$

…

2.m $x_m^{t+1} \sim p(x_m | x_1^{(t)}, x_2^{(t+1)}, \ldots, x_{m-1}^{(t+1)}$

This procedure is repeated a number of times until the samples begin to converge to what would be sampled from the true distribution. Though convergence is guaranteed theoretically with Gibbs Sampling, it is not possible to know the number of iterations required to reach the stationary distribution. Therefore, diagnosing convergence is a real problem with the Gibbs Sampling approximate inference method. Many authors however note that Gibbs sampling is pretty powerful in practice and has fairly good performance. Most applications provide a calculation of the log-likelihood which is an acceptable estimation of convergence.

Many authors for example Darling (2011) have published on theoretical details of probabilistic topic modeling which can aid readers in understanding the mathematical underpinnings of topic modeling.

2.7.3 LDA Adaptations and Improvements

Topic models based on LDA assume a predefined vocabulary which is reasonable in batch settings but not reasonable for streaming and online settings. LDA is extended

by drawing topics from a Dirichlet process whose base distribution is a distribution over all strings rather than from a finite Dirichlet (Zhai, 2013).

Since its inception, a number of research has also been conducted on LDA so as to adapt and improve it to deal with, for example, supervised learning tasks (Mcauliffe & Blei, 2008); time and memory efficiency issues (Porteous et al., 2008; Yao et al., 2009; Newman et al., 2009) and large or streaming data settings (Gaber, 2012).

Modupe et al., (2013) investigated the use of Latent Dirichlet Allocation (LDA) to extract latent features arising from mobile Short Messaging Service (SMS) communication for automatic discovery of user interest. This involved integrating temporal ordering of SMS into a generative process in an iterative manner. The mobile SMS documents were partitioned into segments, wherein the discovered topics in each segment were propagated to influence the discovery of latent features. The proposed technique filters malicious mobile SMS communication and showed that topic models can effectively detect distinctive latent features to support automatic content filtering overtime.

Among the implementations and improvements, some have been explored for speeding up the LDA model like: Newman et al., (2009) proposed two versions of LDA where the data and the parameters are distributed over distinct processors: Approximate Distributed LDA model (AD-LDA) and Hierarchical Distributed LDA model (HD-LDA). In AD-LDA, they simply implement LDA on each processor and update global parameters after a local Gibbs sampling iteration. HD-LDA can be viewed as a mixture model with PLDA, which optimizes the correct posterior quantity but is more complex to implement and slower to run. Porteous et al., (2008) presented a new sampling scheme, which produces exactly the same results as the standard sampling scheme but faster.

An asynchronous distributed version of LDA (Async-LDA) was introduced by Asuncion et al., (2009). In Async-LDA, each processor performs a local collapsed Gibbs sampling step followed by a step of communicating with another random processor to gain the benefits of asynchronous computing. Async-LDA has been

improved in GraphLab, a graph-based parallel framework for machine learning, which was proposed by Low et al., (2014).

Yan et al., (2009) presented parallel algorithms of collapsed Gibbs sampling and collapsed variational Bayesian for LDA model on Graphics Processing Units (GPUs), which have massively built-in parallel processors with shared memory. The research proposed a novel data partitioning scheme to overcome the shared memory limitations.

Xiao & Stibor (2010) proposed a novel dynamic sampling strategy to significantly improve the efficiency of collapsed Gibbs sampling and presented a straight-forward parallelization to further improve the efficiency.

Wang et al., (2009) implemented collapsed Gibbs sampling on Message Passing Interface (MPI) and MapReduce called PLDA, then Liu et al., (2011) enhanced the implementation and called the new approach PLDA+. Ihler & Newman (2012) presented a modified parallel Gibbs sampler, which obtains the same speedups as AD-LDA, but provides an online measure of the approximation quality compared to a sequential sampler.

In dealing with speedup issues, our study will focus on (1) reducing the number of iterations in the looping and (2) Optimize the starting value of parameter in the iterative process of sampling to reduce the number of unnecessary iterations.

2.7.4 Existing LDA Software Implementations

There are various text analysis software tools and frameworks available for use and which work upon different methods of topic modelling. Each one requires different skills for use. Some of the more popular software tools for Topic Modelling are discussed in this section. Table 2.2 shows a list of various tools that implements the latent Dirichlet. allocation model.

Table 2.2 Tools Implementing Latent Dirichlet Allocation

Tool	Description
LDA-C	The software is written in C and provides implementation of Variational inference of LDA. It was one of the earlier software programs available so does not have as many options to tailor the analysis.
GibbsLDA++ Phan and Nguyen (2007)	GibbsLDA++ is a C/C++ implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling technique for parameter estimation and inference. It is very fast and is designed to analyze hidden/latent topic structures of large-scale datasets including large collections of text/Web documents. According to Sharma and Sharma (2017). GibbsLDA++ has little room for customization.
Gensim Rehurek (2015)	Developed in python programming language. The software provides implementation based on the papers "Online learning for Latent Dirichlet Allocation (Hoffman, Bach, & Blei, 2010)" and "Online variational inference for the Hierarchical Dirichlet Process (Wang, Paisley & Blei, 2011)". This tool allows for more customisation. There are support materials on how to run a topic model analysis. The preferred file formats are plain text files.
Lda-1.0.5	Implements LDA using collapsed Gibbs sampling in python. It is fast and is tested on Linux, OS X, and Windows. The interface follows conventions found in scikit-learn.
Matlab topic modelling toolbox	This is a Matlab implementation developed by Mark Steyvers. It is available online from <http://psiexp.ss.uci.edu/research/programs_ data/toolbox.htm> and provides implementation of Gibbs sampling version of LDA based on the paper "Finding Scientific topics" by
Mallet topic modelling	Mallet is a JAVA implementation of LDA developed by Andrew McCallum and available from <http://mallet. cs.umass.edu/ topics.php>. This software provides an implementation of "Probabilistic Topic Models" by Steyvers

	and Griffiths (2007). Further to the original implementation of Mallet, there is a package to be used from R and a complete tutorial on its usage from R is provided at Topic modelling in R
R topic modeling packages.	Mallet, Topic Models, and LDA are three packages capable of doing topic modelling analysis in R. The mallet package in R is based on the Mallet software package but rather than being capable of running a large selection of text analysis algorithms it can only do topic modelling analysis. It provides an interface to the Java implementation of latent Dirichlet allocation
Stanford Topic Modelling Toolbox	The Stanford Topic Modeling Toolbox was written at the Stanford NLP group by: Daniel Ramage and Evan Rosen, first released in September 2009. It is available online at <http://nlp.stanford.edu/software /tmt/tmt-0.4/>
GraphLab Create	GraphLab Create is a proprietary Python library, backed by a C++ engine, for quickly building large-scale, high-performance data products. In text analytics GraphLab Create can be used for automated text analytics including detecting user sentiment regarding product reviews, creating features for use in other machine learning models and understanding large collections of documents. GraphLab implements Collapsed Gibbs Sampling in LDA inference.
SCIKIT learn	Scikit-learn is a free Python programming machine learning library which features classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with NumPy and SciPy. Scikit-learn implements Online Latent Dirichlet Allocation with variational inference.

2.7.5 Challenge of Implementing Gibbs Sampling

Gibbs sampling is a popular method applied to approximate intractable integrals in probabilistic generative models such as latent Dirichlet. allocation. This method has however the crucial drawback of high computational complexity due to its iterative nature and its also difficult to assess convergence, which limits its application in huge data environments. The good news however is that the approach converges to the true posterior.

2.7.6 LDA Parameter Space

When implementing LDA, the following parameters need to be specified:

a) Hyper-Parameters

The predictive performance of LDA can be affected by the choice of hyperparameters. Often users must try different values of hyperparameters and select the best model. An alternative to setting the hyperparameters in advance is to infer them from the data. As with the other parameters of the model, the hyperparameters can be treated as random variables and have their posterior inferred (Wallach, 2008). Rather than full posterior inference, however, a more common practice is to perform MAP estimation with optimization algorithms, which has been empirically shown to be a competitive approach to sampling the hyperparameter values (Wallach et al., 2009).

b) Number of Topics (K)

One challenging issue in application of LDA is to select the optimal number of topics (Wang et al., 2014; Zhao et al., 2015). Several approaches exist, but ultimately, the appropriate number of topics must depend on both the corpus itself and user modeling goals. This section presents a topic selection method which models the minimum perplexity against number of topics for any given dataset within three iterations of the Collapsed Gibbs sampling application in graphlab create. Evaluation results show that the proposed method can generate an optimal number of topics automatically within three iterations thereby improving on hyper parameter inferential speed.

2.7.7 Approaches to Setting Number of Topics

The first is to set manually via “trial and error”. If there is a human in the loop, it may be simplest to try multiple values of K within some reasonable range (e.g., $K \in \{5; 10; 25; 50; 100; 200\}$). The user can then quickly scan the learned topics associated with each value of T and select the value which seems most appropriate to the corpus.

The second is to use domain knowledge. If the topics are meant to correspond to known entities in the world (e.g., in a vision task each topic may be a type of object), then K can be set equal to the true number of entities being modelled.

The third approach optimizes performance on secondary task. If the learned topics are to be used as input to another task, then it follows that K should be chosen according to performance on the ultimate task of interest. For example, if the topics are going to be used for document classification, K could be chosen to optimize classifier performance on a held-aside validation set.

Finally in the fifth approach the likelihood of held aside documents is optimized. Given a means of evaluating the probability $p(w|K)$ of a validation set of held aside documents, topics can be learned for different values of K and values which maximizes the likelihood of the held-aside validation set chosen (Griffiths & Steyvers, 2004; Wallach et al., 2009). Plotting these values leads to a pattern of $p(w|K)$ increasing with larger K up to a point, beyond which the model overfits (Mitchell, 1997) the data and $p(w|K)$ on the held-aside documents begins to fall. This thesis precisely uses mathematical theory to model this behavior with an aim of reducing the time taken to estimate K.

2.7.8 LDA Evaluation: Likelihood and Perplexity

There are two ways of evaluating LDA. One is by measuring performance on some secondary task, such as document classification or information retrieval, and the second is by estimating the probability of unseen held-out documents given some training documents (Wallach 2009). This second approach is the most common way used to evaluate a probabilistic model and is achieved by measuring the log-likelihood of a held-out test set.

When applying the second approach the dataset is split into two parts: one for training, the other for testing. For LDA, a test set is a collection of unseen documents $\boldsymbol{w}_d$, and the model is described by the topic matrix $\boldsymbol{\Phi}$ and the hyper-parameter $\boldsymbol{\alpha}$ for topic-distribution of documents. This leads to the following function that should be evaluated in order to compute the log-likelihood:

$$\mathcal{L}(\boldsymbol{w}) = \boldsymbol{log}\ \boldsymbol{p}\ (\boldsymbol{w}|\boldsymbol{\Phi}, \boldsymbol{\alpha}) = \sum_d \boldsymbol{log}\ \boldsymbol{p}\ (\boldsymbol{w_d}|\ \boldsymbol{\Phi}, \boldsymbol{\alpha}) \qquad (2.12)$$

The computed likelihood of unseen documents can be used to compare models with a higher likelihood implying a better model. A further definition leads to perplexity measure which according to research is the most typical measure for evaluating LDA models (Blei et al., 2003; Bao & Datta, 2014)

Perplexity measures the modeling power by calculating the inverse log-likelihood of unobserved documents. Traditionally perplexity is used which is defined interms of likelihood as follows:

$$perplexity(test\ set\ w) = exp\left\{-\frac{\mathcal{L}(\boldsymbol{w})}{count\ of\ tokens}\right\}$$

Perplexity is a decreasing function of the log-likelihood $\mathcal{L}(w)$ of the unseen documents $\boldsymbol{w_d}$ and lower perplexity corresponds to higher likelihood. This means that better models have lower perplexity which suggests less uncertainties about unobserved documents. The likelihood $p(\boldsymbol{w_d}|\Phi, \alpha)$ of one document however is intractable, which makes the evaluation of $\mathcal{L}(w)$, and therefore the perplexity, intractable as well.

The advantage of perplexity as a measurement is that it is normalized to the size of the data and can thus be compared across different datasets.

2.8 Topic Interpretability and Coherence

If topic models are used as exploratory tools, it is important that the inferred topics are meaningful to humans. Indeed, sometimes inferred topics are assumed to be so meaningful that they are directly used in analyses, for example to visualize historical trends in scientific topics (Hall et al., 2008) or literary topics (Mimno, 2012). In such cases, it is important to be able to choose models that have high interpretability. This subsection describes various methods for evaluating interpretability.

Perhaps the best way to measure the interpretability of a topic model is to directly ask humans. Topic model researchers have therefore conducted a variety of quality

evaluation experiments using human subjects. One approach is to ask subjects to rate the quality of a topic, e.g., on a Likert scale (Newman et al., 2009; Paul & Dredze, 2012), or to vote on topics aligned across models (Xing & Paul, 2018; Paul & Girju, 2009). This is a direct measurement of topic quality, but a subjective one that should be averaged across judgments of many different annotators.

A less subjective evaluation is to ask subjects to perform a task whose difficulty depends on the topic quality. This is the idea behind intrusion tasks for topic models. Chang et al.,(2009) introduced two types of intrusion tasks. The first (and perhaps more widely adopted) is word intrusion, in which subjects are shown in random order several top words from one topic, and a top word from a different, randomly selected topic. Subjects are asked to identify the word from the "intruding" topic. If the topics are coherent, it should be obvious which word came from a different topic. If the topics are noisy, subjects will perform poorly, and thus performance at this task is indicative of the interpretability of the topics.

The second task is topic intrusion, in which users are shown a document and the most probable topics within that document along with a topic that has low probability in the document. As with word intrusion, the task is to identify the low-probability topic. Interestingly, Chang et al., (2009) showed that human judgments of quality across different models were in fact negatively correlated with perplexity measurements, which means that perplexity measurements are not a good proxy for topic quality. This thesis therefore evaluates the topic models in with measures of both predictiveness and coherence.

It is time-consuming and costly to conduct experiments with human subjects. As an alternative, researchers have proposed automatic metrics of topic coherence that have been shown to be highly correlated with human judgments. Most coherence metrics measure the degree to which the words that appear within a topic also co-occur together in documents. If words rarely co-occur, they are probably unrelated, and should not be grouped in the same topic. The coherence of individual topics can be measured (e.g., for identifying and removing incoherent topics), and the average coherence across all topics can be used to evaluate the topic model overall (Mimno et

al., 2011; Lau et al., 2014; Roder et al., 2015) provide summaries and empirical comparisons of additional coherence metrics.

2.9 Model Design, Development and Validation

2.9.1 Model Design and Development

The most fundamental question in model building is determining what the model should predict. A large number of machine learning techniques have been developed, with names such as logistic regression (Chen et al., 2013; Feng et al., 2014), neural networks (Mikolov et al., 2014; Sutskever et al., 2014; Schmidhuber, 2014; Varian, 2014), decision trees, support vector machines (Claesen et al., 2014; Kremer et al., 2014), Kalman filters (Singh & Sinha, 2013; Nkomo et al., 2013) and many others. Contributions to this multi-disciplinary effort have come from the fields of statistics, artificial intelligence, optimization, signal processing, speech, vision and control theory, as well as from the machine learning community itself.

In the traditional approach to solving new machine learning problems, practitioners must select a suitable algorithm or technique from the set with which they are familiar, and then either make use of existing software, or write their own implementation. If the technique requires modification to meet the particular requirements of their specific application, then they must be sufficiently familiar with the details of the software to make the required changes (Bishop, 2013).

Skeletal tracking system is an example of traditional machine learning using Microsoft *Kinect* which uses the signals from a depth video camera to perform real-time tracking of the full human skeleton on low-cost hardware (Shotton et al., 2011). This system is based on a technique known as *random forests of decision trees*, and the training data consists of one million depth images of human body poses, each of which is labelled with body parts (right hand, left knee, etc.). This example shows that model design must take into account training data set and labelling. In the design of training, the parameters of the system which are the selected features and thresholds at the nodes as well as the depths of the trees themselves, are determined in the laboratory during the training phase. Once the performance of the system is

satisfactory, the parameters are then fixed, and identical copies of the trained system are provided to its millions of users.

General approaches to algorithm design: Divide and conquer, Greedy method, Dynamic Programming, Basic Search and Traversal Technique, Graph Theory, Linear Programming, Approximation Algorithm, NP Problem. The complexity of an algorithm is simply the amount of work the algorithm performs to complete its task. RAM model has one processor, executes one instruction at a time, each instruction takes "unit time", has fixed-size operands, and has fixed size storage (RAM and disk).

What's more important than performance? Modularity; Correctness; Maintainability; Functionality; Robustness; User-friendliness; Programmer time; Simplicity; Extensibility; Reliability (Nilagupta, 2015). Measures the efficiency of an algorithm or its implementation as a program as the input size becomes very large We evaluate a new algorithm by comparing its performance with that of previous approaches Comparisons are asymtotic analyses of classes of algorithms We usually analyze the time required for an algorithm and the space required for a data structure.

In lieu of some standard benchmark conditions under which two programs can be run, the algorithm's performance is estimated based on the number of key and basic operations it requires to process an input of a given size For a given input size *n* the time T to run the algorithm is expressed as a function *T(n)*. Concept of growth rate allows comparison of running time of two algorithms without writing two programs and running them on the same computer (Nilagupta, 2015).

2.9.2 Model Validation

Validation is the process of determining the degree to which a model and its associated data are an accurate representation of the real world from the perspective of the intended uses of the model. The validation phase of research focuses on the agreement between the observed behaviour of elements of a system with the corresponding elements of a simulation model of the system and on determining whether the differences are acceptable given the intended use of the model. If a

satisfactory agreement is not obtained, the model is adjusted to bring it in closer agreement with the observed behaviour of the actual system (or errors in observation/experimentation or reference models/analyses are identified and rectified). The substantiation that a model is valid, i.e., performing model validation and verification, is generally considered to be a process and is usually part of the (total) model development process (Sargent, 2013).

Models can be confirmed by the demonstration of agreement between observation and prediction. Models can only be evaluated in relative terms, and their predictive value is always open to question. The term validation does not necessarily denote an establishment of truth (although truth is not precluded). Rather, it denotes the establishment of legitimacy, typically given in terms of contracts, arguments, and method.

a) Common Validation Techniques

There are various validation techniques and tests used in model validation by many researchers. In this section of study some of the most common techniques, which are also found in the literature, are selected although some may be described slightly differently. They can be used either subjectively or objectively. A combination of techniques is generally used. These techniques are used for validating and verifying the sub-models and overall model.

Animation: The model's operational behaviour is displayed graphically as the model moves through time. For example the movements of parts through a factory during a simulation run are shown graphically.

Comparison to other models: Various results (e.g., outputs) of the simulation model being validated are compared to results of other (valid) models. For example, (1) simple cases of a simulation model are compared to known results of analytic models, and (2) the simulation model is compared to other simulation models that have been validated.

Face Validity involves asking individuals knowledgeable about the system whether the model and/or its behavior are reasonable. For example, is the logic in the conceptual model correct and are the model's input-output relationships reasonable.

Historical Data Validation: If historical data exist (or if data are collected on a system for building or testing a model), part of the data is used to build the model and the remaining data are used to determine (test) whether the model behaves as the system does. This testing is conducted by driving the simulation model with either samples from distributions or traces (Sargent, 2009)

The most common method of validation involves a comparison of the measured response from in situ testing, lab testing, or natural analogs with the results of computational models that embody the model assumptions that are being tested. But the agreement between any of these measures and numerical output in no way demonstrates that the model that produced the output is an accurate representation of the real system. Validation in this context signifies consistency within a system or between systems. Such consistency entails nothing about the reliability of the system in representing natural phenomena.

b) Comparative Validation Metrics

This study have used comparative algorithm running time to validate developed algorithm against existing approach. Many factors affect the running time of a program. Some relate to the environment in which the program is compiled and run. Such factors include the speed of the computer's CPU, bus, and peripheral hardware. Competition with other users for the computer's (or the network's) resources can make a program slow to a crawl. The programming language and the quality of code generated by a particular compiler can have a significant effect. The "coding efficiency" of the programmer who converts the algorithm to a program can have a tremendous impact as well. All the above factors can be relevant if you need to get a program working within time and space constraints on a particular computer. To eliminate bias when comparing two programs derived from two algorithms for solving the same problem, they should both be compiled with the same compiler and

run on the same computer under the same conditions. As much as possible, the same amount of care should be taken in the programming effort devoted to each program to make the implementations "equally efficient". In this sense, all of the factors mentioned above should cancel out of the comparison because they apply to both algorithms equally. (Cooper et al., 2014).

Of primary consideration when estimating an algorithm's performance is the number of basic operations required by the algorithm to process an input of a certain size. The terms "basic operations" and "size" are both rather vague and depend on the algorithm being analyzed. Size is often the number of inputs processed. For example, when comparing sorting algorithms, the size of the problem is typically measured by the number of records to be sorted. A basic operation must have the property that its time to complete does not depend on the particular values of its operands. Adding or comparing two integer variables are examples of basic operations in most programming languages. Summing the contents of an array containing n integers is not, because the cost depends on the value of n (i.e., the size of the input) (Shaffer, 2011).

2.10 Research Gaps

In conclusion the literature review identified the following gaps:

a) The most widely used posterior inference methods in Bayesian nonparametric models are the Markov Chain Monte Carlo (MCMC). MCMC methods are guaranteed to converge to the posterior with enough samples. A research gap exists to speed MCMC samplers for application in large data settings since the samplers must be run for many iterations before convergence.

b) Much of existing published work is in the architecture of topic models and that there are other components which are required for a complete topic modelling process. There is therefore a need to develop a framework that helps to bridge the gap between the scattered literature involving model architecture theory and definitions provided in research publications and other practical steps believed to affect the quality of model output in the deployment of topic

models. Ultimately, the framework developed in this research provides a holistic view of any topic model deployment task.

These two major gaps identified in literature have not been addressed adequately in the past.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter describes the general research methodology used in this thesis. Research methodology links theory and the data for purposes of analysis thereby gaining insights into the research problem. The research foci and the philosophical stance on knowledge are key inputs into the methodology. Further the methodology adopts a strategy and methods that are well aligned for the research study.

3.2 Research Design

The study presented in this work involved identifying key features in deployment of topic models, existing inference algorithms and implementation of the Latent Dirichlet Allocation (LDA). The implementation of LDA assisted in determination of topic model identified features' behaviour in large data settings. Further, the study developed and validated a framework and parameter estimation method to address the lack of a topic model deployment framework and slow parameter computation time as identified in the thesis problem statement.

For framework development, the researcher conducted a state-of-the-art academic research on topic modelling. Based on a systematic review of literature published over the past ten years, the research synthesized various perspectives into a comprehensive generic framework that guides topic model deployments.

After framework development, an experimental environment was set up in order to study the trend of topic model input parameter values. Different existing datasets were run in the environment and the output of each dataset closely evaluated to detect any possible patterns that can be modelled. The researcher used these datasets as secondary data together with some codes and tools in frameworks mainly from github for purposes of experimentation.

The research first set up GraphLab environment (Low, et al., 2014) on jupyter notebook (Kluyver, et. al., 2016) in the Google cloud compute engine's custom machine and then developed python code to run on this environment. The code was used to manipulate the data in the desired way and iterated many times on given datasets. Results were displayed on a two dimensional graph using python code. This allowed the research to observe the behaviour of parameters of interest under different datasets, thereby conceptualising the set modelling goals. This procedure was repeated for different datasets with an aim of generalisation.

3.3 Design of Experiment

Experimentation was done under the Google cloud compute engine's custom machine accessed with a personal laptop computer. Python 3.6 from Continuum Analytics Anaconda was set up in the created virtual environment and scikit-learn, gensim and graphlab create libraries configured to work in jupyter notebook. After logging in the gl-env, jupyter notebook was accessed through the browser and all developed python codes typed and executed. A full version of these codes are contained in the appendix of this thesis.

Coding started by importing the dataset and CountVectorizer function from scikit-learn. The CountVectorizer function which is contained in sklearn.feature_extraction module of scikit-learn library was used to convert collection of text documents to a matrix of token counts to produce a sparse representation of the counts using scipy.sparse.csr_matrix.

3.4 Research Procedure

Four main phases were used for this research, namely, to identify topic model features and inference algorithms, to develop a generic topic model deployment framework, to implement LDA in python programming environment and to validate the developed framework and Method. The figure 3.1 summarizes the used procedure.

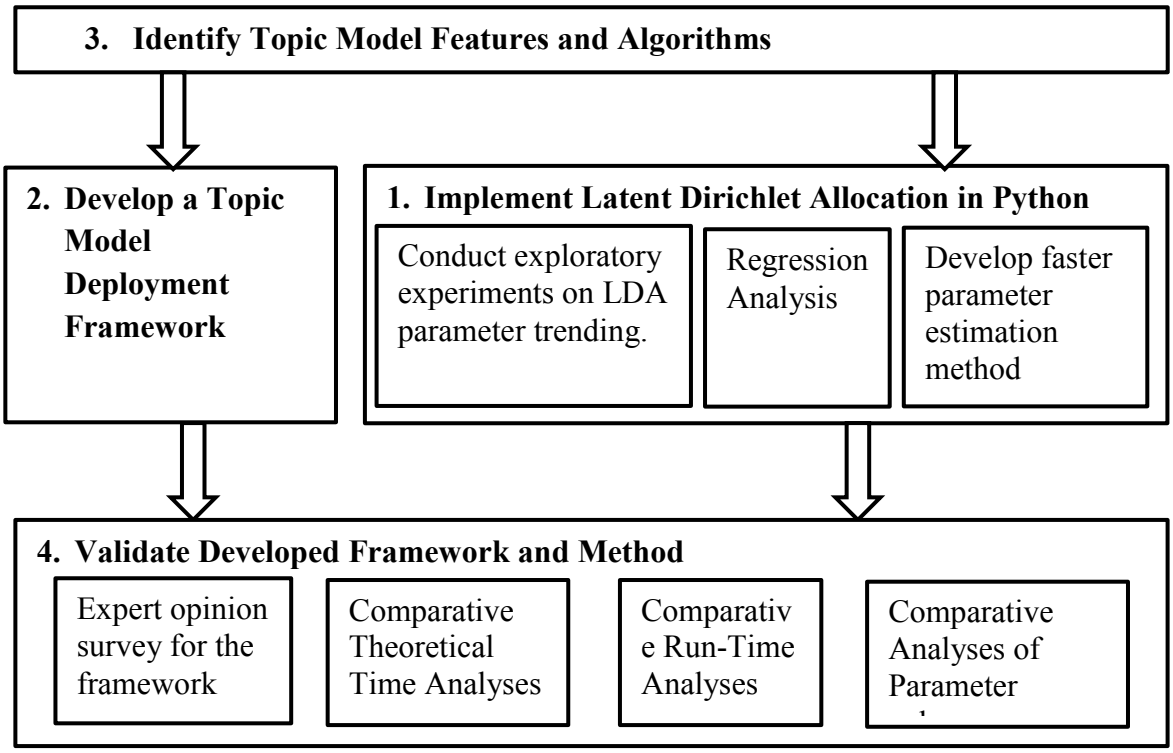


Figure 3.1: Diagram Showing Used Research Procedure

3.4.1. Identification of Topic Model Features and Algorithms

The following key features of topic modelling were identified by analysing existing publications: probability distributions of prior, likelihood and posterior; Set of parameters associated with the topic model, and methods of estimating input parameter values to fit the model; Mathematical notation used in analysis; how to interpret data obtained from the model and how data can be visualised for further interpretation. Two main categories of inference algorithms, that is sampling and optimization approaches were also considered in this study. Analysed content included mathematical and statistical research findings, machine learning algorithmic implementations and their published results.

3.4.2. Developing Topic Model Deployment Framework

This step involved review of current scientific knowledge about the process of implementing topic models, summarizing the reviews and representing the summary in a formal directed network diagram. This network diagram is presented in *Figure 4.1*. The following major components of topic model deployment were identified, described and connected by arcs to show a logical flow of the process: (1) Data pre-processing activities to mainly consist of: corpus collection, tokenization, normalization, spelling correction, conversion of corpus and definition of the execution environment input parameters and procedures; (2) Topic model architecture to mainly describe the interface, define the inference method and associated model parameters, algorithms and optimization, approaches to select optimal parameter and visualization for better utility; And, (3) the topic model products for example news summarizers and sentiment analysis. An expert opinion survey was carried out to validate this framework.

3.4.3. Implementation of LDA in Python

This step conducted exploratory experiments on LDA parameter trending and developed a faster parameter estimation method. To accomplish parameter trending experiments, the study first set up a virtual machine on Google cloud and installed python development environment. This was followed by identification of python modules implementing LDA and the frameworks containing these modules. The research further learned how to call identified procedures, loading data and interpreting results. Further codes were developed in jupyter notebook to assist in manipulating input and analysing the output.

After identifying a key trend when optimising for number of topics input parameter, regression analysis was done to estimate the strength and direction of the relationship between perplexity and number of topics. The researcher conducted both linear and quadratic regression in order to determine which model fitted the data better. The R function *lm()* was used in both cases. According to Dalgaard (2008) the R function *lm()* can be used for polynomial regression to include second-order and higher

powers of a variable in the model along with the original linear term. Quadratic regression in this study involved squaring the independent variable and fitting the *lm()* function on the resulting polynomial. The following line of code presented in Dalgaard (2008) was used for quadratic regression:

$$> summary(lm(Y \sim X + X^2)),$$

where Y is the dependent variable and X is the independent variable in the mathematical relationship. Details of the R code used for quadratic regression analysis is presented in Appendix VI of this thesis.

Substitution method of solving quadratic simultaneous equations together with differentiation principles at a turning point was used to develop a faster parameter estimation method. Python code to test this method for speed against existing method was developed and implemented.

3.4.4. Validating the New Framework and Method

This section of the chapter presents the methodology adopted to validate the developed Topic Model Deployment Framework and Quadratic Topics Approximation Method for Large Text Analytics. The section provides an expert opinion survey used to validate the framework and algorithm time analysis used to validate the method.

a) Expert Opinion Survey

An expert opinion survey was undertaken as part of the study, in order to validate the developed generic topic model deployment framework. The goal of the survey was to investigate expert opinion concerning the framework on two dimensions: Completeness and Dependability.

The target population was identified as consisting of active researchers in any area of text analytics evidenced by presence of at least five published journal papers in domains such as language intelligence, information retrieval, natural language processing, and topic modelling. A sample was taken from the population using

judgmental and convenient sampling. Google scholar was used to identify researchers who have published at least five cited publications in the area of topic modelling within the last five years. Convenient sampling was used to filter only researchers whose email addresses could be obtained from the Internet. The framework, letter of introduction and a link to the questionnaire were sent to the obtained email address.

Data for the survey was collected from experts using a questionnaire administered online. A survey tool called Survey Monkey was used to develop the online version of the questionnaire. Respondents were asked to provide scores on various issues on a five point Likert scale. Descriptive statistics was used to analyse collected data and results discussed.

b) Asymptotic Time Analysis

Asymptotic time analysis was used to compare the time complexity of the new algorithm against the existing algorithm. To achieve this, the time complexity of the existing code used to find optimal value of K was computed and compared with time complexity of the new code used to find the same parameter. The results were presented and discussed.

c) Empirical Analyses of Run-time

Empirical validation of run-time was done through a set of experiments done to compare the run-time of the new method against existing method. The code section used to optimise *K* was modified to include CPU time execution for both the new and existing method. CPU execution time was recorded for different datasets and descriptive statistics used to analyse resulting data.

3.5 Conclusion

This chapter described the methodology used to achieve the objectives of the research. The methodology involved descriptions of topic model features and algorithms, how a generic topic model deployment framework was developed and validated, the procedure used to implement LDA in python and how a new inference method was developed and validated for estimating number of topics parameter faster.

CHAPTER FOUR

TOPIC MODEL DEPLOYMENT FRAMEWORK

4.1 Introduction

This chapter is dedicated to solving objectives one and two of the thesis. The said objectives were to investigate topic model's components and to develop a topic model deployment framework for large text analytics. The chapter starts by defining a framework, identifying attributes of topic models, describing in details the components of the Topic Model Deployment Framework (TMDF) and finally validation of the developed framework. TMDF is a logical structure intended to provide a comprehensive representation and categorization of topic model deployment process.

4.2 Framework Requirements

A Topic Model as discussed in literature is used to automatically extract what topics are contained in large volumes of text. Some examples of large text could be feeds from social media, customer reviews of hotels, movies, user feedbacks, news stories and e-mails of customer complaints. A critical review of literature reveals that the main challenge in Topic Modeling is how to extract good quality of topics that are clear, segregated and meaningful. LDA's approach to topic modeling is to consider each document as a collection of topics in a certain proportion. Each topic is treated as a collection of keywords in a certain proportion.

Once the algorithm is provided with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution. A topic is a collection of dominant keywords that are typical representatives. Just by looking at the keywords, the model user can identify what the topic is all about.

Topic modelling is a use case of natural language processing (NLP). There are some identified challenges that need to be tackled for NLP and by extension applies to topic modeling. The first popular challenge is application domain. Higher accuracy is

needed for specific domains compared to generic domains. This calls for more components being included in the model and a careful selection of model evaluation metrics and their associated hypothesis rejection criteria. The second is the language in which the corpus exists. English being an international language has received the most attention and therefore there are many components realised as tools for different programming environments. This topic highlights some of the most popular tools and demonstrates how they are applied to data in section 4.4.5. Lastly is the medium. Processing speech is more difficult than processing text. To resolve some of these challenges, a user must become familiar with existing data and understand the challenges of the particular use case. The two basic questions are: How will the data be acquired? How will the quality of the data be validated?

In topic modelling, the following are key factors to obtaining good segregation topics: (1) Quality of text pre-p`rocessing; (2) Variety of topics the text talks about; (3) Choice of topic modeling algorithm; (4) Number of topics fed to the algorithm; (5) Algorithms tuning parameters. These key factors are the basis of designing the deployment framework.

4.3 Topic Models Attributes

This section presents some key features of topic models based on Latent Dirichlet Allocation (LDA). It has been noted in literature that LDA is based on a simple premise that there are *latent topics* in a collection of documents (corpus). There is a *generative process* to create the documents and the *inference algorithm* to extract the topics as well as their proportion and word assignment in each document from the input of the observed collection of documents. The basic intuition of LDA is threefold: First, the corpus, such as the articles in a sample journal or their abstracts, covers a hidden or latent set of K topics. The number of topics, K, is the key parameter of LDA. Second, each topic is defined by a probability distribution over words of the vocabulary that co-occur frequently, i.e. there are words that occur with high probability in the given topic, while others are extremely unlikely. Typically the words with high probability will specify the topic. For instance the topic of *missing data* may contain terms such as *missing, impute, list wise* with high probability.

Third, each document, specified by its words, covers typically a mix of topics with varying proportions for each topic. The number of topics per document with substantial proportion can vary.

4.4 Framework Architecture

A framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful (Trienekens et al., 2008). It is a layered structure indicating what kind of programs can or should be built and how they would interrelate. A framework may be for a set of functions within a system and how they interrelate; the layers of an operating system; the layers of an application subsystem or how communication should be standardized at some level of a network.

Drawing on the framework requirements and topic model attributes, a topic modelling framework is conceptualised as the process that starts with pre-processing of raw data which is fed into a statistical process called topic model architecture to give an output of topic model products. This process happens in an environmental context. Both the data pre-processing and statistical process influences the quality of output. The statistical process concerns determination of probability distribution functions to describe the data. These distribution functions are guided by some parameters which determine the quality of the output.

4.4.1 Data Pre-Processing

Practical work in Natural Language Processing (NLP) typically uses large bodies of linguistic data, or corpora. Data preprocessing is often the first step in the pipeline a NLP system, with potential impact in its final performance. Despite its importance, text preprocessing has not received much attention in the topic modeling literature. The goal of this section is to identify preprocessing techniques and later describe how they can be achieved in Python programming.

a) Pre-processing Overview

Language is complex but not all of language's complexity is necessary to effectively analyze texts (Grimmer & Stewart, 2013). In this stage, a recipe for transforming text into quantitative data is presented. Each of the steps are designed to retain information that will be used by the automated methods, while discarding information that will likely be unhelpful, ancillary, or too complex for use in a statistical model. The recipe is easy to apply, with several pieces of freely available software able to apply the sequence of described steps. A recipe for representing text quantitatively is also presented which emphasize that any one of the presented steps should, be modified. More important than following any individual recipe is to think carefully about the particular problem at hand, test different approaches, and validate the results.

Throughout, the unit of analysis is referred to as a *text* or *document*. This applies to the following example texts: a tweet, Facebook status, spoken utterance, press briefing, sentence, or paragraph. We refer to the population of texts to be analyzed as the *corpus* and a collection of these as corpora. Some methods will work better on different types of tasks or documents of different lengths, but most methods begin in the same way with a series of pre-processing steps to reduce the awe inspiring diversity of language to a manageable set of features.

The order of words in the documents is discarded into a bag of words format, where order does not inform desired analyses. While it is easy to construct sample sentences where word order fundamentally changes the nature of the sentence, empirically these sentences are rare. A simple list of words, called unigrams, is often sufficient to convey the general meaning of a text. If this assumption is unpalatable, some word order can be retained by including bigrams (word pairs) or trigrams (word triples) in the analysis (Jurafsky & Martin, 2009). This allows the differentiation for example of the "White House" from the color and the domicile. In practice, for common tasks like measuring sentiment, topic modeling, or search, n-grams do little to enhance performance (Hall et al., 2008; Hopkins & King, 2010; Grimmer & Stewart, 2013).

After discarding word order, resulting vocabulary is simplified with stemming. Stemming removes the ends of words to reduce the total number of unique words in the data set, or reduce the dimensionality of text. Stemming reduces the complexity by mapping words that refer to the same basic concept to a single root. For example, family, families, families', and familial all become famili. Stemming is actually an approximation to a linguistic concept called lemmatization, which seeks to reduce words to their base forms (Hall et al., 2008; Jurafsky & Martin, 2009). The critical difference is that a lemmatizer uses context and dictionaries to help discover (for example) that good is the base form of better and best. The stemmer is a much cruder algorithm, but considerably faster. The performance does not seem to matter for most applications, so the majority of applied research uses stemming. There are numerous stemming algorithms available that vary in the extent and frequency of word truncation. But the Porter stemming algorithm (Porter, 2008) is commonly employed because of its moderate approach to word simplification.

In addition to discarding word order, the following are also discarded: punctuation, capitalization, very common words (often "stop" words are removed, or function words that do not convey meaning but primarily serve grammatical functions), and very uncommon words (words that appear only once or twice in the corpus and thus are unlikely to be discriminating). Typically words which appear in less than 1% and more than 99% of documents in the corpus are removed, although these choices need to be made contingent both on the diversity of the vocabulary, average length of the document, and the size of the corpus (Quinn et al., 2010; Hopkins & King 2010).

The result of the pre-processing steps is that each document i (i=1, , N) is represented as a vector that counts the number of times each of the *M* unique words occur, $(W_i = W_{i1}, W_{i2}, \ldots, W_{iM})$ Each W_{im} counts the number of times the *m-th* word occurs in the *i-th* document. The collection of count vectors into a matrix is often called the document-term matrix. For a moderate volume of documents without a particularly specialized vocabulary, this matrix will have between three thousand and five thousand features or terms and will contain mostly zeroes (a condition called sparsity).

The data pre-processor eliminates uninformative part of the corpus and creates a statistical object for analysis. Mathematically this statistical object is a high-dimensional, sparse feature vector where the elements of the vector are the frequencies of specific words and phrases in the corpus while geometrically it is a point in a high-dimensional space. This means that anything you can do with points, you can do with documents. In text analytics, data pre-processing involves two main activities: document gathering and text pre-processing. In the document gathering step, the text documents are collected which are present in different formats (Vandana & Namrata, 2012). The document might be in form of pdf, word, html, doc, css etc. These documents are passed to the second step.

In the pre-process step, the given input document is processed for removing redundancies, inconsistencies, separate words and stemming. Documents are then prepared for next step (Vandana & Namrata, 2012; Sagayam et al., 2012). The stages performed are as follows: (a) Tokenization: This stage involves breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing (Gupta & Lehal, 2009). (b) Removal of Stop words. In this step the removal of common words that do not contribute to thematic sense of documents like ‘a’, ‘an’, ‘but’, ‘and’, ‘of’, ‘the’ etc. is done (Johnson et al., 2002). Stop words are a division of natural language. The motive that stop-words should be removed from a text is that they make the text look heavier and less important for analysts. Removing stop words reduces the dimensionality of term space. The most common words in text documents are articles, prepositions, and pro-nouns, etc. that does not give the meaning of the documents. These words are treated as stop words. Example for stop words: the, in, a, an, with, etc. Stop words are removed from documents because those words are not measured as keywords in text mining applications (Kannan & Gurusamy, 2014). (c) Stemming: Stemming involves matching the morphological variants of words to their root word. e.g. if a document pertains words like resignation, resigned, resigns then it will be consider as resign after applying stemming method (Singh & Gupta, 2017).

b) Pre-Processing Techniques

Pre-processing the data is the process of cleaning and preparing the text for analytics. Texts usually contain lots of noise and uninformative parts such as HTML tags, scripts and advertisements in online cases. In addition, on words level, many words in the text do not have an impact on the general orientation of it. Keeping those words makes the dimensionality of the problem high and hence the classification more difficult since each word in the text is treated as one dimension. Here is the hypothesis of having the data properly pre-processed: to reduce the noise in the text should help improve the performance of the algorithms and speed up the classification process, thus aiding in real time sentiment analysis (Haddi, et al., 2013)

For word-based contexts, there are a number of associated tasks that must happen before any semantic processing takes place. The first one is tokenization which involves splitting the text into individual words. Second is token normalization, which depending on the task can be as simple as case folding (discarding information about letter casing) or as complex as lemmatization. The third item is correction which deals with ambiguous spelling variations such as “Samsung S8” vs. “Sumsung S-8” vs. “Sumsung S 8”, or “2 thou” vs. “2k”, “don’t” vs. “dont” and “state-of-the-art” vs. “state of the art” etc.

Depending on the application, the desired course of action may be to use the exact form as it appears in the text, or normalize the tokens to a single canonical form (either during the tokenization phase or later at query resolution). In the fourth pre-processing task, multi-word expressions, more complex lexical units are handled for example include dates (“12.4.2010”), IP addresses (“147.251.48.1”), ranges (“10–55”), abbreviations or compound tokens in general, including emoticons (“:-)”, “<3”) (Rehurek, 2011).

All of these choices interact and affect a system’s performance. They are also crucial in the sense that errors at this fundamental level are very costly to correct later on. These pre-processing techniques are shown in Figure 4.1.

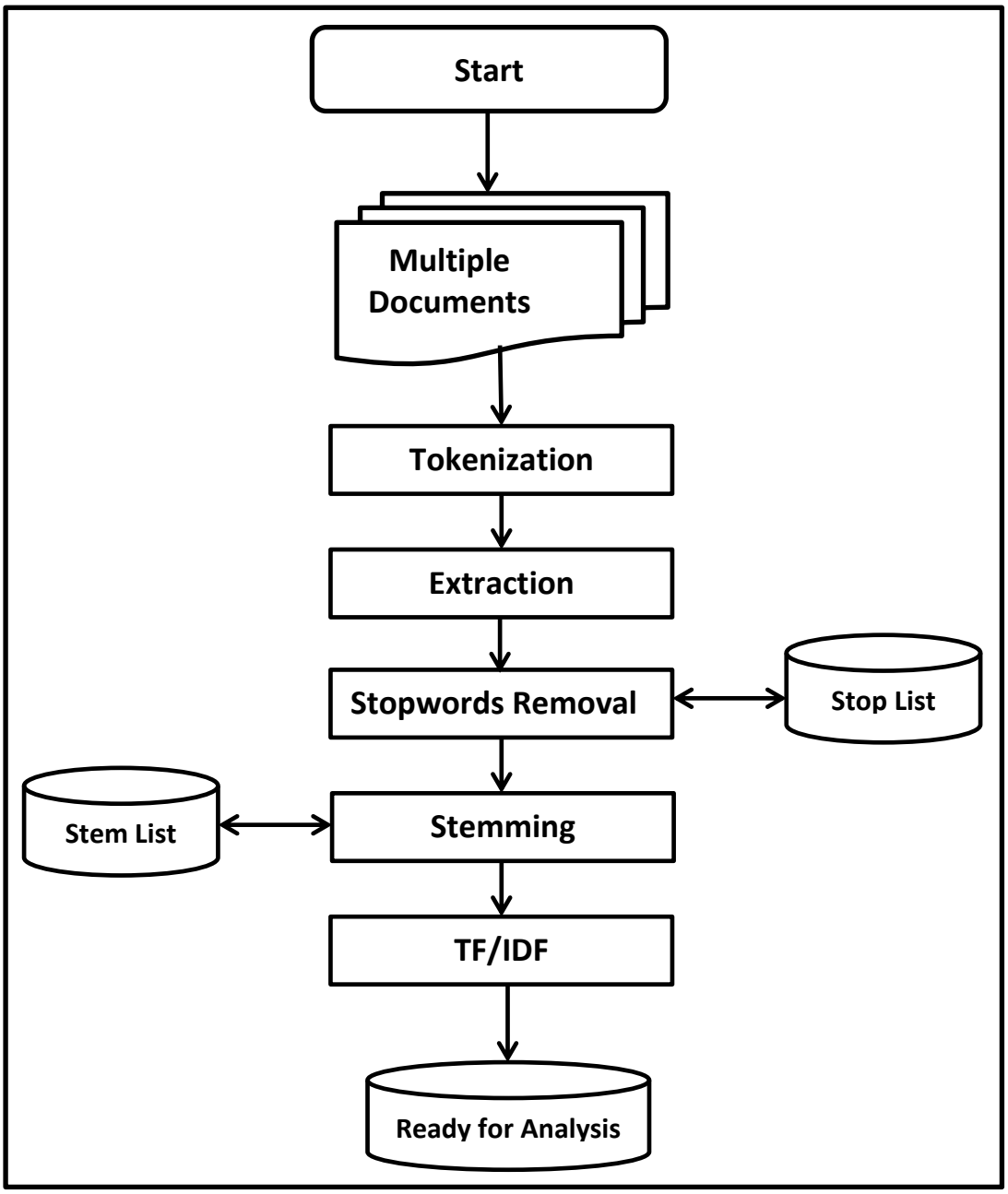


Figure 4.1: Text Analytics Pre-Processing Techniques
(Source: Vijayarani et al.,., 2015)

4.4.2 Topic Model Architecture

In the architecture of a topic model, various components are of interest. The first is the user interface which is an important function of any type of software regardless of whether it is an operating system or an application. A user-friendly interface should make it possible for the user to use the software without having to read the entire documentation first. An interface that is attractive to use will also encourage users to use the software. A good human-computer or user interface needs to observe important characteristics at design time in terms of user-friendliness, attractiveness, effectiveness and ease of use. The second is identification of suitable probability distribution functions that define the statistical process and associated model parameters. These are described in details in the literature review chapter of this thesis. In most cases, these functions are complex multinomials whose inference leads to intractable to solve equations thereby necessitating sampling or approximation methods. This thesis approximates a sampling approach in estimating hyper-parameters.

4.4.3 Topic Model Products

Topic models help users summarize large document collections and to find general themes present in the collections outside the context of any specific information need. This approach stands in contrast to traditional information retrieval systems, which retrieve relevant documents given users' explicit information needs. Following are some areas where topic models are applied.

4.4.4 Topic Model Deployment Pipeline and Framework

Following the discussion in sections 4.1, 4.2, 4.3 and 4.4 of this thesis, the following Topic Model (TM) deployment pipeline and TM deployment framework are conceptualised. The conceptual pipeline describes a high level topic modelling process in which the output of one module feeds into the input of the next module. This pipeline can be used to simplify TM applications design by breaking down the complexity of the task into a smaller set of less complicated tasks. This provides an

easier understanding of the requirements of TM. The pipeline is further expanded to the framework presented in Figure 4.3 for detailed explanations.

a) Topic Modelling Deployment Pipeline

Figure 4.2 shows the conceptualised *Topic Model Deployment Pipeline* that summarizes topic modelling deployment exercise. Working on a topic modelling task, requires that a set of processes and activities be performed. This set of processes constitutes the Topic Modelling Deployment pipeline presented on Figure 4.2.

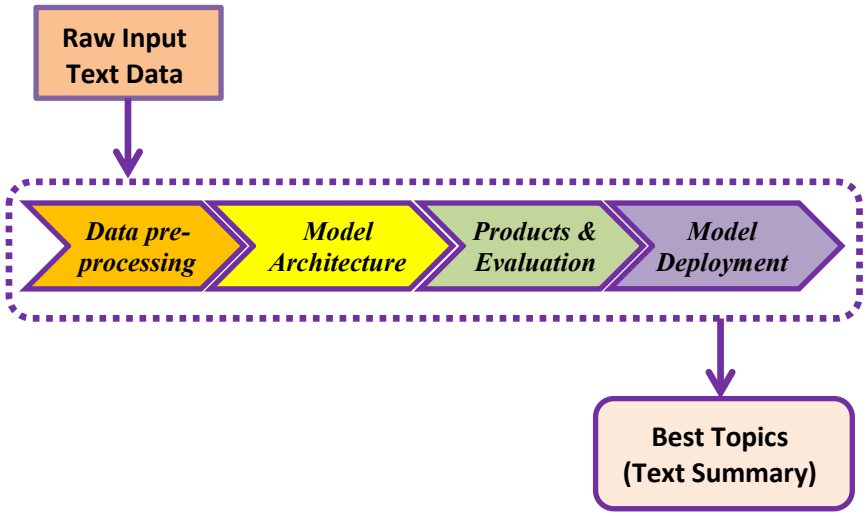


Figure 4.2: Topic Model Deployment Pipeline

b) Topic Model Deployment Framework

Figure 4.3 represents the Topic Model Deployment Framework, a framework for machine learning used in this study. It identifies details of various components of a topic model deployment task and their interactions. This framework simplifies the conception of topic modelling and helps data science researchers and topic model application developers easily focus on particular components of the broad area.

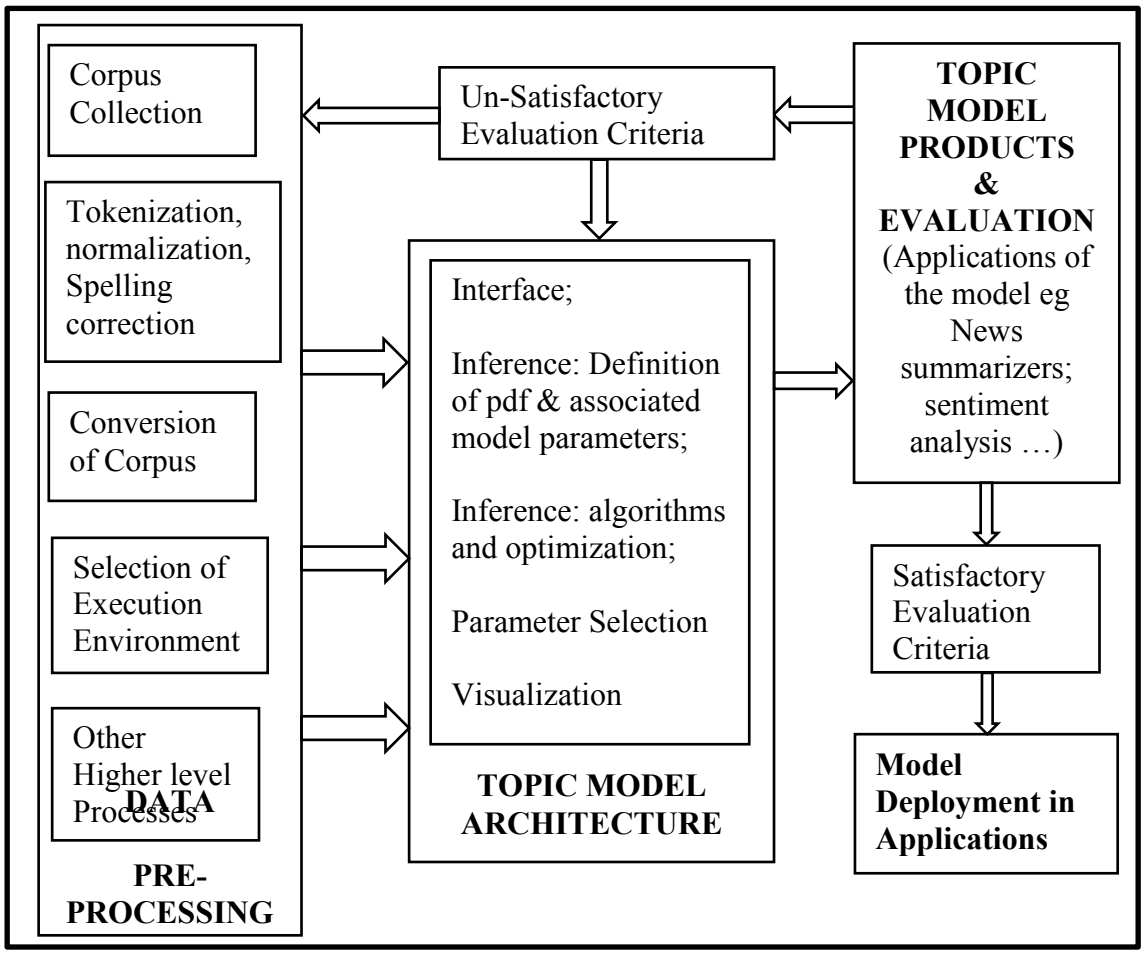


Figure 4.3: Topic Model Deployment Framework

4.5 Validating Topic Model Deployment Framework

Validation is defined as the assessment of an action, decision, plan, or transaction to establish that it is correct, complete, being implemented as intended, and delivering the intended outcome. A model is usually developed to analyse a particular problem and may therefore represent different parts of the system at different levels of abstraction. As a result, the model may have different levels of validity for different parts of the system across the full spectrum of system behaviour (Sargent, 2013).

Broadly speaking there are three approaches to model validation and any combination of them may be applied as appropriate to the different aspects of a particular model. These approaches are: expert opinion; theoretical analysis and real system measurements (Mellor-Crummey, 2005).

The aim of validating the framework was to achieve dependability and completeness of the said framework, in the context of large text analytics. In topic modeling, dependability and completeness are crucial to the success of product development. A missing requirement can mean a missing attribute in complex products, and a missing requirement can mean a major source of implementation defects in applications. The completeness of requirements is such a need, both in sense of whether the requirements' set contains all requirements and whether all requirements are completely specified (Zenun & Loureiro, 2013).

From a formal point of view, correctness is usually meant to be the combination of consistency and completeness. Consistency refers to situations where a specification contains no internal contradictions, whereas completeness refers to situations where a specification entails everything that is known to be "true" in a certain context. From a practical point of view, however, correctness is often more pragmatically defined as satisfaction of certain framework goals. This indeed is the kind of correctness which is more relevant to a framework consumer. In order to validate the framework, a survey was carried out to seek expert opinion concerning the framework on the two dimensions mentioned above: Completeness and Dependability.

4.5.1 Expert Opinion Survey

This section presents the design and administration of the text analytics expert opinion survey which was undertaken as part of the study, in order to validate the developed generic topic model deployment framework. Appendix IX presents the details of introduction letter used to introduce the survey to prospective respondents. Appendix X is the Generic Topic Model Deployment Framework which was sent together with the introduction letter. Appendix XI presents the Questionnaire used to collect expert opinion and Appendix XII presents a detailed sample individual response.

a) Goals of the survey

The goal of the survey was to investigate expert opinion concerning the framework on two dimensions: Completeness and Dependability. This validation was motivated by a need provide researchers and practitioners with a reliable framework that can be used in the design and deployment of topic model applications.

b) Survey Design

The design of the survey is constrained by the following goals: the survey instrument to be as straightforward as possible; outcomes to be suited to automatic processing; and completion by an individual respondent to be performed within a 15 minute period. These goals are intended to support the attraction of as wide a body of responses as possible.

4.5.2 Target Population and Sampling Plan

The target population for the study consisted of active researchers in any area of text analytics who were considered as experts. This was evidenced by presence of at least five published journal papers in domains such as language intelligence, information retrieval, natural language processing, and topic modelling.

The study took a sample from the population using judgmental and convenient sampling. Google scholar was used to identify researchers who have published at

least five cited publications in the area of topic modelling within the last five years. Convenient sampling was used to filter only researchers whose email addresses could be obtained from the Internet. The framework, letter of introduction and a link to the questionnaire were sent to the obtained email address.

A sample list of experts is shown in the Table 4.1. For purposes of assured confidentiality, only general data is included in the table.

Table 4.1 Sample List of Experts

SN	Name	Qualification	Publication credentials
1.	AA	Specialization: Computational Linguistics, head of the Computational Linguistics Lab in a university, deputy head of Cognitive Science research group, in a University.	Citations:- 232; h-index:- 6; i10-index:- 4.
2.	BB	Specialization: University Professor of Computer Science, director of the Language Intelligence and Information Retrieval (LIIR) research lab; bias in Natural language processing, machine learning, information retrieval, multimedia and text mining	Citations:- 5256; h-index:- 40; i10-index:- 107

4.5.3 Data Collection Procedure and Instruments

Data for the study was collected from experts using a questionnaire administered online. A survey tool called Survey Monkey was used to develop the online version of the questionnaire. Respondents were asked to provide scores on various issues on a five point Likert scale. Likert-type scale is easy to construct in comparison to Thurstone-type scale (Thurston, et al., 2014) and can be performed without a panel of judges. Likert type is also considered more reliable because the respondents answer all indicated questions.

4.5.4 Expert Survey Results

This section discusses the framing of the questions asked in each section of the survey, tabulates the responses received, and discusses the findings. As far as possible the tables show the exact wording of each question on the form, and all options presented on the form are listed even if no respondents selected them.

a) Respondents by highest academic qualification

Table 4.2 presents respondents by highest academic qualification of response received from the expert opinion survey.

Table 4.2 Respondents by Highest Academic Qualification

Academic Qualification	Number of Respondents (Frequency)	Percentage Response (%)
Bachelor's Degree	0	0
Master's Degree	8	44
PhD	10	56
Others	0	0
TOTAL	**18**	**100**

The pie chart in Figure 4.4 represents respondents highest academic qualification.

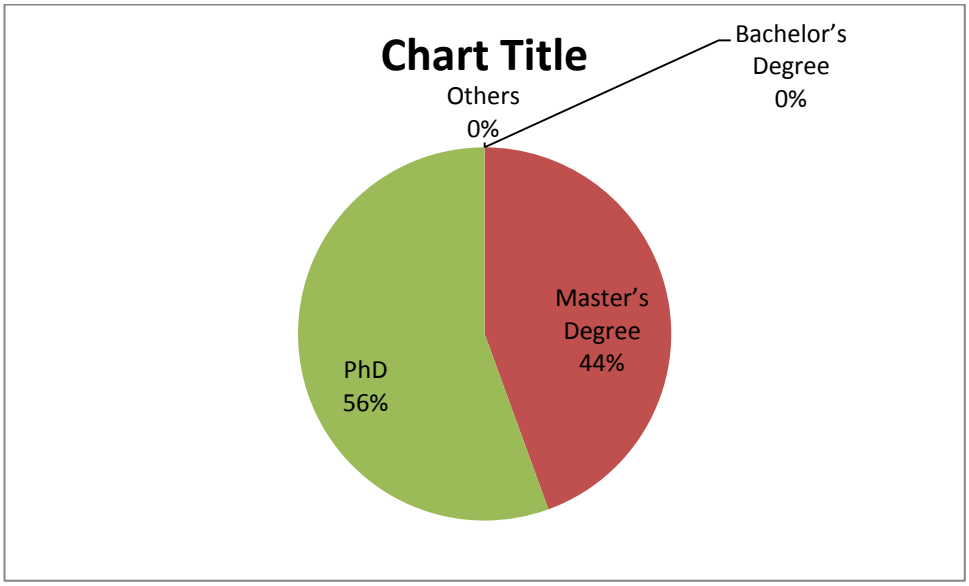


Figure 4.4 Respondents by Highest Academic Qualification

Figure 4.4 shows that PhD holder respondents had the highest percentage (56%) of inclusion in the sample of respondents. This is because a high percentage of questionnaires were sent to PhD holders. The other respondents were Master's Degree holders (44%). This result indicates that the highest percentage of respondents had academic qualifications that suggested they are experts in the field of topic modelling.

b) Respondents' Knowledge of Text Analytics

Table 4.3 Respondents' Knowledge of Text Analytics

Knowledge of Text Analytics Rank	Very Poor	Below Average	Average	Above Average	Excellent
Frequency	0	0	9	6	3
Percentage	0%	0%	50.00%	33.33%	16.67%

The Bar Chart in Figure 4.5 presents respondents' knowledge of text analytics.

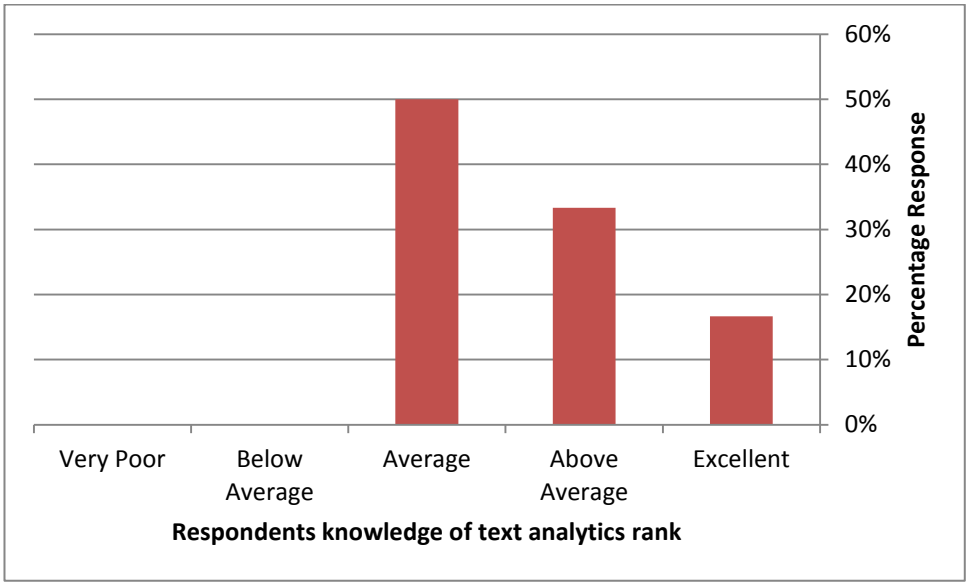


Figure 4.5: Respondents' Knowledge of Text Analytics

Figure 4.5 shows that the highest percentage of respondents (50.00%) considered themselves to have average knowledge in the field of text analytics, 33.336% of the respondents considered themselves above average in the knowledge of text analytics while 16.67% considered themselves excellent in the field of text analytics. This result means that all respondents considered themselves to be above average in their knowledge of text analytics.

c) Respondents' years of experience in Text Analytics

Table 4.4 presents respondents years of experience in text analytics.

Table 4.4 Respondents' Years of Experience in Text Analytics

Years of Experience	0-2	3-5	6-8	9-10	Above 10	Weighted Average
Frequency	0	6	6	4	2	3.11
Percentage Response (%)	0.00%	33.33 %	33.33%	22.22%	11.11%	

The bar chart in Figure 4.6 presents respondents' years of experience.

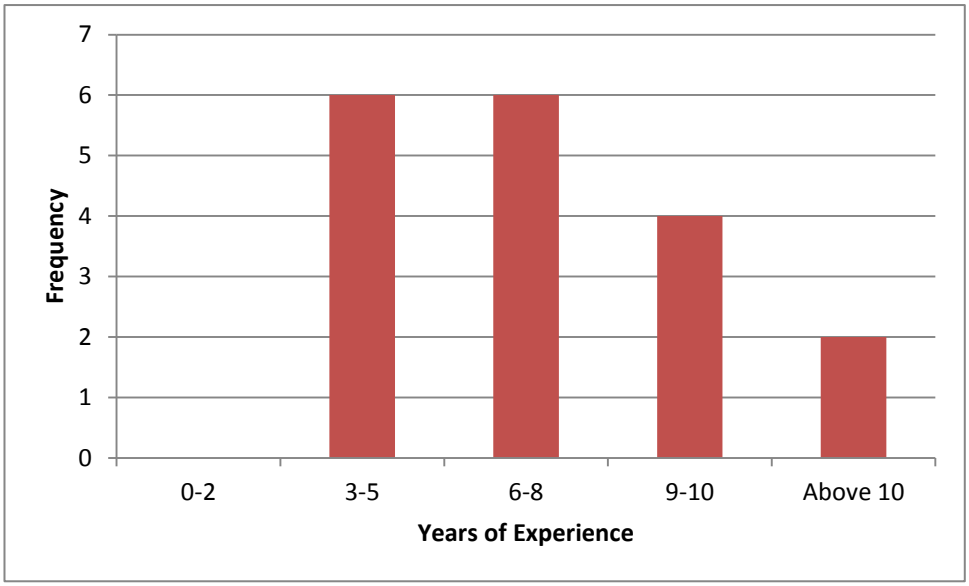


Figure 4.6: Respondents' Years of Experience in Text Analytics

The bar chart shows that respondents the category "3-5" and "6-8" recorded the highest frequency. Respondents in the category "Above 10" had the lowest frequency. Respondents weighted average is 3.11 where category "0-2" is weighted as 1, "3-5" is weighted as 2, "6-8" is weighted as 3, "9-10" is weighted as 4 and "Above 10" is weighted as 5. This means that the respondents weighted average years of experience is in the category "6-8" years.

d) Framework Constructs

On the developed framework, respondents were asked to state the importance of precisely determining identified constructs in the development and deployment of text analytics applications, on a five point likert scale. The results are presented in Table 4.5.

Table 4.5 Framework Constructs

CONSTRUCT		NOT IMPORTANT	SLIGHTLY IMPORTANT	MODERATELY IMPORTANT	IMPORTANT	VERY IMPORTANT	TOTAL	WEIGHTED AVERAGE
	WEIGHT	**1**	**2**	**3**	**4**	**5**		
Data preprocessing	Percentage Response	0.00%	5.56%	5.56%	44.44%	44.44%		
	Frequency	0	1	1	8	8	18	4.28
User Interface	Percentage Response	0.00%	0.00%	33.33%	66.67%	0.00%		
	Frequency	0	0	6	12	0	18	3.67
Selection of statistical model	Percentage Response	0.00%	5.56%	5.56%	16.67%	72.22%		
	Frequency	0	1	1	3	13	18	4.56
Selection of model parameters	Percentage Response	0.00%	0.00%	5.56%	22.22%	72.22%		
	Frequency	0	0	1	4	13	18	4.67
Expected products	Percentage Response	0.00%	0.00%	38.89%	55.56%	5.56%		
	Frequency	0	0	7	10	1	18	3.67

The bar chart in Figure 4.7 shows expert opinion on framework constructs survey results.

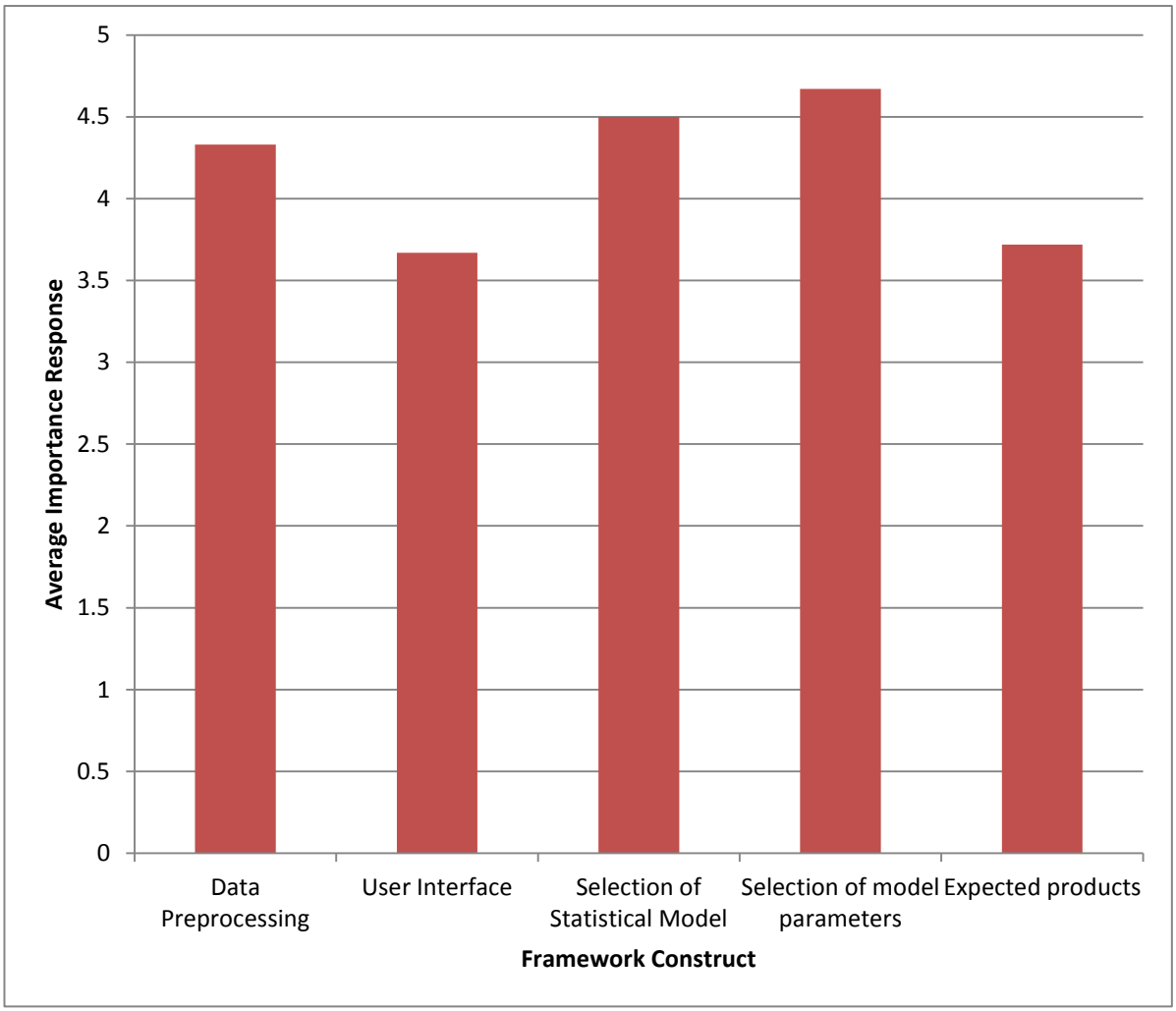


Figure 4.7: Expert Opinion on Framework Constructs

The weighted average for all framework constructs is beyond 3.5 mark on the 5-point likert scale. Data pre-processing has 4.33 indicating that it is an important construct, Selection of statistical model and Selection of model parameters have 4.50 and 4.67 respectively which corresponds to very important on the Likert scale. On the other hand User Interface and Expected products have 3.67 and 3.72 respectively which corresponds to important on the Likert scale. The overall expert opinion survey on framework constructs suggests that all the developed constructs are key.

4.6 Conclusion

In this chapter, the following components of topic model deployment were identified: Pre-processing activities, Topic model architecture and Topic model products and evaluation. Researchers and practitioners can use the framework to easily understand the process of topic model deployment to also identify areas of bottleneck.

The chapter further presents results of an expert opinion survey conducted to validate the developed framework. The results of this survey indicate that the developed framework comprehensively describes the process of topic model deployment. Respondents response show that all identified constructs are key for the deployment process.

CHAPTER FIVE

QUADRATIC TOPICS APPROXIMATION METHOD

5.1 Introduction

Implementation of LDA requires specification of the following three parameters: α and β which identify the Dirichlet priors and K which represent the number of topics in the corpus. In order to increase the robustness of topic models and improve model fitting, the parameters must be optimised. At optimal value, the parameters will minimize the model's perplexity on the held-out data.

This chapter contains an account of practical work involving LDA parameter trending, undertaken in this study, results obtained, design of a faster method to estimate optimal number of topics parameter and validation of the developed method. It starts by demonstrating implementation of Latent Dirichlet Allocation (LDA) using Collapsed Gibbs Sampling (CGS) inference approach. The chapter further carries out tests to determine the optimal parameters alpha, beta and K for three different datasets. The results for these experiments are presented on tables and graphs.

5.2 Experimental Dataset

20 Newsgroups dataset used in this experiment is a collection of approximately 20,000 newsgroup text documents, partitioned evenly across 20 different groups. The data can be found at GitHub and was originally collected by Lang (1995) for the 'learning to filter netnews' paper. The dataset has become popular for experiments in text applications of machine learning techniques, such as text classification and text clustering as evidenced in a lot of published work (Albishre & Li, 2015; Mekala et al., 2017; Dasgupta et al., 2007; Harish & Revanasiddappa 2017; Feng et al., 2017).

5.3 Experimental Environment

These experiments were done under the Google cloud compute engine's custom virtual machine with a specification of 8 CPUs, 8 GB memory and 64 GB boot and

local disk. The machine was accessed with a personal laptop with a CPU specified as Intel(R) Core(TM) i5-4210U CPU @ 1.70 GHz 2.40 GHz and a memory of 4.00 GB. The capacity of personal laptop hard drive is 500 GB with Microsoft Windows 8.1 single Language 64-bit Operating System, x64-based processor. The researcher setup Python 3.6 from Continuum Analytics Anaconda and configured scikit-learn, gensim and graphlab create libraries to work in jupyter notebook. These libraries provided different modules for the experimentation as required.

5.4 Training and Test Data

After data pre-processing and converting the data into a bag-of-words format, the utility for performing a random split for text data was called as follows:

g_train_data, g_test_data = gl.text_analytics.random_split(sa_train)

The utility uniformly partitions each (word, count) pair in a particular element to either a training set or a test set. The output is two data sets in bag-of-words format, where the combined counts are equal to the counts in the original data set.

5.5 Estimating Optimal Parameters

After passing the data file through preprocessing steps, the researcher performed data processing that are required for a topic modelling task. First optimal model parameters are experimentally determined by looping through different values and selecting the one with lowest perplexity. This experiment was repeated for beta and number of topics.

After several runs and working with different values of alpha holding beta and K constant at beta=0.01 and K=375, the researcher established that alpha has an optimal value between 0.06 and 0.4. The range was then subdivided equally into 100 parts by setting the variable num to 100. The graph in Figure 5.4 which illustrates the behaviour of alpha in the range mentioned was obtained.

5.6 Trending Experiments for LDA Input Parameters

Section 5.8 presents parameters trending experimental results after implementing LDA using CGS inference method on the politics subset of Newsgroup dataset. Appendix I provides details of python code used for analysing politics dataset using CGS in graphlab environment.

5.6.1 General Trend for Perplexity against Alpha

For this experiment, alpha was restricted within the values (0.01, 0.5) and num set to 100 in order to obtain sufficient data points to give the trend. The following general trend was obtained:

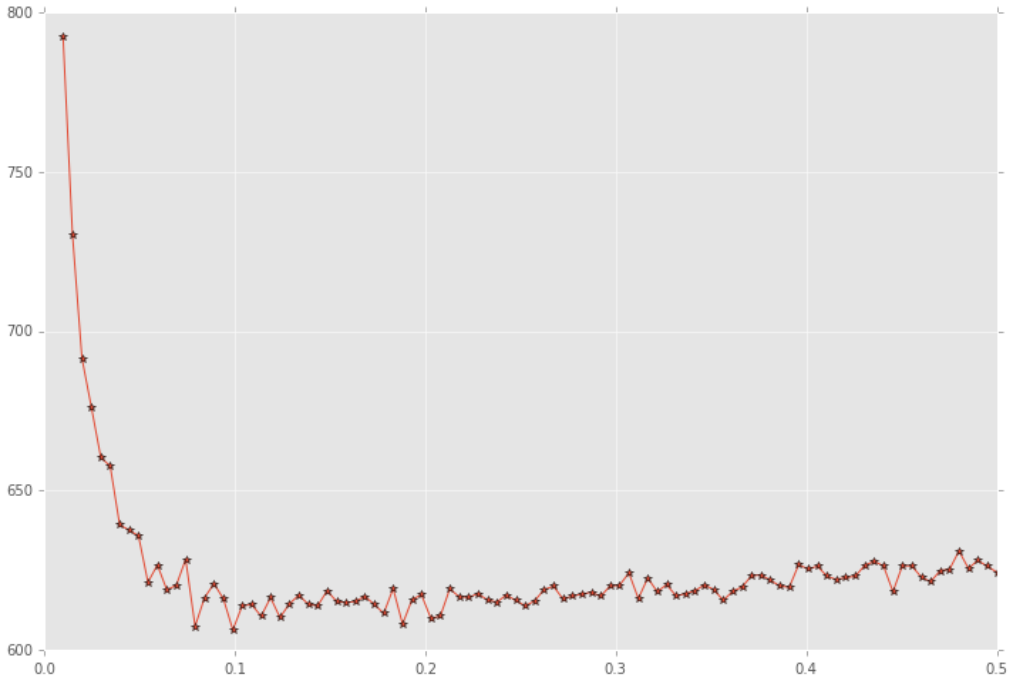


Figure 5.1: Perplexity against Alpha for 0.01<alpha<0.5

It is clear from the trend that an optimal value exists between 0.05 and 0.3.

5.6.2 Perplexity against Alpha for 0.05< alpha<0.3

For this experiment, alpha was restricted within the values (0.05, 0.3) and num set to 100 in order to obtain sufficient data points for modelling the trend. Figure 5.2 shows a line graph for the data obtained. Appendix III presents raw data obtained in the experiment where Number of topics (K) was fixed at 420, beta 0.01 and alpha starting at 0.05 to a maximum value of 0.3.

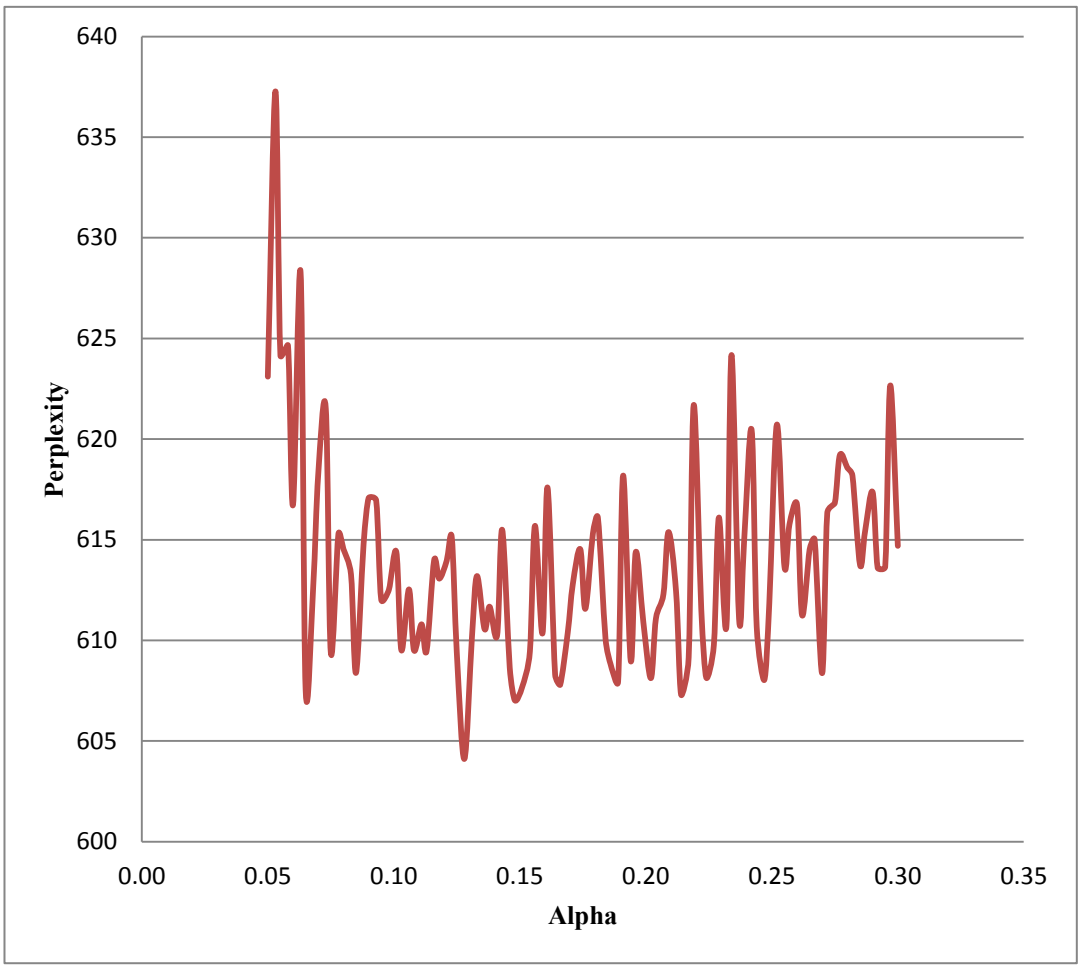


Figure 5.2: Perplexity against Alpha for 0.05< alpha<0.3

This graph shows that the value of alpha that minimises perplexity for this dataset is 0.15. This indicate that lower values of alpha yield the best result for perplexity.

5.6.3 Perplexity against Beta for 0.0001< Beta<0.02

After testing on many values, it was noted that beta has a maximum between 0.0001 and 0.02 ie (0.0001, 0.02, num=100). This is shown in Figure 5.3.

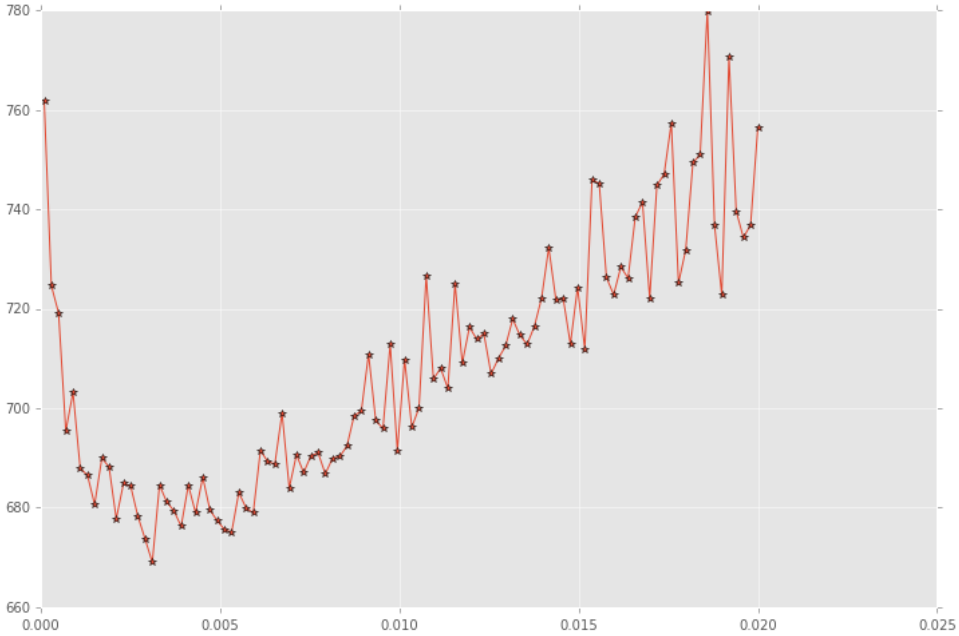


Figure 5.3: Perplexity against Beta for 0.0001< Beta<0.02

The trend shown in Figure 5.6 indicates that an optimum value of perplexity exists in the range of 0.002<beta<0.005 for this dataset.

5.6.4 Perplexity against Number of Topics (K) for 50< K<10,000

In this experiment, alpha was set at 0.1 and beta at 0.01 and K at 10,000, a sufficiently large value of K in order to determine the 'infinite' trend of perplexity against number of topics (K). The detailed data for this experiment is presented in Appendix IV. The graph presented in Figure 5.4 indicates that the value of perplexity increases monotonically beyond a point of minimum perplexity as K increases.

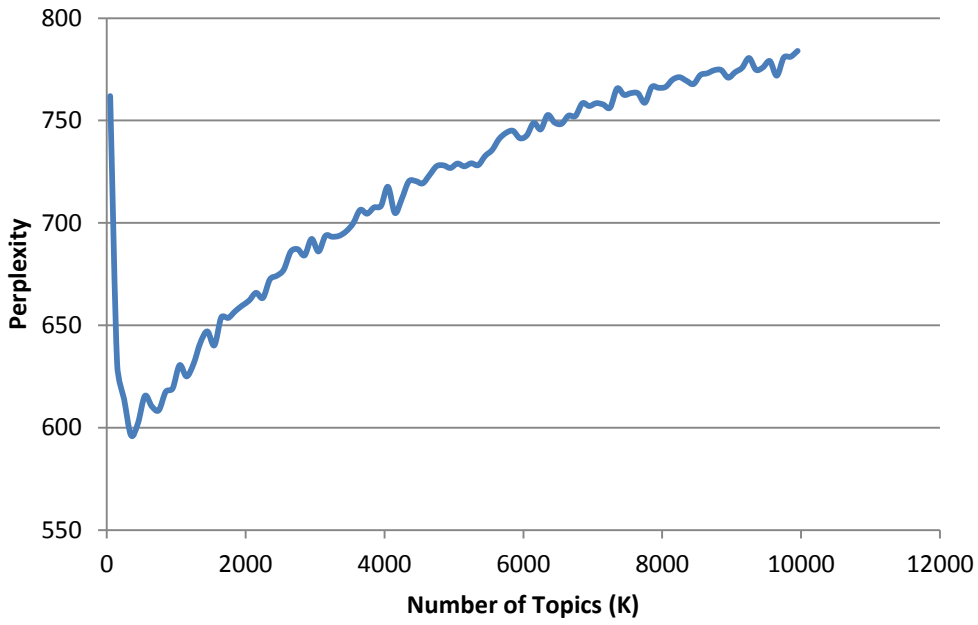


Figure 5.4: Perplexity against K for 50< K<10,000

From the general trend, it was observed that there is a minimum perplexity between a K value of 50 and for purposes of symmetry, an estimate of 690.

5.6.5 Autos Data Perplexity against Topics for 50< K<690

The experimental set up in section 5.6.2 was repeated in this section 5.6.3 changing the range of number of topics (K) to start at 50 and have a maximum of 690 with a step of 5 so as to investigate the behaviour of the parameter number of topics around the turning point. The step of five was chosen in order to increase precision in modelling K for this range. The table in Appendix VI and Figure 5.5 presents the results obtained in this experiment.

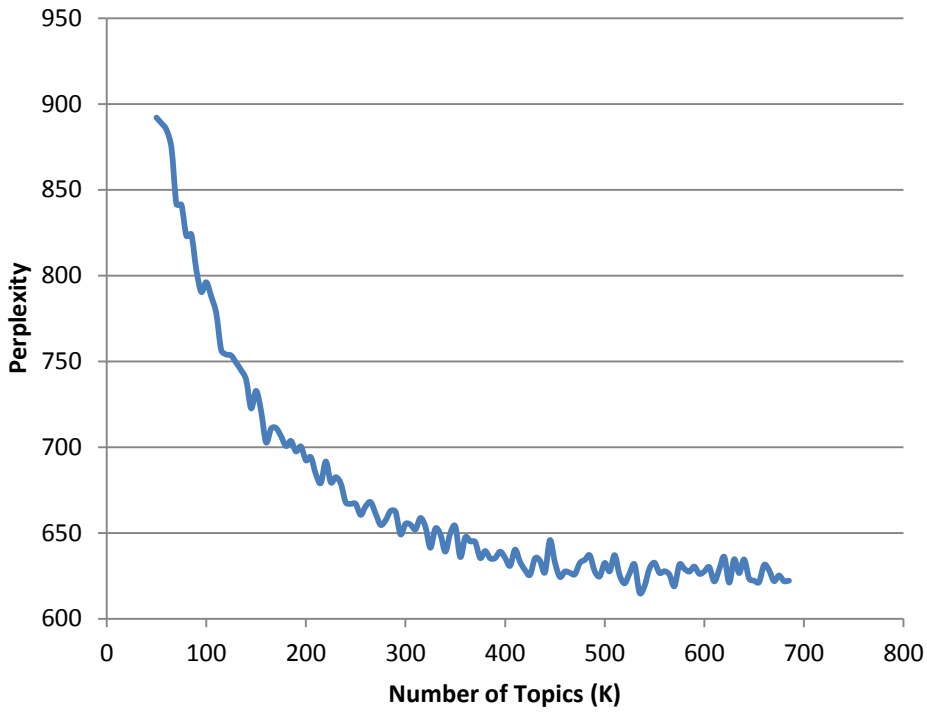


Figure 5.5: Autos Perplexity against Topics for 50< Topics<690

The results presented in Figure 5.5 indicates that a graph of perplexity against number of topics (Topics) have a strong quadratic behaviour around a turning point which opens upwards. This means the graph has a point of minimum perplexity that optimizes choice of Number of Topics (K). In order to ascertain the claim that the relationship between perplexity and number of topics is quadratic, regression analysis was carried out.

5.6.6 NYtimes Data Perplexity against Topics for 50< K<695

A similar experiment to the one in section 5.6.5 above was conducted using a different dataset, NYtimes, downloaded from *<https://static.turi.com/datasets/nytimes>* in order to confirm the results obtained from the first dataset. Following is a line graph for the data obtained for the range parameter values of 50<Topics<695

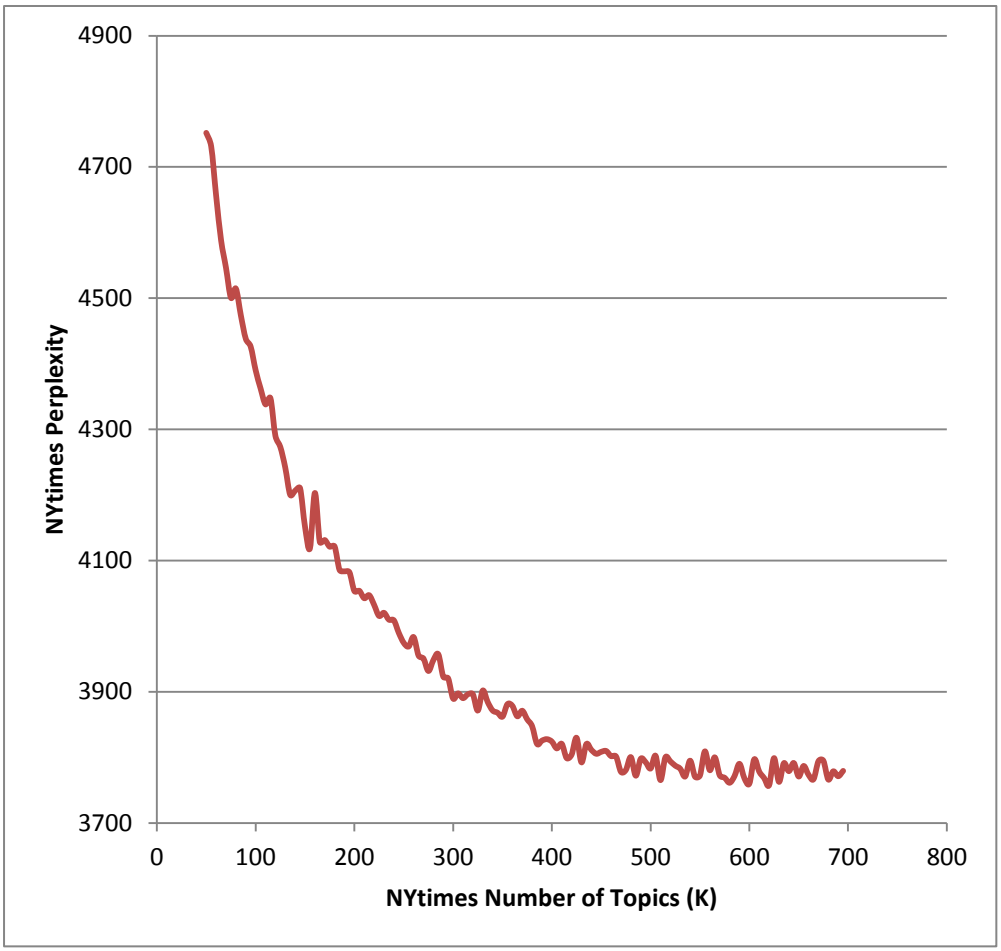


Figure 5.6: NYtimes Perplexity against Topics for 50< Topics<695

The results presented in Figure 5.6 indicates that a graph of perplexity against number of topics (Topics) have a strong quadratic behaviour around a turning point which opens upwards. This result is similar to the result obtained in Figure 5.5.

5.6.7 Autos and NYtimes on Same Axis

The perplexity results in sections 5.6.5: Autos Data Perplexity against Topics for 50< K<690 and 5.6.6: NYtimes Data Perplexity against Topics for 50< K<695 were normalised so that they can be put on the same scale for comparative purposes.

The following formula was used to normalize perplexity for both datasets:

$$NewPerplexity = \frac{Perplexity\ Value - Average\ of\ Perplexity\ Values}{Standard\ Deviation\ of\ Perplexity\ Values}$$

After normalisation, the data was plotted on the same axis as shown in Figure 5.10.

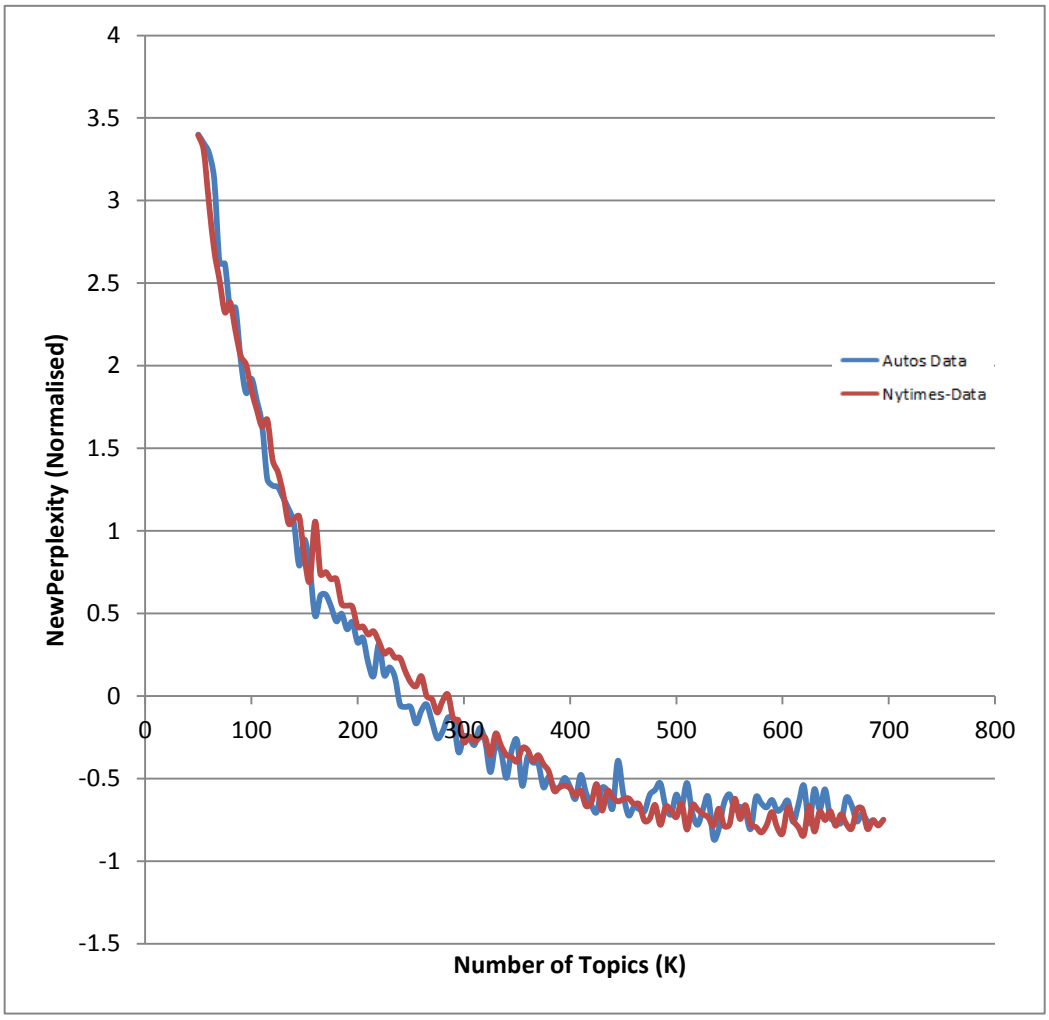


Figure 5.7: Autos and NYtimes on Same Axis

The graph in figure 5.7 strengthens the claim that the trending behaviour of perplexity against Number of topic is quadratic.

5.7 Regression Analysis of Perplexity vs Topics

Section 5.6.5 and 5.6.7, claimed that the graph of perplexity against number of topics (K) is quadratic. In order to ascertain this relationship, a comparison of linear and quadratic regression was carried out and results presented on table 5.1. A statistical test of significance was further carried out involved setting an appropriate hypothesis, finding the best fit equation for the data in section 5.6.5 and 5.6.6, and fitting a quadratic model in RStudio. The R code that was used is presented in Appendix VI of this thesis.

5.7.1. Test of significance:

To test the significance of highest order term, the null and alternative hypothesis was created based on the general quadratic equation.

The general quadratic equation is:

$$y = ax^2 + bx + c$$

Where x and y are independent and dependent variables representing number of topics and corresponding perplexity values respectively.

The null hypothesis (H_0) is:

$$H_0: a = 0$$

The alternative hypothesis (H_1) is:

$$H_1: a \neq 0$$

The test is to reject the null hypothesis if the data fits the model and the alternative hypothesis for the opposite.

5.7.2. Fitting Autos and NYtimes Data in a Quadratic

Figure 5.8 and 5.9 presents results obtained on fitting Autos and NYtimes perplexity against number of topics (K) trending results in RStudio. Table 5.1 presents statistics used to provide decision criteria for hypothesis testing.

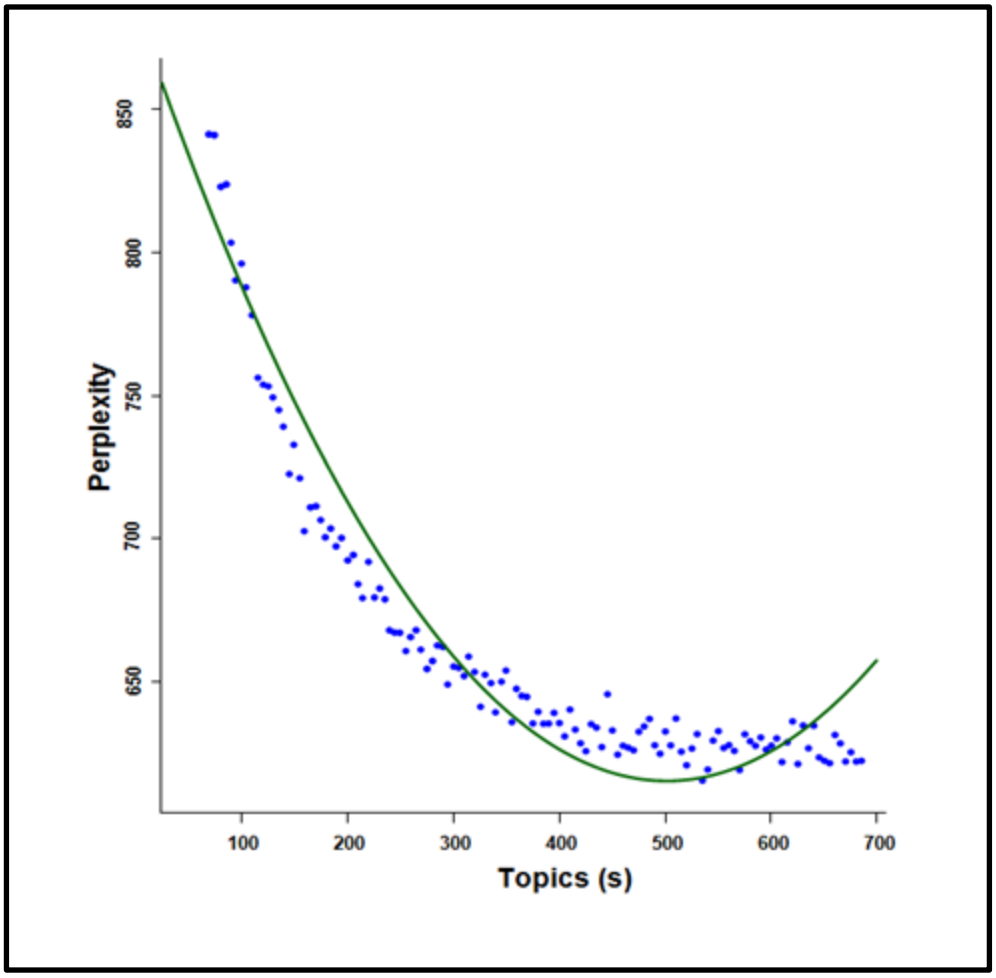


Figure 5.8: Fitting Autos Data in a Quadratic and Results

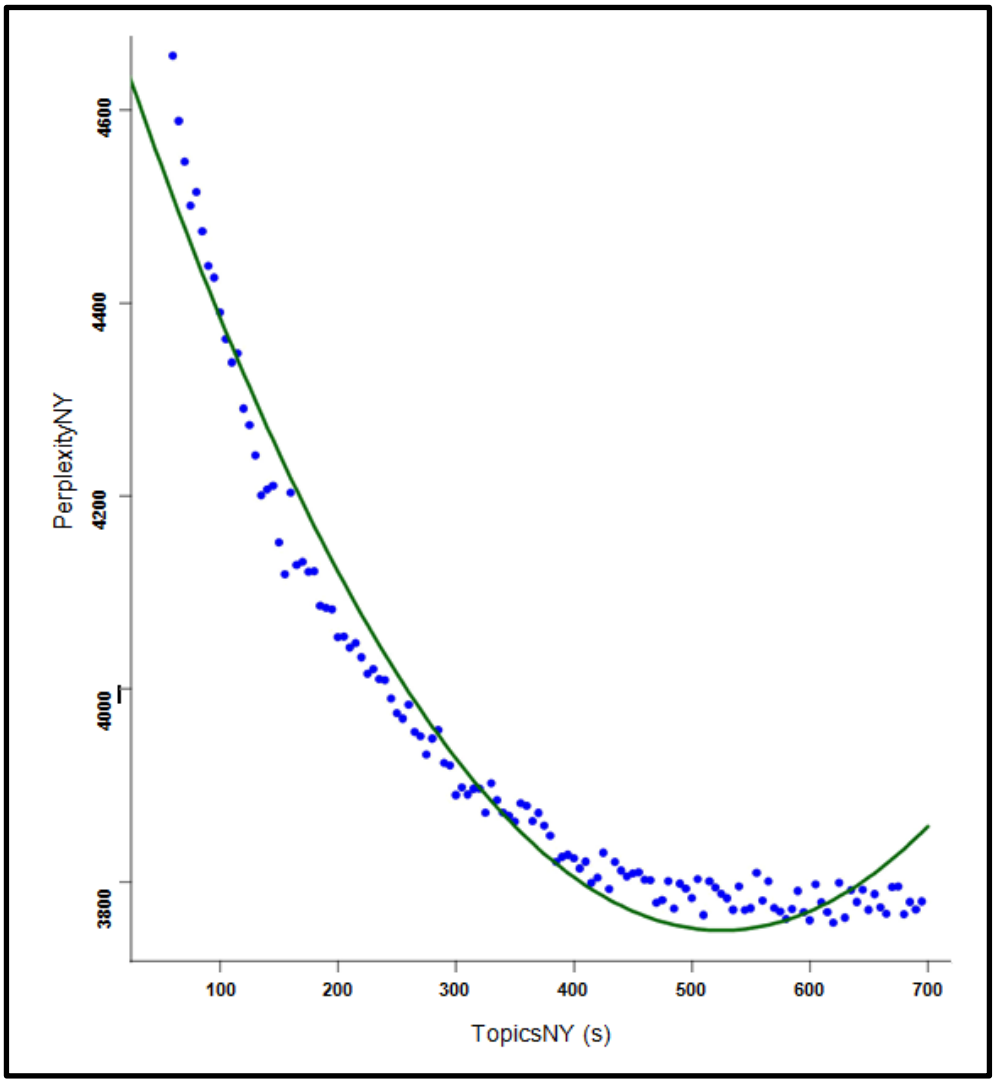


Figure 5.9: Fitting NYtimes Data in a Quadratic and Results

Table 5.1 presents regression analysis statistics of Autos and NYtimes datasets. Details of regression datasets and full results are presented on Appendix VII and Appendix VIII of this thesis.

Table 5.1: Regression Analysis Statistics

STATISTIC	REGRESSION RESULT *AUTOS* DATASET		REGRESSION RESULT *NYTIMES* DATASET	
	LINEAR	**QUADRATIC**	**LINEAR**	**QUADRATIC**
Multiple R	0.81900964	0.962333233	0.85712757	0.978319683
R Square	0.67077679	0.926085252	0.73466767	0.957109403
Adjusted R Square	0.66816391	0.924902616	0.73259476	0.95643396
Standard Error	37.5514460	17.86394042	121.800492	49.16297203
Observations	128	128	130	130

Results presented on table 5.1 indicates better values of coefficient of multiple correlation, coefficient of determination and standard error for quadratic regression compared with linear regression. This means that quadratic regression is more preferred to model the data as opposed to linear regression.

Table 5.2: Coefficient of Determination and p-value Statistics

Statistic	Autos	NYTimes
Multiple R-squared	0.9261	0.9571
Adjusted R-squared	0.9249	0.9564
p-value	< 2.2e-16	< 2.2e-16

Table 5.2 presents a summary of the Coefficient of Determination and p-value Statistics for purposes of discussion. Results on the table 5.2 indicates that computed p-value of less than 2.2e-16 in Table 5.1, is significantly less than 0.005 and a Multiple R-squared value of more than 0.9 for both datasets indicates that the model explains all the variability of the response data which means that the model fits the data.

On the basis of these results, the null hypothesis is therefore rejected and the conclusion that the graph of Perplexity against Number of Topics follows a quadratic

model is supported. This statistical result was used to design the Quadratic Topics Approximation Method (QTAM).

5.8 Design of Quadratic Topics Approximation Method

One of the sub-components of the conceptual framework is model parameters whose value should be estimated from data in order to fit the topic model. The Bayesian approach which is based on Bayes theorem has parameters of the posterior and parameters of the prior. In LDA, parameters of the prior are called hyper parameters.

Literature and explorative experimentation on two datasets whose results are presented under section 5.6 and confirmed under section 5.7 indicates that a graph of perplexity against number of topics (K) has a strong quadratic behaviour around a turning point which opens upwards. This means the graph has a minimum perplexity point that optimizes K. Section 5.8 models the behaviour of K and to guide the calculation of the optimal value with fewer iterations thereby making a contribution to the body of knowledge. This enhanced the speed of estimating the parameter number of topics (K) hence speeding the model for application in large text data analytics.

In order to model this point of minimum perplexity, the following quadratic equation was set up:

$$p=aK^2 + bK + C \qquad (5.1)$$

where p is the perplexity, K is the number of topics, a, b and C are constants for the general quadratic function.

To find the optimal value of *K*, the quadratic equation 5.1 was differentiated, resulting derivative equated to zero and on solving the resulting equation the optimal value of *K* was obtained as follows:

$$\frac{dp}{dK}=2ka + b \qquad (5.2)$$

⇨ $\frac{dp}{dK}=0$ at a minimum point from differential calculus

⇨ $2Ka + b = 0$ at minimum.

⇨ $K = \frac{-b}{2a}$ *(5.3)*

Modelling the evaluation of unknowns *a* and *b* for any dataset of interest was carried out and those values used to calculate the optimal value of *K* in a general perspective as shown on equation 5.3. To accomplish this task, three general quadratic equations, *equation 5.4, equation 5.5* and *equation 5.6* were setup and solved solved by method of substitution. Three equations were required for the solution since the setup had three unknowns: *a, b* and *C*. Three data points (k_1, p_1), (k_2, p_2) and (k_3, p_3) therefore are required to estimate optimum *K* for any dataset as opposed to the previous approach where many iterations on data has to be performed.

$$p_1 = ak_1^2 + bk_1 + C \quad (5.4)$$

$$p_2 = ak_2^2 + bk_2 + C \quad (5.5)$$

$$p_3 = ak_3^2 + bk_3 + C \quad (5.6)$$

To solve equations 5.4, 5.5 and 5.6, the unknown *a* is made the subject of equation 5.4 in 5.7, substituted in 5.5 as shown in 5.8 and the two values of the unknown *a* equated as shown in 5.9. This effectively eliminates *a* from the equation and enables expression of *b* in terms of *C* only as shown in 5.11a. To reduce the complexity of the resulting equations, new arbitrary variables *d, e, f, g* and *h* are introduced and defined as indicated in 5.10, 5.12, 5.13, 5.17, 5.18.

From equation 5.4, $a = \frac{p_1 - bk_1 - C}{k_1^2}$ (5.7)

From equation 5.5, $a = \frac{p_2 - bk_2 - C}{k_2^2}$ (5.8)

➔ $\frac{p_1 - bk_1 - C}{k_1^2} = \frac{p_2 - bk_2 - C}{k_2^2}$ (5.9)

➔ $k_2^2(p_1 - bk_1 - C) = k_1^2(p_2 - bk_2 - C)$

➔ $k_2^2 p_1 - k_2^2 b k_1 - k_2^2 C = k_1^2 p_2 - k_1^2 b k_2 - k_1^2 C$

➔ $k_2^2 p_1 - k_1^2 p_2 + k_1^2 b k_2 - k_2^2 b k_1 = k_2^2 C - k_1^2 C$

➔ $(k_2^2 p_1 - k_1^2 p_2) + b k_1 k_2 (k_1 - k_2) = C(k_2^2 - k_1^2)$

➔ $b k_1 k_2 (k_1 - k_2) = C(k_2^2 - k_1^2) - (k_2^2 p_1 - k_1^2 p_2)$

➔ $b k_1 k_2 (k_1 - k_2) = C(k_2^2 - k_1^2) + (k_1^2 p_2 - k_2^2 p_1)$

Let d = $(k_1^2 p_2 - k_2^2 p_1)$ (5.10)

➔ $b k_1 k_2 (k_1 - k_2) = C(k_2^2 - k_1^2) + k_1^2 p_2 - k_2^2 p_1$

➔ $b = \frac{(k_1^2 p_2 - k_2^2 p_1) + C(k_2^2 - k_1^2)}{k_1 k_2 (k_1 - k_2)} = \frac{k_1^2 p_2 - k_2^2 p_1}{k_1 k_2 (k_1 - k_2)} + \frac{C(k_2^2 - k_1^2)}{k_1 k_2 (k_1 - k_2)}$ (5.11a)

➔ b = e + Cf (5.11b)

Where

$e = \frac{d}{k_1 k_2 (k_1 - k_2)}$; (5.12)

$f = \frac{k_2^2 - k_1^2}{k_1 k_2 (k_1 - k_2)}$; (5.13)

$\therefore b = \frac{k_1^2 p_2 - k_2^2 p_1}{k_1 k_2 (k_1 - k_2)} + \frac{C(k_2^2 - k_1^2)}{k_1 k_2 (k_1 - k_2)}$ (5.14)

From equation 5.7,

➔ a = $\frac{p_1 - (\mathrm{e} + \mathrm{Cf}) k_1 - C}{k_1^2}$ (5.15)

➔ a = $\frac{p_1 - (\mathrm{e} + \mathrm{cf}) k_1 - C}{k_1^2} = \frac{p_1}{k_1^2} - \frac{e k_1}{k_1^2} - \frac{Cf}{k_1^2} - c k_1^2$

➔ a = $\frac{1}{k_1^2}(p_1 - e k_1) - C\left(\frac{f}{k_1^2} + k_1^2\right) = g - C(\frac{f + k_1^4}{k_1^2})$

➔ a = g – Ch (5.16)

where $g = \frac{1}{k_1^2}(p_1 - ek_1)$ (5.17)

and

$h = \frac{f+k_1^4}{k_1^2}$ (5.18)

Equation 5.4 can therefore be restated as follows:

$$p_3 = (\text{g} - \text{Ch})k_3^2 + (\text{e} + \text{Cf})k_3 + \text{C} = gk_3^2 - Chk_3^2 + ek_3 + Cfk_3 + C \quad (5.19)$$

➔ $Chk_3^2 - Cfk_3 - C = gk_3^2 + ek_3 - p_3$

➔ $C(hk_3^2 - fk_3 - 1) = gk_3^2 + ek_3 - p_3$

➔ $C = \frac{gk_3^2+ek_3-p_3}{hk_3^2-fk_3-1}$

Arbitrary variables *d, e, f, g* and *h* were removed in order to find the generic values of *a, b* and *C* as follows:

$$C = \frac{(\frac{1}{k_1^2}\left(p_1-(\frac{d}{k_1k_2(k_1-k_2)})k_1\right))k_3^2+(\frac{d}{k_1k_2(k_1-k_2)})k_3-p_3}{(\frac{k_2^2-k_1}{k_1k_2(k_1-k_2)})+k_1^4)k_3^2-(\frac{k_2^2-k_1^2}{k_1k_2(k_1-k_2)})k_3-1} \quad (5.21)$$

$$a = \frac{1}{k_1^2}\left(p_1 - (\frac{k_1^2p_2-k_2^2p_1}{k_1k_2(k_1-k_2)})k_1\right) - C(\frac{f+k_1^4}{k_1^2}) \quad (5.22)$$

$$b = \frac{k_1^2p_2-k_2^2p_1}{k_1k_2(k_1-k_2)} + \frac{C(k_2^2-k_1^2)}{k_1k_2(k_1-k_2)} \quad (5.23)$$

Which can be stated as follows when using arbitrary variables d, e, f, g and h

$$\text{C} = \frac{gk_3^2+ek_3-k_3}{dk_3^2-fk_3-1} \quad (5.20)$$

a = $g - ch$

b = $e + cf$

where:

$d = k_1^2 p_2 - k_2^2 p_1$;

$e = \frac{d}{k_1 k_2 (k_1 - k_2)}$;

$f = \frac{k_2^2 - k_1}{k_1 k_2 (k_1 - k_2)}$

$g = \frac{p_1 - e k_1}{k_1^2}$

$h = \frac{f + k_1^4}{k_1^2}$

The method developed in section 5.10 requires three data points (k_1, p_1), (k_2, p_2) and (k_3, p_3) to calculate the optimum value of *K*. Existing MCMC based Gibbs Sampling requires that many iterations are done to obtain the value that optimises *K*. The new method therefore enhances *K* inference speed by calculating *K* after only three iterations.

Further the following algorithm for estimating topic model hyperparameters was developed:

Algorithm for the proposed estimator

Initialise α, β and K

FOR iteration i=1, 2, 3

do

READ k_i , p_i

END FOR

$C \leftarrow \frac{g k_3^2 + e k_3 - k}{d k_3^2 - f k_3 - 1}$

$a \leftarrow g - ch$

$b \leftarrow e + cf$

$K \leftarrow \frac{-b}{2a}$

end function

where:

$d = k_1^2 p_2 - k_2^2 p_1$; $e = \frac{d}{k_1 k_2 (k_1 - k_2)}$; $f = \frac{k_2^2 - k_1}{k_1 k_2 (k_1 - k_2)}$; $g = \frac{p_1 - e k_1}{k_1^2}$ *and*

$h = \frac{f + k_1^4}{k_1^2}$

The method developed under section 5.8 is called Quadratic Topic Approximation Method (QTAM) since it approximates number of topics (*K*) input parameter for LDA using quadratic approach. This major contribution for the study significantly reduce the time needed to estimate optimal value of *K*.

5.9 Validating Quadratic Topics Approximation Method

Validation is defined as the assessment of an action, decision, plan, or transaction to establish that it is correct, complete, being implemented as intended, and delivering the intended outcome. A model is usually developed to analyse a particular problem and may therefore represent different parts of the system at different levels of abstraction. As a result, the model may have different levels of validity for different parts of the system across the full spectrum of system behaviour (Sargent, 2013).

Broadly speaking there are three approaches to model validation and any combination of them may be applied as appropriate to the different aspects of a particular model. These approaches are: expert opinion; theoretical analysis and real system measurements (Mellor-Crummey, 2005). In this study, the specific objective on validation was two fold: Validation of the developed topic model deployment framework and validation of the quadratic topics approximation method.

The aim of validating the framework was to achieve dependability and completeness of the said framework, in the context of large text analytics. In topic modeling, dependability and completeness are crucial to the success of product development. A missing requirement can mean a missing attribute in complex products, and a missing requirement can mean a major source of implementation defects in applications. The completeness of requirements is such a need, both in sense of whether the requirements' set contains all requirements and whether all requirements are completely specified (Zenun & Loureiro, 2013).

From a formal point of view, correctness is usually meant to be the combination of consistency and completeness. Consistency refers to situations where a specification contains no internal contradictions, whereas completeness refers to situations where a specification entails everything that is known to be "true" in a certain context. From a practical point of view, however, correctness is often more pragmatically defined as satisfaction of certain framework goals. This indeed is the kind of correctness which is more relevant to a framework consumer. In order to validate the framework, a survey was carried out to seek expert opinion concerning the framework on the two dimensions mentioned above: Completeness and Dependability.

In validating the method, on the other hand, the study sought to establish if the developed parameter approximation mechanism is faster than the existing CGS approach and on correctness whether the new method maintains the same standard of output. The study used two approaches to accomplish this. One is theoretical time analysis and two is analysis of real system implementation output measurements. In both cases, output was presented and compared for validation conclusions.

5.9.1. Theoretical Time Analysis to Validate QTAM

The time-complexity (aka "Big O" or "Big Oh") definition of various operations in CPython defined in the Python Wiki was used in this chapter of the study, available at <*https://wiki.python.org/moin/TimeComplexity*>. Python Wiki is a community place to gather and organize all things about Python. According to the wiki, other Python implementations (or older or still-under development versions of CPython)

may have slightly different performance characteristics. However, it is generally safe to assume that they are not slower by more than a factor of O(log n).

a) Time Complexity of CGS inference for number of topics (K)

The following code is part of graphlab implementation of LDA model which iteratively computes an estimate of K heuristically from data. We compute the time complexity of the code in order to compare with a similar analysis of our developed code for estimating K using the formula developed in the previous chapter.

b) Section of Code for Implementing CGS

The following python code which is presented in a generic fashion was used to implement Collapsed Gibbs Sampling algorithm in graphlab create.

```
alpha = p #doc_topic_prior
beta = q #topic_word_prior
graphlab_model_K = defaultdict(list)
for k in range(a, n, b):
g_m=gl.topic_model.create(g_train_data,num_topics=k,alpha=alpha,beta=beta,num_
iterations=l)
   perpelexity = g_m.evaluate(g_train_data, g_test_data)
   graphlab_model_K[k] = [k,perpelexity['perplexity']]
   print(k,perpelexity)
```

Table 5.3 presents the code used to implement Collapsed Gibbs Sampling as used in Graphlab Create, cost of each code statement, statement worst-case run time, the product of cost and time and the total of product of cost and time. This is used for theoretical comparative analysis of time complexity of CGS against QTAM inference for number of topics (K).

Table 5.3: Code for Implementing CGS

Code Statement	Cost of Statement	Statement worst-case run time	Product of Cost and Time
alpha = *p* #doc_topic_prior	C_1	1	C_1
beta = *q* #topic_word_prior	C_2	1	C_2
graphlab_model_K = defaultdict(list)	C_3	1	C_3
for k in range(*a, n, b*): #*a* is initial value, *n* is maximum and *b* is the step.	C_4	n	nC_4
g_m=gl.topic_model.create(g_train_data,num_topics=k,alpha=alpha,beta=beta, num_iterations=*l*)	C_5	1	C_5
perpelexity = g_m.evaluate(g_train_data, g_test_data)	C_6	1	C_6
graphlab_model_K[k] = [k,perpelexity['perplexity']]	C_7	1	C_7
print(k,perpelexity)	C_8	1	C_8
TOTAL Cost			**A+nB**

In order to simplify the equation for total cost was let to be $C_1 + C_2 + C_3 + C_5 + C_6 + C_7 + C_8 = A$ and $C_4 = B$ since each C_i is a constant representing cost of a statement. Therefore the total cost which is the sum of product cost and time is given by **A + nB** where n is unknown. This means that in big O notation, the algorithm has a *linear* time complexity represented as O(n). That is, when *r* arguments are passed, it will take *r* times as long as when 1 argument is passed, where *r* is an integer.

c) Time Complexity of QTAM inference for number of topics (K)

The following code is part of graphlab implementation of LDA model which iteratively computes an estimate of K heuristically from data. Time complexity of the code was computed in order to compare with a similar analysis of the developed code for estimating K using the formula developed in section 5.8. Appendix II presents the entire code used in implementing QTAM and its run-time on 20Newsgroup Dataset.

d) Section of Code for Implementing QTAM

Table 5.4 presents python code used to implement QTAM and a general fashion of cost of each code statement, statement worst-case run time, the product of cost and time and the total of product of cost as used to implement QTAM in GraphLab create. The total of product of cost was used to compare with time complexity of CGS presented in Table 5.3.

Table 5.4: Code for Implementing QTAM

Code Statement	**Cost of Statement**	**Statement worst-case run time**	**Product of Cost and Time**
alpha = *p* #doc_topic_prior	C_1	1	$\mathbf{C_1}$
beta = *q* #topic_word_prior	C_2	1	$\mathbf{C_2}$
graphlab_model_K = defaultdict(list)	C_3	1	$\mathbf{C_3}$
for k in range(*a, a+2b, b*): *#a* is initial value and *b* is the step.	C_4	3	$\mathbf{3C_4}$
g_m=gl.topic_model.create(g_train_data,num_topics=k,alpha=alpha,beta=beta,num_iterations=*l*)	C_5	1	$\mathbf{C_5}$
perpelexity = g_m.evaluate(g_train_data, g_test_data)	C_6	1	$\mathbf{C_6}$
graphlab_model_K[k] = [k,perpelexity['perplexity']]	C_7	1	$\mathbf{C_7}$
print(k,perpelexity)	C_8	1	$\mathbf{C_8}$
TOTAL Cost			**M**

Similarly, the total cost which is the sum of product cost and time is given by $C_1 + C_2 + C_3 + 3C_4 + C_5 + C_6 + C_7 + C_8 = M$, where C_i and M are arbitrary constants. This means that in big O notation, the algorithm has a constant time complexity represented as O(1). That means that it takes a constant time, like *s* seconds, or *m* minutes no matter the amount of data in the set.

5.9.2. Experimental Validation of QTAM

This section describes an experimental analysis of the proposed QTAM algorithm, with direct comparison to the baseline Collapsed Gibbs Sampling algorithm implemented in GraphLab hereafter referred to as GCGS. As well as performing a running time analysis on selected text datasets.

Python code was created for the deployment of QTAM algorithm and GCGS as described in GraphLab Create Python library. We explored the run-time for both GCGS and the developed QTAM algorithms on the following four text datasets: Autos, Politics, Sports and NYTimes.

a) Properties of Validation Datasets

Table 5.5 presents the properties of datasets used to validate QTAM through comparative run time analysis.

Table 5.5: Validation Dataset Properties

DATASET	NUMBER OF DOCUMENTS	VOCABULARY SIZE
Autos	1192	886
Politics	1575	1926
Sports	1197	1999
Nytimes	10000	55234

To compare the algorithms fairly, all the datasets were evaluated in the same environmental setup and same programming language Python on a machine setup in Google cloud.

b) Validation Results

For validation of the Quadratic Topics Approximation Method (QTAM) Graphlab's topic model based on Collapsed Gibbs Sampling (CGS) was used as the baseline model and labeled GCGS. QTAM and GCGS were then compared and results

presented. CGS was chosen as the baseline model because it is guaranteed to converge to the true posterior distribution (Magnusson et al., 2017).

To collect run-time data, a variable for timing the execution of developed code section for the new algorithm was included in the designed code. This was also done for the old implementation of CGS. Collected data was analysed in excel worksheet and presented in Table 5.6 and Figure 5.10.

Further analysis was carried out to compare the value of parameter K obtained using the CGS method and also using the QTAM method. These results are presented in Table 5.7.

Table 5.6: Run Time Results for CGS

DATASET	TIME TAKEN TO ESTIMATE K (RUN TIME)			
	CR1	CR2	CR3	CAvg
Autos	21.24	21.32	20.51	21.02333
Politics	30.948	28.635	31.876	30.48633
Sports	40.487	42.0675	41.0239	41.1928
Nytimes	473.1696	462.8857	468.6187	468.2247

Table 5.6 presents the run time results for the Collapsed Gibbs Sampling (CGS) where the labels CRi represents CGS-Run i, where i = 1,2,3 meaning three runs were done on each dataset. CAvg is the average run time of the three runs and is used for comparative purposes in the graphs, Figure 5.10 and Figure 5.11.

Table 5.7 Run Time Results for QTAM

DATASET	QR1	QR2	QR3	QAvg
Autos	6.71	6.281	6.57	6.52
Politics	10.457	10.319	10.319	10.37
Sports	16.019	15.683	14.436	15.38
Nytimes	156.735	154.415	154.396	155.2

Table 5.7 presents the run time results for the QTAM method where the labels QRi represents QTAM-Run i, where i = 1,2,3 meaning three runs were done on each

dataset. QAvg is the average run time of the three runs and is used for comparative purposes in the graphs, Figure 5.10 and Figure 5.11.

c) Comparative Average Run-time for All Datasets

Figure 5.10 presents a comparative run-time analysis for the four datasets presented on table 5.6 and table 5.7.

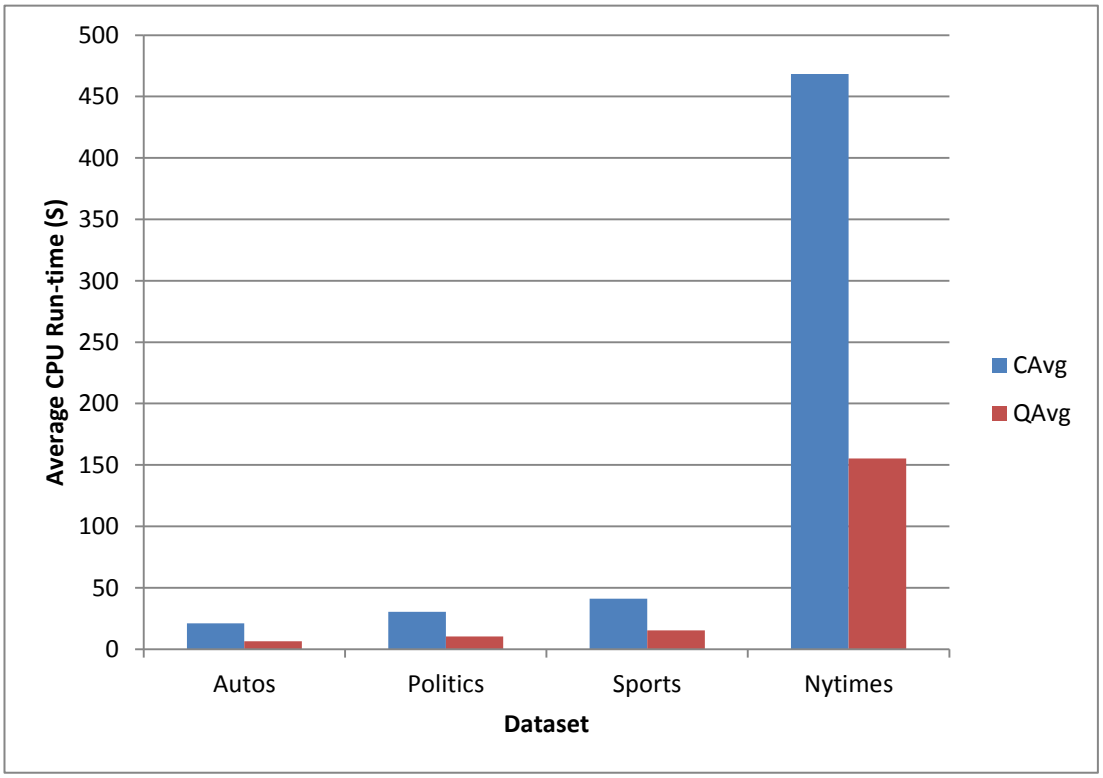


Figure 5.10: Run-time for All Datasets

Nytimes being a larger dataset took more time to execute. The other three datasets are not very clear when plotted on the same axis with this dataset. The researcher therefore plotted the other three datasets on a different graph, as shown in Figure 6.6.

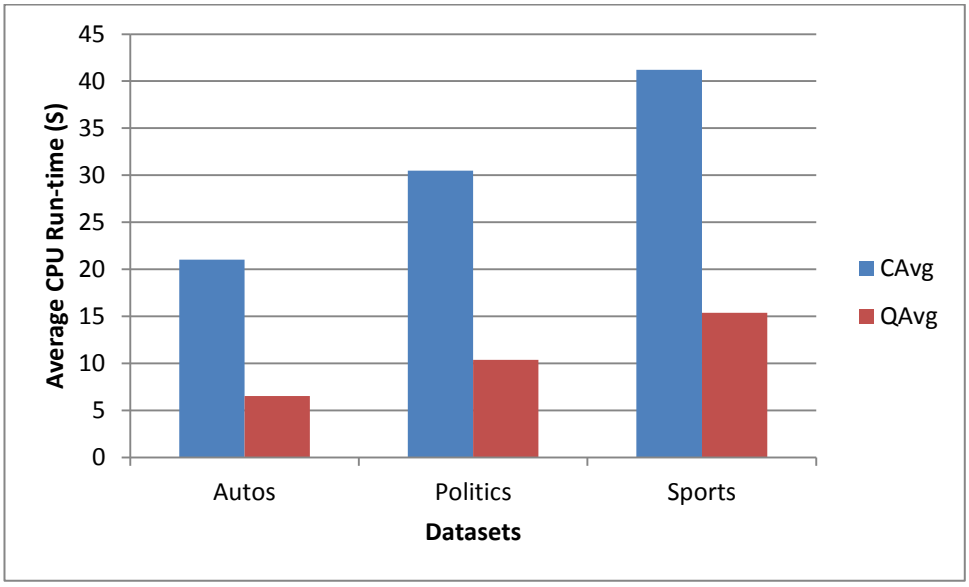


Figure 5.11: Average Run-time for Autos, Politics and Sports Datasets

The graphs in Figure 5.10 and Figure 5.11 indicates that the Average CPU Run-time for QTAM labelled QAvg on the graph is significantly less than the CPU time for GCGS labelled CAvg on the graph, for all the four datasets: Autos, Politics, Sports and NYTimes.

Table 5.8: Value of Parameter K for CGS and QTAM on Four Datasets

DATASET	VALUES OF PARAMETER K					
	KCR1	KCR2	KCR3	KQR1	KQR2	KQR3
Autos	340	344	336	322	346	352
Politics	506	480	478	482	495	468
Sports	320	294	307	303	296	287
Nytimes	802	757	853	776	705	892

Table 5.8 shows the values of parameter K obtained for the four datasets, Autos, Politics, Sports, Nytimes, used in this experiment. KCRi represents the value of K for CGS-Run i, where i = 1,2,3 meaning three runs were done on each dataset. KQRi represents the value of K for QTAM-Run i, where i = 1,2,3 meaning three runs were done on each dataset for QTAM. Obtained values were compared using linear regression presented in section 5.9.2.4

d) Comparison of Parameter K for CGS and QTAM

The python function scipy.stats.linregress(x, y) was used to perform a linear regression of K values obtained using the CGS and QTAM methods for similarity comparative purposes. The function computed a least-squares regression for the two sets of measurements. The results are as shown on the Table 5.9.

Table 5.9: Correlation Coefficient and Coefficient of Determination for K

PARAMETER	DESCRIPTION	VALUE OBTAINED
Slope type float	slope of the regression line	0.99437
intercept type float	intercept of the regression line	-5.02158
r-value type float	correlation coefficient	0.9932286
p-value type float	two-sided p-value for a hypothesis test whose null hypothesis is that the slope is zero.	1.10849
stderr type float	Standard error of the estimate	0.03678
r-squared (square of r-value)	coefficient of determination	0.9865

A correlation coefficient value of 0.9932 indicates a strong positive linear correlation between K values obtained by CGS and those obtained by QTAM. Similarly a coefficient of determination value of 0.9865 means that 98.65% of the variation in KCGS can be explained by KQTAM. The Standard error of the estimate provides an accuracy of 0.03678 for a two tailed test which means that the results are to a confidence level of 98.161%.

5.10 Conclusion

Chapter five described experimental dataset, test environment and software used in details. The chapter further described experimental setup, and presented the procedure used for experimentation, how regression analysis was carried out, results and their interpretation. Finally a fast method of inferring the optimal number of topics (K) using quadratic approach and differential calculus is developed.

This chapter also validated a method for increasing the speed of approximating the optimal number of topics parameter, K used in the implementation of LDA Gibbs sampling. The said validation used two approaches: Theoretical Time Analysis and Experimental CPU run-time analysis.

Results of Correlation Coefficient and Coefficient of Determination for K means that deployment of CGS and QTAM in determination of K, results in closely comparable estimates. Other results indicated a major speed improvement for QTAM compared to CGS. It means then that QTAM would be a preferred choice in estimating the number of topics K for developing topic modelling applications.

CHAPTER SIX

CONCLUSION AND RECOMMENDATIONS

6.1 Summary of Findings

This chapter concludes the thesis with a discussion. The research questions are revisited, arguing that topic modelling deployment framework design and parameter optimisation in the area of large text data analytics offers a handy solution for both the implementer and researcher in terms of reducing execution time yet maintaining the quality of the model.

In the information era, with the rapid development of computer technologies, web pages, emails, databases, digital libraries and other electronic texts have been increasing fast. Most of the data is in an unstructured or semi-structured form. It is very difficult to discover the knowledge from it, but it is quite significant. How to find the useful and required information quickly and accurately from lots of information systems has become a challenge in the fields of information science and technology at present. Probabilistic topic models such as Latent Dirichlet Allocation provide an effective tool for text analysis. These models treat the documents as the probability distribution on topics so that the probability model is established for the text generation process and the probability model parameters are trained. Briefly, the model can extract the latent topic information from text effectively.

In LDA, the core procedure is to estimate the posterior probability distribution of topics which depends on the prior according to Bayes theorem. The main approximation methods include the variational inference based on the EM algorithm and Markov Chain Monte Carlo method based on Gibbs sampling. The advantage of the former is that the variational parameters and iteration are used to approximate the Dirichlet parameters. Because its convergence speed is fast, the number of iterations is less. But, the calculation in each iteration is complex, especially the computing overhead is enormous when dealing with massive text. The advantage of the latter is to construct a Markov Chain of the posterior probability and of hyper-parameters. The global statistics is updated by sampling, and there is a predefined fixed number

of iterations. The sampling algorithm in the iteration is simple, and the computing overhead is small. But, according to the nature of Markov chains, lots of iterations are required to achieve a stable state. These two algorithms have their own advantages and disadvantages respectively.

To implement topic model in applications like graphlab, the parameters alpha, beta and number of topics (K) need to be specified. This thesis has established that an optimal value for alpha can be obtained by the equation alpha = 50/K and beta = 0.01. Further to this, it was noted that in applying CGS to infer the number of topics (K), the function is iterated for some pre-defined number of times and also the number of topics varied within a given range of values. The value of K that minimises perplexity is taken to be the optimal value and can be used to estimate hyper-parameter alpha for use in estimating the prior probability. However, this approach introduces iterations at two levels: one at the level of learning the topic model and the other one at the level of varying the number of topics. To illustrate this, if K is varied in an interval that results in *n* values of K where *n* is a positive integer and the model iterated *m* times for each K, the resulting number of iterations required to estimate K will be *n x m*. This thesis specifically reduces the *m* iterations to only three thereby significantly reducing the overheads yet maintaining the quality of the model. This enhances the resultant model for application in big data settings.

6.2 Revisiting the Research Questions

This study established that there are three major components of a topic model deployment task namely: pre-processing; topic model architecture; and topic model products. The topic modelling framework presented a conceptualised process that starts with pre-processing of raw data which is fed into a statistical process called topic model architecture to give an output of topic model products. This process happens in an environmental context. Both the statistical process and environmental context influences the quality of output. The statistical process concerns determination of probability distribution functions to describe the data. These distribution functions are guided by some parameters which determine the quality of the output.

The pre-processing informs the topic model architecture by cleaning data and conforming it to an acceptable format. On the other hand, the topic model architecture which is the engine outputs the topic model products. If the topic model products are not satisfactory, the system can query the topic model architecture or the pre-processing stage.

This study established that the three most common inference algorithms are: variational inference, Gibbs Sampling and online variational Bayes algorithm. Many researchers have improved different aspects of these techniques to enhance topic modelling as noted in literature.

Gibbs Sampling mechanisms usually rely on Markov Chain Monte-Carlo (MCMC) based algorithms. Though these samplers are guaranteed to converge to the true value, they must be run for many iterations. This major drawback is addressed by this study in as far as estimating the number of topics (K) model parameter.

The experiments in the study showed high effectiveness of the developed method for selecting number of topics (K) and provided empirical evidence to support an optimal value of Beta at 0.01 and Alpha at 50/K for topic modelling tasks. These improvements should hold in many current and future applications of topic model applications.

This thesis used three main approaches to validate the developed framework and parameter selection method. These approaches are: expert opinion survey; theoretical time analysis and parameter selection method implementation comparative run-time analysis.

6.3 Recommendations for Future Work

This study mainly focused on the LDA probability topic model and the Gibbs sampling algorithm, which is one of existing inference methods. Further studies can be carried out to test the applicability of the developed method in other topic models which was beyond the scope of this study.

This study have shown the importance of pre-processing and selection of topic model parameters. Further study can be carried out to determine the best combination of the preprocessing steps and topic model parameters not covered in this thesis.

The study also recommends replica experiments to further enhance the results of the new method as an industry standard.

REFERENCES

Abdullah, M., & Zamil, M. G. (2018). The Effectiveness of Classification on Information Retrieval System. Retrieved from https://arxiv.org/ftp/arxiv/papers/1804/1804.00566.pdf

Albishre, K., Albathan, M., & Li, Y. (2015). Effective 20 newsgroups dataset cleaning. *Web Intelligence and Intelligent Agent Technology, 3*(1), 98-101.

Alghamdi, R. & Alfalqi, K. (2015). A Survey of Topic Modelling in Text Mining, *International Journal of Advanced Computer Science and Applications (IJACSA)*, *6*(1), 147-153.

Alpaydin, E. (2014). *Introduction to machine learning*. Cambridge, Massachusetts: MIT press.

AlSumait, L., Barbara, D., & Domeniconi, C. (2008). On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on* (pp. 3-12). IEEE.

Alwidian, S. A. A., Bani-Salameh, H. A., & Alslaity, A. A. N. (2015). Text data mining: a proposed framework and future perspectives. *International Journal of Business Information Systems*, *18*(2), 127-140.

Angelino, E., Kohler, E., Waterland, A., Seltzer, M., & Adams, R. P. (2014). Accelerating MCMC via parallel predictive prefetching. Retrieved from http://auai.org/uai2014/proceedings/individuals/286.pdf

Asuncion, A., Welling, M., Smyth, P., & Teh, Y. W. (2009). On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 27–34).

Bao, Y., & Datta, A. (2014). Simultaneously discovering and quantifying risk types from textual risk disclosures. *Management Science*, *60*(6), 1371-1391.

Bell, G., Hey, T., & Szalay, A. (2009). Beyond the data deluge. *Science*, *323*(5919), 1297-1298.

Benkhelifa, R., & Laallam, F. Z. (2016). Facebook Posts Text Classification to Improve Information Filtering. *Web Information Systems and Technologies 1*(1), 202-207.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York City, USA: Springer.

Bishop, C. M. (2013). Model-based machine learning. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *371*(1984), 1-17.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, *3*, 993–1022.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

Blei, D. M. (2014). Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, *1*, 203-232.

Blei, D. (2015). Probabilistic Topic Models and User Behavior. Retrieved from Columbia University. Website: http://www.cs.columbia.edu/~blei/talks/Blei_User_Behavior.pdf

Bo, L., Ren, X., & Fox, D. (2013). Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics* (pp. 387-402). Springer International Publishing.

Bolstad, W. M. (2010). *Understanding Computational Bayesian Statistics*. New Jersey, USA: John Wiley & Sons.

Boyd-Graber, J. (2010). *Linguistic extensions of topic models*. (Doctoral dissertation). Retrieved from https://www.cs.colorado.edu/~jbg/docs/2010_jbg_thesis.pdf

Cao, L., & Fei-Fei, L. (2007, October). Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (pp. 1-8). IEEE.

Chemudugunta, C., Smyth, P., & Steyvers, M. (2007). Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in neural information processing systems* (pp. 241-248).

Chen, W., Chen, Y., Mao, Y., & Guo, B. (2013). Density-based logistic regression. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 140–148).

Christen, J. A., & Fox, C. (2005). Markov chain Monte Carlo using an approximation. *Journal of Computational and Graphical statistics*, *14*(4), 795-810.

Claesen, M., De Smet, F., Suykens, J. A. K., & De Moor, B. (2014). EnsembleSVM: A library for ensemble learning using support vector machines. *The Journal of Machine Learning Research*, *15*(1), 141–145.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, *12*, 2493–2537.

Cooper, M. L., Shaffer, C. A., Edwards, S. H., & Ponce, S. P. (2014). Open source software and the algorithm visualization community. *Science of Computer Programming*, *88*, 82-91.

Dahlin J (2016). Accelerating Monte Carlo Methods for Bayesian Inference in Dynamical Models. (Doctoral dissertation). Retrieved from http://user.it.uu.se/~thosc112/dahlinphd2016.pdf.

Dalgaard, P. (2008). *Introductory statistics with R*. New York, USA: Springer

Darling, W. M. (2011). A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 642-647).

Dasgupta, A., Drineas, P., Harb, B., Josifovski, V., & Mahoney, M. W. (2007). Feature selection methods for text classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 230-239). ACM.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391-407.

Feng, J., Xu, H., Mannor, S., & Yan, S. (2014). Robust Logistic Regression and Classification. In *Advances in Neural Information Processing Systems* (pp. 253–261).

Feng, X., Liang, Y., Shi, X., Xu, D., Wang, X., & Guan, R. (2017). Overfitting Reduction of Text Classification Based on AdaBELM. *Entropy*, *19*(7), 330.

Firth, J. R. (1957). A synopsis of linguistic theory 1930-55. *Studies in linguistic analysis*, 1952-59, 1-32.

Fu, Y., Xiang, R., Liu, Y., Zhang, M., & Ma, S. (2007). Finding experts using social network analysis. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 77-80). IEEE Computer Society.

Gaber, M. M. (2012). Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 79-85.

Gaikwad, S. V., Chaugule, A., & Patil, P. (2014). Text mining methods and techniques. *International Journal of Computer Applications*, *85*(17).

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2014). *Bayesian data analysis*. Florida, United States: Taylor & Francis.

Gershman, S. J., & Blei, D. M. (2012). A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, *56*(1), 1–12.

Ghahramani, Z. (2013). Bayesian non-parametrics and the probabilistic approach to modelling. *Phil. Trans. R. Soc. A*, *371*(1984), 20110553.

Glowacka, D. (2012). *Learning from interaction: models and applications*. (Doctoral dissertation). Retrieved from http://discovery.ucl.ac.uk/1369566/1/phd.pdf.

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp. 1764–1772).

Greene, D., O'Callaghan, D., & Cunningham, P. (2014). How many topics? stability analysis for topic models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 498-513). Springer, Berlin, Heidelberg.

Griffiths, T. L., Steyvers, M., Blei, D. M., & Tenenbaum, J. B. (2004). Integrating topics and syntax. In *Advances in neural information processing systems* (pp. 537–544).

Grimes, S. (2005). The Word on Text Mining. Presentation. Portals, Collaboration, and Content Management. Retrieved from: http://altaplana.net/TheWord.pdf

Grimes, S. (2011). Text/content analytics 2011: user perspectives on solutions and providers. Washington DC, USA: Alta Plana.

Grimmer, J., & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, *21*(3), 267-297.

Gupta, P., & Narang, B. (2012). Role of text mining in business intelligence. *Gian Jyoti E-Journal*, *1*(2), 1-13.

Gupta, V., & Lehal, G. S. (2009). A survey of text mining techniques and applications. *Journal of emerging technologies in web intelligence*, *1*(1), 60-76.

Gyori, B. M., & Paulin, D. (2016). Hypothesis testing for Markov chain Monte Carlo. *Statistics and Computing*, *26*(6), 1281-1292.

Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, *17*, 26-32.

Hall, D., Jurafsky, D., & Manning, C. D. (2008). Studying the history of ideas using topic models. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 363-371). Association for Computational Linguistics.

Hamming, R. W. (1962). Numerical analysis for scientists and engineers. *Dover Publications*. New York, USA: Dover Publications.

Harish, B. S., & Revanasiddappa, M. B. (2017). A Comprehensive Survey on various Feature Selection Methods to Categorize Text Documents. *International Journal of Computer Applications*, *164*(8).

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, *14*(1), 1303-1347.

Hoffman, M., Bach, F. R., & Blei, D. M. (2010). Online learning for latent dirichlet allocation. In *advances in neural information processing systems* (pp. 856-864).

Hoffman, M., Bach, F. R., & Blei, D. M. (2010). Online learning for latent Dirichlet allocation. In *advances in neural information processing systems* (pp. 856-864).

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 50–57).

Hu, D. J. (2009). Latent Dirichlet allocation for text, images, and music. Retrieved from http://cseweb.ucsd.edu/~dhu/docs/ research_exam09.pdf.

Huang, C., Wang, Q., Yang, D., & Xu, F. (2018). Topic mining of tourist attractions based on a seasonal context aware LDA model. *Intelligent Data Analysis*, *22*(2), 383-405.

Ihler, A., & Newman, D. (2012). Understanding errors in approximate distributed latent Dirichlet. allocation. *Knowledge and Data Engineering, IEEE Transactions on*, *24*(5), 952-960.

Ishikiriyama, C. S., Miro, D., & Gomes, C. F. S. (2015). Text Mining Business Intelligence: A small sample of what words can say. *Procedia Computer Science*, *55*, 261-267.

Jelodar, H., Wang, Y., Yuan, C., & Feng, X. (2017). Latent Dirichlet Allocation (LDA) and Topic Modeling: Models, Applications, a Survey. *Multimedia Tools and Applications, 78*(11), 15169–15211.

Johnson, D. E., Oles, F. J., Zhang, T., & Goetz, T. (2002). A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal*, *41*(3), 428-437.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kang, M., Ahn, J., & Lee, K. (2018). Opinion mining using ensemble text hidden Markov models for text classification. *Expert Systems with Applications*, *94*, 218-227.

Kannan, S., & Gurusamy, V. (2014). Preprocessing Techniques for Text Mining.

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). Natural Language Processing: State of The Art, Current Trends and Challenges. Retrieved from https://arxiv.org/ftp/arxiv/papers/1708/1708.05148.pdf.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., ... & Ivanov, P. (2016). Jupyter Notebooks-a publishing format for reproducible computational workflows. In *ELPUB* (pp. 87-90).

Kobayashi, V. B., Mol, S. T., Berkers, H. A., Kismihok, G., & Den Hartog, D. N. (2017). Text classification for organizational researchers: A tutorial. *Organizational research methods*, *21*(3), 766-799.

Kremer, J., Steenstrup Pedersen, K., & Igel, C. (2014). Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(4), 313–326.

La Rosa, M., Fiannaca, A., Rizzo, R., & Urso, A. (2015). Probabilistic topic modeling for the analysis and classification of genomic sequences. *BMC bioinformatics*, *16*(6), 2-9.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning* (Vol. 10, pp. 331-339).

Lasserre, J. A. (2008). *Hybrids of Generative and Discriminative Methods for Machine Learning*. (Doctoral dissertation). Retrieved from https://pdfs.semanticscholar.org/fe6c/5f37ed0fa2dd6aeecd101c757880efeb2c13.pdf?_ga=2.172403875.168267036.1563511213-276576430.1549871027.

Lauscher, A., Nanni, F., Ruiz Fabo, P., & Ponzetto, S. P. (2016). Entities as topic labels: combining entity linking and labeled lda to improve topic interpretability and evaluability. *IJCol-Italian journal of computational linguistics*, *2*(2), 67-88.

Letouze, E. (2012). Big data for development: challenges & opportunities. Retrieved from http://www.unglobalpulse.org/sites/default/files/BigDataforDevelopment-UNGlobalPulseJune2012.pdf.

Li, S., Xia, R., Zong, C., & Huang, C. R. (2009). A framework of feature selection methods for text categorization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2* (pp. 692-700). Association for Computational Linguistics.

Liu, Z., Zhang, Y., Chang, E. Y., & Sun, M. (2011). Plda+: Parallel Latent Dirichlet Allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 26.

Low, Y., Gonzalez, J. E., Kyrola, A., Bickson, D., Guestrin, C. E., & Hellerstein, J. (2014). Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*.

Low, Y., Gonzalez, J. E., Kyrola, A., Bickson, D., Guestrin, C. E., & Hellerstein, J. (2014). Graphlab: A new framework for parallel machine learning. *arXiv Preprint arXiv:1408.2041*.

Luo, X., Yu, J. X., & Li, Z. (Eds.). (2014). *Advanced Data Mining and Applications: 10th International Conference, ADMA 2014, Guilin, China, December 19-21, 2014, Proceedings* (Vol. 8933). Springer.

Maclaurin, D., & Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with Subsets of Data. In *UAI* (pp. 543-552).

Magnusson, M., Jonsson, L., Villani, M., & Broman, D. (2017). Sparse Partially Collapsed MCMC for Parallel Inference in Topic Models. *Journal of Computational and Graphical Statistics*, *27*(24), 449-463.

Mahendran, A., Duraiswamy, A., Reddy, A., & Gonsalves, C. (2013). Opinion Mining for text classification. *International Journal of Scientific Engineering and Technology*, *2*(6), 589-594.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). *Big data: The next frontier for innovation, competition, and productivity*. San Francisco, Northern California: McKinsey Global Institute.

Mcauliffe, J. D., & Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems* (pp. 121-128).

McLellan, C. (2015). The internet of things and big data: Unlocking the power. Retrieved from https://www.gsma.com/iot/wp-content/uploads/2015/12/cl_iot_bigdata_11_15-004.pdf.

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

Mekala, D., Gupta, V., Paranjape, B., & Karnick, H. (2017). SCDV: Sparse Composite Document Vectors using soft clustering over distributional representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 670-680).

Mellor-Crummey, J. (2005). Analyzing Simulation Results. Retrieved from https://www.cs.rice.edu/~johnmc/comp528/lecture-notes/Lecture20.pdf

Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., & Ranzato, M. (2014). Learning longer memory in recurrent neural networks. Retrieved from https://arxiv.org/pdf/1412.7753.pdf.

Mimno, D. (2012). Computational historiography: Data mining in a century of classics journals. *Journal on Computing and Cultural Heritage (JOCCH)*, *5*(1), 3.

Mimno, D., Wallach, H., Talley, E., Leenders, M., and McCallum, A. (2011). Optimizing semantic coherence in topic models. In EMNLP.

Modupe, A., Olugbara, O. O., & Ojo, S. O. (2013). Investigating topic models for mobile short messaging service communication filtering. In *Proceedings of the World Congress on Engineering* (Vol. 2).

Moreno, A., & Redondo, T. (2016). Text analytics: the convergence of big data and artificial intelligence. *IJIMAI*,*3*(6), 57-64.

Nallapati, R., Cohen, W., & Lafferty, J. (2007, October). Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability. In *icdmw* (pp. 349-354). IEEE.

Newman, D., Asuncion, A., Smyth, P., & Welling, M. (2009). Distributed algorithms for topic models. *The Journal of Machine Learning Research*, *10*, 1801–1828.

Nkomo, R., Kabundi, A., & others. (2013). *Kalman Filtering and Online Learning Algorithms for Portfolio Selection*.

Paul, M., & Girju, R. (2009, August). Cross-cultural analysis of blogs and forums with mixed-collection topic models. *Association for Computational Linguistics, 3*(3), 1408-1417.

Phan, X. H., & Nguyen, C. T. (2007). GibbsLDA++: AC/C++ implementation of latent Dirichlet allocation.

Phan, X. H., Nguyen, L. M., & Horiguchi, S. (2008, April). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web* (pp. 91-100). ACM.

Piepenbrink, A., & Gaur, A. S. (2017). Topic models as a novel approach to identify themes in content analysis. In *Academy of Management Proceedings* (Vol. 2017, No. 1, p. 11335). Briarcliff Manor, NY 10510: Academy of Management.

Pinoli, P., Chicco, D., & Masseroli, M. (2014). Latent Dirichlet allocation based on Gibbs sampling for gene function prediction. In *Computational Intelligence in*

Bioinformatics and Computational Biology, 2014 IEEE Conference on (pp. 1-8). IEEE.

Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., & Welling, M. (2008). Fast collapsed gibbs sampling for latent Dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 569–577).

Porter, M. (2008). The Porter stemming algorithm, 2005. *See http://www. tartarus. org/~ martin/PorterStemmer*.

Puschmann, D., Barnaghi, P., & Tafazolli, R. (2018). Using LDA to Uncover the Underlying Structures and Relations in Smart City Data Streams. *IEEE Systems Journal*, *12*(2), 1755-1766.

Puurula, A. (2016). Scalable Text Mining with Sparse Generative Models. *arXiv preprint arXiv:1602.02332*.

Rehurek, R. (2011). *Scalability of Semantic Analysis in Natural Language Processing* (Doctoral dissertation, Masarykova univerzita, Fakulta informatiky).

Roberts, M., Stewart, B., & Tingley, D. (2014). Navigating the Local Modes of Big Data : The Case of Topic Models. *Scholar.Harvard.Edu*.

Rosen-Zvi, M., Griffiths, T., Steyvers, M., & Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence* (pp. 487-494). AUAI Press.

Sagayam, R., Srinivasan, S., & Roshni, S. (2012). A survey of text mining: Retrieval, extraction and indexing techniques. *International Journal of Computational Engineering Research*, *2*(5).

Sargent, R. G. (2009). Verification and validation of simulation models. In *Simulation Conference (WSC), Proceedings of the 2009 Winter* (pp. 162-176). IEEE.

Sargent, R. G. (2013). Verification and validation of simulation models. *Journal of simulation*, *7*(1), 12-24.

Sathya, R., & Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *Int J Adv Res Artificial Intell*, *2*(2), 34-38.

Shaffer, C. A. (2011). *Data Structures & Algorithm Analysis in Java*. Courier Corporation.

Sharma, H., & Sharma, A. K. (2017). Study and Analysis of Topic Modelling Methods and Tools–A Survey. *American Journal of Mathematical and Computer Modelling*, *2*(3), 84-87.

Singh, J., & Gupta, V. (2017). A systematic review of text stemming techniques. *Artificial Intelligence Review*, *48*(2), 157-217.

Singh, K., & Sinha, A. K. (2013). Face Recognition using Extended Kalman Filter based Machine Learning. *International Journal of Computer Applications*, *66*(16), 43-50.

Speh, J., Muhic, A., & Rupnik, J. (2013). Parameter estimation for the latent Dirichlet allocation.

Sravani, K., & Srinivasu, P. (2014). Comparative study of machine learning algorithm for intrusion detection system. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013* (pp. 189-196). Springer International Publishing.

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

Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. *Handbook of Latent Semantic Analysis*, *427*(7), 424-440.

Sumbaly, R., Vishnusri, N., & Jeyalatha, S. (2014). Diagnosis of Breast Cancer using Decision Tree Data Mining Technique. *International Journal of Computer Applications*, *98*(10), 16–24.

Sutskever, I., Vinyals, O., & Le, Q. V. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems, 4,* 3104–3112.

Taghavifar, H., & Mardani, A. (2014). Application of artificial neural networks for the prediction of traction performance parameters. *Journal of the Saudi Society of Agricultural Sciences*, *13*(1), 35-43.

Tang, J., Zhang, M., & Mei, Q. (2014). " Look Ma, No Hands!" A Parameter-Free Topic Model. *arXiv preprint arXiv:1409.2993*.

Tated, R. R., & Ghonge, M. M. (2015). A survey on text mining-techniques and application. *International Journal of Research in Advent Technology*, *1*, 380-385.

Teh, Y. W., Newman, D., & Welling, M. (2007). A collapsed variational Bayesian inference algorithm for latent Dirichlet. allocation. In *Advances in neural information processing systems 19,* 1353-1360.

Tian, W. D., & Zhao, Y. D. (2014). *Optimized cloud resource management and scheduling: theories and practices*. Burlington, Massachusetts: Morgan Kaufmann.

Tipping, M. E., & Bishop, C. M. (2006). *U.S. Patent No. 7,106,914*. Washington, DC: U.S. Patent and Trademark Office.

Trienekens, J., Hvolby, H. H., Steger-Jensen, K., & Falster, P. (2008). Architectural frameworks for business information system analysis and design. In *Lean Business Systems and Beyond* (pp. 413-421). Springer, Boston, MA.

Tsai, C. F. (2012). Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, *2012*.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, *37*, 141-188.

Varian, H. R. (2014). Big data: New tricks for econometrics. *The Journal of Economic Perspectives*, 3–27.

Veerachai, G., Chandraprakaikul, W., & Kiattisin, S. (2010). An Application of Neural Networks for Forecasting Container Throughput at Bangkok Port. In *Proceedings of the World Congress on Engineering* (Vol. 1).

Vijayarani, S., Ilamathi, M. J., & Nithya, M. (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7-16.

Vulic, I., & Moens, M. F. (2015). Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (Vol. 2, pp. 719-725).

Wallach, H. M., Murray, I., Salakhutdinov, R., & Mimno, D. (2009). Evaluation methods for topic models. In *Proceedings of the 26th annual international conference on machine learning* (pp. 1105-1112). ACM.

Wang, B., Liu, Y., Liu, Z., Li, M., & Qi, M. (2014). Topic selection in latent Dirichlet allocation. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on* (pp. 756-760). IEEE.

Wang, C., & Blei, D. M. (2012). Truncation-free Stochastic Variational Inference for Bayesian Nonparametric Models.

Wang, C., Paisley, J., & Blei, D. (2011, June). Online variational inference for the hierarchical Dirichlet process. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 752-760).

Wang, Y., Bai, H., Stanton, M., Chen, W. Y., & Chang, E. Y. (2009). Plda: Parallel Latent Dirichlet Allocation for large-scale applications. In *Algorithmic Aspects in Information and Management* (pp. 301-314). Springer Berlin Heidelberg.

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

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 681-688).

Xiao, H., & Stibor, T. (2010). Efficient Collapsed Gibbs Sampling for Latent Dirichlet Allocation. In *ACML* (pp. 63–78).

Xing, L., & Paul, M. J. (2018). Diagnosing and Improving Topic Models by Analyzing Posterior Variability.

Xu, X., Hospedales, T. M., & Gong, S. (2017). Discovery of shared semantic spaces for multiscene video query and summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, *27*(6), 1353-1367.

Yan, F., Xu, N., & Qi, Y. (2009). Parallel inference for Latent Dirichlet Allocation on graphics processing units. In *Advances in Neural Information Processing Systems* (pp. 2134–2142).

Yut, L., Zhang, C., Shao, Y., & Cui, B. (2017). LDA: a robust and large-scale topic modeling system. *Proceedings of the VLDB Endowment*, *10*(11), 1406-1417

Zeimpekis, D., & Gallopoulos, E. (2006). TMG: A MATLAB toolbox for generating term-document matrices from text collections. In *Grouping multidimensional data* (pp. 187-210). Springer, Berlin, Heidelberg.

Zhai, K., & Boyd-Graber, J. (2013). Online Latent Dirichlet Allocation with Infinite Vocabulary. *Journal of Machine Learning Research (JMLR), 28*(1), 561-569

Zhang, Y., Sow, D., Turaga, D., & Van der Schaar, M. (2014). A fast online learning algorithm for distributed mining of bigdata. *ACM SIGMETRICS Performance Evaluation Review*, *41*(4), 90–93.

Zhao, W., Chen, J. J., Perkins, R., Liu, Z., Ge, W., Ding, Y., & Zou, W. (2015). A heuristic approach to determine an appropriate number of topics in topic modelling. *BMC bioinformatics*, *16*(13), 2-10.

Zhu, F., Patumcharoenpol, P., Zhang, C., Yang, Y., Chan, J., Meechai, A. & Shen, B. (2013). Biomedical text mining and its applications in cancer research. *Journal of biomedical informatics*, *46*(2), 200-211.

Zikopoulos, P., & Eaton, C. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. New York, USA: McGraw-Hill Osborne Media.

APPENDICES

Appendix I: Python Code for Politics Data Implementing CGS in graphlab Topic Model

The following python code was used to implement LDA using CGS inference method to trend parameters on different datasets. Appendix I presents python code for implementing LDA on politics dataset of the 20Newsgroup dataset with different parameter setting. In the first case, alpha is fixed at 0.1, beta at 0.01 and K iterated in the range 50<K<690 with steps of 5 within the range. In the second case, beta is fixed at 0.01, K is fixed at the value that had minimum perplexity in the case where alpha was fixed at 0.1, beta at 0.01 and alpha iterated within a range of 0.05<alpha<0.3. In the third case, beta and K are fixed at the value that had minimum perplexity in the case where alpha was fixed at 0.1, while beta is iterated within a range of 0.0001<alpha<0.03. In each of the cases, code for plotting the resulting graph is presented. The code concludes by presenting the best values of number of topics (K), beta and alpha.

Python Code

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import fetch_20newsgroups
from collections import defaultdict
import pickle
from operator import itemgetter
import os
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('ggplot')
plt.rcParams['figure.figsize']= (12,8)
# import graphlab
import graphlab as gl
# function to remove \n
```

```
def clean_text(line):
    text = []
    for word in line.split(" "):
        text.append(word.strip())
    return " ".join(text)
def read_data(path=None):
    if not path:
        data = fetch_20newsgroups(shuffle=True,random_state=1,\
                                  remove=('headers', 'footers', 'quotes'),\
                                  categories=['rec.autos','rec.motorcycles'],
                                  subset='train',\
                                  data_home="data/")
        data = [clean_text(line) for line in data.data]
        return data
    else:
        all_lines = []
        with open(FILE_PATH+path, 'r') as afile:
            for line in afile.readlines():
                new_line = line.strip()
                if new_line:
                    all_lines.append(new_line)
        return all_lines
# read the data
data = read_data()
# lets see some list values
for line in data[:5]:
    print(line)
# create the features
g_vectorizer                                                                    =
CountVectorizer(stop_words='english',max_features=2000,max_df=0.95,min_df=0.0
1) # bag of words model
g_count_model = g_vectorizer.fit_transform(data) # generate the vocabulary and the
matrix/BOW
```

```
#http://scikit-
learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.h
tml
g_feature_names = g_vectorizer.get_feature_names()# the vocabilatirs or the features
g_train = np.array(g_count_model.todense())
# this gives the numbetr of rows/documents and the number of columns
g_train.shape
sa_train = gl.SArray(dtype=dict) # the whole of sarray dataset, gl-graphlab
# create sarray data structre for gl , see the dicumentation of gl topic_model create
for row in g_train[:]:
   temp_list = []
   temp_dict = {}
   #sa_train = gl.SArray(data=[{g_feature_names[index]:arow} for index,arow in
enumerate(row)])
   for index,arow in enumerate(row):
      temp_dict[g_feature_names[index]] = arow
   temp_list.append(temp_dict)
   sa_row = gl.SArray(data=temp_list,dtype=dict)
   sa_train = sa_train.append(sa_row)
   temp_list = []
   temp_dict = {}
type(sa_train)
# split data into training ans testing sets see the documentation
g_train_data, g_test_data = gl.text_analytics.random_split(sa_train)

#Case One
alpha = 0.1 #doc_topic_prior
beta = 0.01 #topic_word_prior
graphlab_model_K = defaultdict(list)
for k in range(50,690,5):
   g_m                                                                    =
gl.topic_model.create(g_train_data,num_topics=k,alpha=alpha,beta=beta,num_iterati
ons=100)
```

```
    perpelexity = g_m.evaluate(g_train_data, g_test_data)
    graphlab_model_K[k] = [k,perpelexity['perplexity']]
    print(k,perpelexity)
with open('gl_data/gl_K_politics.pickle','wb') as pickled:
    pickle.dump(graphlab_model_K,pickled)
def get_min_perplexity(dict_values):
    return sorted(dict_values.items(),key=lambda v:v[1][1])[0][0]
# results of above experiment
sorted(graphlab_model_K .items())
x,y = [],[]
for v  in sorted(graphlab_model_K.items(),key=lambda v:v[1]):
    x.append(v[1][0])
    y.append(v[1][1])
plt.plot(x,y,'*-')
plt.show()
```

#Case Two

```
beta = 0.01 #topic_word_prior
num_topics = get_min_perplexity(graphlab_model_K)
graphlab_model_A = defaultdict(list)
for a in np.linspace(0.05, 0.3,num=100):
    g_m =
gl.topic_model.create(g_train_data,num_topics=num_topics,alpha=a,beta=beta,num_
iterations=50)
    perpelexity = g_m.evaluate(g_train_data, g_test_data)
    graphlab_model_A[str(a)] = [a,perpelexity['perplexity']]
    print(a,perpelexity)
with open('gl_data/gl_A_politics.pickle','wb') as pickled:
    pickle.dump(graphlab_model_A,pickled)
# Here are the results of the experiment
sorted(graphlab_model_A.items())
x,y = [],[]
for v  in sorted(graphlab_model_A.items(),key=lambda v:v[1]):
```

```
    x.append(v[1][0])
    y.append(v[1][1])
plt.plot(x,y,'*-')
plt.show()

#Case Three
graphlab_model_B = defaultdict(list)
num_topics = get_min_perplexity(graphlab_model_K)
alpha = float(get_min_perplexity(graphlab_model_A))
for b in np.linspace(0.0001, 0.03,num=19):
    g_m =
gl.topic_model.create(g_train_data,num_topics=num_topics,alpha=alpha,beta=b,nu
m_iterations=50)
    perpelexity = g_m.evaluate(g_train_data, g_test_data)
    graphlab_model_B[str(b)] = [b,perpelexity['perplexity']]
    print(b,perpelexity)
with open('gl_data/gl_B_politics.pickle','wb') as pickled:
    pickle.dump(graphlab_model_B,pickled)
# Here are the results of the above experiment
sorted(graphlab_model_B.items())
x,y = [],[]
for v in sorted(graphlab_model_B.items(),key=lambda v:v[1]):
    x.append(v[1][0])
    y.append(v[1][1])
plt.plot(x,y,'*-')
plt.show()
topics = get_min_perplexity(graphlab_model_K)
beta = get_min_perplexity(graphlab_model_B)
alpha = get_min_perplexity(graphlab_model_A)
print("number of topics: {}, Beta: {}, and Alpha: {}".format(topics,beta,alpha))
```

Appendix II: Code for Implementing QTAM and Its Run-Time on 20Newsgroup Dataset

The following python code was used to implement LDA using QTAM topics (K) inference method. The code is further enhanced to compute and output total running time.

Python Code

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import fetch_20newsgroups
from collections import defaultdict
import pickle
from operator import itemgetter
import os
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from __future__ import division
import time
plt.style.use('ggplot')
plt.rcParams['figure.figsize']= (12,8)
# import graphlab
import graphlab as gl
# function to remove \n
def clean_text(line):
    text = []
    for word in line.split(" "):
        text.append(word.strip())
    return " ".join(text)
def read_data(path=None):
    if not path:
        data = fetch_20newsgroups(shuffle=True,random_state=1,\
```

```
                        remove=('headers', 'footers', 'quotes'),\
                        categories=['rec.autos','rec.motorcycles'],
                        subset='train',\
                        data_home="data/")
        data = [clean_text(line) for line in data.data]
        return data
    else:
        all_lines = []
        with open(FILE_PATH+path, 'r') as afile:
            for line in afile.readlines():
                new_line = line.strip()
                if new_line:
                    all_lines.append(new_line)
        return all_lines
# read the data
data = read_data()
# lets see some list values
for line in data[:5]:
    print(line)
# the total number of records
len(data)
# create the features
g_vectorizer =
CountVectorizer(stop_words='english',max_features=2000,max_df=0.95,min_df=0.0
1) # bag of words model
g_count_model = g_vectorizer.fit_transform(data) # generate the vocabulary and the
matrix/BOW
#http://scikit-
learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.h
tml
g_feature_names = g_vectorizer.get_feature_names()# the vocabilatirs or the features
g_train = np.array(g_count_model.todense())
```

```
# this gives the number of rows/documents and the number of columns
g_train.shape
sa_train = gl.SArray(dtype=dict) # the whole of sarray dataset, gl-graphlab
# create sarray data structre for gl , see the dicumentation of gl topic_model create
for row in g_train[:]:
    temp_list = []
    temp_dict = {}
    #sa_train = gl.SArray(data=[{g_feature_names[index]:arow} for index,arow in
enumerate(row)])
    for index,arow in enumerate(row):
        temp_dict[g_feature_names[index]] = arow
    temp_list.append(temp_dict)
    sa_row = gl.SArray(data=temp_list,dtype=dict)
    sa_train = sa_train.append(sa_row)
    temp_list = []
    temp_dict = {}
type(sa_train)
# split data into training ans testing sets see the documentation
g_train_data, g_test_data = gl.text_analytics.random_split(sa_train)
start_time = time.time()
keys = []
values = []
alpha = 0.1 #doc_topic_prior
beta = 0.01 #topic_word_prior
graphlab_model_K = defaultdict(list)
for k in range(300,450,50):
    g_m                                                                    =
gl.topic_model.create(g_train_data,num_topics=k,alpha=alpha,beta=beta,num_iterati
ons=100)
    perpelexity = g_m.evaluate(g_train_data, g_test_data)
    graphlab_model_K[k] = [k,perpelexity['perplexity']]
for key,value in graphlab_model_K.items():
```

```
    keys.append(key)
    values.append(value[1])
K1 = keys[0]
K2 = keys[1]
K3 = keys[2]
P1 = values[0]
P2 = values[1]
P3 = values[2]
f = ((K2**2) - K1)/((K1*K2)*(K1-K2))
h = (f+(K1**4))/(K1**2)
d = ((K1**2)*P2)-((K2**2)*P1)
e = d/(K1*K2*(K1-K2))
g = (P1-(e*K1))/(K1**2)
c = (g*(K3**2)+(e*K3)-K3)/(d*(K3**2)-(f*K3)-1)
a = g - (c*h)
b = e+(c*f)
K = int(-b/(2*a))
alpha = 50/K
beta = 0.01
end_time = time.time()
running_time = end_time -start_time
print('Total running time {} seconds'.format(running_time))
print("number of topics: {}, Beta: {}, and Alpha: {}".format(K,beta,alpha))
```

Appendix III: Perplexity against Alpha for 0.05< alpha<0.3 Raw Data

This table presents raw data obtained in the experiment where Number of topics (K) was fixed at 420, beta 0.01 and alpha starting at 0.05 to a maximum value of 0.3. Figure 5.5 represents a graph for the results.

Alpha	Perplexity
0.05	623.1029
0.053	637.2744
0.055	624.1588
0.058	624.65298
0.06	616.738
0.063	628.2885
0.065	607.568
0.068	612.6599
0.07	618.252
0.073	621.521
0.075	609.385
0.078	615.2748
0.08	614.5397
0.083	613.2808
0.085	608.3915
0.088	614.7825
0.09	617.07488
0.093	616.9546
0.095	612.026
0.098	612.50069
0.101	614.3863
0.103	609.5129
0.106	612.5379
0.108	609.5099
0.111	610.803
0.113	609.478
0.116	613.983
0.118	613.0676
0.121	613.9686
0.123	615.1246
0.125	609.375
0.128	604.1125
0.131	610.1488
0.133	613.199

0.136	610.5635
0.138	611.678
0.141	610.2617
0.143	615.4925
0.146	608.646
0.148	607.029
0.151	607.712
0.154	609.451
0.156	615.691
0.159	610.35
0.161	617.5896
0.164	608.307
0.166	607.792
0.169	610.3646
0.171	612.7717
0.174	614.529
0.176	611.5698
0.179	615.312
0.181	616.075
0.184	610.098
0.186	608.826
0.189	607.954
0.191	618.189
0.194	608.99485
0.196	614.386
0.199	610.9306
0.202	608.1236
0.204	611.08278
0.207	612.3045
0.209	615.385
0.212	612.4305
0.214	607.303
0.217	609.143
0.219	621.688
0.222	611.5609
0.224	608.145
0.227	609.7659
0.229	616.103
0.232	610.729
0.234	624.179
0.237	611.0737
0.239	614.9109
0.242	620.448

0.244	610.683
0.247	608.0335
0.249	611.845
0.252	620.71776
0.255	613.674786
0.257	615.763
0.26	616.74178
0.262	611.257259
0.265	614.51601
0.267	615.0019163
0.27	608.377619
0.272	616.3305858
0.275	616.824843
0.277	619.2071
0.28	618.5684
0.282	618.1856
0.285	613.75134
0.287	615.411475
0.29	617.36821
0.292	613.599649
0.295	613.659436
0.297	622.66469
0.3	614.6957

Appendix IV: Perplexity against Number of Topics (K) for 50< K<10,000 Raw Data

This table presents raw data obtained in the experiment where alpha was set at 0.1 and beta at 0.01 and K starting at 50 to a maximum value of 10,000. Figure 5.7 represents a graph for the results.

K	Perplexity
50	761.9409227
150	631.1640023
250	614.0658502
350	596.3329169
450	602.2221942
550	615.5426122
650	610.4311755
750	608.5421276
850	617.5921429
950	619.2309949
1050	630.5567232
1150	625.0702442
1250	631.1206141
1350	641.6331048
1450	647.042073
1550	640.2347766
1650	653.9182584
1750	653.6303033
1850	656.8801837
1950	659.6291967
2050	662.0738103
2150	665.8956666
2250	663.424717
2350	672.429812
2450	674.2233097
2550	677.2051947
2650	685.9710737
2750	687.1908529
2850	684.1712787
2950	692.2030346
3050	686.0923802
3150	693.6923639
3250	693.2423567
3350	693.744548

3450	695.9436744
3550	699.8504712
3650	706.3618151
3750	704.5312076
3850	707.5833391
3950	708.3097966
4050	717.6608628
4150	704.8862352
4250	711.811489
4350	720.2975242
4450	720.4512775
4550	719.3090505
4650	723.4267601
4750	727.6883623
4850	728.1151745
4950	726.840862
5050	728.9945483
5150	727.652409
5250	729.1268564
5350	728.2332244
5450	732.7097764
5550	735.5904789
5650	740.9573477
5750	743.9397861
5850	744.9813304
5950	741.3658223
6050	742.7528552
6150	748.8725117
6250	745.6743216
6350	752.6788712
6450	749.0204149
6550	748.4003287
6650	752.3849447
6750	752.2183392
6850	758.4250679
6950	757.0865439
7050	758.4857649
7150	757.8337685
7250	756.3240376
7350	765.6708267
7450	762.5231628
7550	763.394985
7650	763.3004042

7750	758.7045687
7850	766.4042687
7950	765.9567064
8050	766.4452404
8150	769.9833806
8250	771.1469135
8350	769.3931837
8450	767.8225088
8550	772.2158585
8650	773.0870304
8750	774.5776306
8850	774.6296452
8950	770.9270217
9050	773.6041483
9150	775.7343136
9250	780.5313666
9350	774.7858699
9450	775.9210763
9550	779.0343078
9650	771.9100828
9750	780.7432353
9850	781.1542776
9950	784.0012065

Appendix V: Perplexity against Topics for 50< K<690 Raw Data

This table presents raw data obtained in the experiment where alpha was set at 0.1 and beta at 0.01 and K starting at 50 to a maximum value of 690. Figure 5.8 represents a graph for the results.

Topics	Perplexity
50	892.0474462
55	888.7235656
60	884.9219076
65	874.5466071
70	841.588147
75	841.3128067
80	823.1291457
85	823.9230082
90	803.2576786
95	790.4250666
100	796.1038392
105	787.661577
110	778.0114159
115	756.4945329
120	754.0069634
125	753.3758491
130	749.1557003
135	744.8250831
140	739.296943
145	722.6011388
150	732.8519674
155	721.1026048
160	702.7902012
165	710.7733436
170	711.1476818
175	706.4511059
180	700.6387815
185	703.6263503
190	697.5271234
195	700.3868532
200	692.2986724
205	694.1341717
210	684.2202691
215	679.1089958

220	691.6478409
225	679.4891369
230	682.5428175
235	678.8628427
240	667.7681338
245	666.9746926
250	666.9187095
255	660.4922576
260	665.5765973
265	667.9559172
270	661.2899191
275	654.6773962
280	657.3466464
285	662.823087
290	662.3238124
295	649.1795366
300	655.2765425
305	654.835065
310	652.0573736
315	658.7103751
320	653.6166528
325	641.3503363
330	652.5831196
335	649.3706455
340	639.089719
345	649.9189376
350	653.8573411
355	635.9778113
360	647.435318
365	645.0612796
370	644.7364661
375	635.3577601
380	639.4819242
385	635.2753213
390	635.3357381
395	639.089255
400	635.4925573
405	630.8169312
410	640.2594826
415	633.1863486
420	628.3613949
425	625.5933557
430	635.11632

435	633.8972809
440	627.0416437
445	645.8157455
450	632.9020958
455	624.364932
460	627.5076786
465	626.8130807
470	625.9051823
475	632.4394091
480	634.3152033
485	636.9542074
490	627.6594481
495	624.6786805
500	632.5264997
505	627.6265435
510	637.089481
515	625.3987009
520	620.6298218
525	626.5229312
530	631.6069441
535	615.1932846
540	619.0961834
545	629.3213024
550	632.6380318
555	626.7168088
560	627.7152003
565	625.7031643
570	618.9447161
575	631.5448649
580	629.0929125
585	627.4755332
590	630.4102679
595	626.1656815
600	627.5337164
605	630.0979766
610	621.7579535
615	628.7025963
620	636.1060807
625	621.0690096
630	634.6533015
635	626.6116602
640	634.56549
645	623.3730247

650	622.2030076
655	621.3577916
660	631.306407
665	628.3263475
670	621.9342561
675	625.2134981
680	621.9262543
685	622.1706704

Appendix VI: Regression Analysis R-Code

Appendix VI presents R-Code that was used for quadratic regression analysis. The variables Topics and Perplexity were R structures created to hold numbers of topics analysis results and Perplexity analysis results for respective datasets. A new variable Topics2 was created and defined as the square of Topics. The data and results after this code implemented in R are indicated in Appendix VII and Appendix VIII.

R-Code

```
> A <- structure(list(Topics = c(x-axis dataset put here separated by commas),
+                     Perplexity = c(Corresponding y-axis dataset put here separated by commas)), .Names = c("Topics", "Perplexity"),
+               class = "data.frame")
>  attach(A)
> names(A)
> Topics2 <- Topics^2
> quadratic.model <-lm(Perplexity ~ Topics + Topics2)
> plot(Topics, Perplexity, pch=16, xlab = "Topics (s)", ylab = "Perplexity", cex.lab = 1.3, col = "blue")
> topicsvalues <- seq(0, 700, 5)
> predictedperplexity <- predict(quadratic.model,list(Topics=topicsvalues, Topics2=topicsvalues^2))
> plot(Topics, Perplexity, pch=16, xlab = "Topics (s)", ylab = "Perplexity", cex.lab = 1.3, col = "blue")
> lines(topicsvalues, predictedperplexity, col = "darkgreen", lwd = 3)
> summary(quadratic.model)
```

Appendix VII: Regression Data and Results for Autos dataset

Appendix VII presents data used for computing linear and quadratic regression statistics for Autos dataset and the results obtained.

Topics	Topics2	Perplexity
50	2500	892.047446
55	3025	888.723566
60	3600	884.921908
65	4225	874.546607
70	4900	841.588147
75	5625	841.312807
80	6400	823.129146
85	7225	823.923008
90	8100	803.257679
95	9025	790.425067
100	10000	796.103839
105	11025	787.661577
110	12100	778.011416
115	13225	756.494533
120	14400	754.006963
125	15625	753.375849
130	16900	749.1557
135	18225	744.825083
140	19600	739.296943
145	21025	722.601139
150	22500	732.851967
155	24025	721.102605
160	25600	702.790201
165	27225	710.773344
170	28900	711.147682
175	30625	706.451106
180	32400	700.638782
185	34225	703.62635
190	36100	697.527123
195	38025	700.386853
200	40000	692.298672
205	42025	694.134172
210	44100	684.220269
215	46225	679.108996
220	48400	691.647841

225	50625	679.489137
230	52900	682.542818
235	55225	678.862843
240	57600	667.768134
245	60025	666.974693
250	62500	666.91871
255	65025	660.492258
260	67600	665.576597
265	70225	667.955917
270	72900	661.289919
275	75625	654.677396
280	78400	657.346646
285	81225	662.823087
290	84100	662.323812
295	87025	649.179537
300	90000	655.276542
305	93025	654.835065
310	96100	652.057374
315	99225	658.710375
320	102400	653.616653
325	105625	641.350336
330	108900	652.58312
335	112225	649.370645
340	115600	639.089719
345	119025	649.918938
350	122500	653.857341
355	126025	635.977811
360	129600	647.435318
365	133225	645.06128
370	136900	644.736466
375	140625	635.35776
380	144400	639.481924
385	148225	635.275321
390	152100	635.335738
395	156025	639.089255
400	160000	635.492557
405	164025	630.816931
410	168100	640.259483
415	172225	633.186349
420	176400	628.361395
425	180625	625.593356

430	184900	635.11632
435	189225	633.897281
440	193600	627.041644
445	198025	645.815746
450	202500	632.902096
455	207025	624.364932
460	211600	627.507679
465	216225	626.813081
470	220900	625.905182
475	225625	632.439409
480	230400	634.315203
485	235225	636.954207
490	240100	627.659448
495	245025	624.678681
500	250000	632.5265
505	255025	627.626544
510	260100	637.089481
515	265225	625.398701
520	270400	620.629822
525	275625	626.522931
530	280900	631.606944
535	286225	615.193285
540	291600	619.096183
545	297025	629.321302
550	302500	632.638032
555	308025	626.716809
560	313600	627.7152
565	319225	625.703164
570	324900	618.944716
575	330625	631.544865
580	336400	629.092913
585	342225	627.475533
590	348100	630.410268
595	354025	626.165681
600	360000	627.533716
605	366025	630.097977
610	372100	621.757954
615	378225	628.702596
620	384400	636.106081
625	390625	621.06901
630	396900	634.653302

635	403225	626.61166
640	409600	634.56549
645	416025	623.373025
650	422500	622.203008
655	429025	621.357792
660	435600	631.306407
665	442225	628.326348
670	448900	621.934256
675	455625	625.213498
680	462400	621.926254
685	469225	622.17067

SUMMARY OUTPUT - AUTOS: LINEAR

Regression Statistics	
Multiple R	0.819
R Square	0.671
Adjusted R Square	0.668
Standard Error	37.551
Observations	128.000

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1.000	362002.412	362002.412	256.719	0.000
Residual	126.000	177673.998	1410.111		
Total	127.000	539676.411			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	777.085	7.390	105.157	0.000	762.461	791.709
X Variable 1	-0.288	0.018	-16.022	0.000	-0.323	-0.252

SUMMARY OUTPUT - AUTOS: QUADRATIC

Regression Statistics	
Multiple R	0.962
R Square	0.926
Adjusted R Square	0.925
Standard Error	17.864
Observations	128.000

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	2.000	499786.365	249893.182	783.069	0.000
Residual	125.000	39890.046	319.120		
Total	127.000	539676.411			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	885.562	6.294	140.703	0.000	873.106	898.018
X Variable 1	-1.078	0.039	-27.660	0.000	-1.155	-1.001
X Variable 2	0.001	0.000	20.779	0.000	0.001	0.001

Appendix VIII: Regression Data and Results for NYTimes dataset

Appendix VIII presents data used for computing linear and quadratic regression statistics for NYTimes dataset and the results obtained.

Topics	Topics2	Perplexity
50	2500	4752.014885
55	3025	4731.274174
60	3600	4656.213086
65	4225	4588.428586
70	4900	4546.267976
75	5625	4500.649228
80	6400	4514.652265
85	7225	4474.033398
90	8100	4438.261129
95	9025	4426.076944
100	10000	4390.220895
105	11025	4362.501848
110	12100	4338.128804
115	13225	4347.653544
120	14400	4290.396142
125	15625	4273.320488
130	16900	4241.851912
135	18225	4200.362854
140	19600	4206.52017
145	21025	4210.508221
150	22500	4151.657846
155	24025	4118.711205
160	25600	4203.216947
165	27225	4128.308811
170	28900	4131.493492
175	30625	4121.200714
180	32400	4121.645617
185	34225	4085.829811
190	36100	4083.711663
195	38025	4082.314807
200	40000	4053.626764
205	42025	4054.090041
210	44100	4042.686273
215	46225	4047.345087
220	48400	4032.59146

225	50625	4015.676666
230	52900	4020.523155
235	55225	4009.952048
240	57600	4009.073315
245	60025	3989.898293
250	62500	3974.722084
255	65025	3969.091263
260	67600	3983.552648
265	70225	3955.252919
270	72900	3950.841501
275	75625	3931.786225
280	78400	3948.439692
285	81225	3957.304585
290	84100	3923.181521
295	87025	3920.54147
300	90000	3889.826644
305	93025	3897.858602
310	96100	3890.372019
315	99225	3896.352189
320	102400	3896.415863
325	105625	3871.44858
330	108900	3902.093069
335	112225	3884.564573
340	115600	3871.637269
345	119025	3868.243275
350	122500	3862.19632
355	126025	3881.245105
360	129600	3878.689568
365	133225	3862.749417
370	136900	3871.289343
375	140625	3858.2416
380	144400	3847.746571
385	148225	3820.593533
390	152100	3825.741108
395	156025	3827.940825
400	160000	3824.223193
405	164025	3813.797172
410	168100	3820.840503
415	172225	3799.014171
420	176400	3804.284909
425	180625	3830.086297

430	184900	3792.481465
435	189225	3820.553704
440	193600	3811.609348
445	198025	3805.598803
450	202500	3808.596526
455	207025	3809.926073
460	211600	3801.997331
465	216225	3801.590672
470	220900	3778.047778
475	225625	3781.055559
480	230400	3800.607263
485	235225	3772.163247
490	240100	3798.013513
495	245025	3792.764711
500	250000	3783.106267
505	255025	3802.700707
510	260100	3765.45116
515	265225	3800.351739
520	270400	3794.05096
525	275625	3787.521773
530	280900	3782.888784
535	286225	3770.949595
540	291600	3795.250728
545	297025	3770.651456
550	302500	3772.546027
555	308025	3809.39885
560	313600	3780.512083
565	319225	3800.288029
570	324900	3772.776164
575	330625	3769.12844
580	336400	3761.397336
585	342225	3771.705362
590	348100	3790.413006
595	354025	3768.288428
600	360000	3760.070515
605	366025	3797.308561
610	372100	3778.525236
615	378225	3768.614786
620	384400	3757.558636
625	390625	3799.087924
630	396900	3762.782924

635	403225	3791.42533
640	409600	3779.05643
645	416025	3791.640235
650	422500	3770.909938
655	429025	3787.151721
660	435600	3773.531211
665	442225	3766.834684
670	448900	3794.351168
675	455625	3795.032346
680	462400	3766.312353
685	469225	3779.025854
690	476100	3771.382721
695	483025	3779.597477

SUMMARY OUTPUT – NY-TIMES: LINEAR

Regression Statistics	
Multiple R	0.85712757
R Square	0.734667671
Adjusted R Square	0.732594762
Standard Error	121.8004922
Observations	130

ANOVA

	df	***SS***	***MS***	***F***	***Significance F***
Regression	1	5257857.552	5257857.552	354.414	0.000
Residual	128	1898926.066	14835.360		
Total	129	7156783.619			

	Coefficients	***Standard Error***	***t Stat***	***P-value***	***Lower 95%***	***Upper 95%***
Intercept	4354.636	23.746	183.382	0.000	4307.650	4401.622
X Variable 1	-1.072	0.057	-18.826	0.000	-1.184	-0.959

SUMMARY OUTPUT - NY-TIMES: QUADRATIC

Regression Statistics	
Multiple R	0.978319683
R Square	0.957109403
Adjusted R Square	0.95643396
Standard Error	49.16297203
Observations	130

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	2	6849824.896	3424912.448	1417.011	0.000
Residual	127	306958.723	2416.998		
Total	129	7156783.619			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	4718.567	17.116	275.683	0.000	4684.698	4752.437
X Variable 1	-3.690	0.105	-35.286	0.000	-3.897	-3.483
X Variable 2	0.004	0.000	25.664	0.000	0.003	0.004

Appendix IX: Expert Opinion Survey Letter to Respondents

GEOFFREY MARIGA WAMBUGU
P.O. BOX 7459-01000
THIKA, KENYA.
Email: *gmariga@mut.ac.ke*

RE: LETTER OF INTRODUCTION

I am a PhD (Information Technology) student at **Jomo Kenyatta University of Agriculture and Technology**, Kenya carrying out a study on deployment of topic modelling in large text datasets. The aim of this questionnaire is to seek for expert opinion concerning a developed text analytics deployment framework (attached) on two dimensions: Completeness and Dependability.

You have been identified to participate in this study as a respondent. Kindly provide your responses as comprehensively as possible. I would want to assure you that the information given will remain confidential and it will be used for the purposes of this research only.

Thank you.

Yours Faithfully,

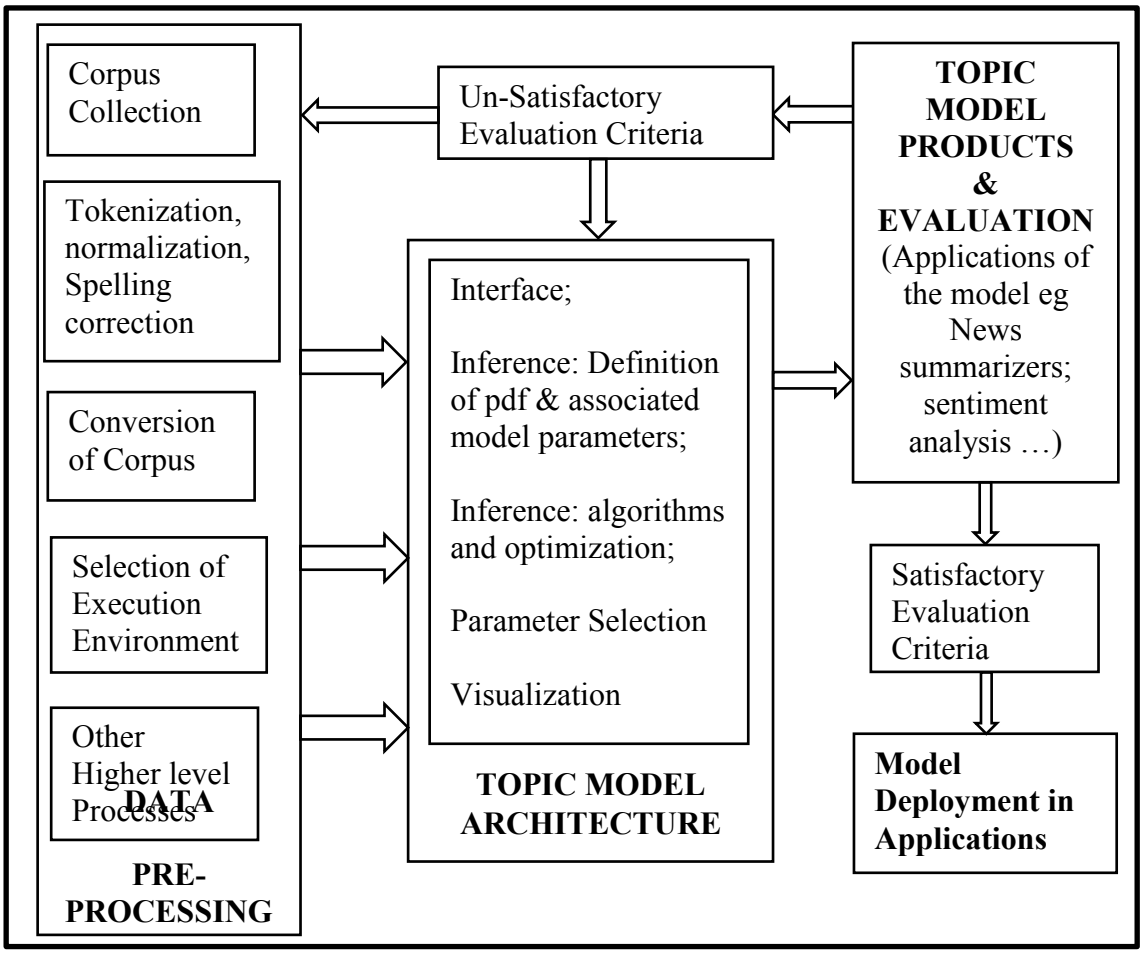
Geoffrey Mariga Wambugu
Student Registration Number:
CS481-4692/2014

Appendix X: Generic Topic Model Deployment Framework

(sent with the letter to respondents)

The topic modelling framework attached is conceptualised as the process that starts with pre-processing of raw data which is fed into a statistical process called topic model architecture to give an output of topic model products. This process happens in an environmental context. Both the statistical process and environmental context influences the quality of output. The statistical process concerns determination of probability distribution functions to describe the data. These distribution functions are guided by some parameters which determine the quality of the output.

The framework identifies various components of a topic modelling task and their interactions. The figure simplifies the conception of topic modelling and helps data science researchers and topic model application developers easily have a holistic view.


Corpus Collection
Tokenization, normalization, Spelling correction
Conversion of Corpus
Selection of Execution Environment
Other Higher level Processes
DATA
PRE-PROCESSING
Un-Satisfactory Evaluation Criteria
Interface;
Inference: Definition of pdf & associated model parameters;
Inference: algorithms and optimization;
Parameter Selection
Visualization
TOPIC MODEL ARCHITECTURE
TOPIC MODEL PRODUCTS & EVALUATION (Applications of the model eg News summarizers; sentiment analysis …)
Satisfactory Evaluation Criteria
Model Deployment in Applications

Appendix XI: Questionnaire for Expert Opinion

A. General Information

1. Please state your highest academic qualification.

a) Bachelor's Degree ☐ b) Master's Degree ☐ c) PhD ☐ d) Others ☐

2. If Others in question one above, specify

_____

3. What is your field of specialization?

_____

4. How do you rank your knowledge of text analytics? Please tick appropriately.

Very Poor	Above Average	Average	Above Average	Excellent

5. Please state your years of experience.

0-2	3-5	6-8	9-10	Above 10

6. Do you know of any comprehensive framework that guides development and deployment of text analytics applications?

a) Yes ☐ b) No ☐

7. If yes in question 6 above, which one and has it been effective in development and deployment of text analytics applications?

_____

_____

B. Relevance and completeness of Framework constructs

8. The following framework constructs represents important aspects which must be precisely determined in the development and deployment of text analytics applications. Please state your agreement with this statement by selecting appropriately.

	Not Important	**Slightly Important**	**Moderately Important**	**Important**	**Very Important**
Data pre-processing					
User Interface					
Selection of statistical model					
Selection of model parameters					
Expected products					

9. In your opinion, do you think the framework has covered all the needs of developing text analytics applications?

a) Yes ☐ b) No ☐

10. Please indicate your level of agreement with the following statement by ticking the appropriate box. KEY: Strongly Disagree (SD); Disagree (D); Undecided (U); Agree (A); Strongly Agree (SA)

STATEMENT	**SD**	**D**	**U**	**A**	**SA**
This framework adequately addresses the task of developing and implementing text analytics applications.					

Thank you for your time and responses

Appendix XII: Sample Individual Response

Appendix XII presents a sampled individual response which was obtained in the expert opinion survey.

- Collector: Social Media Post 1 (Facebook Link)
- Started: Tuesday, June 12, 2018 11:22:58 AM
- Last Modified: Tuesday, June 12, 2018 11:29:14 AM
- Time Spent: 00:06:15

Page 1

Q1

Please state your highest academic qualification?

- PhD

Q2

If Others in Q1 above, please specify

Respondent skipped this question

Q3

What is your field of specialization?

Respondent skipped this question

Q4

How do you rank your knowledge of text analytics? Please tick appropriately.

- (no label)

Average

Q5

Please state your years of experience.

- (no label)

6-8 Years

Q6

Do you know of any comprehensive framework that guides development and deployment of text analytics applications?

- No

Q7

If yes in Q6 above, which one and has it been effective in development and deployment of text analytics applications?

Respondent skipped this question

Q8

The following framework constructs represents important aspects which must be precisely determined in the development and deployment of text analytics applications. Please state your agreement with this statement by selecting appropriately.

- Data preprocessing

  Important
- User Interface

  Important
- Selection of statistical model

  Very Important
- Selection of model parameters

  Very Important
- Expected products

  Moderately Important

Q9

In your opinion, do you think the framework has covered all the needs of developing text analytics applications?

- Yes

Q10

Please indicate your level of agreement with the following statement by ticking the appropriate button.

- This framework adequately addresses the task of developing and implementing text analytics applications.

  Agree